

Universiteit Leiden

Master Computer Science

Learning When to Switch:
Adapting Risk Attitude to Reach a Goal Under Time Constraints

Name:	Lorenzo Madiai
Student ID:	s4069838
Date:	23/06/2026
Specialisation:	Artificial Intelligence
1st supervisor:	Lilian de Martin (TNO)
2nd supervisor:	Jan van Rijn (LIACS)

Master's Thesis in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Preface

This thesis marks the final chapter of my academic journey at Leiden University. Looking back, I realize how much I have learned and grown over the past two years, both academically and personally. Living far from home, adapting to a new environment, and facing unfamiliar challenges have been transformative experiences that have shaped me in ways I could not have anticipated when I first arrived in Leiden.

First of all, I would like to express my sincere gratitude to both of my thesis supervisors, Lilian de Martin and Jan van Rijn. I would like to thank Lilian for the time, guidance, and trust she dedicated to me throughout this journey. My internship at TNO represented my first professional experience, and her support encouraged me to take on ambitious challenges and step outside my comfort zone, teaching me that growth often comes from facing difficult problems.

I would also like to thank Jan for his continuous guidance and constructive feedback throughout this project. His insights consistently challenged me to think critically and analyze problems independently, encouraging me to explore directions that initially seemed intimidating but ultimately became valuable learning experiences.

Finally, I would not have been able to complete this journey without the people who accompanied me along the way. I am deeply grateful to the friends I met during my time in Leiden and to those who continued to support me from afar despite the distance. Last but not least, I also owe a special thanks to my family for their constant encouragement, unconditional support, and advice whenever I needed it.

The memories, experiences, and relationships built over these two years will remain among the most meaningful aspects of this chapter of my life, and I will always cherish them.

Grazie a tutti.

Lorenzo Madiai
Leiden, June 2026

Abstract

Artificial Intelligence systems deployed in mission-critical settings must balance, among other things, task performance, safety, and time constraints. In these scenarios, risk is inherently context-dependent, since the acceptable level of risk changes during a task or operation, especially when time pressure is involved. However, most reinforcement learning (RL) and safe reinforcement learning (SafeRL) methods rely on fixed trade-offs or static safety constraints, which limit their ability to adapt to time-varying operational requirements.

Motivated by this limitation, this thesis studies a time-constrained reinforcement learning setting where time and risk must be considered jointly: when sufficient time is available, safer behavior should be the priority, whereas tighter deadlines may require more risk-tolerant actions to preserve task success.

To address this issue, we propose a two-level policy-switching framework in which a high-level controller selects between two pre-trained low-level policies: a conservative policy that prioritizes safety and an aggressive policy that prioritizes task completion. Experiments in a Safety Gym navigation task show that this hierarchical separation enables more adaptive behavior across different time-budget regimes. The proposed method substantially decreases risk exposure relative to a simple standard policy, while simultaneously enhancing task performance by 10% compared with a risk-aware policy.

The source code used to generate the experimental results presented in this thesis is available at: `https://github.com/lorenzomadiai/msc-thesis`.

Key words and Phrases: *Reinforcement Learning; Safe Reinforcement Learning; Risk-Aware Control; Time Constraints; Policy Switching; Hierarchical Decision Making; Safety Gym*.

Contents

1 Introduction **1**
- 1.1 Motivation . . . 1
- 1.2 Problem Statement . . . 2
  - 1.2.1 Research Questions . . . 3
- 1.3 Acknowledgements . . . 3
  - 1.3.1 Ethical Considerations . . . 3
  - 1.3.2 Use of Artificial Intelligence tools . . . 4

2 State of the Art **5**
- 2.1 Problem Context . . . 5
  - 2.1.1 Risk definitions . . . 5
  - 2.1.2 The interplay between time and military effectiveness . . . 6
- 2.2 Existing Reinforcement Learning Approaches . . . 8
  - 2.2.1 Standard Reinforcement Learning . . . 8
  - 2.2.2 Safe Reinforcement Learning . . . 9
- 2.3 Time-Awareness in Reinforcement Learning . . . 11
- 2.4 Research Gap . . . 13

3 Problem Formulation **15**
- 3.1 Time-Budget Augmented State Space . . . 16
- 3.2 Success and Deadline Failure Events . . . 17
- 3.3 Cumulative Reward, Safety Cost, and Success Probability . . . 18
- 3.4 Low and High-Budget Regimes . . . 19
- 3.5 CVaR-Based Risk Bound in the High-Budget Regime . . . 19
- 3.6 Optimization Problem . . . 21

4 Proposed Method **23**
- 4.1 Motivation and Design Rationale . . . 23
- 4.2 Method Overview . . . 25
- 4.3 Defining the Risk-Attitudes . . . 25
- 4.4 Optimal Stopping Formulation . . . 26

5 Implementation **29**
- 5.1 Experimental Setup . . . 29
- 5.2 Metrics . . . 31
  - 5.2.1 Classifier Performance Metrics . . . 31
  - 5.2.2 Agent Performance Metrics . . . 33
- 5.3 Baseline Methods . . . 33
  - 5.3.1 SAC: Aggressive Baseline . . . 34
  - 5.3.2 SAC: Flat Baseline . . . 34
  - 5.3.3 WCSAC: Conservative Baseline . . . 34

5.3.4 Training Configuration 35
5.4 Proposed Method 35
5.4.1 Computation of the Q-values 35
5.4.2 Supervised dataset construction 37
5.4.3 Training of the switching classifier 38
5.4.4 Deployment procedure 39

6 Results **41**
6.1 Evaluation Protocol 41
6.2 Main Results 42
6.2.1 Performance Comparison 42
6.2.2 Effect of Time Budget on Performance and Risk 43
6.3 Classifier Evaluation 44
6.3.1 Validation of the switching-controller 44
6.3.2 Effect of the Decision Threshold 45
6.4 Discussion 46

7 Conclusions **49**

References **51**

Chapter 1

Introduction

1.1 Motivation

Artificial intelligence (AI) is rapidly transitioning from concepts and prototypes to operational components embedded within military systems and missions. Advances in machine learning, autonomy, and data-driven decision making promise significant gains in speed, precision, and adaptability across domains such as intelligence, surveillance and reconnaissance, command and control, logistics, and weapon systems. However, the move from laboratory prototypes to deployed, mission-critical capabilities remains a central challenge. This transition, often referred to as the operationalization of AI, describes the effort to move traditional AI solutions, typically developed to achieve specific statistical performance thresholds (e.g., recall or precision), into the domain of algorithmic warfare, where the ultimate objective is the accomplishment of a military mission [1] [2].

A standard military mission can be understood as a continuous effort "to capitalize on opportunities and maximize gains, while avoiding threats and minimizing costs or losses, by selecting the alternative from a set of options that maximizes "expected utility" [3]. In this context, utility (or opportunity) and risk constitute a persistent and structural trade-off, not only at the strategic level of the mission but also throughout its execution [4].

However, traditional approaches to risk definition and assessment, well established in engineering and other safety-critical domains, are insufficient for AI-enabled systems. In such systems, "many risks remain inherent and unavoidable, given the complex, dynamic, and adversarial nature of the environments in which these systems operate" [5].

A key challenge lies in defining what constitutes risk for military AI. Since the spectrum of risks for AI systems ranges from the risk of technical failures, such as model inaccuracies, data drift, or cyber vulnerabilities, to the risk of misalignment between system behaviour and operational intent, degraded human–machine teaming, ethical or legal violations, and unintended escalation dynamics [6].

As a result, risk must be defined across multiple layers, ranging from algorithmic and system-level properties to operational and mission-level effects. This multi-layered nature complicates both the identification of relevant risk factors and the assessment of their likelihood and impact.

At the mission level, risk becomes substantially more difficult to define and assess, because it cannot be separated from the broader set of competing operational objectives that together determine the military utility of an AI system. In this context, the value of an AI system cannot be reduced to isolated technical metrics, but must instead be understood as a compound function of mission-level objectives, such as desired outcomes, schedule, cost, and risk [4].

1.2 Problem Statement

In this work, we focus on reinforcement learning (RL), as it provides a natural framework for handling dynamic trade-offs between competing objectives under uncertainty. RL is of growing interest in the defence domain [7], and its integration into autonomous platforms, including military vehicles and drones, has already been explored [8]. As highlighted by the authors of [8], the adoption of robot and autonomous systems (RAS) in military settings requires adaptability, particularly in highly dynamic scenarios where mission conditions evolve unpredictably, and the dominance of risks shifts with the mission state.

To study how existing RL methods behave when risk tolerance varies over time, we focus on scenarios in which mission requirements impose changing trade-offs between performance and safety.

This naturally motivates the use of safe reinforcement learning (SafeRL), in which agents are required not only to achieve task objectives but also to satisfy explicit safety requirements, such as "not visiting some states considered dangerous in the environment and not taking some actions that could cause danger"[9].

However, many SafeRL formulations rely on fixed safety constraints or predefined notions of hazardous behaviour. This limits their relevance for dynamic and context-dependent decision-making problems. In operational settings, risk cannot be assessed in isolation from the broader context in which the system is deployed. Instead, it must be evaluated together with competing objectives, such as mission success, safety, and adherence to operational requirements. These objectives jointly determine the utility of the system and may change in relative importance over time.

A similar need has been recognised in prior work on reinforcement learning. For instance, the authors of [10] suggest the possibility of learning, in parallel through multi-policy methods, a set of policies that are optimal under different risk-sensitivity preferences. However, they do not provide a concrete method for constructing or selecting such policies in settings where risk preferences vary with the operational context.

In this work, we argue that a utility-based formulation provides a more expressive framework than a risk-based formulation alone. Rather than modelling risk only through hazardous events or fixed safety constraints, our framework represents the acceptability of outcomes as a function of the operational context.

This raises a fundamental question: can existing standard or safe reinforcement learning methods represent and adapt to mission-level notions of utility, where safety must be balanced against other operational objectives rather than enforced as a static requirement?

In this work, we address this question and demonstrate that existing approaches offer only limited support for such adaptability. Consequently, we propose a

method that enables our system to explicitly adjust its risk attitude as a function of the mission context.

1.2.1 Research Questions

Taking into consideration what has been raised in the Problem Statement, the high-level research questions of this project are:

- ◇ What are the limitations of current reinforcement learning approaches based on a fixed trade-off between performance and risk in applications that require both safety guarantees and performance guarantees?
- ◇ Can current risk-aware reinforcement learning methods adequately handle scenarios in which the acceptable level of risk varies with the operational conditions?
- ◇ How can we design a simple mission in which the optimal behaviour requires a dynamic risk attitude?

1.3 Acknowledgements

1.3.1 Ethical Considerations

This work is motivated in part by decision-making problems that arise in military and other time-critical domains, particularly those involving trade-offs between risk and task completion under time constraints. Because decision-making under risk and time constraints can arise in sensitive application domains, potential ethical implications were considered throughout the design and evaluation of this work. Furthermore, the proposed method is domain-agnostic and could equally be applied to non-military settings such as robotics, autonomous navigation, and industrial automation.

Specifically, the scope of this work is limited to an abstract and controlled research setting on the trade-off between risk and time. This trade-off is not limited to military applications; it is a broader decision-making problem that has attracted significant attention across the reinforcement learning and planning literature.

The contribution of this thesis is methodological: it studies whether an RL agent can adapt its risk attitude under time constraints by switching between pre-trained policies. The work is intended to improve understanding of safety, risk-awareness, and context-dependent behaviour in reinforcement learning. The chosen environment is the standard academic benchmark for safe reinforcement learning, Safety Gym [11], and both the proposed method and the baseline policies are evaluated only on a simulated Safety Gym navigation task. The results should be interpreted only within this simplified benchmark setting.

In conclusion, this project does not develop, deploy, or evaluate an operational military system. Its contribution is limited to a simulated reinforcement learning benchmark, and the results should therefore be interpreted as methodological research rather than as a directly deployable capability.

1.3.2 Use of Artificial Intelligence tools

During this project, I used OpenAI's ChatGPT-5.5 and Anthropic's Claude Sonnet 4, accessed through GitHub Copilot, as supplementary tools to support various stages of the thesis process.

ChatGPT was mainly used as a starting point to broaden my understanding of unfamiliar topics and to identify potentially relevant research directions. Suggested papers were not automatically included; instead, each publication was independently assessed for relevance and quality before being considered. The literature review was conducted through a combination of AI-assisted suggestions, searches on Google Scholar, and backward snowballing by examining the references of relevant papers. This process allowed me to progressively build a comprehensive understanding of the research field while maintaining full responsibility for the selection and evaluation of all sources included in the thesis. ChatGPT was also occasionally used to discuss theoretical concepts encountered during the development of the project.

This AI tool was also used to improve grammar, sentence structure, and the overall readability of existing text when I thought it was necessary, but the ideas, analyses, and conclusions presented in this thesis were developed through my own critical thinking.

Regarding Claude Sonnet 4, it was primarily used to assist with debugging, environment configuration, and the development of auxiliary functions and analysis scripts, such as those used for generating plots. Again, all the methodological design, experimental setup and implementation of the proposed approach come from personal thinking. This AI tool was particularly helpful in enabling me to identify and resolve implementation errors more efficiently, accelerate debugging activities, and assess faster whether the development process was progressing in the intended direction.

Overall, the use of AI tools helped the literature review process, facilitated the exploration of relevant research directions, improved writing efficiency, reduced the time required for debugging, and helped with the implementation of the proposed method. This allowed me to devote more time to methodological design, result analysis and interpretation.

In conclusion, since AI-generated suggestions can occasionally contain inaccuracies, incomplete explanations, or references that are not directly relevant, all AI-generated outputs were critically evaluated and independently verified before being used.

Chapter 2

State of the Art

2.1 Problem Context

Before discussing existing reinforcement learning approaches, it is necessary to clarify the operational context that motivates the problem addressed in this thesis, with particular emphasis on the interplay between risk, time constraints, and military utility.

2.1.1 Risk definitions

Numerous scholars have observed that the term "risk" presents significant challenges due to its wide-ranging and often ambiguous connotations [3, 12, 13]. When attempts are made to define risk with greater specificity, it becomes clear that alternative terms already exist that capture the same concepts.

For some, risk is a synonym of probability. For example, consider an autonomous military vehicle that needs to choose a course of action between crossing the river or detouring to find a bridge. One could ask, what is the risk of crossing the river? In reality, a more accurate question would be, what is the probability that the vehicle gets stuck when attempting to cross the river?

Another definition of risk often used in risk assessment methods involves a combination of probability (P) and magnitude (M) of loss associated with the occurrence of a particular event or type of event [13] . This definition of risk is problematic when dealing with AI systems operating in complex environments because P and M are not easy to define or obtain.

Finally, risk is sometimes used interchangeably with volatility. More precisely, volatility refers to the degree of variability or dispersion of a quantity over time, particularly when fluctuations are frequent, rapid, or difficult to predict. In the financial literature, volatility has been extensively studied as a central component of risk measurement. For instance, the authors of [14] analyze how different volatility measures affect Value-at-Risk (VaR) forecasts, thereby framing volatility as a fundamental driver of distribution-based risk models.

