Master in Intelligent Interactive Systems
Universitat Pompeu Fabra

Robustness in reinforcement learning under task-uncertainty

Svetlichnyi Nikodim Aleksandrovich

Supervisor: Dr. Miguel Calvo-Fullana

July 2025

Master in Intelligent Interactive Systems

Universitat Pompeu Fabra

Robustness in reinforcement learning under task-uncertainty

Svetlichnyi Nikodim Aleksandrovich

Supervisor: Dr. Miguel Calvo-Fullana

July 2025

Contents

Abstract

Reinforcement learning (RL) agents often face challenges in real-world scenarios where the task is not known in advance. This thesis tackles the problem of task-uncertainty by developing agents that can identify and adapt to the current objective in real-time, using only reward signals as feedback. Instead of a monolithic meta-policy, we propose a modular framework based on a committee of pre-trained "expert" policies, each specialized for a single known task.

We develop and analyze two distinct online adaptation mechanisms: a "Dual-Lambda" algorithm, derived from a game-theoretic max-min formulation using Lagrangian duality, which finds a robust policy mixture and offers formal guarantees; a pragmatic Predictive Control (MPC-style) algorithm that selects the best expert at each step through short-horizon simulations in the true environment.

The performance of these algorithms is rigorously evaluated in a custom 2D navigation environment through a three-phase protocol of increasing complexity, culminating in a zero-shot generalization test with novel, unseen obstacle geometries. The results demonstrate that both approaches significantly outperform baseline methods, successfully adapting to the active task. The analysis reveals a trade-off: the Dual-Lambda method provides inherent conservatism and theoretical robustness, while the predictive approach offers greater practical flexibility and emergent behaviors, such as autonomously assigning specialized roles to experts in complex scenarios.

A crucial finding is that the performance of both algorithms is fundamentally bounded by the expressive capacity of the initial expert policy set, highlighting that while the adaptation mechanism is critical, its success is contingent on the diversity of the underlying skills. This work provides a comprehensive analysis of two modular solutions for task-uncertain RL and establishes a foundation for developing more flexible and robust autonomous systems.

Chapter 1

Introduction

1.1 Introduction to the Problem

This thesis addresses the problem of designing reinforcement learning (RL) policies in settings where the agent does not know in advance what task it will have to perform. Unlike classical RL settings, where the environment and the agent's goal are fixed, this thesis assumes that the agent is faced with many possible tasks (or variations of one task) and must not only learn to perform them, but also quickly determine which of them is relevant at the moment.

It is known that classical RL methods [1] allow training policies on a variety of heterogeneous environments to increase their robustness in real-world settings. However, such approaches require significant computational resources and complex neural network architectures, which limits their practical application. This thesis explores alternative methods that provide adaptability to task uncertainty at lower computational costs. Moreover, the proposed approach allows the agent to dynamically switch between tasks even in settings where their parameters (e.g., the agent's target position) are unknown in advance.

The relevance of this approach is due to its potential application in robotics [2], autonomous systems [3] and adaptive control [4], where the agent must solve various problems depending on the context. For example, a robot operating in a variable environment may need to adapt to different types of surfaces (asphalt, sand, grass), each of which requires its own movement strategy. Classical RL methods involve preliminary training for all of the possible scenarios, which is ineffective in unpredictable conditions and especially in transitions between them. In contrast, the proposed approach is based on a paradigm in which the agent not only performs tasks, but also determines which task needs to be solved, using rewards as a source of information to identify the current goal.

1.2 Objectives

The primary research question of this thesis is to determine whether a reinforcement learning agent can be taught to effectively identify and adapt to its designated task in real-time under conditions of uncertainty. To address this question, we posit a central hypothesis: *the mechanism introduced in this paper demonstrates strong capability to recognize and adjust to its assigned task within environments characterized by task uncertainty, and it surpasses traditional baseline methods in both efficiency and adaptability under uncertain conditions.*

To systematically test this hypothesis and structure the research, the following objectives are established:

1. To develop an algorithmic framework that allows an RL agent to dynamically identify and adapt to the current task by analyzing reward signals.
2. To conduct a comparative analysis of the proposed method against baseline MetaRL approache in a simulated environment with task uncertainty.
3. To evaluate the computational efficiency and robustness of the developed approach, demonstrating its advantages for resource-constrained applications.

Successfully achieving these objectives will provide a comprehensive answer to the main research question and will demonstrate the practical viability and potential advantages of our proposed approach for building more flexible and efficient autonomous agents.

1.3 State of the Art

Our research on task-uncertain reinforcement learning does not exist in a vacuum, rather it is positioned at the intersection of several mature and rapidly evolving fields of study. This section provides an overview of these domains, highlighting the foundational concepts that inform our methodology and clarifying the unique contributions of this thesis.

Meta-Reinforcement Learning (MetaRL) [5, 6, 7] is a primary field addressing similar challenges, it focuses on developing agents that can "learn to learn" [8] and rapidly adapt their policies to new tasks with minimal data. Generally, MetaRL methods aim to acquire a generalized knowledge base across a distribution of tasks during a meta-training phase, which can then be fine-tuned for a specific task. Prominent approaches often rely on learning a latent context vector that captures task-specific information, which then modulates a single, general policy [9, 10]. For instance, some methods explicitly decompose reward and dynamics functions to better infer this context, thereby improving adaptation, especially when both aspects of the environment change simultaneously [11]. While our work shares the overarching goal of rapid adaptation, our methodology diverges fundamentally. Instead of training a single, complex meta-policy that implicitly encodes task information, we propose a more modular and interpretable framework: we train a set of distinct "expert" policies, one for each potential task, and develop a separate, lightweight mechanism to dynamically select the appropriate expert during execution based on direct environmental feedback.

Our approach is also deeply rooted in the formal study of uncertainty in reinforcement learning. Robust agents must effectively manage uncertainty, which is often categorized into two types: aleatoric (inherent environmental stochasticity) and epistemic (the agent's lack of knowledge) [12]. A significant body of work uses estimates of epistemic uncertainty, often derived from methods like Bootstrapped DQN [13] or MC-Dropout [14], to drive more efficient exploration. This thesis reframes the application of this concept. Instead of using uncertainty to guide exploration of the state-action space, we leverage the reward signal itself as a tool to reduce the agent's epistemic uncertainty about the current task identity. The formal taxonomy [12] was instrumental in formulating our core hypothesis that rewards can serve this dual purpose: simultaneously acting as a performance metric and an information source for task inference.

Furthermore, our methodology draws theoretical inspiration from Constrained and Safe Reinforcement Learning (CRL) [15, 16, 17]. This domain focuses on algorithms that maximize a reward function while adhering to a set of predefined constraints, which is critical for real-world applications. Seminal theoretical results have shown that despite the non-convex nature of many CRL problems, they often possess a zero duality gap [18]. This allows for the development of effective primal-dual algorithms that can find optimal solutions by automatically balancing objectives and constraints without manual weight tuning [19, 20, 21, 22]. More recent advancements in Safe RL have introduced methods that provide formal safety guarantees, for example, through the ergodic relaxation of probabilistic invariants [23] or by augmenting the state space with Lagrange multipliers to directly embed constraint satisfaction into the policy dynamics [24]. While our work does not formulate task uncertainty as a classical CRL problem, it is inspired by its principles. The mathematical rigor of CRL, which enables precise adaptation to complex requirements, supports our premise that a similarly precise adaptation to an unknown task is feasible. Specifically, the idea of using entropy metrics to assess an agent's confidence, proposed in the context of safety guarantees [23], is repurposed in our framework to evaluate the certainty of task hypotheses and guide the switching strategy.

Finally, our work incorporates principles from research aimed at improving learning efficiency, particularly from the domain of Action Advising. These methods seek to accelerate learning by making judicious use of limited resources, such as an expert demonstrator. For example, some algorithms allow the agent to request advice only in states of high epistemic uncertainty, thus optimizing the use of the expert's budget [25]. Related approaches leverage demonstrations or Bayesian methods to improve exploration and sample efficiency [26, 27]. Our framework internalizes this principle of on-demand assistance. Instead of relying on an external expert, our agent maintains an internal "committee" of pre-trained expert policies. When faced with uncertainty about the current task, it effectively "consults" this committee, using the reward feedback to determine which expert's advice is most relevant.

Thus, current research convincingly shows that the integration of meta-learning, methods for dealing with uncertainty and constrained RL creates fundamentally new opportunities for the development of robust agents. Our work develops these directions by proposing a universal adaptation mechanism that differs fundamentally from existing analogs in three key aspects: first, instead of an external expert or fixed switching rules, an internal evaluation system through entropy metrics is used; second, adaptation occurs at the level of reward interpretation, which allows the agent to "rethink" its goals in the process of work; third, the architecture maintains computational efficiency even in the presence of many potential tasks. These features make our approach especially promising for real-world applications, where requirements for flexibility and resource efficiency are critical.

Chapter 2

Methods

2.1 Preliminaries

2.1.1 Basics of Reinforcement Learning

To emphasize on the key algorithm of this thesis, it is necessary to define reinforcement learning in general and understand the basic principles. Reinforcement learning (RL) is a machine learning paradigm in which an agent learns to make decisions by interacting with the environment. This approach is fundamentally different from other machine learning methods in that the agent must independently discover optimal behavior strategies through a process of trial and error.

Unlike classical supervised learning, where the labeling for each example is available in advance, in RL the agent receives information about the quality of its actions solely in the form of rewards, which can be rare and delayed in time. This feature creates a fundamental time lag problem—the agent must associate current actions with long-term consequences, which requires complex mechanisms for evaluation and planning.

Fundamental components of almost any RL algorithm include:

- Agent—a learning system that makes decisions and is able to perceive the states of the environment and perform actions

- Environment—the entire external system with which the agent interacts, including the dynamics of transitions and feedback mechanisms
- State—a formal description of the current situation in the environment, containing all the information necessary for decision-making
- Action—possible choices that the agent can make in a given state from a set of admissible alternatives
- Reward function—a numerical score received by the agent for the actions performed, serving as a feedback signal about the quality of behavior
- Policy—a strategy used by the agent to select actions at each step, which can be deterministic or stochastic

The learning process in RL is based on iterative improvement of the policy through the accumulation of experience in interacting with the environment, where the agent seeks to find a balance between exploring new possibilities and exploiting already known successful strategies.

2.1.2 Markov Decision Processes

The RL process is typically modeled using a Markov Decision Process (MDP) formulation, where the agent's goal is to maximize the total (or averaged, or discounted) reward over its "lifetime" in the environment. MDPs provide a mathematically rigorous framework for formalizing sequential decision problems in stochastic settings.

A key property of MDPs is the Markov property, which states that future states depend only on the current state and the chosen action, and not on previous history. This property greatly simplifies analysis and allows the use of dynamic programming techniques to find optimal solutions.

Formally, an MDP is defined by a tuple (S, A, P, R, γ), where S is a set of states, A is a set of actions, $P(s'|s, a)$ is a transition function that determines the probability of transitioning to state s' given an action a in state s, $R(s, a)$ is a reward function, and γ is a discount factor that determines the importance of future rewards relative to current ones.

A learning agent explores various strategies to discover the sequences of actions that lead to the best outcome, and then uses this knowledge to maximize reward in the future. The learning process includes two main phases: exploration, where the agent tries new actions to gain information about the environment, and exploitation, where the agent uses its accumulated knowledge to maximize reward.

2.1.3 Solving Reinforcement Learning Problems

To properly analyze the specific variations of reinforcement learning considered in this work, it is first necessary to establish the foundational principles of the classical problem. The standard framework for this is the Markov Decision Process (MDP) described above.

The problem unfolds over discrete time steps, indexed by $t \in \mathbb{N} \cup \{0\}$. We define the state space as a compact set $S \subset \mathbb{R}^n$, which contains all possible configurations of the agent's environment. Similarly, the action space is a compact set $A \subset \mathbb{R}^d$, representing all actions available to the agent.

The dynamics of the environment are described by a transition probability function P. This function defines the probability of moving to a new state given the current state and the action taken:

$$p : S \times A \to \Delta(S),$$

where $\Delta(S)$ is the set of probability distributions over the state space S. For any state-action pair (s_t, a_t), the expression $P(\cdot|s_t, a_t)$ gives the distribution over the next state s_{t+1}.

The agent makes a sequential choice of actions in accordance with its policy, denoted by π. A policy is a mapping from states to a probability distribution over the set of actions, $\pi : S \to \Delta(A)$. At each time step t, the agent observes its current state

$s_t \in S$ and selects an action $a_t \in A$ according to its policy.

Upon executing an action and the subsequent transition to a new state, the agent receives a numerical estimate of the effectiveness of its choice. This mechanism is formalized by a reward function R, which assigns a real-valued reward to each state-action pair:

$$R : S \times A \to \mathbb{R}.$$

The specific reward received at time $t+1$ is denoted as $r_{t+1} = R(s_t, a_t)$.

The central objective in classical reinforcement learning is to identify an optimal policy, π^*, that maximizes a cumulative measure of these rewards over time. Two common objectives are considered:

1. **Average Reward:** This formulation is often used for continuing tasks without a clear endpoint. The goal is to maximize the long-term average reward per time step. The value of a policy π is given by:

$$V_{\text{avg}}(\pi) = \limsup_{T \to \infty} \frac{1}{T} \mathbb{E}_\pi \left[\sum_{t=0}^{T-1} R(s_t, a_t) \right]$$

2. **Discounted Reward:** This is the more common formulation for episodic tasks or when future rewards are considered less valuable than immediate ones. The value of a policy π is the expected sum of discounted future rewards:

$$V_{\text{disc}}^{\gamma}(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t) \right], \quad \text{where } \gamma \in (0,1) \text{ is the discount factor.}$$

Thus, the core task in reinforcement learning is to solve for the optimal policy π^* by solving the following optimization problem:

$$\pi^* = \arg\max_{\pi} V(\pi),$$

where $V(\pi)$ represents either the average reward $V_{\text{avg}}(\pi)$ or the discounted reward $V_{\text{disc}}^{\gamma}(\pi)$, depending on the specific problem formulation.

2.1.4 Transition to Tasks-Uncertainty

The emphasis of this thesis is on the problem of agent learning under uncertainty, i.e. in a situation where at the training stage the agent knows many possible objective functions, but does not know which of them will be relevant at the execution stage. This problem statement reflects many real-life scenarios where the system must be prepared for different requirements or changing operating conditions.

Unlike standard RL problems, where the objective reward function is fixed and known in advance, problems with goal uncertainty require the development of adaptive strategies that can effectively operate under multiple potential optimization criteria. This introduces an additional layer of complexity, as the agent must not only learn the optimal policy for each possible target, but also learn to quickly identify the active target and adapt its behavior accordingly.

This problem formulation is particularly relevant in the context of mobile robotics, autonomous vehicles, resource management, and other areas where the system may face different operational requirements or changing priorities. For example, an autonomous vehicle may be configured to minimize travel time under normal conditions, but switch to prioritize safety under adverse weather conditions.

Mathematically, this can be formalized as an optimization problem, where the agent must find a strategy that maximizes performance for an unknown function from a given set of possible objective functions, using only the observed reward signals to identify the active target.

2.2 Addressing Task-Uncertainty via Robust Reinforcement Learning

2.2.1 Mathematical Statement and Theoretical Derivation

As it was clarified above, this thesis is focused on the problem of agent learning under uncertainty, i.e., in a situation where at the learning stage the agent knows a set of possible objective functions, but does not know which of them will be relevant at the execution stage. Formally, the agent operates in an environment described as a multi-task MDP, where the state $s \in S$, the action $a \in A$, the dynamics are determined by the probabilistic transition $p(s_{t+1}|s_t, a_t)$, and the objective function is determined by the set $\{r_i : S \times A \to \mathbb{R}\}, \quad i = 1, \dots, m$.

The agent's task is to find a policy that maximizes the minimum accumulated reward across all possible objective functions, since it is unknown in advance which reward will be relevant at execution time. This robustness is formalized through the following max-min reinforcement learning problem with m different reward functions:

$$P^* = \max_{\pi \in \Pi} \min_{i \in [m]} V_i(\pi), \tag{2.1}$$

where Π is the set of admissible policies and $V_i(\pi)$ is the average or discounted reward over r_i under the policy π.