In addition to the ambiguities associated with the term "risk", military operational environments are characterized by competing objectives. An example is shown in Fig. 2.1, in which several aspects, such as military effectiveness and

suitability, aggregate to provide military utility. It is interesting to observe that two aspects of military effectiveness is "compliance with risk objectives" and "compliance with objectives". Some scholars fold every dimension of the mission into a risk; for instance, the risk of not producing the desired outcome or the risk of not complying with schedule.

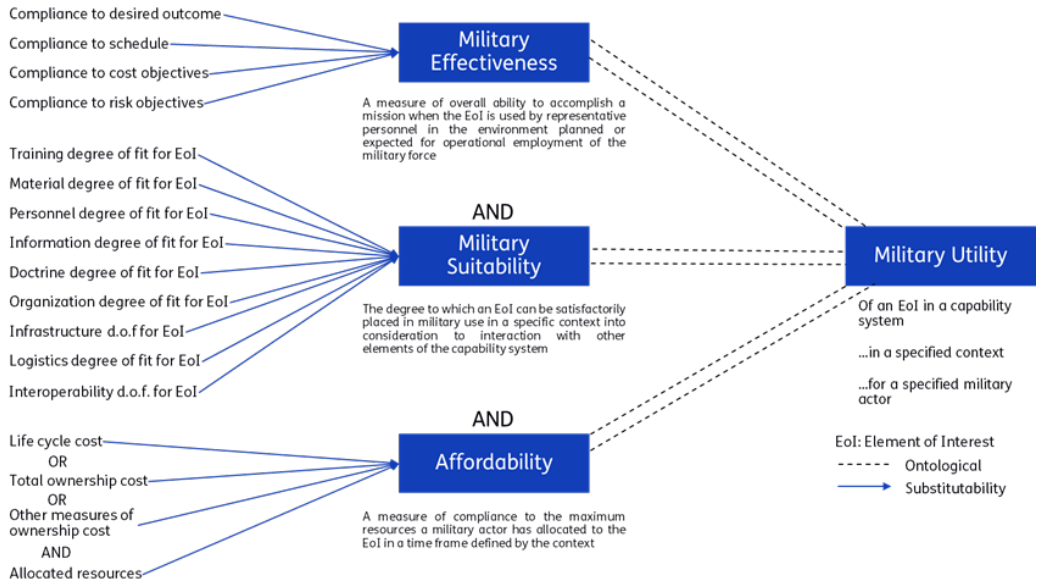


Figure 2.1: The military utility concept introduced by the authors of [4] for the study of the use of technology in military operations.

By following the taxonomy of 2.1, we will consider that the military effectiveness of a AI system is given by:

- ◇ Compliance with the desired outcome, which represents the primary purpose of the mission
- ◇ Compliance with schedule requirements, which capture the temporal constraints under which the mission must be accomplished
- ◇ Compliance with cost objectives, such as constraints on personnel, financial resources, or energy consumption
- ◇ Compliance with risk objectives, such as limiting exposure to hazards or other adverse consequences associated with mission accomplishment

In this work, we focus on outcome, schedule, and risk, and do not model compliance with cost objectives explicitly. We therefore treat the utility of the considered AI system as a simplified operational proxy for military effectiveness, defined as

$$U = f(\text{outcome}, \text{schedule}, \text{risk}). \tag{2.1}$$

2.1.2 The interplay between time and military effectiveness

Time is a fundamental yet often under-specified dimension of risk in military operations. Mission risk is not static; rather, it must be understood through a

"constant awareness of the changing risks associated with the operation" [15]. As mission objectives, environmental conditions, and operational constraints evolve, decisions that may be acceptable early in a mission can become untenable later, while options initially judged too risky may become necessary to prevent mission failure. This is especially true in military settings, where decision-makers are often required to make complex decisions under "hard deadlines" [15]. In such contexts, risk cannot be understood independently of time. The consequences of a poor decision are shaped not only by the decision itself, but also by how much time remains to detect, correct, or recover from it [16]. As a result, the cost of failure may increase as the mission progresses and the opportunity for recovery narrows. Time, therefore, is not merely an external constraint on action, but a core factor that shapes how risk is perceived, prioritized, and managed throughout the mission.

At the mission level, time scarcity directly reshapes the risk landscape. In sequential decision-making problems, the value of actions and associated risks depend on the remaining time and future opportunities for recovery [16] [17]. In military contexts, this effect is amplified by operational constraints which require decision-makers to continuously reassess risk as the mission unfolds [15]. When sufficient time is available, conservative strategies that emphasize safety are often preferred. Conversely, as deadlines approach, the dominant risk may shift toward mission failure itself, making the acceptance of higher immediate risk operationally justified.

From a risk definition perspective, time introduces the notion of risk dominance: at any given moment, certain risks exert a greater influence on mission success than others. This dominance can shift continuously or abruptly as time progresses, external events unfold, or new information becomes available. Effective mission-level risk assessment must therefore account not only for the presence of multiple risks, but for how their relative significance evolves over time. Capturing this evolution is essential for aligning AI behavior with operational intent.

To enable the effective operationalization of AI systems, it is necessary to explicitly represent risk, in contrast to standard methods that collapse it into a fixed trade-off between performance and safety [6, 18, 19]. At the same time, such systems should not rely on the assumption of a constant, global safety constraint or a fixed risk level applied uniformly throughout the mission. Rather, they should account for the fact that the dominant risk may shift over time and across mission phases, with different hazards becoming more or less critical as the operational context evolves. This, in turn, requires adaptive behaviour and "the need of multiple behaviours with a diverse range of different sensitivities to risk" [10].

Time and risk-aware reinforcement learning systems are essential because many operational decisions involve trade-offs that evolve as a function of remaining time rather than static objectives. In mission-oriented settings, the acceptability of risk depends not only on the nature of the action but also on when it is taken within the mission timeline. Making an RL agent time-aware, for example by embedding time in the observation state of the agent, is a relatively simple approach [20].

However, when performance and safety are competing objectives, this design can implicitly induce an uncontrolled trade-off between them, yielding an agent whose level of risk sensitivity cannot be explicitly controlled or adapted to mission demands.

The core challenge, therefore, is not merely to incorporate time, but to enable an agent to reason jointly about time and risk, so that its behavior remains consistent with the mission's expected outcome, schedule, and risk requirements.

2.2 Existing Reinforcement Learning Approaches

Existing reinforcement learning approaches address sequential decision-making under uncertainty from different perspectives, including performance optimization, explicit safety constraints, risk-sensitive decision-making, and temporal adaptation. However, each family of methods relies on assumptions that limit its applicability to operational settings where the acceptable level of risk depends on the evolving mission context.

2.2.1 Standard Reinforcement Learning

Standard reinforcement learning (RL) models sequential decision-making as a Markov Decision Process (MDP) [21, 22], defined by the tuple

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma) \tag{2.2}$$

where $\mathcal{S}$ is the state space, $\mathcal{A}$ is the action space, $P(s'|s,a)$ defines the transition dynamics, $R(s,a)$ is the reward function, and γ is the discount factor.

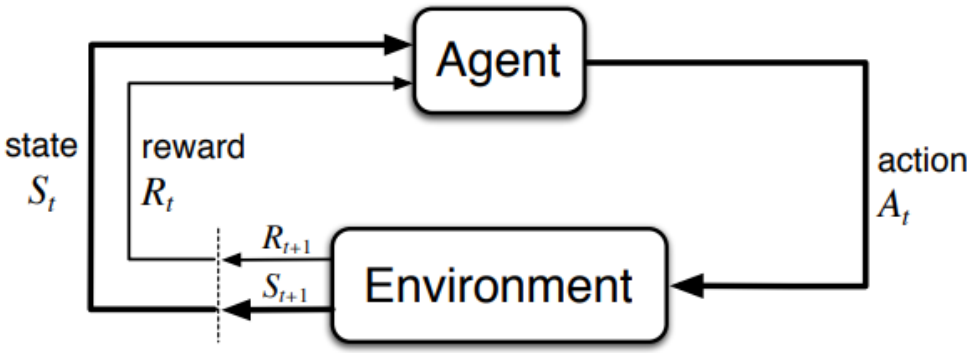


Figure 2.2: Reinforcement learning framework introduced in [22]: at each time step t, the agent selects an action A_t based on the current state S_t, and the environment returns the next reward R_{t+1} and the next state S_{t+1}.

The RL framework is described as a sequential decision-making process, with each decision occurring at a specific timestep t. The number of decision steps can be either infinite or finite for each episode. An episode refers to a complete interaction between the agent and the environment, ending when a terminal state is reached. So with S_t, R_t, A_t we refer respectively to the state, the reward and the action taken at each time step t. As illustrated in Fig. 2.2, at each time step the agent selects an action according to a policy $\pi(a|s)$. A policy can be implemented as a neural network that takes as input the current state S_t and

gives as output the action A_t, intending to maximize the expected cumulative return:

$$J_R(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \right] \tag{2.3}$$

leading to the optimization problem:

$$\pi^* = \arg\max_\pi J_R(\pi) \tag{2.4}$$

For further algorithmic details, we refer the reader to [22]. This formulation assumes that all relevant objectives can be encoded within a single scalar reward function. As a consequence, any trade-off between competing objectives such as task performance and safety is handled implicitly through reward shaping.

To represent safety, additional information complementary to the reward is introduced into the environment in the form of a cost or penalty C_t, intended to capture unsafe or dangerous actions.

The expected cumulative cost is defined as:

$$J_C(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t C_{t+1} \right] \tag{2.5}$$

Consequently, the policy π will maximize an implicit trade-off between performance and safety, for example:

$$\max_\pi \; J_R(\pi) - \lambda J_C(\pi), \tag{2.6}$$

where $J_R(\pi)$ denotes the expected performance of policy π, $J_C(\pi)$ denotes the expected safety-performance, and $\lambda \geq 0$ controls the relative importance assigned to safety in that specific problem.

While the standard formulation is effective in many domains, it becomes problematic in safety-critical settings. A policy that maximizes expected reward may accept unsafe behaviors if they improve performance, and no explicit guarantees can be provided regarding constraint satisfaction.

Furthermore, the learned behavior reflects a fixed trade-off determined during reward design, making adaptation to changing operational requirements difficult.

A representative example of this class of methods is Soft Actor-Critic (SAC), an off-policy actor-critic algorithm based on the maximum-entropy reinforcement learning framework, where the entropy term encourages stochasticity and improves exploration during learning [23].

2.2.2 Safe Reinforcement Learning

While standard reinforcement learning optimizes performance through reward maximization, safety-critical applications require explicit control over undesirable behaviors. In standard RL, safety considerations are typically incorporated indirectly through reward shaping, which provides no formal guarantee that unsafe behaviors will remain within acceptable limits. This limitation motivated the

development of *Safe Reinforcement Learning* (SafeRL), a field concerned with learning policies that achieve high performance while explicitly satisfying safety constraints during training and deployment [9].

Expectation-based Optimization

A common formulation for SafeRL is the *Constrained Markov Decision Process* (CMDP) [24], which extends the standard MDP by introducing explicit cost signals and safety requirements. A detailed introduction to this formulation is given in Section 3. The resulting optimization problem is:

$$\max_{\pi} J_R(\pi) \quad \text{s.t.} \quad J_c^{(i)}(\pi) \leq d_i, \quad i = 1, \ldots, m \tag{2.7}$$

where $J_R(\pi)$ denotes the expected cumulative reward, and $J_C^{(i)}(\pi)$ the expected cumulative cost associated with the i-th safety constraint. The values d_i represent the corresponding safety budgets, each defining the maximum admissible level for the associated expected safety cost.

This formulation makes safety explicit rather than embedding it indirectly into the reward function, and it restricts the space of the admissible policies. The goal is to find a policy that maximises the objective function while at the same time satisfying the safety constraint defined a priori. This limits the possible behaviours that the policy can perform after the training, especially when, during the deployment, we would like to have an agent capable of performing multiple behaviours.

To overcome this limitation of deterministic policies trained for a single safety constraint, the authors of [25] propose an algorithm based on the Budgeted Markov Decision Process (BMDP). The BMDP formulation was introduced in [?] as an extension of the CMDP formulation. The idea is to enable a policy to have online control over the budget β, defined within an interval $\mathcal{B} \subseteq \mathbb{R}$ of admissible budgets/safety costs. The problem formulation can be written as:

$$\max_{\pi} J_R(\pi) \quad \text{s.t.} \quad J_C(\pi) \leq \beta, \quad \beta \in \mathcal{B} \tag{2.8}$$

Broadly speaking, this method allows for obtaining a single policy with different behaviours conditioned by the different values of β.

However, for both these formulations, the safety constraint is formulated in terms of expected cumulative cost. As a result, the obtained policy obtains an average safety performance but does not explicitly account for high-cost outcomes in the tail of the cost distribution [9]. Therefore, we argue that characterizing these distinct policies as reflecting different attitudes toward risk would be misleading, since they do not explicitly incorporate risk considerations, in contrast to the methods presented in the following section.

Risk-Aware Optimization

Several scholars have emphasised the importance of training policies with explicit control over the risk associated with the cost distribution, in order to reduce the likelihood of unexpected or unacceptable outcomes. For instance, in [26], the authors motivate their work by noting that standard policy executions may lead to excessively high costs, which can be unacceptable in safety-critical applications

such as military operations. This further supports our view that, in such domains, policies should explicitly account for risk rather than only controlling expected cost.

Consequently, the field of risk-sensitive reinforcement learning extends the optimization objective from expectation-based criteria to a risk-sensitive one.

An example of a risk-sensitive formulation is the one proposed in [26], where safety is considered as the probability that the cumulative cost remains below a predefined threshold, rather than minimizing the expected cost. The problem in this scenario is formulated as

$$\pi^* = \arg\max_{\pi} \Pr\left(C_\pi \leq \theta\right), \tag{2.9}$$

where C_π denotes the cumulative cost induced by policy π, and θ is a user-defined budget threshold. Although this formulation introduces a risk-aware objective by considering the probability of remaining below a cost threshold, it still leads to the computation of a single policy for a fixed risk-bound.

More recent approaches typically define the safety costraints through explicit risk measures. Among these, *Conditional Value-at-Risk* (CVaR) has become a widely used criterion for modeling worst-case outcomes, since it captures the expected cost within the most adverse portion of the cost distribution [27, 28, 29]. A more detailed exposition of CVaR is provided in Section 3.

One of the earliest works to incorporate the CVaR risk measure into the MDP framework is [29]: the idea of combining constrained optimization with tail-risk control leads to *risk-aware SafeRL* formulations, where safety constraints are imposed not on the expected cost, but on a risk measure of the cost distribution. The resulting optimization problem becomes:

$$\max_{\pi} J_R(\pi) \quad \text{s.t.} \quad \mathrm{CVaR}_\alpha(C_\pi) \leq d \tag{2.10}$$

where C_π denotes the cumulative cost distribution under policy π, α defines the tail confidence level, and d is the admissible risk threshold.

Building on this formulation, methods such as Worst-Case Soft Actor-Critic (WC-SAC) operationalize CVaR-based safety by learning a distributional safety critic and explicitly constraining tail risk during policy optimization [30, 31]. Compared to expectation-based SafeRL methods, these approaches provide stronger robustness against rare high-impact failures.

Nevertheless, the acceptable level of risk remains fixed throughout the task, as the safety threshold is globally defined and independent of the operational context. This assumption becomes restrictive in mission-oriented settings, where the appropriate balance between safety and performance may evolve dynamically as time constraints and mission objectives change.

2.3 Time-Awareness in Reinforcement Learning

Time-awareness in reinforcement learning refers to the ability of an agent to condition its behaviour on the available or remaining time. This is particularly

relevant in finite-horizon tasks, where the same action may have different value depending on how much time is left to complete the objective. [32] is one of the earliest significant works in this area; it extends the famous Q-learning algorithm, called QT-learning, by augmenting the state with the remaining budget, highlighting that ignoring time leads to suboptimal behaviour.

This idea was stressed years later again by the authors of [20], where they argue that although time is naturally accounted for in dynamic programming and optimal control through finite-horizon formulations [17], it has often been underspecified in reinforcement learning benchmarks and algorithms.

In both these works [33] and [20], time was included directly in the agent's input representation. This allows the policy to condition its decisions on the remaining horizon and can prevent the agent from treating states with different temporal urgency as equivalent. Other works extend this idea by learning policies that explicitly depend on the desired horizon. For example, [34] propose horizon-aware policies of the form $\pi(a|s, g, h)$, where the agent learns to reach a goal g from state s within at most h steps. This enables different behaviours depending on the available time, as illustrated in Fig. 2.3.

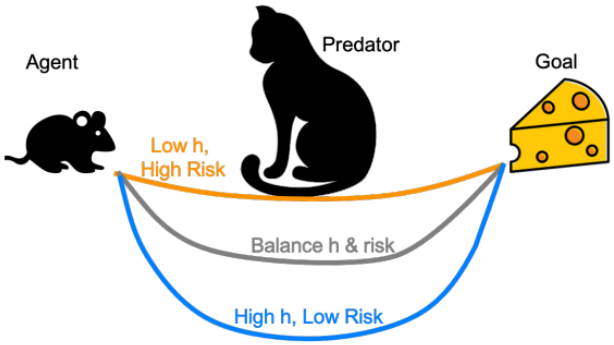


Figure 2.3: Illustration presented by the authors of [34] to show the trade-off between time and risk in navigation. The agent can reach the goal either by taking a shorter, high-risk path close to the predator (orange trajectory) or a longer, low-risk path that avoids it (blue trajectory). The gray trajectory represents an intermediate strategy that balances time and risk.

Related approaches also consider temporal information at the algorithmic or policy-selection level. For instance, in [35], the authors incorporate information about the optimization horizon into meta-learned RL algorithms, while in [36], they introduce Time-Adaptive Markov Decision Processes, where different stationary policies are learned for different temporal regimes and selected at runtime according to the current time budget. Similarly, [37] is an example in which deployed policies must adapt to dynamically changing budget constraints without being retrained from scratch. A recent contribution toward time-aware reinforcement learning is presented in [38], where the authors adopt an imitation learning approach to incorporate temporal awareness into a punctual robotic system, allowing for managing the robot speed and precision through an external input.

Beyond the reinforcement learning literature, the interaction between risk-taking and time pressure has also been studied in economic decision-making, where this

interconnection is extremely relevant. The authors of [39] investigate a finite-horizon problem as a multi-armed bandit problem in which an agent must decide whether to continue pursuing a risky but potentially fast solution (action *doing*) or invest effort in exploring less risky alternatives under a deadline (action *thinking*). Their analysis shows that increasing time pressure can rationally induce more aggressive decisions as the deadline approaches. While their work does not consider sequential decision-making or reinforcement learning, it highlights the fundamental trade-off between risk and remaining time that also motivates our problem setting.

Overall, these works show that time can substantially affect optimal behaviour and that policies should not ignore the remaining horizon. However, most time-aware RL methods focus primarily on adapting task performance to the available time. They do not explicitly address how time changes the acceptable level of risk, nor do they enforce risk-aware constraints only in the regimes where such constraints are operationally meaningful.

2.4 Research Gap

As discussed in the problem context, in mission-critical settings the acceptability of risk is not static, but depends on the operational context, including remaining time, mission objectives, and recovery opportunities. A strategy that is appropriate early in a mission may become suboptimal or even detrimental as temporal constraints tighten and mission failure becomes the dominant risk.

Table 2.1 summarizes the limitations of the most relevant reinforcement learning paradigms. Standard RL methods provide adaptive behaviour, but safety and risk are handled only implicitly through reward design, offering no explicit control over risk exposure. Risk-aware methods address this limitation by introducing explicit safety or tail-risk constraints, but typically assume a fixed risk attitude enforced uniformly throughout the task.

Table 2.1: Comparison of RL approaches for performance–safety optimization.

Aspect	**Performance**	**Performance–Safety Trade-off**	**Safety**
Class of methods	Standard RL	Standard RL	Safe RL
Objective focus	Maximize performance only	Optimize a fixed performance–safety trade-off	Maximize performance under an explicit safety constraint
Risk representation	Not explicitly constrained	Implicit through the penalty term	Explicit through the safety constraint
Risk attitude	Risk-seeking	Risk-neutral	Risk-aware
Safety constraint	×	×	✓
Dynamic behaviour	×	✓	×
Optimization problem	$\max_{\pi} \mathbb{E}[R_\pi]$	$\max_{\pi} \mathbb{E}[R_\pi] - \lambda \mathbb{E}[C_\pi]$	$\max_{\pi} \mathbb{E}[R_\pi]$ s.t. $\mathrm{CVaR}_\alpha(C_\pi) \leq d$
Safety guarantee	×	×	✓
Main limitation	Static risk attitude; ignores safety	No guarantee on safety satisfaction	Static risk attitude; ignores time pressure

Time-aware reinforcement learning addresses the importance of temporal context

by conditioning behaviour on the remaining horizon or selecting policies optimized for different time budgets. However, these methods primarily adapt performance-oriented behaviour and generally do not reason explicitly about changing risk tolerance.

Budget-conditioned methods are perhaps the closest line of work to our research question, as they enable a single policy to exhibit different behaviours by conditioning its decisions on a safety budget. Different safety budget values correspond to different behaviours. However, to the best of our knowledge, existing budget-conditioned approaches are formulated in terms of expected cumulative costs and therefore regulate safety through the average cost incurred by the policy. For example, the framework of [25] learns a policy conditioned on a budget parameter that constrains the expected cost. As a result, it controls average cost but does not explicitly account for the tail of the cost distribution, unlike risk-aware methods.

The main difference between their work and our contribution is that their framework treats safety in expectation, whereas our work models safety through an explicit risk measure. This distinction is important because we argue that describing a policy in terms of risk attitude is not fully justified unless the policy is trained to satisfy a risk-sensitive constraint. Moreover, our work emphasizes temporal pressure as a key driver of behavioural adaptation, while their work does not explicitly consider time-dependent settings to motivate the need for different behaviours.

The gap in the literature is therefore the limited study of risk-attitude adaptation under temporal pressure. While existing work addresses risk-aware decision-making with single deterministic policies or budget-conditioned policies with safety considered in expectation, in this work, we study how to obtain an agent whose risk attitudes must evolve in response to temporal pressure.

A limitation of the present work is that we do not provide an empirical comparison against budget-conditioned approaches such as [25]. Such a comparison would require a non-trivial adaptation of the original Budgeted Fitted-Q framework to our setting. In particular, the scalable implementation proposed in [25] relies on a finite action space, whereas our environment involves continuous control actions. Therefore, applying this baseline would require either discretising the action space or developing a continuous-action variant of the budgeted Bellman update, which is beyond the scope of this work.

Chapter 3

Problem Formulation

We formalize the problem as a mission-level decision-making task, where the agent must balance competing objectives related to task success, time constraints, and a safety requirement.

The objective is to design an agent that solves a goal-reaching task under a variable mission time, while satisfying an explicit safety requirement when sufficient time is available and maximising the performance over the overall mission-time distribution.

Since the available mission time varies across episodes, the agent cannot rely on a single fixed risk profile. When sufficient time is available, it should behave conservatively to satisfy the safety requirement; under more restrictive time budgets, it may need to adopt riskier actions in order to preserve task success.

We formulate the problem within a CMDP framework [28]. Within this framework, we introduce a high-budget regime, defined as the set of episodes in which sufficient time budget remains, and impose the safety requirement only in this regime. The environment is defined as

$$\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, r, c_{hazard}, \gamma), \tag{3.1}$$

where:

- $\mathcal{S}$ denotes the state space, i.e., the set of all possible states of the environment available to the agent;
- $\mathcal{A}$ denotes the action space, i.e., the set of actions that the agent can select;
- $P(s' \mid s, a)$ is the transition kernel, which defines the probability of reaching state $s' \in \mathcal{S}$ after taking action $a \in \mathcal{A}$ in state $s \in \mathcal{S}$;
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is the reward function, which quantifies task performance by assigning a scalar reward to each state–action pair;
- $c_{hazard} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}_{\geq 0}$ is the hazard cost function, which assigns a non-negative penalty to state–action pairs associated with unsafe behaviour or exposure to hazards;
- $\gamma \in (0, 1]$ is the discount factor, which determines the relative importance of immediate and future rewards and costs.

3.1 Time-Budget Augmented State Space

The proposed problem is formulated as a finite-horizon sequential decision-making task in which the agent must complete the mission within a limited amount of available time. Unlike standard episodic reinforcement learning, where the horizon is fixed across all trajectories, we consider missions with variable temporal urgency.

To model this variability, each episode is assigned an initial time budget sampled from a distribution over admissible mission durations:

$$b_{mission} \sim p_B, \quad p_B : [B_{\min}, B_{\max}] \tag{3.2}$$

where $b_{mission}$ denotes the initial time budget assigned to the episode, expressed in discrete decision steps. Therefore, $b_{mission}$ represents the maximum number of actions the agent may execute before the mission deadline is reached.

This formulation introduces a critical modeling issue. If the agent observes only the original environment state s_t, then two identical physical states reached under different temporal conditions become indistinguishable, even though the optimal action may differ substantially depending on the available time. For example, a state encountered early in a mission may allow conservative navigation, whereas the same state near the deadline may require a more aggressive action to preserve mission success.

As a consequence, the original state s_t does not contain all information required for optimal decision-making. To address this issue, we augment the observation with explicit temporal features that encode both the remaining urgency within the current episode and the overall time budget assigned to the mission.

As discussed in [22], an environment satisfies the Markov property if:

$$P(s_{t+1} \mid s_{0:t}, a_{0:t}) = P(s_{t+1} \mid s_t, a_t) \tag{3.3}$$

that is, if the current state contains all information necessary for optimal decision-making.

Following the time-aware formulation proposed in [20], we restore the Markov property by augmenting the observation space with explicit temporal information. Formally, the augmented state space becomes:

$$\tilde{\mathcal{S}} = \mathcal{S} \times \mathcal{T} \tag{3.4}$$

where $\mathcal{S}$ denotes the original environment state space and $\mathcal{T}$ represents the temporal features associated with the mission budget.

Rather than providing the raw remaining budget directly, we introduce two normalized temporal features for each timestep t:

- ◇ the normalized remaining time:

$$b_{\text{left}} = 2\left(\frac{b_{mission} - t}{b_{mission}}\right) - 1 \in [-1, 1] \tag{3.5}$$

which encodes the fraction of the current mission budget that remains available. A value of 1 indicates the beginning of the episode, while -1 corresponds to deadline exhaustion.

- $\diamond$ the normalized mission budget:

$$b_{\text{norm}} = \frac{b_{mission}}{B_{\max}} \in [0, 1] \tag{3.6}$$

which provides information about the overall temporal regime of the mission.

The first feature captures the urgency within the current episode, while the second allows the agent to distinguish between missions with intrinsically different initial time budgets.

The augmented observation at timestep t is therefore:

$$\tilde{s}_t = (s_t, b_{\text{left}}, b_{\text{norm}}) \tag{3.7}$$

where s_t denotes the original environment observation.

This construction ensures that the agent can condition its behaviour both on the immediate temporal urgency and on the broader mission context, while preserving the Markov property required by the underlying MDP/CMDP formulation.

Following the standard RL framework, we can define a trajectory ω generated by a policy π as the sequence of states and actions:

$$\omega = (\tilde{s}_0, a_0, \tilde{s}_1, a_1, \dots), \tag{3.8}$$

where actions $a_0, a_1, ...$ are sampled according to a policy π:

$$a_t \sim \pi(\cdot \mid \tilde{s}_t). \tag{3.9}$$

For notational simplicity, in the remainder of this section, we denote the augmented state $\tilde{s}_t$ simply by s_t.

3.2 Success and Deadline Failure Events

The introduction of a time budget induces the following termination conditions, an episode terminates either:

- $\diamond$ when the agent reaches the goal region $\mathcal{G}$, or
- $\diamond$ when the time left b_{left} is exhausted.

Let the goal region $\mathcal{G}$ be a subset of the original environment state space:

$$\mathcal{G} \subset S \tag{3.10}$$

An episode is considered successful if the state enters $\mathcal{G}$, i.e., if there exists a time step $t \geq 0$ such that $s_t \in \mathcal{G}$.

To capture deadline violations, we introduce the set of terminal failure states corresponding to trajectories that exhaust the time budget before reaching the goal:

$$\mathcal{F} := \{(s, 0) \in \tilde{S} : s \notin \mathcal{G}\} \tag{3.11}$$

We define the deadline-violation indicator as

$$I_t^{\text{deadline}} := \mathbf{1}\{s_t \in \mathcal{F}\} \tag{3.12}$$

which takes the value 1 if and only if the next state belongs to the terminal failure set $\mathcal{F}$, and 0 otherwise.

3.3 Cumulative Reward, Safety Cost, and Success Probability

The reward function is designed to capture three components: (i) dense feedback from the environment, (ii) a reward for successful goal completion, and (iii) a penalty for failing to reach the goal before the available time budget is exhausted.

Formally, we define the reward as

$$r(s_t, a_t) = r_t = r_t^{\text{dense}} + r_t^{\text{goal}} - I_t^{\text{deadline}} \cdot r_{\text{deadline}}. \tag{3.13}$$

The term r_t^{dense} provides continuous feedback from the environment, encouraging progress toward the goal. The term r_t^{goal} represents a terminal reward granted upon successful completion of the task. The last term penalizes trajectories that terminate due to deadline violation, as captured by the indicator I_t^{deadline}.

The deadline penalty is necessary to encode the trade-off between completing the mission on time and avoiding exposure to the hazard. In this way, the agent is encouraged not only to reach the goal safely, but also to account for the operational cost of delaying the mission beyond the available time budget. In this way, the reward definition is consistent with the formal notions of success and deadline failure introduced above.