To facilitate the analysis, we reformulate the problem using an epigraphic transformation [28] by introducing an auxiliary scalar variable t:

$$P^* = \max_{\pi \in \Pi,\, t \in \mathbb{R}} \quad t \tag{2.2}$$

$$\text{subject to} \quad V_i(\pi) \geq t, \quad \forall i \in [m],\, \pi \in \Pi. \tag{2.3}$$

This constrained formulation allows us to analyze the problem through the lens of Lagrangian duality. Let us introduce non-negative Lagrange multipliers $\lambda = (\lambda_1, \dots, \lambda_m)$, where $\lambda_i \geq 0$, one for each constraint. The Lagrangian of the problem is:

$$\mathcal{L}(\pi, t, \lambda) = t + \sum_{i=1}^{m} \lambda_i (V_i(\pi) - t) = t \left(1 - \sum_{i=1}^{m} \lambda_i\right) + \sum_{i=1}^{m} \lambda_i V_i(\pi). \tag{2.4}$$

The corresponding dual function $d(\lambda)$ is obtained by maximizing the Lagrangian over the primal variables π and t:

$$d(\lambda) = \max_{\pi \in \Pi, t \in \mathbb{R}} \mathcal{L}(\pi, t, \lambda). \tag{2.5}$$

We can observe that the Lagrangian is linear in t. For the maximum over t to be bounded (and not diverge to infinity), the coefficient of t must be zero. This imposes a key constraint on the dual variables:

$$\sum_{i=1}^{m} \lambda_i = 1. \tag{2.6}$$

This condition, combined with $\lambda_i \geq 0$, means that the vector λ must belong to the probability simplex Δ^m. When this condition holds, the term with t vanishes, and the dual problem becomes:

$$D^* = \min_{\lambda \in \Delta^m} \max_{\pi \in \Pi} \sum_{i=1}^{m} \lambda_i V_i(\pi). \tag{2.7}$$

A crucial result for this class of problems is that duality holds, meaning the optimal value of the primal problem equals the optimal value of the dual problem. This is true despite the non-convex nature of the policy space Π [18]. Therefore, we have:

$$P^* = D^*. \tag{2.8}$$

This reformulation is equivalent to the original max-min problem, as maximizing t under the constraint that all value functions exceed t is the same as maximizing the minimum value across all $V_i(\pi)$.

Given that each $V_i(\pi)$ can be optimized independently, one can obtain policies $\pi_i^* \in \arg\max_{\pi \in \Pi} V_i(\pi)$ for each reward $i \in [m]$. Denote $V_i^* := V_i(\pi_i^*)$ as the optimal value attainable for reward i alone.

However, the max-min optimality seeks a policy π^* that equalizes the smallest value across all rewards, i.e., ensuring that $P^* = \min_{i \in [m]} V_i(\pi^*)$ is as large as possible.

2.2.2 Two-Phase Algorithmic Approach

In contrast to classical robust settings—which look for a policy maximizing the minimum reward across all possible tasks—or classical MetaRL, where fast adaptation is the focus, we leverage the structure of the reward functions to allow self-identification of the task "on the fly" during execution, without explicit knowledge of the environment parameters.

The architectural paradigm is to split the core methodology into two phases: a separate Training phase and an Execution phase, directly following the insights from the theoretical derivation. In this regard, further in the work the algorithm will often be called the "dual-lambda algorithm".

Training Phase

During the Training phase, the agent independently learns individually optimal policies for each potential reward function r_i. That is, for each task, the classical RL problem is solved:

$$\pi_i^* \in \arg\max_{\pi \in \Pi} V_i(\pi), \quad \forall i \in \{1, \dots, m\}.$$

In practice, this means running parallel RL trainings for each r_i, using the chosen base algorithm. The choice of PPO (Proximal Policy Optimization) [29] as the base algorithm for this work is justified by its stability and robustness when learning complex policies using gradient methods. PPO scales well to tasks with a large number of states and actions and does not require careful hyperparameter tuning, which is important for multi-task settings.

The result of the first stage is a set of independent policies $\{\pi_i^*\}$, each of which is optimal with respect to its reward function. This approach was chosen due to the simplicity of implementation (there is no need for a single, super-complex universal policy) and the possibility of reusing expertise for new tasks by simply expanding m.

Execution Phase

At the second stage—the Execution phase—the agent's task is to adaptively select (or form a mixture of) policies from the set $\{\pi_i^*\}$ to maximize the actual reward, without knowing in advance which function r_i is currently being implemented. At each moment of time, the agent randomly selects a policy π_i^* proportional to a vector of weights-coefficients $\lambda \in \Delta_m$, where Δ_m is the standard $(m-1)$ simplex such that $\lambda_i \geq 0$ and $\sum_i \lambda_i = 1$.

It is theoretically shown that executing a random mixture of policies (where a policy is drawn from $\{\pi_i^*\}$ with weights λ_i) leads to an average performance on each of the tasks: $V_j(\pi) = \sum_{i=1}^m \lambda_i V_j(\pi_i^*)$ for any j. The optimal vector λ is selected by solving an optimization problem, where the goal is to maximize the expected reward with respect to the active objective function. In practice, this vector is updated adaptively: after each short episode (rollout), the values of λ are adjusted using a gradient step based on the received rewards, and then projected back onto the simplex. This ensures fast and robust adaptation of the agent to the current task, automatically concentrating the weight on the appropriate policy without the need for explicit task identification.

Algorithm 1 Training phase

```
Output: Trained policies $\{\pi_i^*\}_{i\in[m]}$
1: Use RL algorithm to obtain policies $\pi_i^* \in \arg\max_{\pi\in\Pi} V_i(\pi)$ for all $i \in \{1,\dots,m\}$
```

Algorithm 2 Execution phase

```
Input: Policies $\{\pi_i^*\}_{i\in[m]}$, step size $\eta_\lambda$, epoch $T_0$
Output: Trajectories $(s_t, a_t)$ achieving $P^*$
1: Initialize: Initial state $s_0$, dual variable $\lambda = \mathbf{1}/m$
2: for $k = 0, 1, \dots$ do
3:     Draw current policy $\pi_k \sim \{\pi_i^*$ with prob. $\lambda_{i,k}\}_{i\in[m]}$
4:     Rollout $T_0$ steps with actions $a_k \sim \pi_t(s_k)$
5:     Update dual variables:
       $\lambda_{i,k+1/2} = \lambda_{i,k} - \frac{\eta_\lambda}{T_0} \sum_{t=kT_0}^{(k+1)T_0-1} r_i(s_t, a_t)$
       $\lambda_{k+1} = \left[\lambda_{k+1/2}\right]_{\mathcal{P}}$
6: end for
```

2.3 An Alternative Approach: Online Task Inference via Predictive Control

2.3.1 Core Principle: Short-Horizon Simulation and Selection

Here, we explore a different paradigm for adaptive decision-making. What is more, this method assumes the agent has access to a set of m pre-trained expert policies, $\{\pi_1, \pi_2, \ldots, \pi_m\}$, where each policy is optimal for a distinct task i. The agent again operates under task uncertainty, where the environment's true reward function r_{true} is one of $\{r_i\}$ but is not known beforehand.

The strategy is built upon an online, step-by-step planning process. At each moment of decision, the agent leverages its expert policies to "look into the future" through short, parallel simulations. By observing the hypothetical performance of each policy in the current, real environment, the agent can infer which expert's strategy is most promising in the immediate future. It then commits to the initial step of that expert's proposed plan.

2.3.2 The Predictive Control Loop

The mechanism can be formalized as a predictive control loop that activates at every time step k. Given the agent's current state s_k, it initiates m independent simulations, or "rollouts," to evaluate each available policy.

Each rollout for a policy π_i begins from the identical starting state s_k and proceeds for a fixed, finite horizon of T_0 steps. This process generates a hypothetical action-state sequence:

$$\tau_i(s_k) = (s'_{k,0}, a'_{k,0}, s'_{k,1}, a'_{k,1}, \ldots, s'_{k,T_0-1}, a'_{k,T_0-1}) \tag{2.9}$$

where the initial simulation state is the real state, $s'_{k,0} = s_k$, and subsequent actions are determined by the policy, $a'_{k,t} = \pi_i(s'_{k,t})$. The transitions follow the environment's dynamics.

The key to task inference lies in how these simulations are evaluated. The rewards for each simulated step are calculated using the true, active reward function of the environment, r_{true}. Thus, the cumulative reward for each rollout reflects how well policy π_i performs on the actual task at hand:

$$R_i(s_k) = \sum_{t=0}^{T_0-1} r_{true}(s'_{k,t}, a'_{k,t}) \tag{2.10}$$

Upon completion of the rollouts, the agent obtains a vector of projected returns, $[R_1(s_k), R_2(s_k), \ldots, R_m(s_k)]$. The policy selection rule is then a greedy maximization over these projected outcomes. The index of the optimal policy for the current step k is chosen as:

$$i_k^* = \arg \max_{i \in \{1,\ldots,m\}} R_i(s_k) \tag{2.11}$$

Following the principle of receding horizon control, the agent does not execute the full trajectory of the winning policy. Instead, it adopts only the very first action from the best-found plan:

$$a_k = a'_{k,0} \quad \text{from the trajectory generated by } \pi_{i_k^*} \tag{2.12}$$

This action a_k is then executed in the real environment, leading to the next state s_{k+1}. At this new state, the entire cycle of simulation, evaluation, and selection is repeated, ensuring continuous adaptation to the task.

Algorithm 1 Training phase

```
Output: Trained policies $\{\pi_i^*\}_{i \in [m]}$
1: Use RL algorithm to obtain policies $\pi_i^* \in \arg\max_{\pi \in \Pi} V_i(\pi)$ for all $i \in \{1, \ldots, m\}$
```

Algorithm 3 Execution via Predictive Policy Selection

```
Input: Pre-trained policies $\{\pi_i\}_{i \in [m]}$, simulation horizon $T_0$
Output: Trajectory of states $(s_0, s_1, \ldots)$
 1: Initialize state $s_0$ from the environment
 2: for $k = 0, 1, 2, \ldots$ do
 3:     Let the current state be $s_k$
 4:     for each policy $\pi_i$ in $\{\pi_1, \ldots, \pi_m\}$ do
 5:         Initialize simulation environment to state $s_k$
 6:         Generate a trajectory $\tau_i(s_k)$ by executing $\pi_i$ for $T_0$ steps
 7:         Calculate the cumulative reward $R_i(s_k)$ for trajectory $\tau_i(s_k)$
 8:     end for
 9:     Find the index of the best policy: $i_k^* \leftarrow \arg\max_i R_i(s_k)$
10:     Select the first action from the best trajectory: $a_k \leftarrow$ first action of $\tau_{i_k^*}(s_k)$
11:     Apply action $a_k$ to the true environment to get the next state $s_{k+1}$
12: end for
```

Chapter 3

Numerical Results

3.1 Experimental Environment and Experimental Design

To test and implement the proposed two-phase task-uncertain RL algorithm, a series of experiments in a specially designed mobile navigation environment with variable task conditions were developed and analyzed. A two-dimensional mobile agent simulator (custom environment based on Gymnasium [30]) was created as an environment, combining features of classical navigation tasks and safety elements. Key characteristics of the environment include space: a discrete square map where the agent starts from one of the fixed points, and the goal is located at a certain position, to achieve which the agent does not necessarily have to approach it closely—it is enough to simply approach the nearby area. In addition to the agent and the goal, there is a stationary obstacle—a round area (collision area) with a variable radius $r \in \{1, 2, 3\}$ depending on the specific task (Figure 1).

The dynamics lies in the fact that the agent is controlled by two actions—linear and angular velocity, which simulates the motion of a robot with a controlled vector and orientation. The agent's state at time t can be defined as $s_t = (x_t, y_t, \theta_t)$, where (x_t, y_t) are its coordinates and θ_t is its orientation. The control action $a_t = (v_t, \omega_t)$

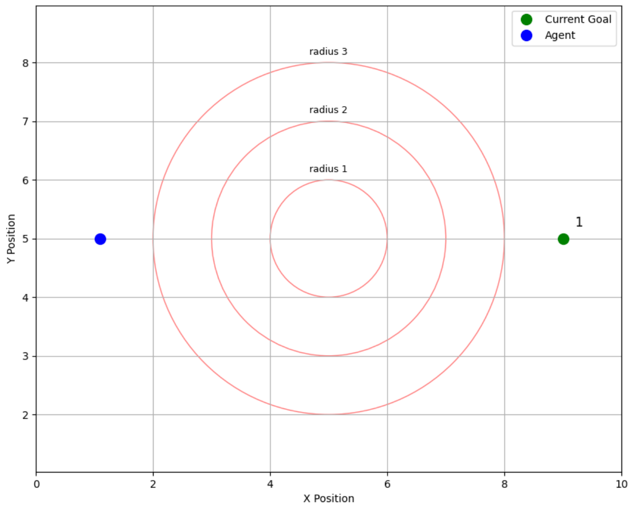


Figure 1: Obstacle radius settings in the training environment: $r = 1$, $r = 2$, $r = 3$.

consists of a linear velocity v_t and an angular velocity ω_t. The state update over a small time interval Δt is described by the following system of equations:

$$\begin{cases} x_{t+1} = x_t + v_t \cos(\theta_t)\Delta t \\ y_{t+1} = y_t + v_t \sin(\theta_t)\Delta t \\ \theta_{t+1} = \theta_t + \omega_t \Delta t \end{cases}$$

The perception system of the agent enables it to "see" the surrounding space within a certain radius and a viewing angle of 60 degrees, as well as the distance and angle to the obstacle when this "bell" crosses the obstacle, which allows for partial observability and realistic tactility, and the agent "sees" the distance to the target (Figure 2).

That is, observation is an array of 4 values: distance to the target, angle to the target relative to the current direction of the agent, distance to the nearest obstacle in the sensor zone, angle to the obstacle relative to the current direction of the agent. Formally, the observation vector o_t at time t can be represented as:

$$o_t = [d_t^{\text{target}}, \phi_t^{\text{target}}, d_t^{\text{obs}}, \phi_t^{\text{obs}}]^T$$

The reward function consists of two components: a negatively proportional distance

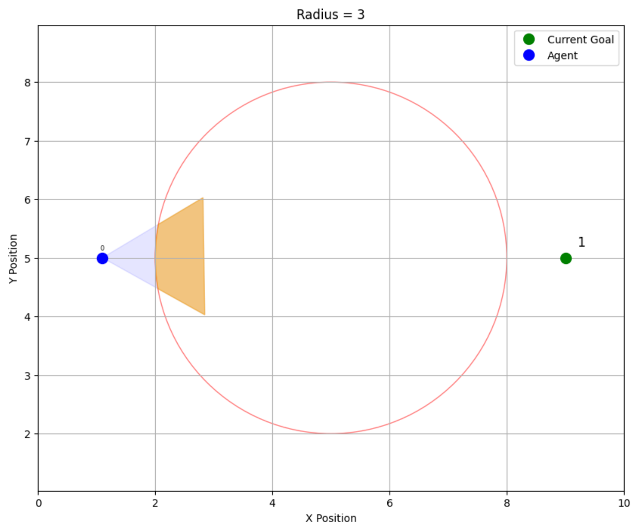


Figure 2: Agent perception dome and obstacle radius intersection in the training environment with $r = 3$.

to the target (reward for approaching) and a large penalty for hitting an obstacle (implementation of safety constraints). Thus, the reward function R for transitioning from state s_t to s_{t+1} can be defined as:

$$R(s_t, a_t, s_{t+1}) = \begin{cases} -C_{\text{penalty}} & \text{if collision with obstacle} \\ -k \cdot d(s_{t+1}, \text{goal}) & \text{otherwise} \end{cases}$$

where C_{penalty} is a large positive penalty constant for a collision, k is a proportionality coefficient, and $d(s_{t+1}, \text{goal})$ is the Euclidean distance from the agent's new state to the goal. Such an arrangement of the environment reflects a typical scenario of mobile robotics, where precise targeting and simultaneous compliance with safety constraints are required.

The key element of task-uncertain RL is the multiplicity of potential reward functions and/or constraints that are unknown to the agent at the execution stage, this is the multitasking nature of the experiment. In this implementation, this is realized through three environment variants that differ in the value of the obstacle radius. Thus, there are three potentially different navigation tasks with different degrees of accessibility of passage and maneuver complexity. At the execution stage, the agent does not know in advance which of the environment variants will be active.

This most closely reflects the conditions of real-life use of mobile robots in a variable environment—for example, with the unpredictable appearance of dangerous zones of different sizes.

3.2 Behavioral Biases of Individually Trained Policies: Illustrative Scenario

The training phase, as previously described, results in a set of individually optimal policies $\{\pi_i^*\}_{i=1}^m$, where each policy is specialized for one of m possible reward parameterizations, in our case, different obstacle radii $r \in \{1, 2, 3\}$ (Figures 3, 4, 5). To build intuition for the subsequent analysis, we first experimentally investigate how these expert policies behave when applied to environments different from those they were trained on.

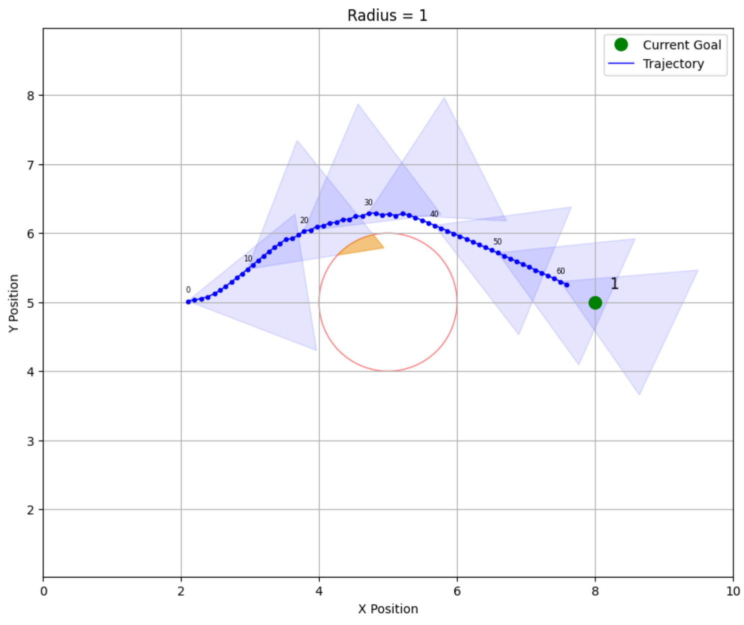


Figure 3: Optimal performance of policy $\pi_{r=1}^*$ on the environment with obstacle $r = 1$.

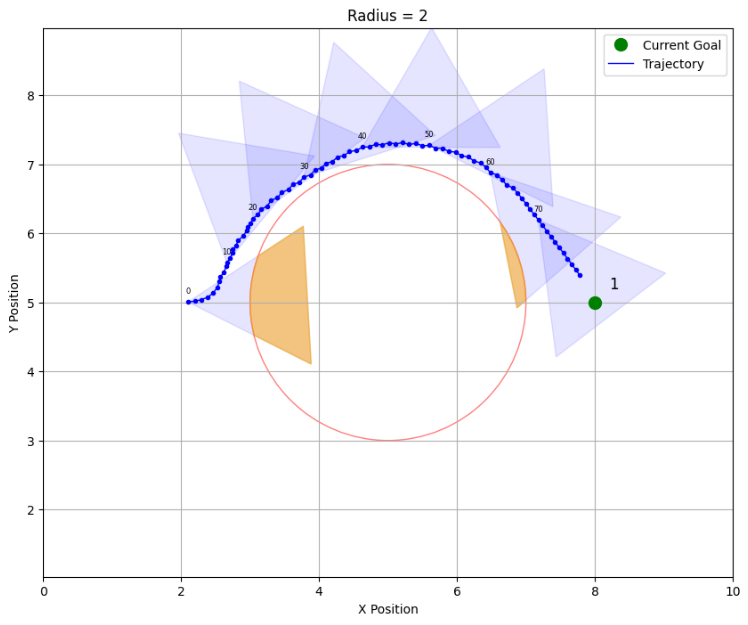


Figure 4: Optimal performance of policy $\pi^*_{r=2}$ on the environment with obstacle $r = 2$.

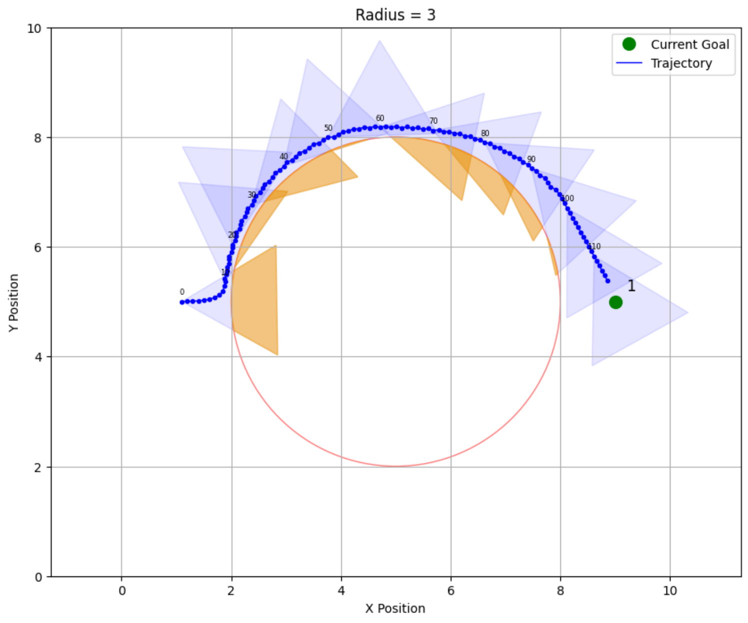


Figure 5: Optimal performance of policy $\pi^*_{r=3}$ on the environment with obstacle $r = 3$.

Each policy π^*_i is evaluated both “in-domain” (i.e., in the environment whose reward function r_i was used for training) and “out-of-domain” (i.e., with a different obstacle radius). This experiment is not designed to assess generalization, but rather to highlight and visualize the qualitative differences in the behavioral priors these policies acquire during training. The focus is to reveal the intrinsic trade-off between pursuit of the goal and adherence to safety constraints, as encoded by the reward shaping.

Key observations:

- **In-domain execution:**
  - Each policy reliably solves its own task. For example, $\pi^*_{r=1}$, trained with a minimal obstacle, learns to approach the target directly, balancing efficiency and minimal collision risk for its scenario. Similarly, $\pi^*_{r=3}$ learns more conservative, detour-prone strategies suitable for navigating the much larger penalty region.
- **Out-of-domain execution: asymmetric bias.**
  - Applying $\pi^*_{r=1}$ to $r = 3$ (large obstacle): The policy, unaware of the now-expanded penalty region, retains its aggressive shortcutting behavior and often attempts to pass through what would be a safe region for $r = 1$, but is now forbidden for $r = 3$. This typically results in the agent moving directly towards the goal and colliding with the obstacle. Figure 6 illustrates such a trajectory, where the agent's path crosses into the collision zone.
  - Applying $\pi^*_{r=3}$ to $r = 1$ (small obstacle): In contrast, the policy trained on the largest obstacle maintains its conservative, detour-based navigation, even when the obstacle is much smaller than anticipated. As shown in Figure 7, the agent needlessly circumvents a wide area, avoiding zones that are actually safe in the current context. Notably, this does not cause direct failures, but results in less efficient trajectories.

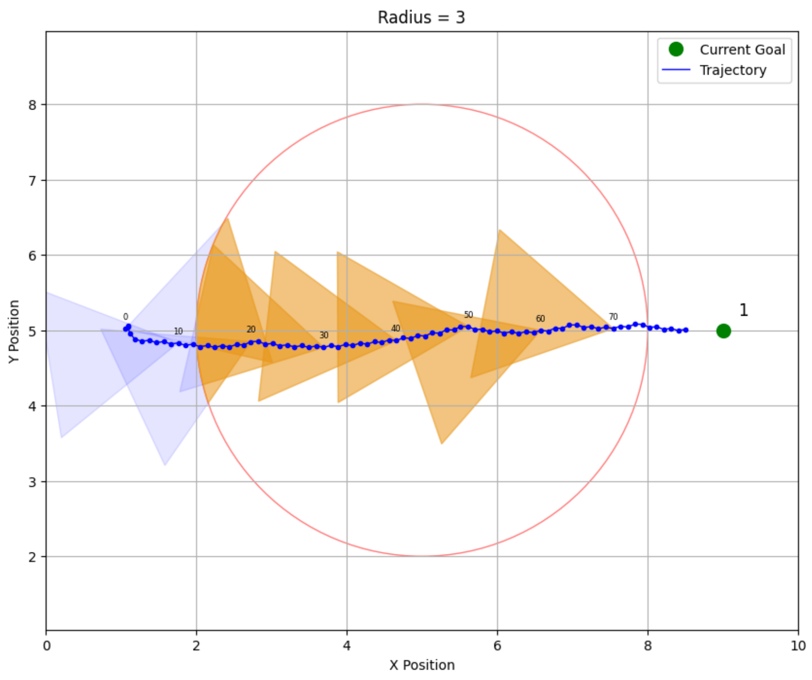


Figure 6: Trajectory of policy $\pi^*_{r=1}$ in environment with $r = 3$: direct movement through the enlarged obstacle, resulting in collision.

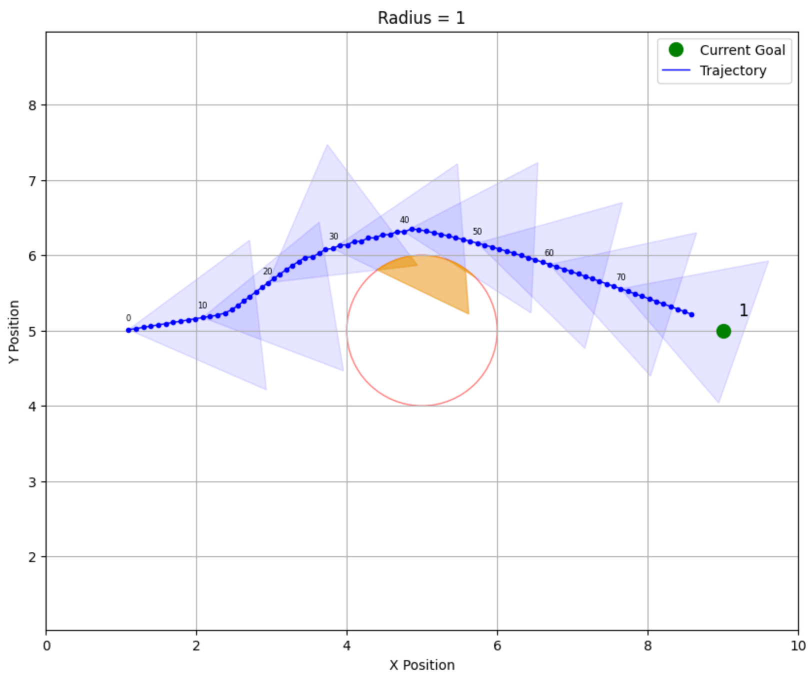


Figure 7: Trajectory of policy $\pi^*_{r=3}$ in environment with $r = 1$: safe but unnecessarily wide detour around the small obstacle.

This pronounced asymmetry is a direct consequence of the reward landscape each policy experienced during training. For $r = 3$, the penalty region occupies a significant fraction of the state space, and the agent's value function is heavily shaped by the need to avoid costly violations. As a result, $\pi^*_{r=3}$ internalizes a "safety-oriented" prior: it is highly averse to entering regions that could possibly carry penalty, erring strongly on the side of caution.

Conversely, for $r = 1$, the relative impact of the penalty term is much smaller; the policy can afford to be "goal-oriented," prioritizing the shortest path to the target with only minor detours. Thus, $\pi^*_{r=1}$ is, in a sense, approximately three times less cautious than $\pi^*_{r=3}$—if we take the proportional sizes of the penalty zones as a heuristic quantification.

Formally, if C_{penalty} is the fixed magnitude of the collision penalty, then the expected penalty encountered per random trajectory for $r = 3$ is substantially higher than for $r = 1$. Therefore, the optimal policies settle into regimes dominated respectively by safety or efficiency.

The constant C_{penalty} was chosen through empirical tuning and multiple trial runs, aiming to achieve a nontrivial trade-off in all three variants. Although it is impossible to equalize behavior across all settings—by design, larger radii enforce stricter avoidance—it is critical that the relative interpretation of the penalty is somewhat harmonized for cross-policy analysis. The reward component for goal-seeking, defined as negative Euclidean distance to the target, is classic for navigation and performed robustly.

This explicit bias in the learned priors is not accidental, but forms a crucial part of the following analysis. By visually and quantitatively contrasting the behaviors of $\pi^*_{r=1}$ and $\pi^*_{r=3}$ in mismatched environments, we set the stage for analyzing how the proposed dual adaptation mechanism exploits or compensates for these different expert strategies when the current task is ambiguous. Ultimately, this mini-experiment motivates the need for adaptive policy selection or mixing, as well as the use of reward feedback for online inference of the active task.

3.3 Performance Evaluation

The efficacy of the proposed dual-lambda algorithm was rigorously evaluated through a series of experiments designed to test its adaptability and robustness under progressively challenging conditions of task-uncertainty. The experimental protocol was structured into three distinct phases, each targeting a different aspect of the agent's performance.

3.3.1 Experimental Design

The evaluation framework was designed to systematically increase the complexity and uncertainty faced by the agent.

- **Phase 1: Baseline Environments.** In the initial phase, the algorithm's performance was benchmarked in the three fundamental environments that were used to train the base policies ($\pi^*_{r=1}, \pi^*_{r=2}, \pi^*_{r=3}$). This phase served to establish a performance baseline and verify that the adaptive mechanism could correctly identify and execute the optimal policy in simple settings.

- **Phase 2: Complex Composite Environment.** The second phase introduced a significant increase in complexity. The agent was placed in a composite environment that integrated elements from all base scenarios, featuring multiple goal locations and a dense field of obstacles with heterogeneous radii. This setup tested the algorithm's ability to navigate a cluttered and unpredictable space where the nature of the immediate challenge changes dynamically.

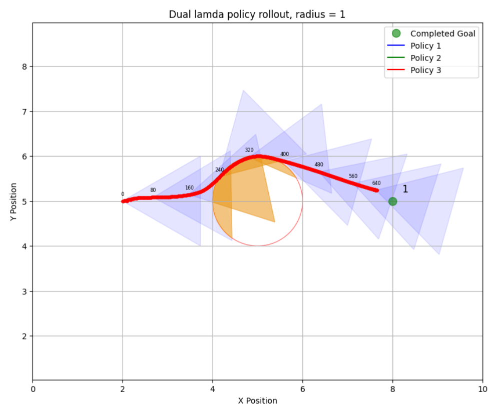
- **Phase 3: Generalization to Novel Obstacle Geometries.** The final and most demanding phase was designed to assess the algorithm's zero-shot generalization capability. The environment contained obstacles of non-standard, polygonal shapes (triangles and squares), which were entirely absent from the training distributions of the base policies. Results in this phase would demonstrate if the adaptive mechanism is not merely interpolating between known tasks but is capable of robustly responding to fundamentally novel environmental features.

It should also be noted in advance that, for the sake of conciseness, the experimental sections will present representative plots of agent behavior under various policies and algorithms, illustrating the most frequently observed patterns across different experimental phase configurations (i.e., for different initial agent positions and target locations). A quantitative analysis, including statistical evaluations based on a large sample of experimental runs to ensure robustness, will be provided in the section with comparison of algorithms. Nevertheless, the conclusions and observations described within each experimental phase for each algorithm are accurate, comprehensive, and are based on the entire set of experiments conducted for the respective phase.

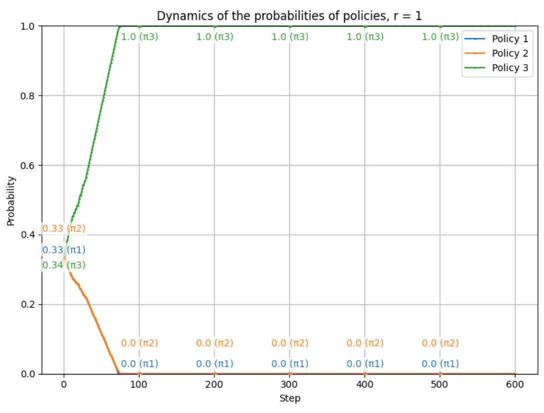
3.3.2 Experimental Phase 1

Dual-Lambda Algorithm

The execution trajectories on the fundamental environments and the corresponding dynamics of the policy probabilities for each of these three phases are presented in Figure 8, Figure 9 and Figure 10 respectively.

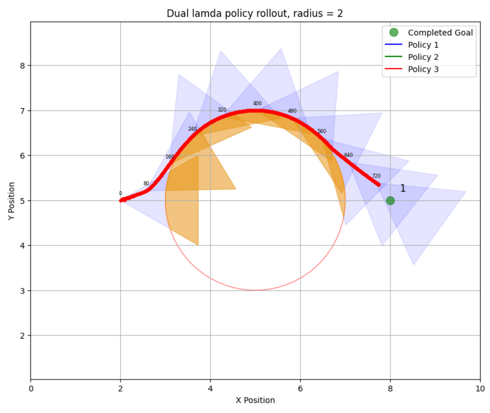


(a) Execution Trajectory

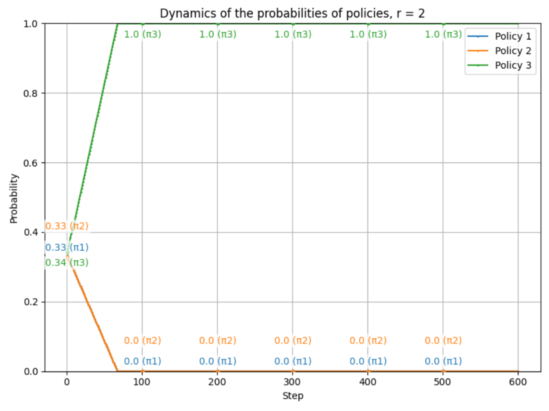


(b) Policy Probability Dynamics

Figure 8: Dual-Lambda Algorithm performance in the baseline environment with obstacle radius $r = 1$.