Following the formulation introduced in the previous Section 2.2, the expected cumulative reward of a policy π is defined as:

$$J_R(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t r_{t+1} \right] \tag{3.14}$$

Similarly, the expected cumulative cost of a policy π is:

$$J_C(\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{T} \gamma^t \, c_{t+1} \right] \tag{3.15}$$

Given a policy π, our objective is to maximize J_R, which captures the expected performance (successfully reaching the goal), while simultaneously minimizing J_C, which reflects the expected cost of the policy (hazard exposure).

3.4 Low and High-Budget Regimes

Imposing a global safety constraint over the entire time-budget distribution would be inconsistent with the objective of the problem, since it would enforce a uniform risk profile across all mission conditions. Instead, the formulation is designed to allow context-dependent behaviour across different time-budget regimes.

Our idea is to divide the ordered distribution of time budgets into two regimes: low-budget and high-budget.

This is justified by the fact that in tasks where we have plenty of time, when we are in the high-budget regime, the agent must satisfy a safety requirement while reaching the goal. Specifically, in the high-budget regime, risk exposure is not treated as an acceptable consequence of time pressure, but as a quantity that must remain below a prescribed risk bound.

On the other hand, when the available time for the task falls within the low-budget regime, the priority is to meet the goal rather than satisfy any safety requirements.

To separate these two regimes, we introduce a parameter based on the ordered discrete time-budget distribution p_B defined in Eq. 3.2. Specifically, let q_β denote the quantile of order β. This threshold partitions the distribution into two regimes: a low-budget regime, containing all time budgets strictly less than q_β, and a high-budget regime, containing all time budgets greater than or equal to q_β.

The idea is that the parameter β controls the position of the threshold within the ordered time-budget distribution. Smaller values of β place the threshold at lower time budgets, thereby increasing the size of the high-budget regime and enforcing the risk bound over a larger portion of the distribution. Conversely, larger values of β move the threshold toward higher time budgets, restricting the high-budget regime to cases with more available time.

We define the set of high-time budgets representing the high-budget regime as

$$\mathcal{B}_{\text{high}}(\beta) = \{b_{mission} \sim p_B \ : \ b_{mission} \geq q_\beta\}\,, \tag{3.16}$$

where q_β denotes the quantile of order β of the time-budget distribution. Basically, this set contains all time budgets that exceed the threshold q_β.

For example, if we set $\beta = 0.75$, $\mathcal{B}_{\text{high}}(0.75)$ will be defined as the upper 25% of the time-budget distribution. Only for these time budgets will the safety constraint be required since the time available is enough to prioritise safety.

3.5 CVaR-Based Risk Bound in the High-Budget Regime

Now we explain how and why we set the risk constraint for the high-budget regime.

Rather than constraining the expected cumulative costs, we adopt a coherent risk measure, namely the Conditional Value-at-Risk (CVaR). The expected cost can mask rare but severe safety violations by averaging them out, as illustrated in Fig. 3.1. In contrast, CVaR focuses explicitly on the tail of the cost distribution and quantifies the expected cost in the worst-case trajectories [27, 30].

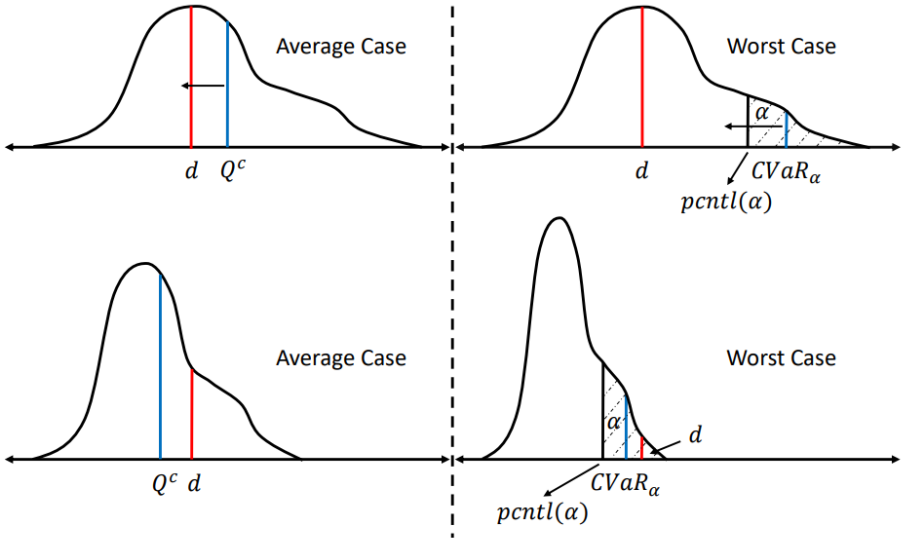


Figure 3.1: Comparison, presented in [30], between average-case (left) and worst-case safety evaluation (right) in constrained reinforcement learning. The top row represents the initial policy cost distributions, while the bottom row shows the reshaped distributions after training. The x-axis represents cumulative cost, while the y-axis represents probability density. The left column illustrates methods based on expected cost minimization, where the policy is optimized such that the expected cumulative cost Q^c (blue line) remains below the safety threshold d (red line). The right column illustrates worst-case risk-sensitive methods, where safety is evaluated using the Conditional Value-at-Risk (CVaR_α) (blue line), which captures the expected cost in the upper tail of the distribution beyond the α-quantile (Value-at-Risk). In the average-case setting, optimization shifts the mean cost below the constraint but leaving significant tail risk. In contrast, worst-case optimization reduces extreme high-cost outcomes and satisfies the risk constraint even in the tail of the cost distribution.

Our choice is consistent with the notion of military effectiveness proposed in [4], where compliance with risk objectives is treated as a key mission requirement alongside outcome, schedule, and cost objectives. In this context, risk reflects the extent to which undesired effects are kept within acceptable operational limits.

Fig. 3.1 illustrates the difference between average-based and risk-based methods in how they shape the cost distribution, and visually highlights why risk-aware methods are necessary.

We follow the formulation introduced in [30]. The only difference is that, instead of considering the cost distribution over all trajectories, we restrict the analysis to the subset of episodes whose initial time budget belongs to the high-budget

regime. Let

$$C_\pi^{\text{high}}(\tilde{s}, a) = \sum_{t=0}^{T} \gamma^t c(\tilde{s}_t, a_t) \;\Bigg|\; \tilde{s}_0 = \tilde{s},\ a_0 = a, \pi \tag{3.17}$$

denote the random cumulative cost under policy π, where $\tilde{s}$ is restricted to initial augmented states, defined in Eq.3.7, corresponding to episodes with $b_{mission} \in \mathcal{B}_{\text{high}}(\beta)$.

Then, as done in [30], we denote by $F_{C_\pi}^{\text{high}}$ the cumulative distribution function (CDF) of the probability distribution of the random variable $C_\pi^{\text{high}}(\tilde{s}, a)$, namely the cost distribution induced by π in the high-budget regime.

Then, we introduce the term $\alpha \in (0, 1]$, which denotes the risk level and determines the fraction of worst-case trajectories considered in the safety evaluation. Smaller values of α correspond to a more conservative risk assessment, since the focus is restricted to increasingly extreme tail events.

The term α is a parameter of our problem, and for the given risk level α, the corresponding Value-at-Risk is defined as

$$\text{VaR}_{1-\alpha}^{\text{high}} := \left(F_{C_\pi}^{\text{high}}\right)^{-1}(1-\alpha) \tag{3.18}$$

where $\left(F_{C_\pi}^{\text{high}}\right)^{-1}$ denotes the inverse cumulative distribution function of the cost distribution. Therefore, $\text{VaR}_{1-\alpha}^{\text{high}}$ is the specific cumulative cost exceeded only by the worst α-fraction of the cost distribution. Broadly speaking, only α-fraction of the episodes performed in the high-budget regime have higher cost than $\text{VaR}_{1-\alpha}^{\text{high}}$.

Safety is then evaluated through the Conditional Value-at-Risk:

$$\text{CVaR}_{\alpha}^{\text{high}} := \mathbb{E}\Big[C_\pi^{\text{high}} \;\Big|\; C_\pi^{\text{high}} \geq \text{VaR}_{1-\alpha}^{\text{high}}\Big], \tag{3.19}$$

which measures the expected cost in the upper tail of the cost distribution beyond the α-quantile. For a more detailed discussion of the CVaR-based constraint formulation, we refer the reader to [30].

The safety requirement in the high-budget regime is therefore expressed through the risk bound

$$\text{CVaR}_{\alpha}^{\text{high}}(\pi) \leq d_{\text{risk}} \tag{3.20}$$

where d_{risk} denotes the maximum acceptable expected cumulative cost in the tail of the cost distribution, defining a risk-bound on the hazard exposure.

3.6 Optimization Problem

We can now define the following constrained optimization problem over the set of admissible policies Π:

$$\begin{aligned} \max_{\pi \in \Pi} \quad & J_R(\pi) - \lambda J_C(\pi) \\ \text{s.t.} \quad & \text{CVaR}_{\alpha}^{\text{high}}(\pi) \leq d_{\text{risk}} \end{aligned} \tag{3.21}$$

The objective function preserves a global penalty on safety-related costs to discourage hazardous behavior across all episodes, whereas the explicit safety constraint is enforced only on episodes endowed with sufficiently large time budgets, for which risk-aware behavior is expected to remain feasible.

In this way, the formulation captures the desired behavior of the agent and may be interpreted as implicitly maximizing mission utility, where military effectiveness results from the alignment of three competing objectives: mission success, compliance with the risk objective, and adaptation to the available schedule. Rather than encoding a fixed trade-off among these factors, the formulation induces context-dependent behavior, whereby their relative importance is determined by the mission conditions.

Chapter 4

Proposed Method

This chapter presents the proposed policy-switching framework, which provides a structured policy approach for the constrained decision-making problem formulated in Chapter 3. Rather than learning a single policy, the proposed method adapts the agent's behaviour to the task context by using a meta-controller to select between two pre-trained policies with complementary risk profiles. Specifically, the framework combines a conservative policy, designed to limit hazard exposure, with an aggressive policy, designed to preserve task completion under restrictive time budgets.

4.1 Motivation and Design Rationale

The proposed approach is motivated by the limitations of both standard reinforcement learning methods and existing risk-aware reinforcement learning approaches in addressing our problem.

Standard methods optimize a single objective function and therefore tend to learn a compromise policy that reflects average behavior across different time–risk regimes. This is not well-suited to our formulation, where a specific risk attitude is required in the high-budget regime.

Conversely, risk-sensitive methods in the literature are typically designed for settings in which safety requirements are imposed globally and under stationary conditions [30, 28].

As for the standard methods, this formulation is also not suitable for our problem, since in our setting, safety cannot be imposed uniformly across all episodes, because the feasibility of risk-aware behaviour depends on the available time budget. When sufficient time is available, the agent is expected to behave conservatively and satisfy the safety requirement. Under tighter time budgets, however, accepting a higher level of risk may be necessary to maximize performance, and therefore the safety requirement can no longer be strictly enforced. Therefore, imposing a single global safety constraint would be overly restrictive, as it would penalise behaviours essential in low-budget settings.

Our method is inspired by policy-switching approaches such as [40, 41] and by the hierarchical separation of decision-making introduced in hierarchical reinforcement learning [42, 43]. Both perspectives are motivated by settings in which a

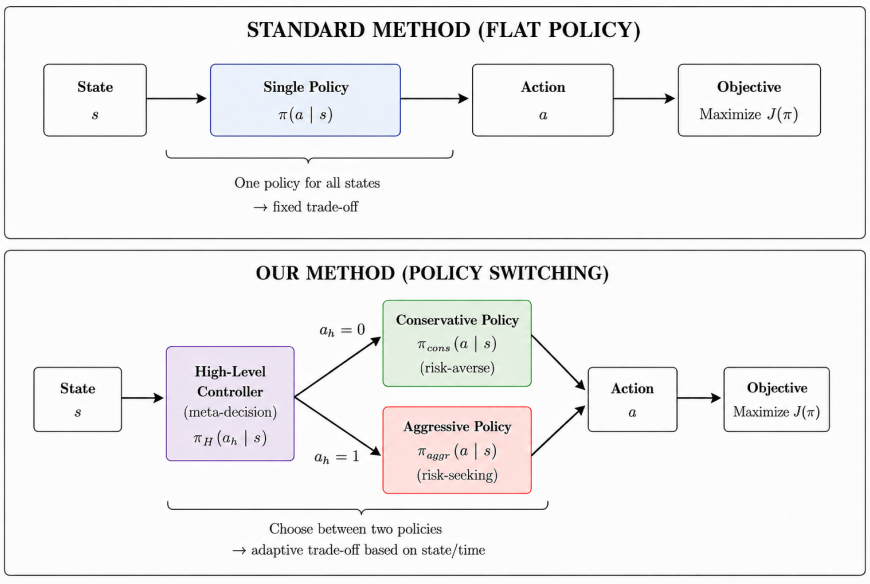


Figure 4.1: Comparison between a standard reinforcement learning approach and the proposed policy-switching approach. Top: a single policy optimizes a fixed trade-off between performance and risk across all states. Bottom: a hierarchical controller selects between a risk-averse and a risk-seeking policy, enabling an adaptive trade-off with explicit behaviours based on the current state and available time.

single policy may be insufficient to capture qualitatively different behaviors required across different operating conditions. In such cases, multiple specialized policies can represent distinct behaviors more effectively, while a higher-level mechanism can select the most appropriate behavior given the current operation context. Namely, as reported in [40], the authors motivate their policy switching approach by noting that "using only a single driving strategy is difficult to meet the multiple requirements of efficiency and safety in multi-scenario autonomous driving tasks." Similarly, when time and risk are jointly involved, more behaviours are necessary to meet the multiple operations' requirements.

Moreover, one major advantage of a hierarchical approach is its potential to improve the explainability of an agent's decision-making process. As highlighted in [44], "As such, agents become increasingly powerful; they will eventually be deployed in real-world situations where the consequences of their actions carry considerable stakes. A crucial requirement would then be our capacity to ensure their alignment, and interpreting their decisions is a key aspect of this challenge."

This idea directly motivates our approach: different policies are used to handle different risk–time regimes of the same problem. In this work, we refer to the standard single-policy architecture as a flat policy, in line with the distinction commonly made in hierarchical reinforcement learning between flat and hierarchical decision-making structures [42]. The term emphasizes that the policy directly maps states to actions without an intermediate high-level controller or decomposition into lower-level behaviours. As illustrated in Fig. 4.1, a flat approach encodes the performance–risk trade-off within a single policy, whereas the proposed method separates high-level risk-profile selection from low-level action execution.

4.2 Method Overview

Building on this idea, we propose a policy-switching framework that adapts the agent's behaviour to the current mission conditions, enabling it to operate across different risk–time regimes, satisfying the risk bound when required while simultaneously maximizing the overall performance–safety trade-off.

In particular, we assume access to two policies with complementary properties: a risk-aware conservative policy, designed to satisfy the safety objective by limiting exposure to hazards, and a performance oriented aggressive policy, optimized to reach the goal as quickly as possible.

At each decision step, a switching mechanism selects which policy to execute based on the current mission context. This allows the agent to behave conservatively when mission conditions make conservative behaviour feasible, and to adopt a more risk-tolerant strategy under tighter time constraints in order to comply with the expected outcome. As a result, the overall behaviour of the agent adapts dynamically across different time-budget regimes.