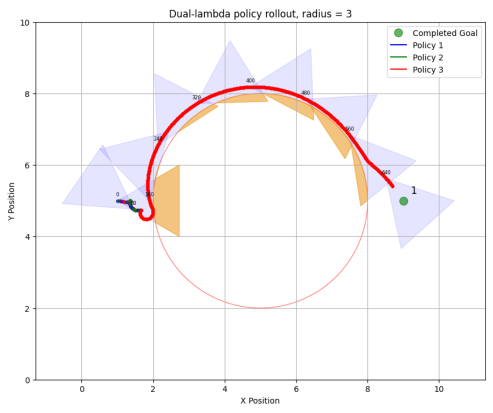


(a) Execution Trajectory

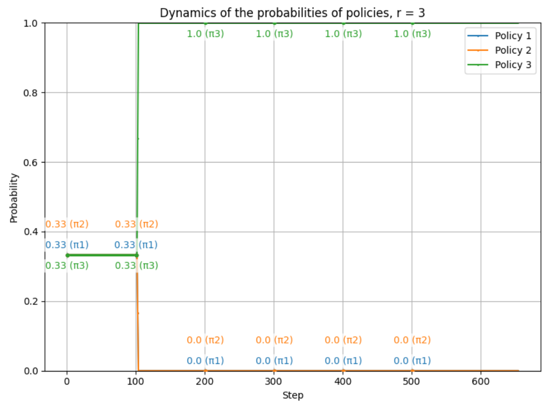


(b) Policy Probability Dynamics

Figure 9: Dual-Lambda Algorithm performance in the baseline environment with obstacle radius $r = 2$.



(a) Execution Trajectory



(b) Policy Probability Dynamics

Figure 10: Dual-Lambda Algorithm performance in the baseline environment with obstacle radius $r = 3$.

The results compellingly demonstrate that the dual-lambda algorithm successfully navigates environments even when the specific task parameters (i.e., obstacle configurations) are unknown a priori.

While the primary goal of this work was to establish the theoretical and empirical validity of the algorithm rather than to optimize its computational performance, a discussion of key hyperparameters that were introduced in discussion of Execution Phase is warranted.

The interplay between the lambda learning rate, η_λ, and the online learning epoch length, T_0, is particularly crucial.

- A high value of η_λ leads to more rapid updates to the policy weights, resulting in an agent that makes more "confident" or decisive steps. However, when combined with a long epoch T_0, this can lead to unstable, "jerky" probability distributions for λ, causing the agent to commit to suboptimal trajectories.

- Conversely, a low η_λ promotes smoother, more gradual changes in λ, which generally yields more optimal and stable trajectories. Yet, if paired with a short epoch T_0, the agent's learning becomes too slow, resulting in overly timid and hesitant movements with very slow progress towards the goal.

Given that computational speed was not the primary concern, we opted for a configuration that prioritizes robust and stable learning: a relatively low learning rate of $\eta_\lambda = 0.001$ was paired with a high epoch length T_0, which varied between $1/5$ and $1/3$ of the total steps required to reach the goal in an obstacle-free path. Furthermore, the agent's linear step was reduced compared to the setting in the "Behavioral Biases" experiment (Section 4.2). This was a deliberate choice to afford the dual variables (λ_i) sufficient time to converge based on the reward signals received during online learning.

It may initially appear inconsistent that the linear step was not also reduced for the experiments in Section 3.2. However, doing so yielded inferior results for the individual policies. Hypothesis for this phenomenon is that a smaller step size increases the number of actions required to reach an obstacle and this, in turn, skews the distribution of received rewards: the agent accumulates a long sequence of small, positive rewards for approaching the goal before encountering the negative rewards from a collision (as seen in Figure 11). This can induced a myopic policy, where the agent becomes overly focused on short-term goal acquisition while failing to adequately account for the delayed, but critical, signal to avoid obstacles—the opposite of what was shown with the larger step (Figure 3).

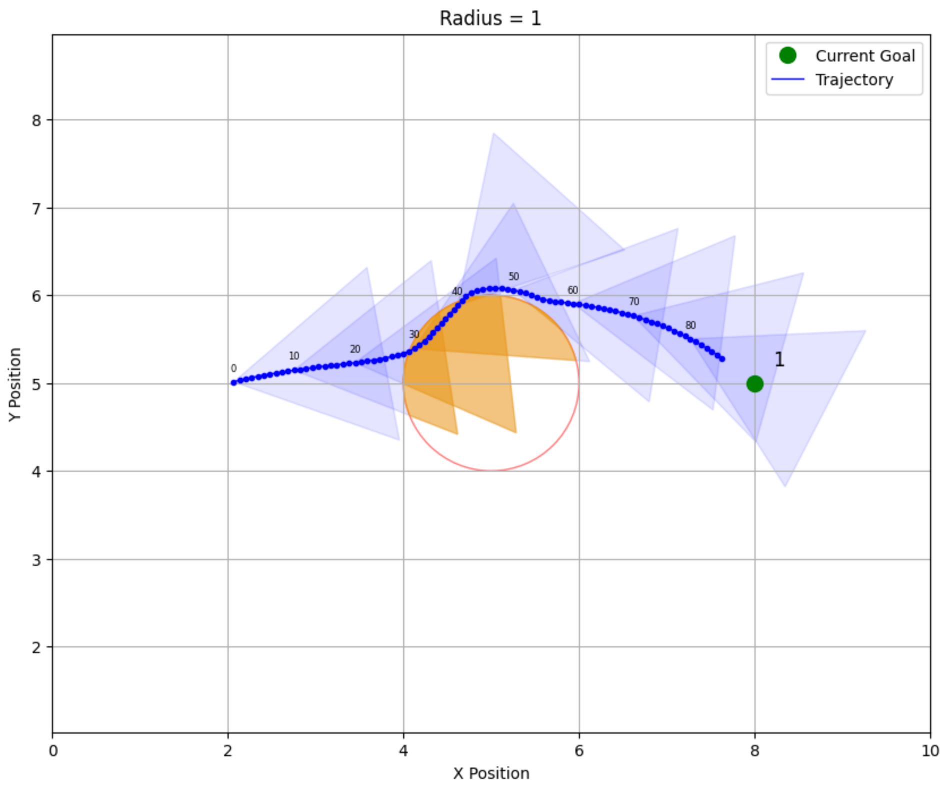
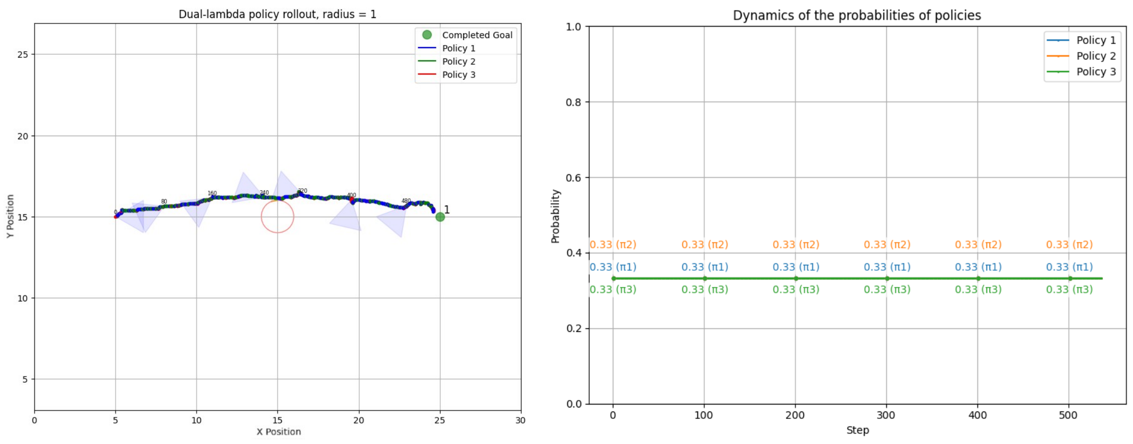


Figure 11: Non-optimal policy of $\pi^*_{r=1}$ in environment with $r = 1$, resulting from a skewed reward signal distribution.

A key observation from the trajectory and policy probability plots (Figures 8, 9, 10) is the algorithm's persistent preference for the most conservative policy (π^*_3, trained on the largest obstacle radius). This might seem counter-intuitive; one might expect the agent to employ an aggressive policy (π^*_1) when far from an obstacle, switch to a cautious one for avoidance, and revert to aggression afterward. The observed conservative behavior can be explained by two complementary factors.

First, this behavior is a direct consequence of the algorithm's underlying **minimax framework**. As formulated in our methodology, the agent's objective is to find a policy that performs best in the worst-case scenario. This is equivalent to a game where, after the agent selects a policy, an adversary (or "nature") chooses the least favorable reward function from the known set $\{r_i\}$. This minimax strategy mathematically guarantees a maximized lower bound on performance, but inherently fosters a conservative decision-making process. The agent chooses the policy that with the least losses will lead to the best result (exactly in that order)—this is so-called "algorithmic conservatism"—and even though π_3^* is the most conservative, we will see later in phases 2 and 3 that for different environments and a full rollout the algorithm will not necessarily choose it.

Second, the **spatial constraints of the environment** amplify this inherent conservatism. The standard experimental grid is relatively small. In such a limited space, the agent almost immediately encounters an obstacle during online learning, which leads to a multi-way decision regarding whether to execute one policy or another and whether it prevails over others.



(a) Execution Trajectory (b) Policy Probability Dynamics

Figure 12: Dual-Lambda Algorithm perfomance in an expanded 30x30 environment with an obstacle of $r = 1$.

To verify the second hypothesis, we conducted an additional experiment in an environment whose size was increased threefold. As illustrated in Figure 12, the agent's behavior changes significantly under these conditions. In the expanded environment, the agent adopts an "indifference" policy, selecting among π_1^*, π_2^*, and π_3^* with equal probability. The distribution over λ does not shift toward any particular policy, as the agent has no prior experience with such a large space and therefore lacks a clear strategy in the absence of visible obstacles. Consequently, none of the policies dominate the others. During training, all three base policies received identical penalties for encountering obstacles and identical rewards for progressing toward the goal. When the agent perceives no obstacles and relies solely on proximity to the goal, the learning algorithm assigns equal value to all policies, as each is capable of reaching the goal and obtaining the same reward without encountering obstacles. Even when approaching or passing near an obstacle, the policies prevent the agent from entering it. Nevertheless, the algorithm continues to choose all policies with equal probability, as none of them are adapted to an environment in which obstacles remain unseen for an extended period (in terms of the number of steps).

Alternative Predictive Control Algorithm

As evidenced by the graphical representations presented in the subsequent Figures 13, 14, 15, the Alternative Predictive Control Algorithm successfully addresses the navigation task. Notably, it exhibits a trajectory that closely approximates optimality across all obstacle radii.

A particularly intriguing observation from the empirical results pertains to the algorithm's strategic selection of fundamental expert policies. For scenarios involving obstacle $r = 1$ and $r = 2$, the alternative algorithm predominantly favors policy π_3^*—the most conservative of the pre-trained experts. However, a remarkable deviation is observed when the obstacle $r = 3$. In this instance, the algorithm unexpectedly gravitates towards policy π_1^*. While both π_1^* and π_3^* share the overarching objective of obstacle avoidance and goal attainment, π_1^* represents the least conservative strategy, effectively being the diametric opposite of π_3^* in terms of risk aversion.

Crucially, unlike the Dual-Lambda algorithm, the alternative algorithm does not employ a mechanism for updating policy weights over time, instead, its operation is founded on a spot-generation approach where trajectories are simulated at each decision step, and the best-performing policy is selected. During the intricate maneuver around the perimeter of the $r = 3$ obstacle, policy π_1^* consistently yielded superior outcomes compared to π_2^* and π_3^* at nearly every step which is counter-intuitive, as policy π_1^* is not theoretically expected to be optimal for navigating around a large $r = 3$ obstacle. This suggests an emergent behavior where the short-horizon predictive capability allows π_1^* to exploit subtle, locally optimal pathways even in scenarios where a globally conservative strategy might be anticipated.

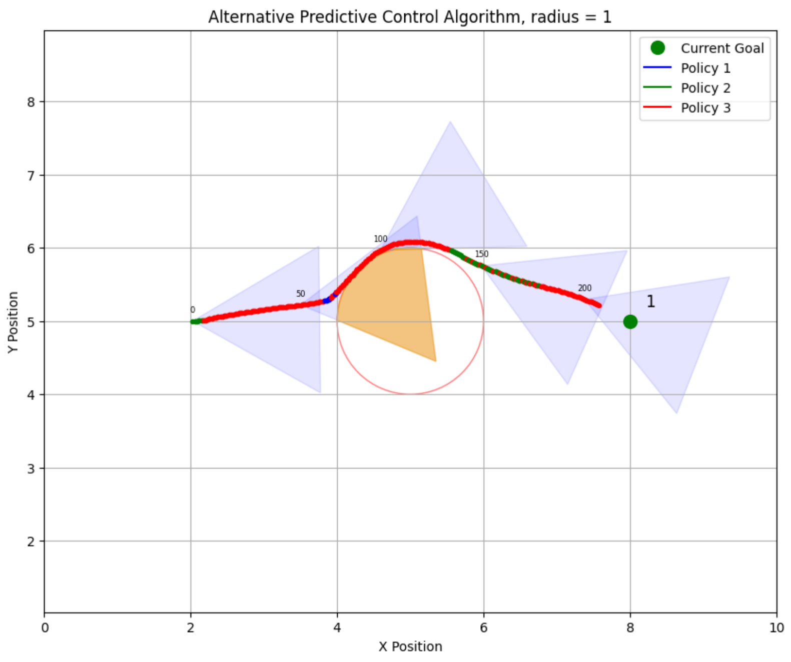


Figure 13: Alternative Predictive Control Algorithm performance in the baseline environment with obstacle radius $r = 1$.

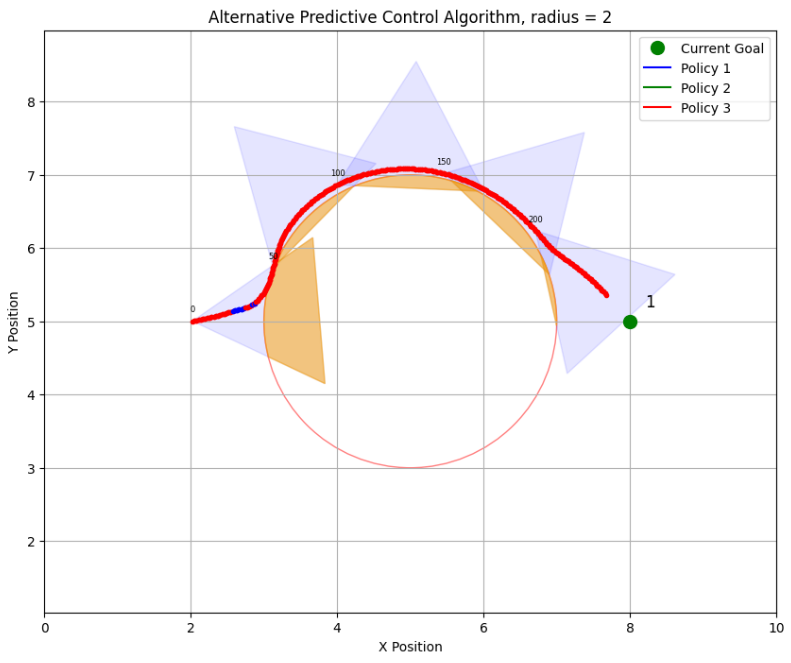


Figure 14: Alternative Predictive Control Algorithm performance in the baseline environment with obstacle radius $r = 2$.

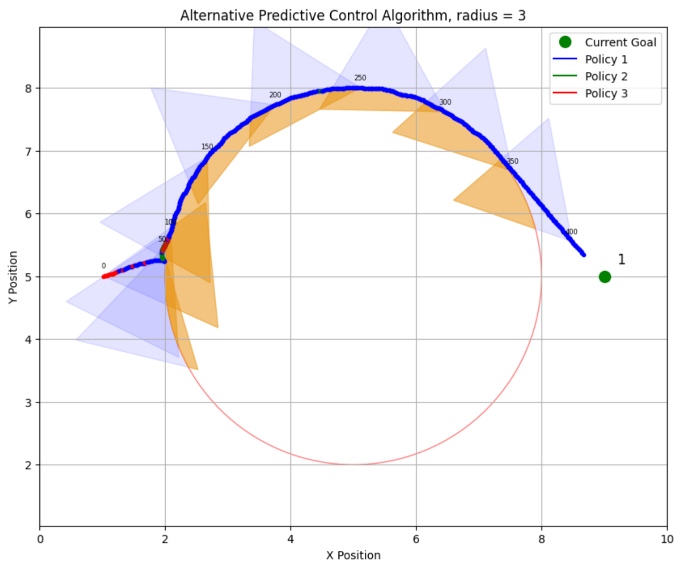


Figure 15: Alternative Predictive Control Algorithm performance in the baseline environment with obstacle radius $r = 3$.

MetaRL

To establish a comprehensive performance baseline against which to evaluate our proposed methods, we implemented a standard approach called Meta-Reinforcement Learning (MetaRL). Unlike our modular frameworks, which leverage a committee of specialized expert policies, the MetaRL agent is trained to develop a single, monolithic policy capable of operating across the distribution of tasks. Specifically, the agent was trained concurrently on the three baseline scenarios with obstacle radii of $r = 1, 2$, and 3. Crucially, the agent receives no explicit information about the active obstacle radius and must infer the task context solely from its observations and the received reward signals so it would learn to adapt to an unknown task from the presented stack.

To ensure a methodologically sound and fair comparison, the training protocol for the MetaRL agent was carefully controlled. The total number of training epochs allocated to the MetaRL policy was set to be the sum of the epochs used to train the three individual expert policies (π_1^*, π_2^*, and π_3^*) for our proposed algorithms. Furthermore, the reward function parameters were kept identical to those used during the training of the base policies. This protocol ensures that all approaches are evaluated under equivalent computational budgets and environmental conditions, providing a robust and unbiased basis for comparison.

As expected, the resulting MetaRL policy demonstrates near-optimal performance within the specific baseline environments on which it was trained (Figures 16, 17, 18).

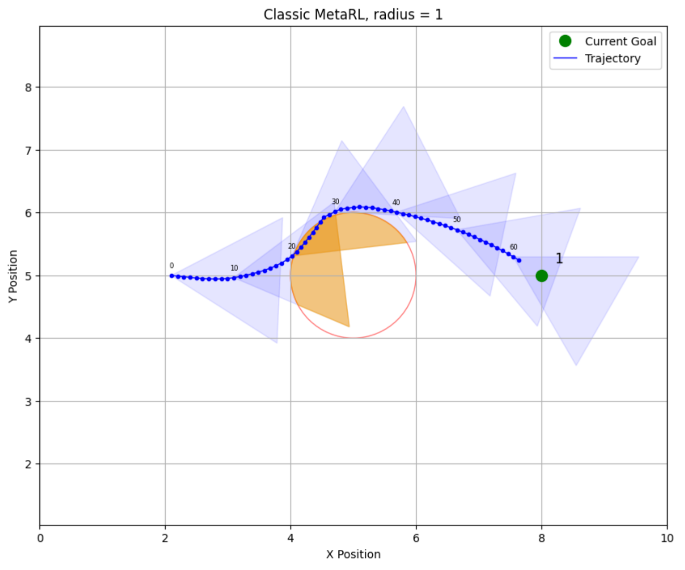


Figure 16: MetaRL performance in the baseline environment with obstacle radius $r = 1$.

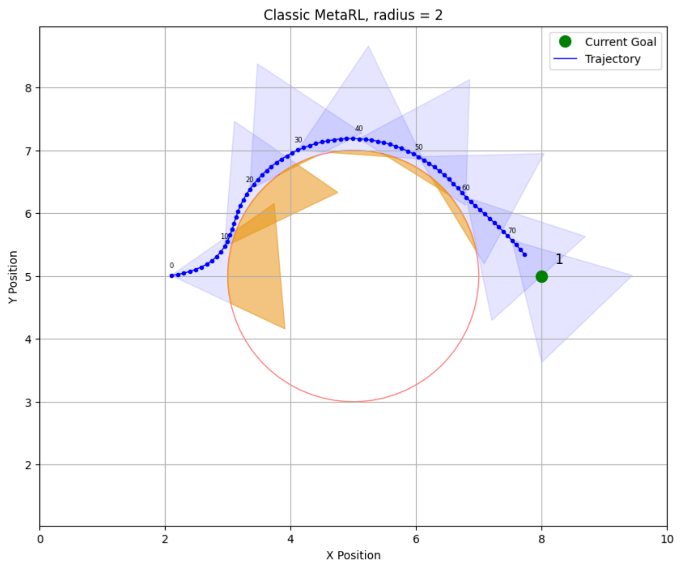


Figure 17: MetaRL performance in the baseline environment with obstacle radius $r = 2$.

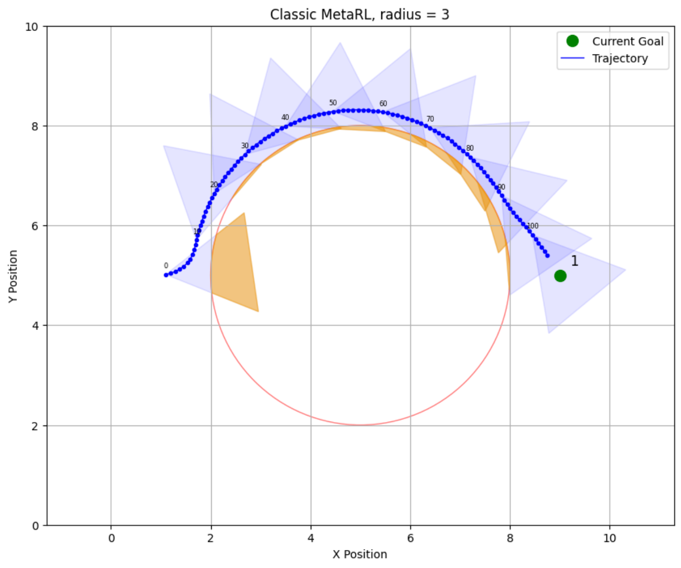


Figure 18: MetaRL performance in the baseline environment with obstacle radius $r = 3$.

3.3.3 Experimental Phase 2

The second experimental phase represents a substantial escalation in task complexity relative to the baseline environments considered previously. As illustrated in the figures below, the environment is transformed into a heterogeneous field populated with multiple obstacles and an arbitrary number of potential goals. This configuration introduces two principal sources of challenge for both the agent and the underlying algorithmic framework.

First, the fundamental expert policies obtained during the training phase have never been exposed to scenarios characterized by an extended sequence of navigation steps—now required to traverse between multiple targets. Second, the diversity and density of obstacle arrangements, particularly cases where two obstacles are positioned in close proximity, give rise to novel perceptual ambiguities for the agent. Notably, while each expert policy has learned to circumvent an individual obstacle, the agent's sensory field—its observation "bell"—has not previously encountered simultaneous perception of multiple obstacles. Consequently, when the agent attempts to navigate around one obstacle and is confronted by another, its learned priors do not readily prescribe an appropriate behavioral response.

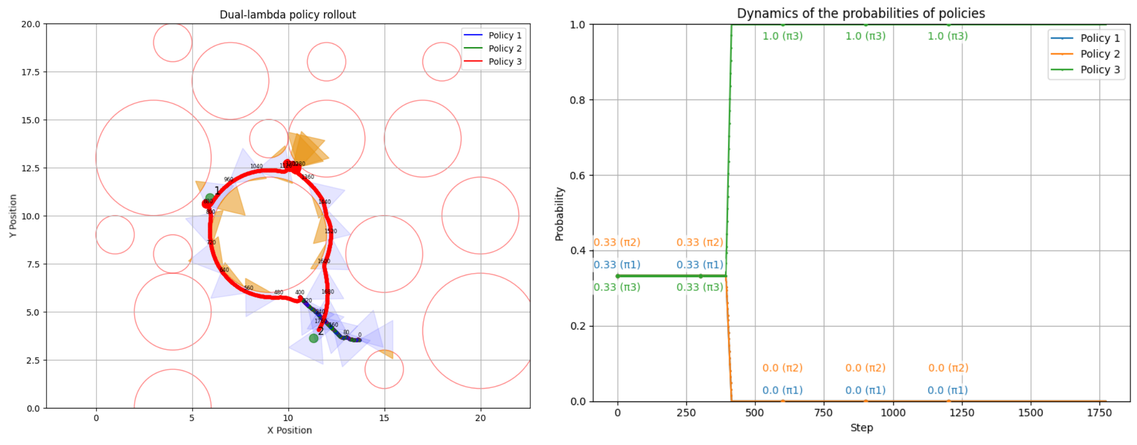
Dual-Lambda Algorithm

To systematically investigate the adaptive capabilities of the proposed algorithm under these compounded uncertainties, we initially constrain the environment to contain exactly two goal locations. This controlled setting enables an analysis of agent performance when faced with both an unfamiliar distribution of obstacles and the requirement to effectively double its mission planning horizon.

Despite encountering significant perceptual uncertainty in the latter part of its trajectory, the agent ultimately succeeded in reaching the goal (Figure 19). As shown in Figure 19a, the initial segment of the rollout is characterized by rapid alternations between policies. This behavior reflects the agent's indifference to policy selection before any obstacles are detected and agent's unfamiliarity with increased dimensions, as all candidate policies yield similar rewards while the agent remains sufficiently distant from environmental hazards.

However, as the agent approaches an obstacle, the dual-lambda mechanism adaptively concentrates probability mass on the most conservative policy (Policy 3). This transition is evident both in the trajectory plot and the corresponding policy probability dynamics (Figure 19b), indicating a sudden and decisive shift toward a safer navigation strategy when risk becomes imminent.

It is important to note that, as discussed in the introduction to this experimental phase, the individual expert policies were not previously exposed to situations where multiple obstacles are simultaneously present within the agent's sensory field. Consequently, when traversing narrow passages between closely-spaced obstacles, the agent exhibits a momentary loss of navigational confidence, manifested as two pronounced oscillations in its trajectory. These 'zig-zag' segments are a direct result of the agent's perceptual ambiguity—its learned policies do not encode behaviors for resolving conflicts when multiple obstacles are concurrently detected. Nevertheless, the dual-lambda algorithm rapidly recovers from these transient uncertainties. By leveraging the reward feedback and updating its policy mixture, the agent is able to adapt online and resume progress toward the goal.



(a) Execution Trajectory (b) Policy Probability Dynamics

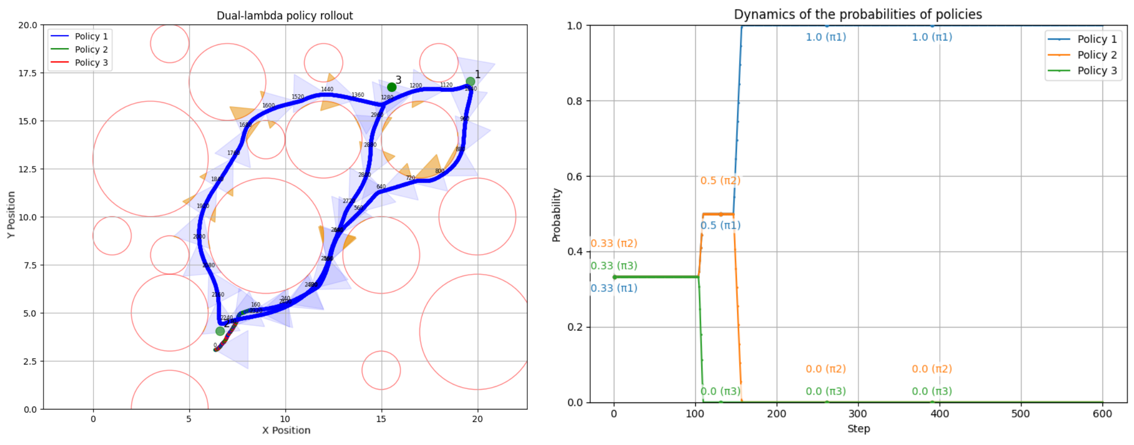
Figure 19: Dual-Lambda Algorithm performance in the environment with distribution of obstacles from baseline environments and 2 targets.

To further challenge the adaptability of the dual-lambda algorithm, we increased the number of goal locations within the environment to three. While this modification preserves the fundamental structure of the second experimental phase, it imposes additional complexity: the agent must now navigate a longer cumulative path, confronting a greater number of obstacles and opportunities for perceptual uncertainty.

As presented in Figures 20a and 20b, the trajectory and the dynamics of policy probabilities reveal a significant shift in the agent's strategy. After initial oscillations between policies, the agent ultimately assigns almost exclusive weight to $\pi^*_{r=1}$. This outcome is in stark contrast to the two-goal scenario, where $\pi^*_{r=3}$ (the most conservative) was dominant throughout most of the trajectory.

This result highlights a critical limitation of the dual-lambda algorithm's underlying mechanism: its performance is highly sensitive to the relative optimality of the constituent expert policies given the structure of the test environment. If one of the expert policies (even if it was not explicitly trained for the current environment) happens to yield substantially higher rewards than the others, the dual-lambda optimizer will overwhelmingly favor it. This behavior is logical given the algorithm's update rule, but it renders the approach vulnerable to environments where such a policy is, by chance, much closer to optimal than its peers. Conversely, if the favored policy becomes suboptimal due to a sudden change in environmental context, the adaptation of the mixture weights—governed by the parameter η_λ—may be insufficiently rapid to redirect the agent towards a more suitable strategy and it still will be choosing prevail one. As a result, the dual-lambda method may exhibit excessive conservatism or inconsistency in policy selection.

Nonetheless, despite these theoretical drawbacks, the three-goal experiment demonstrates that the dual-lambda framework is generally robust: the agent is still able to reach all designated goals, successfully navigating the more complex environment. However, the findings underscore the necessity of careful policy set design and, potentially, adaptive mechanisms for more rapid weight adjustment in highly nonstationary or diverse task settings.



(a) Execution Trajectory (b) Policy Probability Dynamics

Figure 20: Dual-Lambda Algorithm performance in the environment with distribution of obstacles from baseline environments and 3 targets.

Alternative Predictive Control Algorithm

In parallel with the dual-lambda framework, we conducted identical experiments using the Alternative Predictive Control Algorithm. As in previous sections, we first analyze performance in the environment with two goal locations (Figure 21).

A key observation is that, although the trajectory generated by the alternative algorithm is more efficient and direct compared to the dual-lambda approach, the overall navigation strategy remains qualitatively similar. The agent does not approach the first goal and turn back immediately; instead, it circumvents the whole obstacle in a wide arc before proceeding to the goal. This is an inherent consequence of the heavy reliance of both algorithms on the underlying expert policies. Since none of the base policies were explicitly trained to reach a goal and then retreat along the same path, such behavior does not emerge from their combination, regardless of the mixture strategy.

The performance of the alternative algorithm in this two-goal environment is primarily governed by $\pi^*_{r=1}$, albeit with notable, intermittent switches to $\pi^*_{r=3}$. This dynamic adaptation highlights the agent's ability to leverage the strengths of conservative policies when confronted with more challenging obstacle configurations.

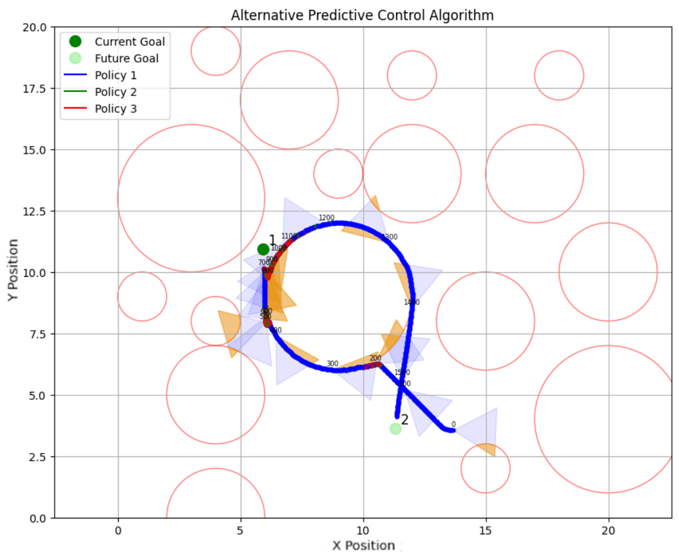


Figure 21: Alternative Predictive Control Algorithm performance in the environment with distribution of obstacles from baseline environments and 2 targets.

Transitioning to the more complex three-goal environment (Figure 22), we observe a significant increase in policy mixing by the alternative algorithm. This is expected due to the increased navigational complexity: the agent must travel a longer path while encountering a greater density and diversity of obstacles. Interestingly, the resulting trajectory is not only significantly shorter than that produced by the dual-lambda algorithm but also turned out to have only two types of obstacles ($r = 2$

and $r = 3$). In contrast, the dual-lambda agent must have successfully navigated around all three obstacle types.