4.3 Defining the Risk-Attitudes

The proposed method builds upon two policies that optimise distinct objective functions and aims to enable a single agent to tackle the problem by dynamically switching between these two opposite behaviours. In our framework, the two specialized policies are assumed to be pre-trained in the same environment.

The purpose of these policies is to represent two distinct behaviours required by the problem. The pre-trained aggressive policy captures a behaviour primarily focused on maximizing the utility associated with mission outcome, which can be interpreted conceptually as

$$\pi_{\text{aggressive}} \propto f(U_{\text{outcome}}) \ .$$

Conversely, the pre-trained conservative policy captures a behaviour that accounts for both mission outcome and risk considerations, conceptually:

$$\pi_{\text{conservativee}} \propto f(U_{\text{outcome}}, U_{\text{risk}}) \ .$$

The role of the controller is then to determine when to switch between these two pre-trained policies according to the current mission context. In this sense, the resulting switching policy can be interpreted as approximating a context-dependent behaviour that accounts for mission outcome, risk, and schedule considerations:

$$\pi \propto f(U_{\text{outcome}}, U_{\text{risk}}, U_{\text{schedule}}) \ .$$

Under this interpretation, choosing one policy over the other corresponds to selecting the utility profile that is most appropriate for the current mission state, with the overall goal of maximizing military effectiveness.

To remain consistent with the proposed formulation and enable a fair comparison with the flat baseline, both approaches optimize the same utility function. However, in the proposed framework, additional control is introduced over the deployed actions through the policy-switching mechanism.

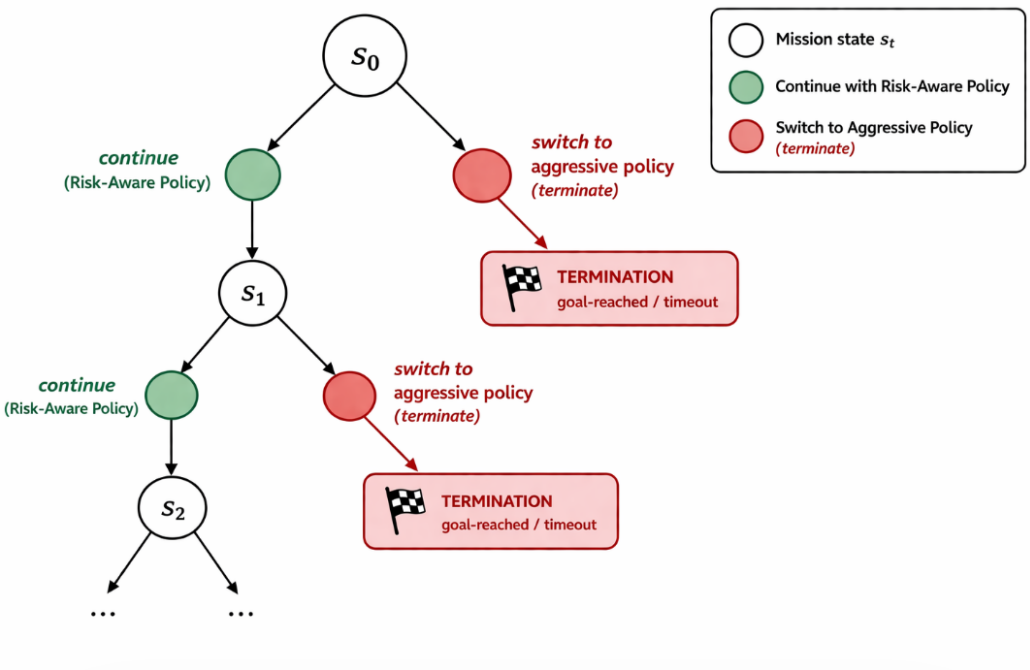


Figure 4.2: Optimal-stopping interpretation of the switching decision. At each mission state $s_0, s_1, ...$, the high-level controller chooses between two meta-actions: *continue*, which keeps the conservative policy active and postpones the switching decision, and *switch*, which terminates the controller's decision process and transfers control to the aggressive policy for the remainder of the episode. The goal is to select the switching time that maximizes the expected utility.

4.4 Optimal Stopping Formulation

Inspired by the optimal stopping formulation in [45], we model the switching decision as an optimal stopping problem, where the objective is to determine the most appropriate time to stop one decision process to maximize the expected utility.

At each time step, the agent must choose between two possible actions:

- ◇ *continue*, which corresponds to continuing the decision process under the conservative policy;
- ◇ *switch*, which corresponds to stopping the current decision process and handing over control to the aggressive policy.

As shown in Algorithm 1, the two actions have fundamentally different consequences. Choosing *continue* does not terminate the decision problem: the agent remains under the conservative policy, the environment evolves to a new state, and a new switching decision is taken at the following time step. Choosing *switch*, instead, terminates the switching process: from that point onward, no further switching decisions are made, and the mission continues under the aggressive policy until termination.

For this reason, the switching mechanism can be naturally interpreted as an

optimal stopping problem: as shown in Fig. 4.2, the agent sequentially observes the mission state and decides whether to continue with the current conservative behaviour or to stop this decision process and commit to the aggressive policy.

As introduced in [45], an optimal switching problem with two available actions can be formulated as a problem of learning the action-value functions associated with the two decisions. In our setting, these correspond to the *switch* and *continue* actions.

Q-values are necessary cause they provide an estimate of how beneficial it is to take a particular action in a given state, taking into account not only the immediate outcome but also its long-term consequences with respect to the optimization objective. Basically, we want to learn this function to decide when to *switch* is more beneficial than to *continue*.

Following the Q-learning routine introduced in [33], the corresponding Q-values can be written as

$$Q(s, a_{continue}) = \begin{cases} r(s, a_{continue}), & \text{if at final time-step } T, \\ r(s, a_{continue}) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'), & \text{otherwise,} \end{cases} \tag{4.1}$$

and

$$Q(s, a_{switch}) = r(s, a_{switch}), \tag{4.2}$$

where a_{continue} denotes the continue action, a_{switch} denotes the switch action, γ the discount factor of the problem and A the set of actions available, in our case $a_{continue}$ and a_{switch}. The term $r(s, a_{switch})$ denotes the final reward obtained after choosing the switch action, while $r(s, a_{continue})$ denotes the reward obtained by following the continue action for one decision step. The term $\max_{a' \in \mathcal{A}} \hat{Q}(s', a')$ denotes the maximum state action value achievable from the next state s'.

A practical explanation of how the Q-values are computed in our setting is provided in the following section.

The switching decision is obtained by comparing the action-value estimates (Q-values) of the two available actions, *switch* and *continue*. These Q-values represent the expected utility of two alternative and opposing future strategies. More specifically, $Q(s, a_{continue})$ denotes the expected utility of continuing the mission under the conservative policy, while $Q(s, a_{switch})$ denotes the expected utility of handing over control to the aggressive policy from the current state onward.

This comparison partitions the state space into two decision regions: one in which switching is preferred and one in which continuing is preferred. The decision boundary corresponds to the set of states where the two actions have equal value. States close to this boundary are characterized by a small difference between the two Q-values, meaning that neither action clearly dominates the other [45].

Formally, the switching region is defined as

$$Q(s, a_{switch}) > Q(s, a_{continue}), \tag{4.3}$$

while the decision boundary is given by

$$Q(s, a_{switch}) = Q(s, a_{continue}). \tag{4.4}$$

The proposed method implements a classifier that acts as meta-controller and has to predict if the input state s belongs to the switching region or not.

Algorithm 1 Policy Switching Execution

```
Initialize state s_0, τ decision threshold
t ← 0
switched ← False
while episode not terminated do
    if not switched then
        Compute p = Classifier(s_t)
        if p > τ then
            switched ← True
        end if
    end if
    if switched then
        a_t ~ π_aggressive(· | s_t)
    else
        a_t ~ π_conservative(· | s_t)
    end if
    Execute a_t and observe s_{t+1}
    t ← t + 1
end while
```

Specifically, for a given state s, we define the utility gap of that state as

$$\Delta(s) = Q(s, a_{\textit{switch}}) - Q(s, a_{\textit{continue}}). \tag{4.5}$$

The sampled state s is then assigned a binary label according to

$$y(s) = \begin{cases} 1, & \text{if } \Delta(s) > 0 \quad \text{(switch-win)}, \\ 0, & \text{otherwise} \quad \text{(continue-win)}. \end{cases} \tag{4.6}$$

This pipeline effectively transforms the switching problem from value comparison into a supervised classification problem. Indeed, the classifier has to predict the value of $y(s)$ given the current state:

$$\hat{y}(s) = f_\theta(x(s)) \approx y(s). \tag{4.7}$$

Broadly speaking, if the classifier predicts 1, the input state is classified as belonging to the switching region, and so the control is passed to the aggressive policy.

Lastly, we introduce the notation k_{best}. Given a sequence of decision steps $k_1, k_2, \ldots, k_{\text{best}}$ denotes the earliest decision step at which the expected utility of the *switch* action exceeds that of the *continue* action. Ideally, the classifier should always identify this decision step as switching time.

In conclusion, the classifier is the only learned component of the framework. Rather than estimating the Q-values of the available actions directly, it learns the sign of their difference, indicating whether switching provides higher utility than continuing. The final goal is to enable the switching decision only when necessary at runtime to preserve the expected outcome (reaching the goal).

Chapter 5

Implementation

In this section, we present the implementation details of the experimental setup, we present the baselines found in the literature to tackle our problem, the training settings used and the specific implementation pipeline of our method.

5.1 Experimental Setup

All experiments are conducted in the Safety Gym benchmark by OpenAI [11], a widely adopted environment for safe reinforcement learning due to its standardized safety costs, lightweight implementation, and reproducibility. Its design makes it particularly suitable for studying the trade-off between task performance and safety under controlled conditions.

For this work, we adopt the `StaticEnv` configuration introduced in [30], illustrated in Fig. 5.1. The task consists of a 2D navigation problem in which a Point robot must reach a fixed goal within a limited mission time while minimizing exposure to hazardous regions whenever operationally feasible.

The goal and hazard locations remain fixed across episodes at $(1.1, 1.1)$ and $(0, 0)$, respectively, while the agent's initial position is randomly sampled at the beginning of each episode. The goal radius is set to 0.3, while the hazard radius is 0.7. The environment consists of a square arena with placement extents $[-1.5, -1.5, 1.5, 1.5]$.

The agent is controlled through two continuous actions in the interval $[-1, 1]$: one controlling angular velocity and one controlling forward/backward motion.

Safety is encoded through the per-step cost function:

$$c_t = \begin{cases} 1 & \text{if the agent is inside the hazard region at time-step } t, \\ 0 & \text{otherwise.} \end{cases} \tag{5.1}$$

where a cost of 1 therefore represents a safety violation.

The reward function follows the original Safety Gym formulation:

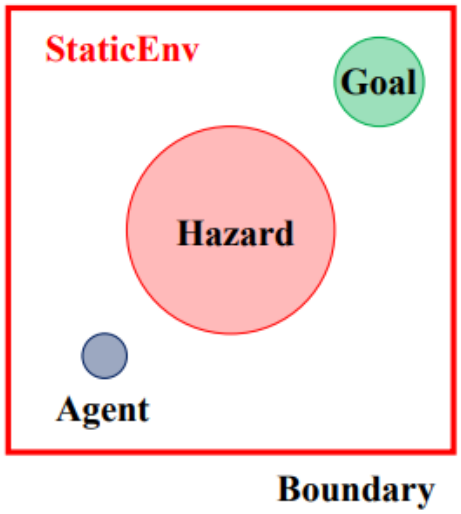


Figure 5.1: Safety Gym `StaticEnv` environment used in the experiments [30]. The hazard and goal remain fixed, while the agent's initial position is randomly sampled at the beginning of each episode.

$$r_t = r_t^{\text{dense}} + r_t^{\text{goal}}, \tag{5.2}$$

where r_t^{dense} encourages progress toward the goal and r_t^{goal} provides a sparse terminal reward upon successful completion.

The observation space consists of LiDAR-based proximity sensing and proprioceptive measurements. Specifically, the Point agent receives 16 LiDAR bins for hazard detection and 16 for goal detection, uniformly covering the full 360° field of view. Each bin reports the normalized distance to the closest detected object in that direction, or zero if no object is present [11]. Additional proprioceptive observations include accelerometer, velocimeter, gyroscope, and magnetometer readings, for a total of 12 dimensions.

To instantiate the time-aware problem formulation introduced in Section 3, each episode is assigned a mission time budget B sampled from a discrete uniform distribution over the interval $[B_{\min}, B_{\max}]$, with step size 5. This models missions with varying temporal urgency.

The upper quantile of the budget distribution defines the safe-feasible regime, where the task remains achievable while satisfying the safety constraint, whereas lower budgets correspond to operational regimes in which stricter trade-offs between safety and mission success emerge.

In our experiments, we set $B_{\min} = 120$ and $B_{\max} = 220$ time steps.

Regarding the risk-aware safety formulation, the CVaR constraint parameters are set to $d = 5$ and $\alpha = 0.1$. The value of d defines a meaningful but non-trivial safety requirement: sufficiently restrictive to penalize unsafe behaviour while still preserving feasibility. The choice $\alpha = 0.1$ focuses risk control on the worst 10% of episodes, enforcing conservative behaviour in scenarios where safety is intended to dominate.

Table 5.1: Environment configuration used in all experiments.

Parameter	**Value**
Robot base	Point robot
Arena extents	$[-1.5, -1.5, 1.5, 1.5]$
Goal size	0.3
Goal keepout	0.305
Goal location	$(1.1, 1.1)$
Hazard constraint	✓
Number of hazards	1
Hazard size	0.7
Hazard keepout	0.705
Hazard location	$(0, 0)$
Goal lidar observation	✓
Hazard lidar observation	✓
Lidar max distance	3
Lidar bins	16

Finally, the utility trade-off parameter is fixed at $\lambda = 0.02$. This parameter determines the relative weighting between performance and safety penalties. Its value is selected based on the observed scale of the signals: episode rewards are bounded around 1, whereas cumulative hazard costs can reach values close to 50 under aggressive behaviour. This choice preserves successful mission completion as the primary objective while ensuring that unsafe behaviour remains meaningfully penalized.

Unless otherwise specified, all remaining environment hyperparameters follow the original `StaticEnv` configuration reported in Table 5.1.

5.2 Metrics

In this section, we define the metrics used to assess the proposed framework. The selected metrics are designed to reflect the mission-level objectives introduced in Section 3, capturing performance in terms of task completion, safety, and adherence to time constraints.

We consider two complementary aspects. First, we assess the quality of the switching mechanism by analyzing the performance of the learned classifier that governs policy transitions. Second, we evaluate the overall behavior of the agents through outcome- and risk-oriented metrics.

5.2.1 Classifier Performance Metrics

We evaluate the policy switching network as a binary classifier that predicts whether a policy switch should occur at a given decision step. More specifically,

given an observation encoding the current mission context, the classifier must determine whether the current state belongs to the *continue-win* region, in which continuing with the current policy is sufficient to preserve mission success, or to the *switch-win* region, in which switching policy is necessary to maximize the expected utility.

This distinction is central to our framework. The classifier does not act directly on the environment, but instead determines which of the two policies should control the agent under the current mission conditions. Its role is therefore to detect those states in which the current policy is no longer adequate and a switch becomes necessary.

It is also important to note that this is an imbalanced classification problem. Most states are labeled as *continue-win*, while only a relatively small subset belongs to the *switch-win* class. For this reason, not all evaluation metrics are equally informative. In particular, we are primarily interested in the ability of the classifier to correctly identify *switch-win* states, since failing to detect them may prevent the agent from switching when required and thus directly compromise mission success. Accordingly, metrics that reflect performance on the minority class are especially relevant for assessing the usefulness of the switching model.

We evaluated the performance of the classifier along two complementary dimensions: standard performance metrics and threshold-based operational analysis.

Classification metrics. To evaluate the predictive quality of the switching classifier, we report standard classification metrics including accuracy, precision, recall, and F1-score. Accuracy provides an overall measure of correct predictions, while precision and recall are particularly important for assessing performance on the *switch-win* class. In our setting, recall is especially relevant, since failing to identify a *switch-win* state means missing a necessary switch and may therefore compromise mission success. The F1-score is reported to summarize the trade-off between precision and recall. All these metrics are computed using a default decision threshold of 0.5, unless otherwise specified.

In addition to these threshold-dependent metrics (accuracy, precision, recall and F1-sccore), we report threshold-independent measures such as ROC-AUC and PR-AUC. ROC-AUC evaluates the classifier's ability to rank positive and negative samples across all possible decision thresholds, whereas PR-AUC is more informative in our imbalanced setting, as it focuses on performance over the minority *switch-win* class.

Since the switching decision is ultimately obtained by thresholding predicted probabilities, it is also important to assess how well these probabilities are calibrated. For this reason, we report binary cross-entropy (BCE) and Brier score. BCE measures the quality of the predicted probabilities from a probabilistic classification perspective, assigning larger penalties to confident but incorrect predictions, while the Brier score evaluates the mean squared error between predicted probabilities and ground-truth labels. Together, these metrics provide an indication of how reliable the classifier's confidence estimates are, which is particularly important when selecting decision thresholds for switching.

Threshold analysis Since the switching decision is inherently temporal, we evaluate the prediction error with respect to the empirical best switching time k_{best} as

$$\delta_k = k_{\text{pred}} - k_{best},$$

which captures whether the model anticipates or delays the switching decision.

This analysis was conducted to validate the classifier prior to deployment and to ensure that its predictions are not biased. This is crucial, as we aim to use the classifier directly with the default decision threshold of 0.5, without relying on additional threshold tuning or post-hoc adjustments.

5.2.2 Agent Performance Metrics

We evaluate the performance of the agents using metrics aligned with the mission-level objectives introduced in Section 3, namely outcome, risk and schedule.

Success rate We measure task performance through the success rate, defined as the probability of reaching the goal within the available time budget. In our setting, this metric jointly captures both the outcome objective (goal achievement) and the schedule objective, since episodes terminate at the sampled time budget.

Risk metrics: Expected cost and CVaR Risk in our problem is defined in terms of exposure to the hazard. To evaluate safety performance, we report both the expected hazard cost and the Conditional Value-at-Risk (CVaR) of the hazard-cost distribution. While the expected cost captures the average level of hazard exposure, CVaR focuses on the upper tail of the distribution, measuring the expected cost under worst-case outcomes.

The expected hazard cost captures the average level of exposure to hazards, while CVaR focuses on the upper tail of the distribution, providing a measure of worst-case risk.

By construction of the problem, feasibility of the risk objective is guaranteed in the high-budget regime, since in that regime the goal can always be reached while maintaining conservative behavior. For this reason, CVaR is evaluated in the high-budget regime while the expected hazard cost is evaluated over all the time budgets distribution.

All metrics are estimated empirically from a set of evaluation episodes generated under the learned policies. Success rate and expected hazard cost are reported as overall performance metrics over the full time-budget distribution. In contrast, following [30], CVaR is computed only on the subset of episodes corresponding to the high-budget quantile defined in the problem formulation, and is evaluated with respect to the safety threshold d introduced in Section 3.

5.3 Baseline Methods

The baseline policies are selected to instantiate the main classes of approaches identified in the research gap analysis in Section 2.4: a SAC policy optimized solely for task performance, a SAC policy optimized for a fixed performance–safety trade-off, and a WCSAC policy optimized for performance under an explicit risk bound.

5.3.1 SAC: Aggressive Baseline

This policy is implemented using the standard Soft Actor-Critic (SAC) algorithm [23], with a reward formulation focused exclusively on mission completion:

$$r_t = r_t^{\text{dense}} + r_t^{\text{goal}} - r_t^{\text{deadline}} \tag{5.3}$$

The resulting optimization objective is:

$$\max_{\pi} J_R(\pi) \tag{5.4}$$

This baseline represents a performance-oriented strategy that prioritizes rapid goal completion without explicit consideration of safety constraints.

5.3.2 SAC: Flat Baseline

To evaluate whether a single stationary policy can solve the proposed problem without explicit policy switching, we introduce a flat baseline based on SAC.

Unlike the aggressive baseline, this policy directly optimizes the scalarized utility objective defined in the problem formulation:

$$r_t = r_t^{\text{dense}} + r_t^{\text{goal}} - r_t^{\text{deadline}} - \lambda c_t^{\text{hazard}} \tag{5.5}$$

leading to the objective:

$$\max_{\pi} \; J_R(\pi) - \lambda J_C(\pi) \tag{5.6}$$

This baseline evaluates whether a single non-hierarchical policy can learn an effective compromise between mission success and safety across all operational regimes. Since the trade-off is fixed by the scalarization parameter λ, the learned behaviour is expected to represent an averaged compromise rather than explicitly adapting its risk attitude to the mission context.

5.3.3 WCSAC: Conservative Baseline

The conservative baseline is implemented using Worst-Case Soft Actor-Critic (WCSAC) [30] and it optimizes the same CVaR-constrained formulation introduced in the problem definition:

$$\max_{\pi} J_R(\pi) \quad \text{s.t.} \quad \text{CVaR}_{\alpha}(\pi) \leq d \tag{5.7}$$

By explicitly constraining the upper tail of the cumulative hazard-cost distribution, this policy represents a strongly risk-averse strategy designed to satisfy the prescribed safety requirement.

However, unlike the proposed adaptive framework, the safety constraint is enforced through a single fixed global threshold. As a result, the policy maintains the same conservative behaviour regardless of the available mission time, even in regimes where increased risk-taking may be operationally justified.

Table 5.2: Problem and reward parameters used in the experiments for each policy.

Parameter	**Aggressive (SAC)**	**Conservative (WCSAC)**	**Flat Policy (SAC)**	**Switching Controller**
Goal reward r_{goal}	1.0	1.0	1.0	1.0
Hazard cost c_t	–	1.0	1.0	1.0
Risk bound d	–	5.0	–	–
Risk level (α)	–	0.1	–	–
Budget min	120	120	120	120
Budget max	220	220	220	220
Deadline penalty	1.0	1.0	1.0	1.0
Hazard weight λ	–	–	0.02	0.02

5.3.4 Training Configuration

This section reports the design parameters and training hyperparameters used for the baseline methods. The corresponding problem-specific parameters are summarized in Table 5.2, while the training hyperparameters are reported in Table 5.3.

All policies are implemented in PyTorch and trained under the same experimental protocol to ensure a fair comparison. Unless otherwise specified, each model is trained for 100 epochs using a single random seed (seed $= 0$), and the final policy used for evaluation corresponds to the last training checkpoint.

During training, actions are sampled from the stochastic policy distribution to preserve exploration, whereas during evaluation actions are selected deterministically by taking the mean of the learned policy distribution.

5.4 Proposed Method

In this section, we describe the implementation of the proposed switching-based approach. The implementation is divided into four main components: (i) the computation of the Q-values, (ii) the construction of the supervised dataset used to learn the switching decision, (ii) the training of the binary classifier, and (iii) the deployment procedure used to integrate the classifier with the pre-trained policies during execution.

5.4.1 Computation of the Q-values

We now describe the procedure used in our framework to estimate the Q-values associated with the respective meta-actions *switch* and *continue* and introduced in Eq. 4.1.

In standard reinforcement learning, the Q-values (also known as state-action values) are defined as expectations over stochastic trajectories and are typically estimated via temporal-difference learning [22].

Table 5.3: Main hyperparameters used for training all baseline policies.

Hyperparameter	**Aggressive (SAC)**	**Conservative (WCSAC)**	**Flat Policy (SAC)**
Algorithm	SAC	WCSAC	SAC
Actor network	MLP $(256, 256)$	MLP $(256, 256)$	MLP $(256, 256)$
Critic network	MLP $(256, 256)$	MLP $(256, 256)$	MLP $(256, 256)$
Variance network	–	MLP $(256, 256)$	–
Batch size	256	256	256
Learning rate	10^{-3}	10^{-3}	10^{-3}
Discount factor γ	0.99	0.99	0.99
Polyak coefficient	0.995	0.995	0.995
Replay buffer size	10^6	10^6	10^6
Reward scale	1	1	1
Training epochs	100	100	100
Steps per epoch	30,000	30,000	30,000
Update frequency	100	100	100
Initial random steps	500	500	500
Updates start after	500	500	500
Seed	0	0	0

In our setting, both the pre-trained policies and the environment dynamics are deterministic: for any given state, each policy always outputs the same action, and each state–action pair always leads to the same next state. Therefore, the trajectory generated from a given state–action pair is unique, the return is non-random, and the action-value function reduces to this return, without requiring an expectation.

As a consequence, the Q-value associated with a given state–action pair can be computed through a single rollout-based evaluation.

For the *switch* meta-action, the value is computed from a single rollout starting from state s and following the aggressive policy from that point onward:

$$Q(s, a_{\text{switch}}) = r(s, a_{switch}) \tag{5.8}$$

For the *continue* meta-action, the value is computed as the immediate utility of continuing plus the discounted continuation value at the next state:

$$Q(s, a_{\text{continue}}) = r(s, a_{\text{continue}}) + \gamma \max_{a' \in \mathcal{A}} \hat{Q}(s', a'), \tag{5.9}$$

where s' denotes the state reached by continuing with the conservative policy. The term $\max_{a' \in \mathcal{A}} \hat{Q}(s', a')$ captures the possibility of switching at a future state. This continuation value is computed through the rollout procedure described below.

Again, we can compute this value trough a rollout since the pre-trained policies are executed deterministically and the environment dynamics are non-stochastic.

The rollout procedure differs depending on whether the episode can be successfully completed under the conservative policy alone, in which case no *switch* action is required, or whether the conservative policy fails to reach the goal, making a *switch* action necessary.

If the episode is *conservative win*, the term $\max_{a' \in \mathcal{A}} \hat{Q}(s', a')$ always corresponds to a complete rollout where at each decision step the *continue* action is selected.

On the other hand, if the episode is *conservative fail*, the term $\max_{a' \in \mathcal{A}} \hat{Q}(s', a')$ corresponds to a rollout in which the *continue* action is selected until the best switching time k_{best} is reached. The underlying idea is that, when the episode cannot be completed by the conservative policy alone, there exists a switching time k_{best} at which the meta-controller should transition to the aggressive policy in order to to maximize the expected utility.

The value of k_{best} is determined through an heuristic search procedure where, to avoid an exhaustive search over all possible switching steps, we use a two-stage coarse-to-fine procedure:

1. The trajectory is partitioned into segments of width `scan_interval`, and the switching return is evaluated at the midpoint of each segment.
2. The top `n_top_zones` segments are retained, and all switching steps within these segments are evaluated to obtain a refined estimate of k_{best}.

Specifically, we set `scan_interval` $= 5$ and `n_top_zones` $= 2$.

Finally, the switching step at which the *switch* action achieves the highest estimated utility is selected as k_{best} for that episode.

5.4.2 Supervised dataset construction

The dataset is constructed from an offline pool of episodes generated by deploying the conservative policy. Episodes are collected until 500 successful and 500 failed episodes are obtained, ensuring a balanced dataset with respect to the outcome under the conservative policy.

Starting from the offline pool of episodes, we construct a supervised dataset in which each data point corresponds to a specific state encountered during the mission. The input x contains the feature representation of that state, while the label y specifies the preferred meta-action: $y = 0$ if *continue* is preferred, and $y = 1$ if *switch* is preferred. Each input vector has dimension 36 and includes two 16-dimensional LiDAR arrays: one encoding distances to hazards and one encoding distances to the goal, the robot velocities along the x- and y-axes, the normalized remaining time, and the normalized time budget.

To construct the dataset, we first sample an episode and then select three observation states, corresponding to three decision steps, from that episode. The procedure used to select these states depends on the episode outcome under the conservative policy, namely whether the mission is successfully completed within the available time budget or not.

If the episode is successfully completed by the conservative policy, the episode is treated as a *conservative win*. In this case, we randomly sample 3 states from the trajectory and label their meta-features according to the rule defined above. Since the conservative policy is already sufficient to complete the mission within

the deadline, switching is not required to improve the expected utility. Therefore, all sampled states from these episodes are assigned the label $y = 0$.

Conversely, when a sampled episode is not successfully completed by the conservative policy, it is treated as a *conservative fail*. In this case, the transition associated with the best time to switch k_{best} is labeled as $y = 1$, since it corresponds to a state where switching provides higher utility than continuing. Additionally, for each failed episode, two extra transitions are randomly sampled and labeled as $y = 0$.

For each conservative-fail episode, only the state corresponding to the utility-maximizing switching time (k_{best}) is labeled as a positive example. Although multiple switching times may still lead to successful mission completion, the objective of the classifier is to identify the switching point that maximizes the estimated utility according to the heuristic search procedure described in the previous subsection. Therefore, (k_{best}) is treated as the unique optimal switching decision for that episode, while all other sampled states are labeled as negative examples.

As a result, the dataset consists of labeled transitions in which positive samples $(y = 1)$ represent states where switching is estimated to be optimal according to the heuristic search described above, whereas negative samples $(y = 0)$ represent states where continuing remains preferable.

To summarize, the supervised dataset used to train the switching classifier is constructed from 1000 episodes generated by the conservative policy, consisting of 500 successful and 500 failed episodes under that policy. From each episode, a subset of states is sampled according to the procedure described above, resulting in a dataset of approximately 3000 samples. The positive samples $(y = 1)$ represent states where switching is estimated to be optimal according to the heuristic search described above, whereas negative samples $(y = 0)$ represent states where continuing remains preferable. The resulting dataset is imbalanced, with a ratio of approximately $5 : 1$ between *negative* and *positive* samples. This is consistent with the fact that the conservative policy is able to complete around 80% of episodes without requiring a switch.

5.4.3 Training of the switching classifier

After constructing the dataset, we train a switching classifier to enable meta-level decision making at runtime.