Most notably, the policy mixture under alternative becomes much richer. While $\pi^*_{r=1}$ and $\pi^*_{r=3}$ remain dominant, $\pi^*_{r=2}$ is now selectively activated—specifically when the agent is forced to traverse narrow passages between obstacles. This is particularly intriguing, as $\pi^*_{r=2}$ was trained on obstacles with a radius only slightly different from those in the base scenarios. Yet, the alternative algorithm autonomously discovers a specialized use for it: $\pi^*_{r=2}$ is predominantly employed in situations where the agent is squeezed between obstacles, revealing an emergent division of labor that was not explicitly encoded during training.

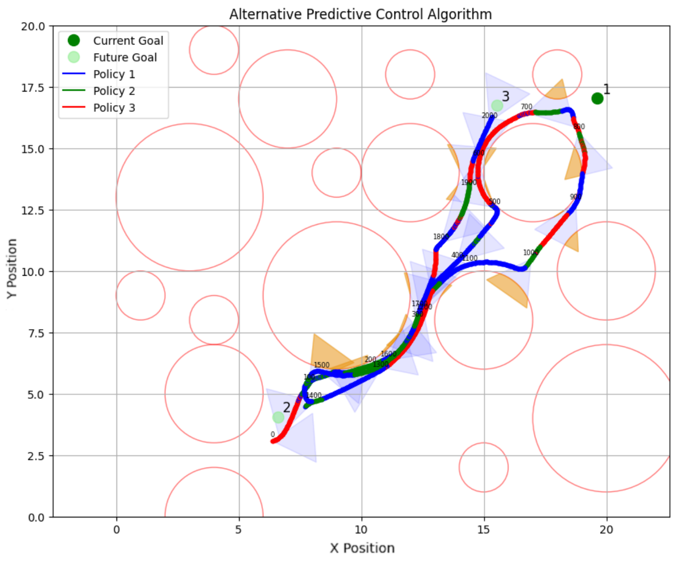


Figure 22: Alternative Predictive Control Algorithm performance in the environment with distribution of obstacles from baseline environments and 3 targets.

Overall, these results demonstrate the flexibility and online adaptability of the Alternative Predictive Control Algorithm in environments with increased task uncertainty and complexity. However, they also reinforce the central insight that, regardless of selection mechanism, the performance ceiling for both approaches is fundamentally limited by the behavioral repertoire of the underlying expert policies.

MetaRL

In the second experimental phase, the performance of the MetaRL agent was markedly deficient, failing to demonstrate effective adaptation to the environment's increased complexity (Figures 23, 24). The monolithic policy proved incapable of effectively balancing the competing demands of navigating towards multiple goals and safely avoiding the dense and heterogeneous obstacle field.

An empiric analysis of the agent's behavior reveals highly suboptimal trajectories. While the agent generally oriented towards the target locations, its paths were inefficient and characterized by a consistent failure to circumvent obstacles. The primary deficiency observed was the emergence of significant behavioral pathologies. In numerous instances, the agent would exhibit indecisive behavior, becoming stalled or entering into cyclic, looping motions. This erraticism was particularly detrimental as it also occurred within the boundaries of an obstacle, causing the agent to accrue substantial penalties that drastically degraded its aggregate performance.

In summary, the MetaRL framework failed to cope with the challenges of task uncertainty posed by the Phase 2 environments. A detailed quantitative analysis corroborating these observations is presented in section of the comparison of algorithms.

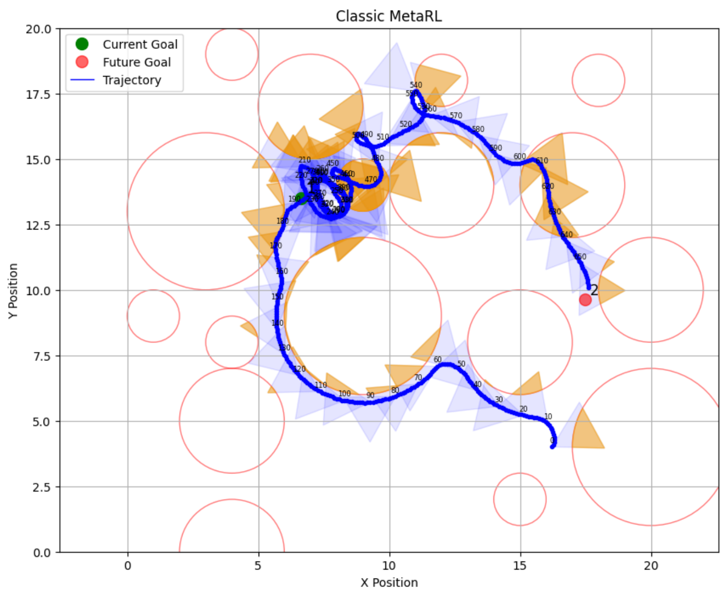


Figure 23: MetaRL performance in the environment with distribution of obstacles from baseline environments and 2 targets.

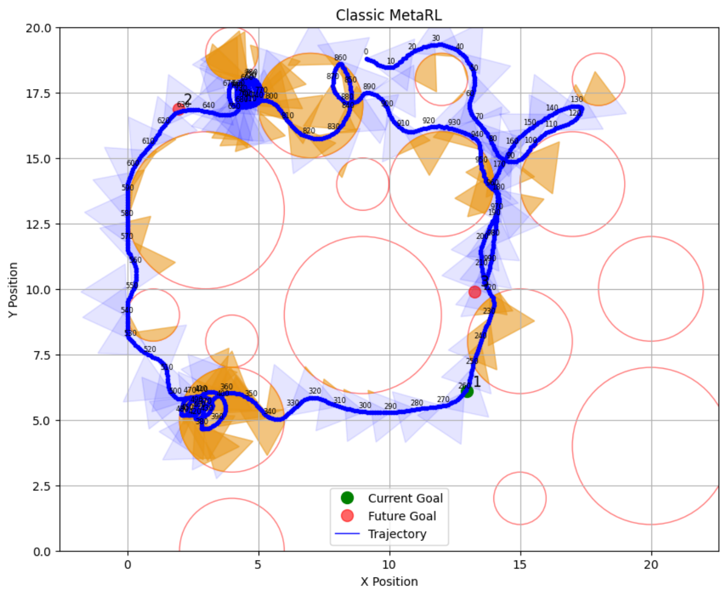


Figure 24: MetaRL performance in the environment with distribution of obstacles from baseline environments and 3 targets.

3.3.4 Experimental Phase 3

The third experimental phase constitutes the most challenging and stringent test of the agent's robustness and generalization capacity. Unlike prior phases, the environment in Phase 3 introduces obstacles whose geometric shapes—specifically, triangles and squares—were never encountered by the agent during training. As a result, the agent's perceptual system and learned policies are fundamentally unprepared to handle such obstacles in a familiar manner, as their underlying sensorimotor statistics and traversal dynamics diverge considerably from the previously witnessed circular obstacles.

Despite retaining the same observation model as before, namely,

$$o_t = \left[d_t^{\text{target}}, \phi_t^{\text{target}}, d_t^{\text{obs}}, \phi_t^{\text{obs}}\right]^T,$$

and being consistently penalized for collisions, the agent's policies are each trained exclusively on round obstacle avoidance. This presents a unique generalization challenge: although the agent can infer, from its sensory inputs, the proximity and bearing of a nearby obstacle, the sequences of actions that constitute a successful avoidance maneuver for circular obstacles may not transfer to polygonal obstacles. The geometry and local gradients of approach are fundamentally altered, so that, for instance, the angle-to-obstacle observed during traversal around a triangle or square will follow distinctly different temporal profiles than those observed for a circle. As such, the trajectory 'templates' associated with 'successful' avoidance in training become misaligned with the requirements of the new scenario.

To further exacerbate the task complexity, the environment in Phase 3 is configured with four distinct goal locations distributed throughout the map. Consequently, the agent must navigate across nearly the full span of the workspace multiple times, repeatedly negotiating novel obstacle shapes from varying approach angles. This environment is thus deliberately constructed to test not only the agent's ability to generalize its learned representations to previously unseen perceptual inputs and geometric configurations, but also its capacity for dynamic re-adaptation in the face of compounding task and sensory uncertainties.

Dual-Lambda Algorithm

In contrast to prior phases, where the performance of the dual-lambda algorithm could be unequivocally characterized as successful, its behavior in Phase 3 presents a more nuanced picture (Figure 25). On the one hand, the algorithm demonstrates a degree of robustness: the agent does not become "lost" or trapped in indecisive loops, as sometimes occurs in reinforcement learning under severe uncertainty. The agent ultimately succeeds in reaching all designated goals, despite being confronted with entirely novel obstacle geometries—specifically, triangular and square shapes absent from the training regime.

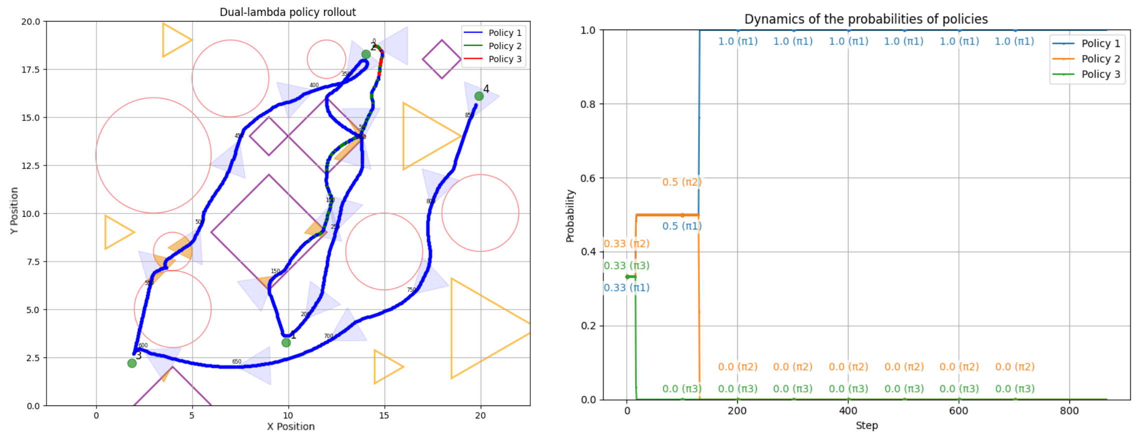
However, the fluid and efficient obstacle avoidance observed in Phases 1 and 2 is no longer present. While the agent consistently avoids naive collisions and does not blindly traverse through obstacles, its avoidance maneuvers are often suboptimal and irregular. Specifically, when navigating around square obstacles, the agent displays a tendency to "skim" closely along the edge in the inner part of the obstacle, rather than executing a smooth detour—in several instances, the agent partially encroaches into the obstacle region, "hugging" the edge as it progresses. Notably, upon approaching the corners of a square, there is a clear attempt to exit the obstacle's region precisely through the vertex.

An intriguing aspect of the agent's adaptive behavior is the increased, though still selective, use of $\pi^*_{r=2}$ in angular situations, while $\pi^*_{r=1}$ remains predominant for most of the trajectory (Figure 25a). The challenge introduced by novel obstacle geometries does not induce the agent to entirely abandon its trained strategies, but rather prompts a localized, context-dependent shift in policy selection.

Furthermore, the agent is capable of traversing narrow passages between previously unseen and familiar shapes without becoming stalled or exhibiting erratic switching. Yet, contrary to expectations, failures occur not only at polygonal obstacles but also with circular ones of $r = 1$ and $r = 2$. Particularly, while the agent attempts to "cut the corner" on the obstacle with $r = 2$, it proceeds straight through the obstacle with $r = 1$ without avoidance.

This behavior is a direct consequence of the agent encountering sequences of observations fundamentally distinct from those experienced during training. In Phase 3, the agent tries sequentially circumvent multiple unknown shapes, predominantly receiving negative rewards, and then attempt to navigate narrow passages adjacent to familiar obstacles. The resultant observation-action-reward trajectory differs substantially from any "successful" pattern encoded in the base policies, thereby leading to such performance anomalies.

In summary, while the dual-lambda algorithm ultimately fulfills the primary objective of reaching all target locations, its performance is marred by frequent, localized failures and unconventional avoidance maneuvers. These results illuminate the fundamental limitations of the dual-lambda adaptation mechanism: generalization potential are tightly coupled to the diversity and representational richness of the underlying policy set. Phase 3 thus highlights both the residual robustness and the intrinsic boundaries of the approach in truly out-of-distribution environments.



(a) Execution Trajectory (b) Policy Probability Dynamics

Figure 25: Dual-Lambda Algorithm performance in the environment with distribution of known and unknown obstacles and 4 targets.

Alternative Predictive Control Algorithm

In the final experimental phase, the Alternative Predictive Control Algorithm was subjected to the most demanding test scenario as well (Figure 26), and in this setting, the alternative algorithm exhibited several notable challenges and behaviors not observed in the earlier phases.

A prominent difficulty was the agent's frequent and pronounced violation of obstacle boundaries, resulting in direct traversal through obstacle regions. While the generated trajectories remained close to optimal in terms of global path efficiency and direction, the agent often failed to avoid penetration into obstacles, mirroring the behavior observed with the dual-lambda algorithm—particularly for the two circular obstacles of $r = 1$ and $r = 2$. In these cases, the agent simply crossed through the obstacles without attempting substantial avoidance.

Paradoxically, the alternative algorithm demonstrated more competent navigation when faced with novel obstacle shapes (squares and triangles). Although it sometimes partially entered these new obstacles, it typically recovered rapidly, exiting along the nearest edge and efficiently skirting around corners—both with squares and triangles. This adaptability suggests an ability to leverage the predictive control horizon to exploit geometric "escape" opportunities not present in the training data.

Another interesting observation is the increased sensitivity of alternative to hyper-parameter tuning under the heightened complexity of Phase 3. Whereas in previous phases satisfactory performance could be achieved with virtually any value of the predictive horizon parameter T_0 (as long as $T_0 \geq 1$), in this phase meticulous experimentation was required to balance execution time and prediction depth. The optimal T_0 was empirically determined to be 50—a significantly longer planning horizon than before—highlighting the growing importance of foresight as the environment's complexity increases.

It is also noteworthy that the inherent flexibility of the alternative algorithm—its ability to rapidly switch between expert policies—conferred an advantage for handling unfamiliar obstacle shapes. Nonetheless, similar to the dual-lambda algorithm, alternative algorithm failed to avoid the circular obstacles near the third goal. This suggests that the difficulties encountered at these specific locations do not stem from the policy selection or adaptation mechanism alone.

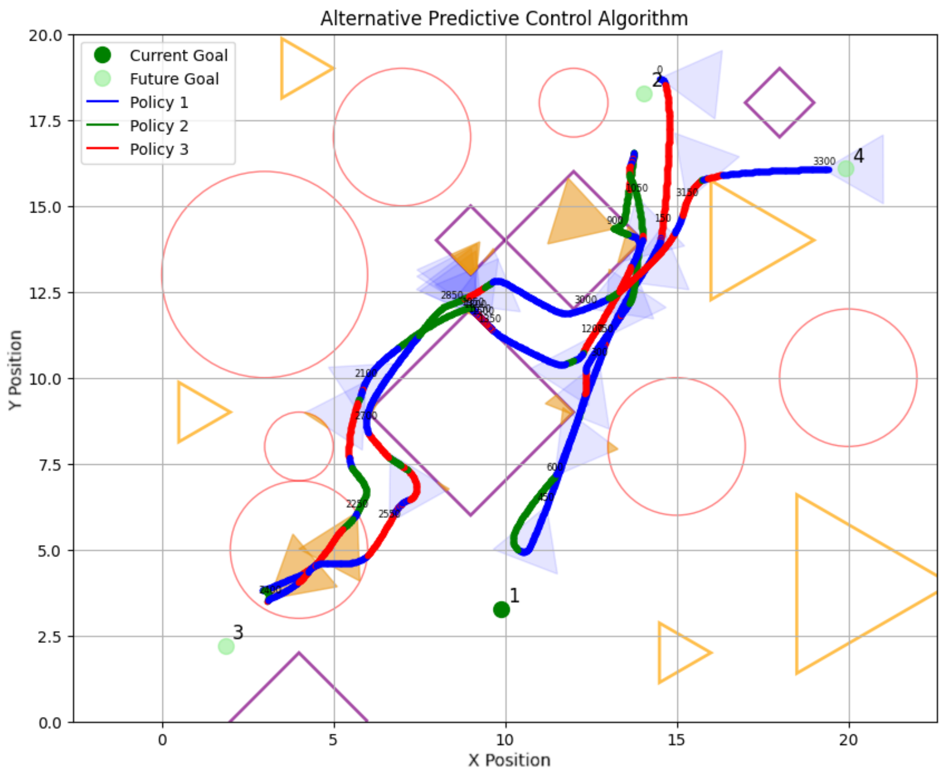


Figure 26: Alternative Predictive Control Algorithm performance in the environment with distribution of known and unknown obstacles and 4 targets.