The classifier is implemented as a lightweight multilayer perceptron (MLP) with a single hidden layer. Received an input feature vector x_t, the network first computes a hidden representation

$$h_t = \mathrm{ReLU}(W_1 x_t + b_1),$$

and then produces the output logit as

$$z_t = W_2 h_t + b_2.$$

The hidden layer contains 32 units, this architecture was intentionally kept small in order to limit overfitting and maintain low computational cost at inference time.

The scalar logit $z_t \in \mathbb{R}$ produced by the classifier is converted into a probability through the sigmoid function:

$$\hat{p}_t = \sigma(z_t) = \frac{1}{1 + e^{-z_t}},$$

where the positive class corresponds to states in which switching is estimated to be preferable.

Broadly speaking, in this classification task the role of the classifier is similar to that of a spam detector: it must identify the relatively rare *switch-win* states among a much larger number of *continue-win* states.

The model is trained using binary cross-entropy loss. Let $y_i \in \{0, 1\}$ denote the ground-truth label of sample i, and let z_i be the corresponding output logit. The training objective is

$$\mathcal{L}_{\mathrm{BCE}} = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log \sigma(z_i) + (1 - y_i) \log(1 - \sigma(z_i)) \right].$$

In practice, this loss is implemented using a numerically stable BCE-with-logits formulation. Optimization is performed using Adam optimizer [46], with a learning rate of 10^{-3} and a mini-batch size of 16. Moreover, to improve training stability, gradient clipping with maximum norm 1.0 is applied.

To improve robustness and encourage better generalization to nearby states, Gaussian noise is injected into the input features during training:

$$\tilde{x} = x + \epsilon, \qquad \epsilon \sim \mathcal{N}(0, \sigma^2 I),$$

with $\sigma = 0.01$. This acts as a simple form of regularization by encouraging the classifier to be less sensitive to small perturbations in the input.

The dataset is randomly shuffled using a fixed seed and split into training, validation, and test sets with proportions 0.8,0.1, and 0.1, respectively. Model selection is based on the validation binary cross-entropy.

The training is run for at most 500 epochs. In practice, the actual number of epochs is often smaller, since early stopping is used to prevent overfitting: the training is monitored on the validation loss, with a patience of 20 epochs, and the model parameters corresponding to the best validation performance are restored at the end of training.

5.4.4 Deployment procedure

At deployment time, the switching classifier is queried at every decision step to determine which of the two pre-trained policies should control the agent. Starting from the initial state, the agent follows the conservative policy as long as switching is not predicted. Once a switch is triggered, control is permanently transferred to the aggressive policy for the remainder of the episode.

More formally, let x_t denote the classifier input at decision step t, and let

$$p_t = \sigma(z_t) \in [0, 1] \tag{5.10}$$

Table 5.4: Training hyperparameters of the switching classifier.

Parameter	Value
Input features	36
Feature layout	[hazards_lidar, goal_lidar, v_x, v_y, time_left_norm, budget_norm]
Model	MLP (1 hidden layer)
Hidden size	32
Trainable parameters	1217
Loss	Binary Cross-Entropy with logits
Optimizer	Adam
Learning rate	10^{-3}
Batch size	16
Max epochs	500
Early stopping patience	20
Feature noise std	0.01
Decision threshold	0.5
Train/Val/Test split	0.8 / 0.1 / 0.1
Base seed	2001

be the predicted probability of the *switch-win* class, obtained by applying the sigmoid function to the classifier logit z_t.

The switching decision is taken at every step by comparing p_t against a fixed threshold $\tau = 0.5$:

$$d_t = \begin{cases} 1, & \text{if } p_t > \tau, \\ 0, & \text{otherwise,} \end{cases} \qquad \tau = 0.5 \tag{5.11}$$

where $d_t = 1$ denotes a switch decision.

Let $m_t \in \{0, 1\}$ denote the active control mode at time step t, where $m_t = 0$ corresponds to the conservative policy and $m_t = 1$ to the aggressive policy. The control mode evolves according to

$$m_0 = 0, \qquad m_{t+1} = \begin{cases} 1, & \text{if } m_t = 1 \text{ or } d_t = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{5.12}$$

This definition makes the switching mechanism irreversible: once the aggressive policy is activated, the system remains in that mode until the end of the episode.

The executed action is then selected as

$$a_t \sim \begin{cases} \pi_{\text{conservative}}(\cdot \mid s_t), & \text{if } m_t = 0, \\ \pi_{\text{aggressive}}(\cdot \mid s_t), & \text{if } m_t = 1. \end{cases} \tag{5.13}$$

Therefore, the switching classifier is evaluated at every decision step until a switch occurs, after which the aggressive policy exclusively controls the agent.

Finally, we report the specific hyperparamters used for our classifier in Table 5.4.

Chapter 6

Results

6.1 Evaluation Protocol

In this section, we describe the evaluation pipeline used to obtain the final results, detailing how the performance of the proposed method and of the baseline approaches is assessed, and how fair comparisons are ensured. All policies are evaluated in the same environment and under the same problem parameters introduced in Section 5.1.

For the final evaluation, each policy is tested over 2000 episodes for each of 3 random seeds (2208, 2306, 3101), resulting in a total of 6000 evaluation episodes per policy. In every episode, the time budget is sampled from the same discrete uniform distribution used during training. The environment configuration is kept identical across methods, while variability is induced by randomly sampled initial states and mission time budgets. During evaluation, policies are executed deterministically, and the reported results are averaged across seeds.

Success rate. The success rate is defined as the fraction of evaluation episodes in which the agent reaches the goal within the available time budget. Let N denote the total number of evaluation episodes, and let $I_i \in \{0, 1\}$ be an indicator variable equal to 1 if the goal is reached within budget in episode i, and 0 otherwise. The success rate is computed as

$$\text{Success Rate} = \frac{1}{N} \sum_{i=1}^{N} I_i. \tag{6.1}$$

Expected hazard exposure. The hazard cost of an episode is defined as the cumulative number of time steps in which the agent is in contact with a hazardous region. Let $c_t \in \{0, 1\}$ denote the per-step hazard cost returned by the environment. The total hazard cost of episode i is then

$$C_i = \sum_{t=1}^{T_i} c_t, \tag{6.2}$$

where T_i denotes the length of episode i. The expected hazard cost is estimated empirically as

$$\mathbb{E}[C] = \frac{1}{N} \sum_{i=1}^{N} C_i. \tag{6.3}$$

CVaR of hazard exposure. To capture worst-case safety behavior, we report the Conditional Value-at-Risk (CVaR) of the episode hazard cost [30]. For a risk level $\alpha \in (0, 1)$, CVaR is computed as the mean cost over the worst α fraction of episodes. Let $C_1, \ldots, C_N$ denote the empirical episode costs, and let $k = \lceil \alpha N \rceil$. Moreover, let

$$C_{(1)} \leq C_{(2)} \leq \cdots \leq C_{(N)}$$

be the sorted episode costs in ascending order. The empirical CVaR is then computed as

$$\mathrm{CVaR}_\alpha(C) = \frac{1}{k} \sum_{j=N-k+1}^{N} C_{(j)}. \tag{6.4}$$

While the expected hazard cost captures average safety performance, CVaR provides a measure of tail risk by focusing on the highest-cost episodes. In our evaluation, CVaR is computed over the subset of episodes belonging to the high-budget regime.

6.2 Main Results

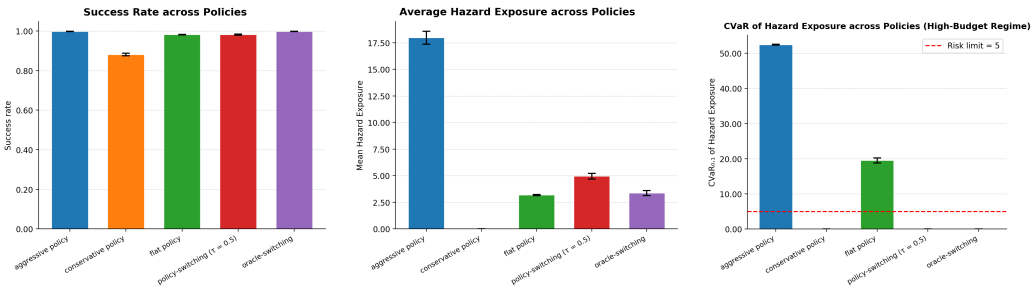
In this section, we evaluate the proposed switching method by comparing it with the baseline policies under the experimental settings defined previously. We first analyze the overall performance across different metrics, and then study how the available time budget affects the trade-off between performance and risk.

6.2.1 Performance Comparison

Figure 6.1 reports the comparison across three metrics: success rate, measuring task completion; average hazard exposure, capturing safety in expectation; and CVaR, reflecting performance under worst-case risk.

The aggressive policy, which adopts a risk-seeking strategy, achieves the highest success rate across all regimes. However, this comes at the cost of significantly higher hazard exposure and risk, as reflected by both the average exposure and the CVaR metric.

In contrast, the conservative policy represents the opposite extreme. It consistently maintains the lowest hazard exposure and CVaR, but achieves the lowest success rate, highlighting the trade-off between safety and performance.



(a) Success rate (b) Mean hazard exposure (c) CVaR

Figure 6.1: Performance comparison across policies in terms of success rate, mean hazard exposure, and CVaR in the high-budget regime. The aggressive and flat policies achieve high success rates but do not satisfy the CVaR risk limit. The conservative policy satisfies the risk bound at the cost of lower success rate, whereas the proposed switching method satisfies the risk bound while maintaining high task performance.

The flat policy optimizes the objective function defined in the problem formulation without explicitly enforcing the risk constraint. It achieves a success rate comparable to the other policies and reduces hazard exposure relative to the aggressive policy. However, it fails to satisfy the risk bound in the high time-budget regime.

The oracle switching policy constitutes an idealized upper bound for switching-based methods, as it is derived a posteriori by selecting the best switching time for each trajectory. Its performance shows that, under an ideal switching mechanism, it is possible to obtain an average hazard exposure comparable to that of the flat policy while fully satisfying the CVaR risk constraint.

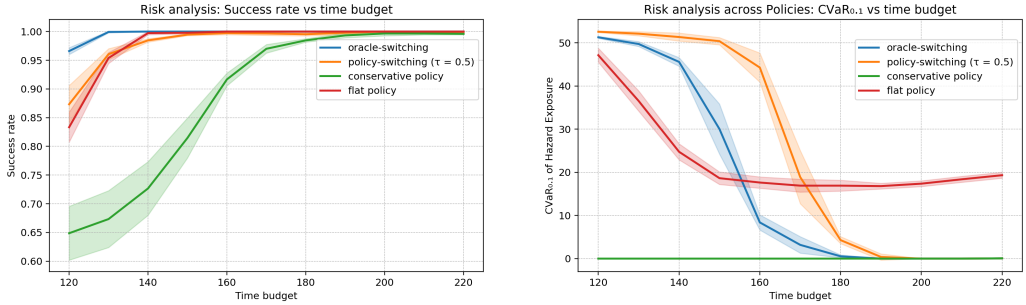
6.2.2 Effect of Time Budget on Performance and Risk

We further investigate how the available time budget affects the risk exposure of the different policies, measured through the CVaR metric. For each time budget value on the x-axis, the same set of episodes is evaluated. Varying the time budget effectively changes the difficulty of the task: lower time budgets induce time pressure, encouraging risk-seeking behavior, while higher budgets allow safer strategies.

As shown in Fig. 6.2, the conservative policy exhibits a nearly constant behavior across all time budgets. It maintains minimal risk exposure, with CVaR close to zero, but this comes at the cost of a significantly lower success rate in low time-budget regimes. This confirms that a fixed conservative strategy fails to adapt to time-critical scenarios.

In contrast, both the flat policy and the proposed switching method display adaptive behavior. As the time budget decreases, both approaches increase their risk exposure in order to maintain high success rates. Indeed, they achieve comparable performance in terms of success rate across all regimes.

However, a key difference emerges as the time budget increases. The proposed switching method progressively reduces its risk exposure, eventually satisfying the risk constraint in high time-budget regimes. The flat policy, instead, reaches



(a) Success rate

(b) Risk analysis

Figure 6.2: Effect of the available time budget on success rate and risk exposure. The same evaluation episodes are assessed under different time budgets. The left plot reports the success rate, while the right plot reports the CVaR of the hazard cost. The flat policy and the proposed method achieve comparable success rates; however, for large time budgets, the flat policy reaches a plateau in risk exposure, whereas the proposed method progressively reduces risk and approaches the conservative policy.

a plateau in risk exposure: even when the available time increases, its CVaR does not decrease further and remains above the desired risk bound.

Again, the oracle policy provides an idealized reference for the best achievable behavior under a switching mechanism. Its performance shows that, even when using only two low-level policies, an appropriate switching strategy can match the flat policy in terms of overall risk exposure across time-budget regimes while achieving a much better control of tail risk for the high-budget regime.

The proposed method follows the same qualitative trend, progressively reducing risk as the time budget increases, although with a slower transition compared to the oracle. This comparison highlights that the limitation of the flat policy does not lie in the objective itself, but in the inability of a single policy to realize the distinct behaviors required across different regimes.

This highlights the key limitation of flat policies: a fixed trade-off between performance and risk cannot capture the state-dependent requirements of the task. In contrast, the proposed switching method enables an explicit state-dependent trade-off, allowing the agent to adapt its behavior to the available time and satisfy the risk constraint when required.

6.3 Classifier Evaluation

6.3.1 Validation of the switching-controller

As introduced in Section 5.2.1, we evaluate model performance using three classification metrics: ROC-AUC, PR-AUC, and the Brier score. In addition, we report the confusion matrix to provide a more detailed overview of classification outcomes.

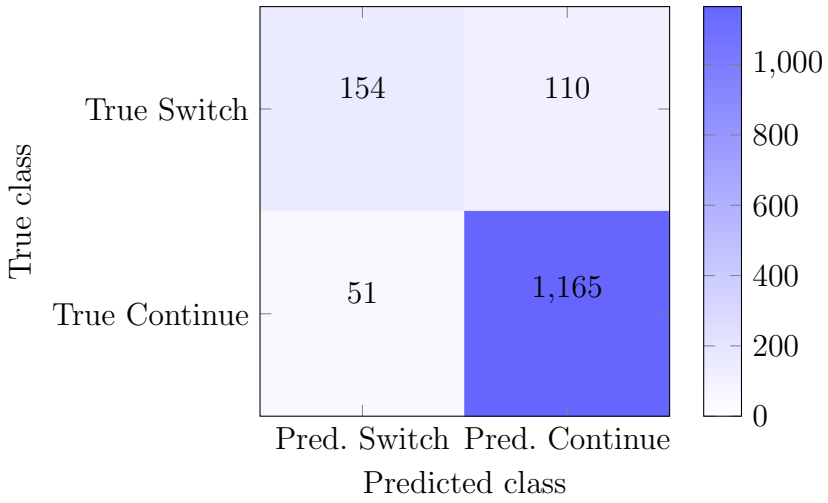


Figure 6.3: Confusion matrix on the test set at threshold 0.5.