In conclusion, although the alternative algorithm ultimately succeeded in reaching all specified goals in Phase 3, its performance was marred by repeated collisions and unconventional obstacle interactions. These results reinforce the principal limitations identified earlier: both the dual-lambda and alternative algorithms are fundamentally constrained by the expressiveness and diversity of their underlying expert policy set. Nevertheless, alternative algorithm's comparatively rapid policy adaptation and emergent strategies in handling polygonal obstacles demonstrate a measure of robustness and flexibility when facing unprecedented environmental complexity.

MetaRL

Consistent with its performance degradation in the more complex Phase 2, the MetaRL agent demonstrated a profound failure to generalize in Phase 3. When confronted with novel, out-of-distribution obstacle geometries, the monolithic policy proved entirely inadequate for the task (Figure 27).

An empiric assessment of the agent's trajectories reveals a stark inability to adapt. While the agent exhibited some residual, albeit inefficient, attempts to navigate around the familiar circular obstacles, it almost completely disregarded the novel polygonal shapes—the agent would proceed directly towards a goal, colliding with square obstacles as if they were non-existent. Furthermore, the behavioral pathologies noted in the previous phase—specifically, periods of indecisive stalling and cyclic motion—persisted, further degrading its performance.

Ultimately, the MetaRL framework failed to solve the concurrent tasks of goal achievement and obstacle avoidance in this zero-shot generalization setting. Crucially, this phase was designed to test zero-shot generalization, and the algorithm demonstrated no adaptive capacity to handle obstacle geometries that were absent from its training distribution.

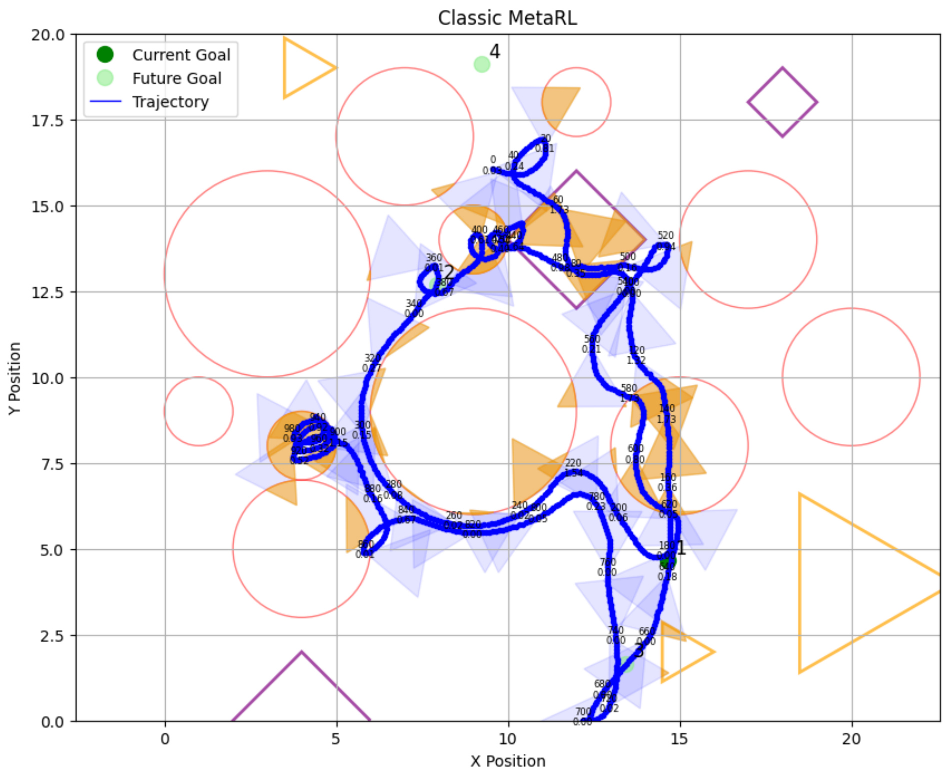


Figure 27: MetaRL performance in the environment with distribution of known and unknown obstacles and 4 targets.

3.4 Comparison of Algorithms

3.4.1 Quantitative Analysis

To conduct a rigorous quantitative comparison of the proposed algorithms against a MetaRL baseline, a dedicated analysis was performed using data from the second experimental phase. This phase was specifically chosen as it provides the most informative basis for comparison. Phase 1, where all algorithms achieve nearly best possible performance, offers little differentiation. Conversely, Phase 3, featuring novel obstacle geometries, poses a significant generalization challenge where all algorithms exhibit degraded performance, making it less suitable for a direct comparison of their core adaptation mechanisms. Phase 2, therefore, represents an ideal middle ground for assessing performance under significant yet manageable task uncertainty.

The analysis is based on 30 experimental runs for each algorithm, comprising five trials for each unique combination of the number of goals and the rollout horizon (T_0). It should be noted that the MetaRL baseline does not utilize the T_0 parameter, as its policy is not predictive, thus, its experiments vary only with number of goals. The results presented are averaged across these trials.

The performance of the algorithms was evaluated using the following metrics and parameters:

- **Number of goals**—the number of target locations the agent must visit.
- T_0—the rollout horizon parameter for the predictive algorithms (Dual-Lambda and MPC-based).
- **Algorithm**—the algorithm being evaluated: Dual-Lambda, MPC-based (Alternative Predictive Control Algorithm) or the MetaRL baseline.
- **Execution Time (s)**—the total wall-clock time in seconds to complete an episode.
- **Reward**—the cumulative reward obtained (cost).
- **Steps**—the total number of steps taken by the agent to complete the episode.
- **Steps Inside Obstacles**—the total number of steps the agent spent within an obstacle's boundaries.
- **Reward per Step**—the average reward (cost) per step.
- **Steps Inside Proportion (%)**—the percentage of total steps spent inside obstacles.

Goals	T0	Algorithm	Exec. T (s)	Reward	Steps	Steps Obs.	Rwd./Step	Steps Obs.%
1	10	Dual-Lambda	46.52	-3945.49	419.6	10.8	-8.99	2.17
1	10	MPC-based	59.88	-3719.21	452.8	0.0	-7.85	0.00
1	0	MetaRL	2.77	-14733.80	933.0	158.4	-15.49	17.16
2	10	Dual-Lambda	71.77	-7157.76	706.6	69.2	-9.96	11.65
2	10	MPC-based	67.62	-6392.72	711.8	18.2	-8.93	4.83
2	25	Dual-Lambda	226.45	-9496.99	846.0	50.6	-10.98	6.49
2	25	MPC-based	170.32	-6907.65	789.4	19.4	-8.38	2.34
2	0	MetaRL	8.83	-41168.39	2849.9	423.8	-14.10	16.62
3	25	Dual-Lambda	320.54	-13991.87	1260.8	122.4	-10.96	9.15
3	25	MPC-based	243.67	-10431.21	1118.6	23.6	-9.30	2.03
3	50	Dual-Lambda	651.45	-13014.19	1127.4	120.8	-11.49	11.51
3	50	MPC-based	439.80	-9304.62	965.0	27.4	-9.54	2.79
3	0	MetaRL	9.33	-41268.79	2749.0	389.1	-14.67	14.25
4	50	Dual-Lambda	806.85	-14151.23	1351.0	54.2	-10.41	4.10
4	50	MPC-based	721.02	-13171.56	1373.4	32.0	-9.48	1.99
4	0	MetaRL	10.66	-43028.00	2705.6	283.2	-15.32	11.33

Table 1: Comparison table of algorithms.

Computational Efficiency

The analysis begins with the secondary metric of computational efficiency. While the MetaRL baseline exhibits the lowest execution time, this is a direct consequence of its non-predictive, single-policy architecture, where runtime scales primarily with the number of steps taken. A more equitable comparison lies between the Dual-Lambda and Predictive Control algorithms. For the simplest scenario (1 goal, $T_0 = 10$), the Dual-Lambda algorithm is more efficient, achieving a lower execution time and, correspondingly, taking fewer steps on average (419.6 vs. 452.8).

However, across all other configurations, the Predictive Control algorithm proves to be more time-efficient, outperforming Dual-Lambda by an average of 19.53%. This performance advantage is not solely attributable to superior pathfinding (i.e., fewer steps). For instance, in the (2 goals, $T_0 = 10$) and (4 goals, $T_0 = 50$) configurations, Dual-Lambda found shorter paths by 0.73% and 1.6% respectively, yet still had a higher execution time. This indicates that computational cost is a function of the interplay between the number of steps and the overhead of the T_0 parameter, rather than being determined by trajectory length alone. On average, the Predictive

Control algorithm required 5.3% fewer steps than Dual-Lambda across the tested scenarios.

Performance and Safety

The primary metrics for evaluating adaptation to task uncertainty are Reward, Reward per Step, and Steps Inside Proportion (%), as they directly reflect the agent's ability to balance goal achievement with safety constraints.

The MetaRL baseline serves as a clear outlier in this comparison. While its performance in the simplest single-goal scenariois comparable to perfomances of other algorithms in the most complicated scenariois, its efficacy collapses as complexity increases. For tasks with two or more goals, the total reward for MetaRL was more than 3 times worse than the poorest result from Dual-Lambda and 3.22 times worse than that of Predictive Control, demonstrating its inability to handle the compounded uncertainty.

Comparing the two proposed algorithms, the Predictive Control algorithm consistently outperforms the Dual-Lambda method in terms of cumulative reward across all configurations. Interestingly, the performance gap is narrowest in the least complex (1 goal, $T_0 = 10$) and most complex (4 goals, $T_0 = 50$) scenarios, at 5.74% and 6.92% respectively. For all other configurations, the Predictive Control algorithm's advantage in cumulative reward is more pronounced, averaging 22.98%. This trend is also reflected in the Reward per Step metric, where Predictive Control remains superior in all cases, though the average performance gap narrows to 14.63% versus 17.41% when calculated relative to the average cumulative reword.

The superiority of the Predictive Control algorithm is most evident in its safety performance. It consistently accumulates fewer Steps Inside Obstacles than both competitors, with the MetaRL baseline performing particularly poorly, entering obstacles at least four times more frequently on average than other presented algorithms. The Predictive Control agent averages 50 fewer steps inside obstacles than the Dual-Lambda agent, making it 71.8% more effective in absolute terms. When normalized by trajectory length, the Steps Inside Proportion (%) metric shows that the Predictive Control algorithm remains 69.0% more effective. Framing this metric as an "error rate" provides a stark comparison: the Predictive Control algorithm erros in 2.33% of its steps, compared to 7.51% for Dual-Lambda and a significant 14.84% for the MetaRL baseline.

Conclusion of Quantitative Analysis

In summary, this quantitative analysis confirms that both the Dual-Lambda and Predictive Control algorithms demonstrate a commanding dominance over the classical MetaRL baseline, which struggles to simultaneously pursue goals and avoid obstacles in complex environments. The Predictive Control algorithm emerges as the more effective and adaptive of the two proposed methods. While the performance margin over the Dual-Lambda algorithm is not always vast, the predictive approach consistently exhibits superior adaptability, particularly in its ability to balance goal-seeking and obstacle avoidance under significant task uncertainty.

3.4.2 Synthesis of Findings

Although both Dual-Lambda and the Alternative Predictive Control Algorithm are inherently predictive and share several architectural similarities—including their reliance on a discrete set of expert policies and overlapping hyperparameters—their core operational paradigms diverge in a critical aspect: how they utilize environment knowledge to predict the agent's next action.

Dual-Lambda computes the next move by simulating alternative environments (i.e., applying each potential environment's reward on the trajectory of a given policy) to determine which expert policy mix is optimal for the current situation. In contrast, the Alternative Predictive Control Algorithm reverses this paradigm: it simulates the progression of each expert policy within the actual environment, forecasting the outcomes and selecting the policy that yields the highest predicted return. This foundational difference in predictive focus shapes both the functional behavior and empirical performance of the two approaches.

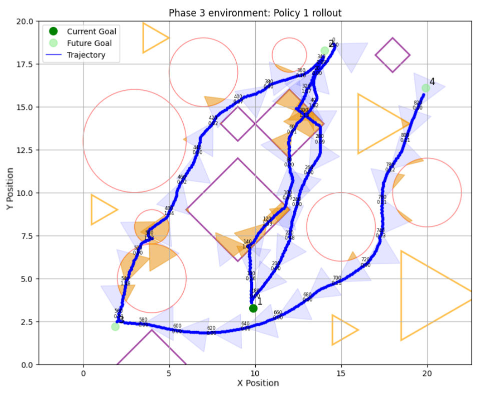
Robustness and Task Uncertainty

The primary metric of interest is the effectiveness and robustness of each method compared to random policy selection. Both algorithms reliably outperformed classic MetaRL approach, demonstrating a strong ability to reach all goal locations while skillfully avoiding obstacles, albeit with varying levels of trajectory efficiency. Also, as expected, naive randomization among policies typically leads to failure (Figure 28) in all but the most favorable cases (e.g., when the "correct" policy is randomly—and fortuitously—chosen and happens to match the task). This finding underscores the central challenge of task uncertainty: in real deployments, the agent cannot know in advance which policy is optimal, so adaptive selection is essential. Both Dual-Lambda and the Alternative Predictive Control Algorithm successfully identify and exploit the best available policies or their mixtures to address the active task, substantiating our core hypothesis.

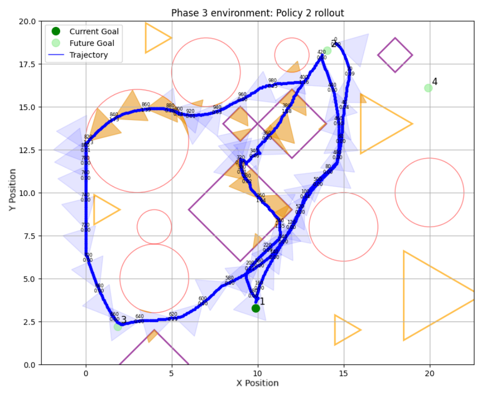
Sensitivity to Hyperparameters

Both algorithms exhibit pronounced dependence on key hyperparameters. For Dual-Lambda, the interaction between the learning rate for the λ coefficients (η_λ) and the rollout horizon (T_0) is pivotal: a large learning rate and long rollout can induce rapid, but unstable, shifts in policy selection, potentially degrading trajectory quality. Conversely, too small a learning rate or too short a horizon can hamper adaptation and cause sluggish convergence. In addition, the step size parameter can both increase the robustness of the algorithm and be a source of overfitting.

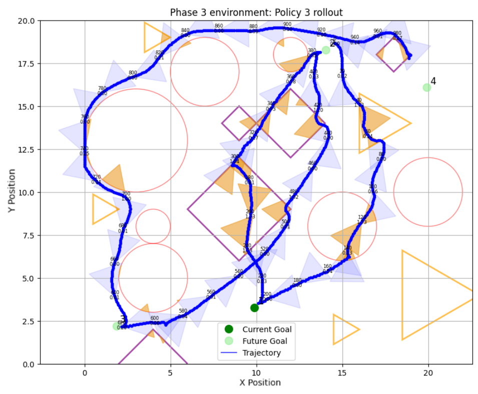
The Alternative Predictive Control Algorithm's main sensitivity is to the predictive horizon T_0: a short horizon limits foresight (and thus the advantage of prediction), while an excessively long horizon dramatically increases computational cost and may lead to overfitting to short-term outcomes. While both methods are relatively robust to hyperparameter variation in simple environments, for complex cases (e.g., Phase 3), performance becomes acutely dependent on careful tuning. Notably, achieving satisfactory outcomes in the most challenging scenarios required increasing T_0 to 50 steps for the Alternative Predictive Control Algorithm—a much longer lookahead



(a) Execution Trajectory $\pi^*_{r=1}$



(b) Execution Trajectory $\pi^*_{r=2}$



(c) Execution Trajectory $\pi^*_{r=3}$

Figure 28: Performance of standard RL static or predefined objective formulations in the environment with distribution of known and unknown obstacles and 4 targets.

than in earlier phases.

Fundamental Limitations

A key limitation of both algorithms is rooted in the expressiveness of the underlying ensemble of expert policies. Neither Dual-Lambda nor the Alternative Predictive Control Algorithm can synthesize fundamentally new behaviors that are absent from the expert policy set. If none of the pre-trained experts encapsulate strategies suitable for a novel task (e.g., negotiating polygonal obstacles or resolving unusual observation-reward sequences), both methods resort to suboptimal avoidance maneuvers, “clipping” obstacles or generating inefficient routes. This limitation was especially evident in Phase 3, where failures at circular obstacles—despite the algorithms' otherwise differing behaviors—were nearly identical and traced directly to the lack of relevant training experience among the experts. Nevertheless, the Alternative Predictive Control Algorithm exhibited greater adaptability with respect to novel obstacle types, on average intruding less deeply and recovering more quickly when it did so.

Theoretical Guarantees vs. Practical Flexibility

Dual-Lambda enjoys formal theoretical guarantees: under the assumption of a stationary environment and a finite task set, it provably converges to a saddle point of the min-max optimization problem, ensuring optimality in the minimax sense over the training distribution. This property is valuable when safety and guarantee of performance across all known scenarios is paramount.

In contrast, the Alternative Predictive Control Algorithm—being an instance of Model Predictive Control (MPC)—lacks such formal convergence or optimality proofs; its empirical robustness is grounded in experimental results rather than theory. This means that while it may excel in practice and adapt rapidly to unforeseen situations, it does not offer the same level of formal performance assurance in all settings.

Chapter 4

Conclusion

This dissertation was devoted to one of the fundamental problems of modern reinforcement learning (RL): ensuring the robustness and adaptability of intelligent agents under conditions of task-uncertainty. While classical RL approaches assume operation in a stationary environment with a fixed reward function, real-world applications, especially in robotics and autonomous systems, require agents to be able to quickly recognize and adapt to changing goals and constraints. The main research question we aimed to answer was whether an RL agent can be taught to effectively identify and adapt to the current task in real-time, using exclusively the feedback in the form of received reward.

To answer this question, we proposed a hypothesis that specialized adaptation mechanisms based on the analysis of the reward signal can achieve higher efficiency and robustness compared to standard RL solutions such as MetaRL. To test this hypothesis, we developed, implemented, and comprehensively analyzed two algorithmic approaches, testing them in a specially created simulation environment modeling the navigation of a mobile agent under various and previously unknown safety constraints.

4.1 Summary of Work and Key Contributions

This research addressed the challenge of ensuring robustness in reinforcement learning agents under task uncertainty. We moved away from the dominant MetaRL paradigm of training a single, monolithic policy. Instead, we proposed a modular framework based on a committee of pre-trained "expert" policies, each specialized for a single known task, reformulating the problem as one of online strategy selection.

The primary contributions of this work are theoretical, algorithmic, and empirical:

- **A Novel Theoretical and Algorithmic Framework:** We developed two distinct algorithms to solve the online selection problem.
  - **The "Dual-Lambda" Algorithm:** We introduced a new, robust algorithm grounded in a rigorous game-theoretic (max-min) formulation. A key theoretical result was using Lagrangian duality to transform the complex, non-convex policy optimization into a tractable min-max problem. The resulting algorithm is on-policy, model-free, and has theoretical guarantees of convergence to the minimax optimum.
  - **The MPC-style Predictive Algorithm:** As a pragmatic alternative, we applied model-predictive control principles to select the best expert at each step. By performing short-horizon simulations in the real environment, this algorithm demonstrates greater practical flexibility and the ability to produce emergent, complex behaviors.
- **A New Experimental Testbed for Robustness:** For empirical validation, we designed and implemented a custom 2D navigation environment with variable task parameters. Our three-phase testing protocol—progressing from baseline scenarios to complex environments and finally to a zero-shot generalization test with novel obstacle shapes (triangles and squares)—provides a clear and reproducible methodology for evaluating algorithms under task uncertainty.

- **In-depth Empirical Analysis and Findings:** Our comprehensive analysis went beyond a simple performance comparison to reveal fundamental properties of the approaches:
  - We clearly demonstrated that the minimax nature of the Dual-Lambda algorithm gives rise to a "healthy conservatism," a valuable property for safety-critical applications.
  - We showed that the predictive approach, while lacking formal guarantees, can find more effective and non-standard solutions by dynamically switching experts based on local context (e.g., using different policies for open spaces versus narrow passages).
  - Crucially, we identified a shared fundamental limitation: the performance of both algorithms is ultimately bounded by the expressive capacity and diversity of the initial expert policy set.

4.2 Response to the Stated Goals and Hypotheses

Based on the results obtained, we can provide detailed answers to the goals and hypotheses formulated in the introduction.

Goal 1: To develop an algorithmic framework that allows an RL agent to dynamically identify and adapt to the current task by analyzing reward signals. This goal was fully achieved. We developed not one, but two such frameworks: the Dual-Lambda algorithm and the alternative predictive algorithm. Both mechanisms successfully use the reward signal received in real-time to adapt their behavior. Dual-Lambda does this indirectly, by updating the weights λ_i to form an optimal stochastic policy, while the predictive approach uses the reward directly to evaluate hypothetical trajectories and select the best action.

Goal 2: To conduct a comparative analysis of the proposed method against baseline RL approaches in a simulated environment with task uncertainty. This goal was achieved. As a baseline approach, we used classic

MetaRL approach which uses single, monolithic, and complex policy for adapting to new tasks. The experiments, especially the results presented in Table 3.1, convincingly show that both of our adaptive algorithms outperform basic approach by orders of magnitude. Where a static or randomly selected policy might fail by being suboptimal (colliding with an obstacle or going into "loop" mode), our algorithms successfully reach all goals, demonstrating meaningful and purposeful adaptation.

Goal 3: To evaluate the efficiency and robustness of the developed approach. Both algorithms demonstrated high robustness in Phases 1 and 2, successfully coping with tasks for which they had "building blocks" in the form of expert policies. However, Phase 3 (generalization to new obstacle shapes) revealed the limits of their robustness, showing that it directly depends on the expressiveness and diversity of the initial set of experts.

Central Hypothesis: *"The mechanism introduced in this paper demonstrates strong capability to recognize and adjust to its assigned task within environments characterized by task uncertainty, and it surpasses traditional baseline methods in both efficiency and adaptability under uncertain conditions."* Our work has generally confirmed this hypothesis. As the comparative analysis showed, adaptive mechanisms are not just an improvement but a necessity for successful operation under task uncertainty. However, the study also clarified and supplemented this hypothesis, showing that robustness is not absolute and has clear boundaries defined by the quality of prior knowledge (expert policies).

4.3 Key Findings from the Experiments

The analysis of experimental data allowed us to draw several important conclusions that go beyond simply confirming the hypotheses.

1. **The trade-off between safety and efficiency is embedded in the architecture.** The Dual-Lambda algorithm, due to its minimax nature, is inherently "cautious." It aims to guarantee an acceptable result in any scenario, which makes it preferable for systems where the cost of an error is high. The

predictive algorithm, in contrast, is an "opportunist"—it seeks a locally optimal solution here and now, which can lead to faster and more efficient trajectories, but without formal guarantees.

2. **Emergent division of labor.** One of the most interesting observations was how the predictive algorithm autonomously "found" a use for the seemingly less useful policy $\pi^*_{r=2}$. In complex scenarios (Phase 2 and 3), it began to selectively use this expert to navigate narrow passages between obstacles or to "handle" the corners of obstacles, although it was not specifically trained for this. This demonstrates the ability of online planning to create complex composite behavior.

3. **Perceptual ambiguity and the limits of generalization.** Phase 3 was a harsh test of generalization and revealed the main insight of the entire work. When the agent encounters completely new stimuli (polygonal obstacles), the sequences of observations it receives are fundamentally different from anything it "saw" during training. None of the expert policies has a ready-made recipe for such a situation. As a result, both algorithms begin to fail: they "scrape" along the edges of obstacles, and sometimes even pass right through them. Notably, failures occurred even with familiar circular obstacles if they were in a new, complex context.

4. **The "performance ceiling" is determined by the set of experts.** This conclusion follows from the previous one and is perhaps the most important. Neither Dual-Lambda nor predictive control can "invent" fundamentally new behavior that is absent from their basic repertoire. They can only combine and switch between existing skills. If no expert knows how to, for example, navigate sharp corners, then their combination will not learn it either. This means that the success of such modular systems in the real world will depend not so much on the switching mechanism as on the quality and diversity of the set of expert policies itself.

4.4 Limitations of the Study

Like any scientific work, our study has a number of limitations that outline its scope of applicability and open up directions for future research.

- **Fixed and small set of experts:** In our work, we used a small ($m = 3$), predefined, and fixed set of expert policies. We did not consider how to optimally form this set, how to add new experts when new tasks appear (Continual Learning), or what to do if the number of potential tasks is continuous.
- **Specificity of the simulation environment:** All experiments were conducted in a two-dimensional simulation with idealized physics and sensorics. Transferring these algorithms to a real robotic platform will face additional difficulties: sensor noise, inaccuracies in actuator performance, delays in the control system, and the sim-to-real gap.
- **Simple policy combination mechanisms:** The Dual-Lambda algorithm implements a probabilistic selection of one policy at each time segment, while the predictive one makes a deterministic choice of one policy. We did not investigate more complex ways of combining them, such as weighted averaging of their output actions (action averaging) or using more complex hierarchical architectures where one policy modulates the outputs of others.
- **Dependence on hyperparameters:** We showed that the performance of both algorithms, especially in complex conditions, is sensitive to the tuning of hyperparameters, such as the learning rate η_λ and the planning horizon T_0. The work did not aim to automatically tune these parameters, which could be critical for practical applications.
- **Computational complexity:** For conducting the experiment as well as for training the expert policies, GPUs were not used, in order to test comparability with robotic systems that also have limited computational capabilities. However, considering the trend of growing computational power of on-board

computing, it can be said that this limitation—not testing the algorithms in their prime—is nevertheless a limitation of the study.

4.5 Prospects for Future Work

The recognition of the limitations of the current study directly points to promising directions for its continuation and development.

- **Dynamic replenishment of the expert set:** The most obvious and important next step is to develop mechanisms that will allow the agent not only to choose from existing policies but also to create new ones "on the fly" when it realizes that none of the existing ones can cope with the current situation. This direction lies at the intersection of reinforcement learning and Continual Learning.
- **Hierarchical and compositional policies:** Instead of simply switching between whole policies, one could investigate more subtle mechanisms for their composition. For example, one could use an attention mechanism that would dynamically weigh and combine the actions proposed by all experts at each step. This could lead to much smoother and more adaptive behavior.
- **Transition to a continuous task space:** Instead of a discrete set of tasks (radius $r \in \{1, 2, 3\}$), one could consider the case where the task parameter can take any value from a continuous range. This would require moving from a set of discrete experts to a parameterized policy, where the task parameter is one of the inputs, and our proposed mechanisms could be adapted for online estimation of this unknown parameter.
- **Testing on real hardware and in more complex tasks:** Testing the developed algorithms on real mobile robots would be the ultimate proof of their practical value. In addition, it would be interesting to apply these approaches to other areas, such as manipulation tasks, where uncertainty might lie in the mass or shape of the object being moved.

- **In-depth theoretical analysis:** For the Dual-Lambda algorithm, it would be useful to investigate its convergence rate depending on the properties of the environment and the algorithm's parameters. For the predictive approach, it would be interesting to develop methods that could provide at least probabilistic guarantees of safety or performance.

- **Hyperparameter Optimization:** As proposed above, the difficulty of selecting hyperparameters can be one of the main limitations of the algorithms, so optimization or automatic selection of hyperparameters would significantly improve the robustness of the methods under consideration.

4.6 Final Word

In conclusion, this dissertation has presented and comprehensively investigated two modular approaches to solving the problem of robustness in reinforcement learning under task uncertainty. We have shown that both a theoretically grounded method based on duality and a pragmatic method based on predictive control significantly outperform standard approaches and allow the agent to successfully adapt to changing conditions in real-time.

The main lesson learned from this work is that in modular systems, there is an inseparable link between the complexity of the adaptive mechanism and the diversity of the basic skills. Our algorithms have demonstrated that even a simple but smart switching mechanism can work wonders if it has good "building blocks." However, they also showed that no adaptation mechanism can save the day if the basic skills are inadequate for a new, unforeseen situation.

This work lays the foundation for further research in creating truly flexible and intelligent autonomous systems. We believe that the future belongs to hybrid architectures that combine the power of deep learning to form basic skills with the elegance of classical methods of control theory and optimization for their meaningful and robust composition.

I would like to express my sincere gratitude to my supervisor, Dr. Miguel Calvo-Fullana, for his invaluable support, wise advice, and inspiring discussions throughout this research. His guidance was key to the successful completion of this work.

List of Figures

List of Tables

Bibliography

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

[2] Kober, J., Bagnell, J. A. & Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research* **32**, 1238–1274 (2013).

[3] Kiran, B. R. *et al.* Deep reinforcement learning for autonomous driving: A survey. *IEEE transactions on intelligent transportation systems* **23**, 4909–4926 (2021).

[4] Kiumarsi, B., Vamvoudakis, K. G., Modares, H. & Lewis, F. L. Optimal and autonomous control using reinforcement learning: A survey. *IEEE transactions on neural networks and learning systems* **29**, 2042–2062 (2017).

[5] Rakelly, K., Zhou, A., Finn, C., Levine, S. & Quillen, D. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International conference on machine learning*, 5331–5340 (PMLR, 2019).

[6] Cervino, J., Bazerque, J. A., Calvo-Fullana, M. & Ribeiro, A. Multi-task reinforcement learning in reproducing kernel hilbert spaces via cross-learning. *IEEE Transactions on Signal Processing* **69**, 5947–5962 (2021).

[7] Beck, J. *et al.* A survey of meta-reinforcement learning. *arXiv preprint arXiv:2301.08028* (2023).

[8] Finn, C., Abbeel, P. & Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135 (PMLR, 2017).

[9] Caruana, R. Multitask learning. *Machine learning* **28**, 41–75 (1997).

[10] Hospedales, T., Antoniou, A., Micaelli, P. & Storkey, A. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence* **44**, 5149–5169 (2021).

[11] Wang, M., Li, X., Zhang, L. & Wang, M. Metacard: meta-reinforcement learning with task uncertainty feedback via decoupled context-aware reward and dynamics components. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, 15555–15562 (2024).

[12] Lockwood, O. & Si, M. A review of uncertainty for deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 18, 155–162 (2022).

[13] Osband, I., Blundell, C., Pritzel, A. & Van Roy, B. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems* **29** (2016).

[14] Gal, Y. & Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, 1050–1059 (PMLR, 2016).

[15] Altman, E. *Constrained Markov decision processes* (Routledge, 2021).

[16] Garcıa, J. & Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal of Machine Learning Research* **16**, 1437–1480 (2015).

[17] Wiesemann, W., Kuhn, D. & Rustem, B. Robust markov decision processes. *Mathematics of Operations Research* **38**, 153–183 (2013).

[18] Paternain, S., Chamon, L., Calvo-Fullana, M. & Ribeiro, A. Constrained reinforcement learning has zero duality gap. *Advances in Neural Information Processing Systems* **32** (2019).

[19] Bhatnagar, S. & Lakshmanan, K. An online actor–critic algorithm with function approximation for constrained markov decision processes. *Journal of Optimization Theory and Applications* **153**, 688–708 (2012).

[20] Tessler, C., Mankowitz, D. J. & Mannor, S. Reward constrained policy optimization. *arXiv preprint arXiv:1805.11074* (2018).

[21] Achiam, J., Held, D., Tamar, A. & Abbeel, P. Constrained policy optimization. In *International conference on machine learning*, 22–31 (PMLR, 2017).

[22] Borkar, V. S. An actor-critic algorithm for constrained markov decision processes. *Systems & control letters* **54**, 207–213 (2005).

[23] Paternain, S., Calvo-Fullana, M., Chamon, L. F. & Ribeiro, A. Safe policies for reinforcement learning via primal-dual methods. *IEEE Transactions on Automatic Control* **68**, 1321–1336 (2022).

[24] Calvo-Fullana, M., Paternain, S., Chamon, L. F. & Ribeiro, A. State augmented constrained reinforcement learning: Overcoming the limitations of learning with rewards. *IEEE Transactions on Automatic Control* **69**, 4275–4290 (2023).

[25] Guo, Y. *et al.* Explainable action advising for multi-agent reinforcement learning. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 5515–5521 (IEEE, 2023).

[26] Thakur, S., van Hoof, H., Higuera, J. C. G., Precup, D. & Meger, D. Uncertainty aware learning from demonstrations in multiple contexts using bayesian neural networks. In *2019 International Conference on Robotics and Automation (ICRA)*, 768–774 (IEEE, 2019).

[27] Lee, S., Seo, Y., Lee, K., Abbeel, P. & Shin, J. Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble. In *Conference on Robot Learning*, 1702–1712 (PMLR, 2022).

[28] Boyd, S. P. & Vandenberghe, L. *Convex optimization* (Cambridge university press, 2004).

[29] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347* (2017).

[30] Towers, M. *et al.* Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032* (2024).