Table 6.1: Main performance indicators of the classifier on the test set.

ROC-AUC	PR-AUC	Brier	Precision	Recall	F1-score
0.947	0.697	0.072	0.751	0.583	0.657

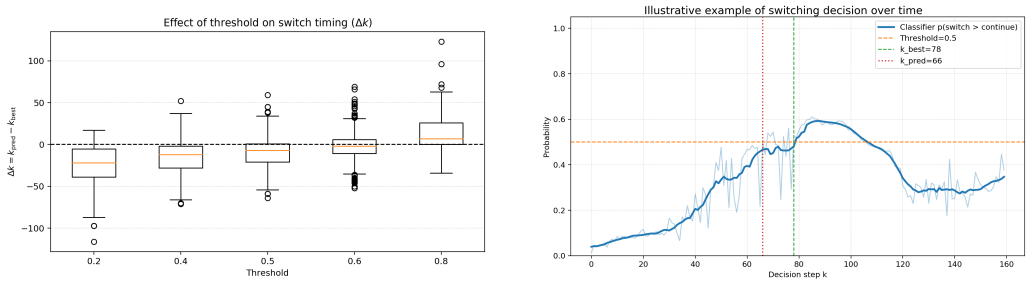
Table 6.1 summarizes the main performance indicators of the classifier on the test set.

The classifier achieves a ROC-AUC of 0.947, indicating strong separability between the *continue* and *switch* classes across different thresholds. The PR-AUC is 0.697, reflecting good performance on the minority class. The Brier score of 0.072 indicates that the predicted probabilities are well calibrated.

Figure 6.3 reports the confusion matrix at threshold 0.5. The classifier correctly identifies 154 switch cases and $1,165$ continue cases, while producing 51 false positives and 110 false negatives. Precision is higher than recall, indicating that predicted switch cases are often correct, while a number of true switch cases are missed.

6.3.2 Effect of the Decision Threshold

Figure 6.4a reports the temporal gap between the predicted switching time and the empirical best switching time k_{best}. The results show that increasing the threshold leads to delayed switching decisions, while lower thresholds result in earlier switching. For a threshold of 0.5, the predicted switching times are centered close to k_{best}, with a slight anticipatory bias.



(a) Switching-time gap over thresholds (b) Switching probability over time

Figure 6.4: Analysis of the switching controller behavior.

Figure 6.4b is a visual example of the evolution of the predicted switching probability over time for a representative trajectory. The increase in the predicted switching probability is consistent with the expected behavior of the controller. As the trajectory approaches the empirical best switching time k_{best}, the observed states become progressively more similar to the switching states used during training, leading the classifier to assign a higher probability to the switch action guaranteeing the completion of the task.

6.4 Discussion

The experimental results show that, when time constraints and risk are jointly considered, a single policy may struggle to learn the different behaviors required to simultaneously achieve high performance and satisfy the risk objective.

This limitation clearly emerges in the behavior of the baseline policies. The aggressive policy achieves high success rates across all regimes, but incurs unnecessary hazard exposure when the time budget is large. Conversely, the conservative policy maintains low risk exposure, but its fixed risk-averse behavior leads to a significant drop in performance in time-constrained scenarios. Both policies rely on a static risk attitude and therefore fail to adapt to the changing requirements induced by the time budget.

The flat policy provides a more flexible baseline, as it optimizes a single objective function that balances performance and safety. However, this trade-off is encoded directly into a single policy through the fixed parameter λ required by the problem setting, resulting in an averaged behavior across different regimes. As a consequence, while the policy can partially adjust its risk exposure depending on the available time, it is not able to satisfy the risk constraint in the high time-budget regime where an explicit risk-aware behaviour is required.

Importantly, the limitation of the flat policy does not arise from the objective function itself, but from the fact that this objective is optimized by a single, non-hierarchical policy. As a result, the learned behavior becomes an averaged compromise between safety and performance across different operating regimes. This prevents the policy from expressing the extreme risk profiles required by the task, such as highly conservative behavior when the risk constraint must be satisfied, or more aggressive behavior when time pressure makes task completion critical.

The proposed method addresses this limitation by decoupling decision-making from policy execution. Instead of learning a single policy that encodes the trade-

off, the same objective function is used at the meta-decision level to determine when it is beneficial to switch between two policies with distinct risk profiles. This mechanism enables a state-dependent decision-making strategy: the agent can adhere to the conservative policy when temporal resources are large or when environmental conditions permit conservative behavior, and transition to the aggressive policy when the available time is insufficient to maintain a conservative setting.

This difference is reflected in the empirical results. In particular, Fig. 6.2b shows that the single flat policy does not fully capture the conservative, risk-aware behaviour required in the high-budget regime. In contrast, our approach explicitly enforces this risk attitude by selecting a conservative policy whenever the mission context permits it. Moreover, we retain explicit control over this policy, as it is trained separately to satisfy the prescribed risk bound.

Regarding the classifier analysis, the classifier exhibits a conservative bias, as it produces fewer false positives than false negatives: this indicates that switching decisions are triggered only when the model is sufficiently confident, reducing unnecessary switches at the cost of missing some opportunities.

This behavior can be partially explained by the imbalance of the dataset and its structure since switching states correspond to a narrow transition region (1 decision step) within the trajectory, making them difficult to identify precisely. As a result, observations close to the switching point may share similar features with non-switch states, leading to lower recall.

At deployment level, the controller shows a slight anticipatory behavior, as observed in Fig. 6.4a for the threshold value 0.5. This can be explained by the similarity between states close to the empirical best switching point: as the trajectory approaches k_{best}, neighboring states may share features with the true switching state, causing the classifier to trigger the switch slightly earlier. This behavior is clearly illustrated in Fig. 6.4b. This anticipatory behavior is not detrimental in our setting. Switching slightly in advance may increase hazard exposure, but it preserves the opportunity to complete the task. In contrast, switching too late can make recovery impossible, leading to mission failure. Therefore, a small anticipatory bias is preferable to delayed switching decisions.

Despite the advantages introduced by the hierarchical approach, the proposed approach presents some limitations. First, the proposed approach depends on the quality of the switching classifier and on the reliability of its predicted probabilities. If the classifier is poorly calibrated or inaccurate, the overall agent performance may be compromised, even when the low-level policies are individually effective. Second, the switching mechanism may introduce undesired effects when policies are applied to states that are outside the distribution encountered during their training. In such cases, the policies may exhibit unpredictable or suboptimal behavior, potentially degrading overall performance.

Overall, the results suggest that, in environments where the desired risk behavior depends on the environment conditions and the time available to complete the task, a single policy optimized with a fixed trade-off between safety and performance could be insufficient and inappropriate. Introducing a switching mechanism provides a principled way to achieve adaptive, state-dependent behavior by decoupling high-level decision-making from low-level action selection. Instead of forcing a single policy to represent all risk profiles, the proposed method allows different policies to specialize in distinct behavioral regimes. This enables the agent to choose the most suitable policy based on the environment conditions

and the time available, leading to a more effective balance between performance and safety.

Chapter 7

Conclusions

This thesis investigated reinforcement learning methods for missions in which the acceptable level of risk is not fixed, but depends on the operational context, in particular on the remaining time budget. The work was guided by three high-level research questions, which are addressed below.

RQ1: What are the limitations of current reinforcement learning approaches based on a fixed trade-off between performance and risk? The results show that a single flat policy trained with a fixed performance–safety trade-off tends to learn an average compromise across different time-budget regimes. While this allows the policy to partially balance success and hazard exposure, it does not fully recover the conservative, risk-aware behaviour required when sufficient time is available. This indicates that encoding the whole performance–risk trade-off into a single fixed policy is restrictive when both safety guarantees and performance guarantees are required under different mission conditions (time available).

RQ2: Can current safe reinforcement learning methods adequately handle scenarios in which the acceptable level of risk varies with the operational conditions? The conservative SafeRL policy satisfies the risk objective globally, but this comes at the cost of reduced adaptability. Since the policy is trained to respect a fixed risk constraint, it remains overly risk-averse even when time is scarce and a more aggressive behaviour would be operationally preferable. Therefore, current SafeRL methods based on fixed global risk requirements provide limited support for scenarios in which the acceptable risk level changes with the mission context.

RQ3: How can we design a simple mission in which the optimal behaviour requires a dynamic risk attitude? To study this limitation, we designed a time-constrained navigation scenario where the agent must adapt its behaviour according to the remaining time budget. In the high-budget regime, the optimal behaviour is conservative and avoids unnecessary hazard exposure. In the low-budget regime, the agent must accept higher risk in order to preserve task performance. This mission therefore makes the need for a dynamic risk attitude explicit.

Proposed solution and final conclusion. To address these limitations, we proposed a policy-switching framework in which a high-level controller selects between two specialized behaviours: a risk-averse conservative policy and a risk-seeking aggressive policy. The controller is trained to maximize the same objective used by the flat policy, but it does so by choosing between distinct risk profiles rather than encoding the entire trade-off in a single policy.

Empirically, this hierarchical approach satisfies the required risk bound in the high-budget regime while maintaining performance comparable to the flat policy. This suggests that, when the desired behaviour depends jointly on temporal requirements and risk considerations, separating high-level risk-profile selection from low-level action execution provides a more suitable structure than relying on a single flat policy or a single risk-aware policy. The proposed framework therefore enables more explicit control over the agent's risk attitude and improves adaptability to the mission context.

Future work Several directions could be explored to extend the work presented in this thesis. First, this analysis could be extended to more challenging environments with dynamic hazards. In such settings, the limitations of a single flat policy may become more pronounced, as the agent would need to adapt to a wider range of time-dependent situations and risk conditions using a single policy.

Second, future work should investigate how the proposed framework can be made more integrated and scalable. The current implementation relies on a sequence of separate stages, including the independent training of low-level policies, offline rollout-based label generation, and supervised training of the switching controller. A natural extension would be to formulate the full architecture as a unified hierarchical reinforcement learning problem, in which both the low-level policies and the high-level controller are learned jointly.

Moreover, future work could move beyond a discrete set of pre-trained policies and investigate whether a single agent can adapt across a continuum of risk profiles. In this thesis, the switching mechanism operates between two extreme behaviours, namely a conservative and an aggressive policy. However, these policies can be interpreted as two endpoints of a broader spectrum of possible risk-sensitive behaviours. A natural extension would be to explicitly model this spectrum by leveraging approach such as *Universal Value Function Approximators* (UVFA). This framework allow the agent to condition its behaviour on additional parameters, such as a preference over reward and cost or a desired risk level. In this setting, the policy can be expressed as $\pi(a \mid s, \lambda)$, where λ encodes the trade-off between performance and safety. By varying λ, a continuum of behaviours can be obtained within a single model, rather than relying on a finite set of separately trained policies.

From this perspective, the approach proposed in this thesis can be seen as a simplified and practical approximation of such a parameterized framework, where only two representative operating points are learned and a high-level controller is used to select between them. Extending this idea to a continuous parameterization could improve flexibility and reduce the need for discrete switching.

Bibliography

[1] Courtney Crosby. Operationalizing artificial intelligence for algorithmic warfare. *Military Review*, 100(4):42–51, 2020.

[2] Emelia Probasco, Helen Toner, Matthew Burtell, and Tim GJ Rudner. AI for military decision-making. *Center for Security and Emerging Technology*, pages 5–26, 2025.

[3] David R Mandel. Toward a concept of risk for effective military decision making. Technical report, 2007.

[4] Kent Andersson, Martin Bang, Carina Marcus, Björn Persson, Peter Sturesson, Eva Jensen, and Gunnar Hult. Military utility: A proposed concept to support decision-making. *Technology in society*, 43:23–32, 2015.

[5] Jonathan Kwik. Iterative assessment for military artificial intelligence (AI) systems. *TMC Asser Institute for International & European Law, ASSER Research Paper*, (2025-03), 2025.

[6] André Steimers and Moritz Schneider. Sources of risk of ai systems. *International Journal of Environmental Research and Public Health*, 19(6), 2022.

[7] Kyle Bunch, Alexander C Hou, Ryan Haberman, Marissa Herron, Anthony Jacques, and Gary J Briggs. *Risk Assessment of Reinforcement Learning AI Systems: Looking Beyond the Technology*. RAND Corporation, 2024.

[8] Bianca Torossian, Frank Bekkers, Tim Sweijs, Michel Roelen, Alen Hristov, and Salma Atalla. *The military applicability of robotic and autonomous systems*. JSTOR, 2020.

[9] Ankita Kushwaha, Kiran Ravish, Preeti Lamba, and Pawan Kumar. A survey of safe reinforcement learning and constrained mdps: A technical survey on single-agent and multi-agent safety. *arXiv preprint arXiv:2505.17342*, 2025.

[10] Peter Vamplew, Cameron Foale, Conor F. Hayes, Patrick Mannion, Enda Howley, Richard Dazeley, Scott Johnson, Johan Källström, Gabriel de Oliveira Ramos, Roxana Radulescu, Willem Röpke, and Diederik M. Roijers. Utility-based reinforcement learning: Unifying single-objective and multi-objective reinforcement learning. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2024*, pages 2717–2721. International Foundation for Autonomous Agents and Multiagent Systems / ACM, 2024.

[11] Alex Ray, Joshua Achiam, and Dario Amodei. Benchmarking safe exploration in deep reinforcement learning. *arXiv preprint arXiv:1910.01708*, 7(1):2, 2019.

[12] Wyatt Hoffman and Heeu Millie Kim Kim. *Reducing the risks of artificial intelligence for military decision advantage*. Center for Security and Emerging Technology, 2023.

[13] D Mandel. Toward a concept of risk for effective military decision making. defence r&d canada—toronto. Technical report, Technical Report. DRDC Toronto TR 2007-124, 2007.

[14] Christian T Brownlees and Giampiero M Gallo. Comparison of volatility measures: a risk management perspective. *Journal of Financial Econometrics*, 8(1):29–56, 2010.

[15] Chris W Johnson. The paradoxes of military risk assessment: Will the enterprise risk assessment model, composite risk management and associated techniques provide the predicted benefits. In *Proceedings of the 25th International Systems Safety Conference, Baltimore, USA. International Systems Safety Society, Unionville, VA, USA*, volume 8, pages 59–69, 2007.

[16] Mykel J Kochenderfer. *Decision making under uncertainty: theory and application*. MIT press, 2015.

[17] Dimitri Bertsekas. *Dynamic programming and optimal control: Volume I*, volume 4. Athena scientific, 2012.

[18] Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine learning*, 49(2):267–290, 2002.

[19] Yun Shen, Michael J Tobia, Tobias Sommer, and Klaus Obermayer. Risk-sensitive reinforcement learning. *Neural computation*, 26(7):1298–1328, 2014.

[20] Fabio Pardo, Arash Tavakoli, Vitaly Levdik, and Petar Kormushev. Time limits in reinforcement learning. In *International Conference on Machine Learning*, pages 4045–4054. PMLR, 2018.

[21] Martin L. Puterman. Markov decision processes. In Daniel P. Heyman and Matthew J. Sobel, editors, *Handbooks in Operations Research and Management Science*, volume 2, pages 331–434. Elsevier, 1990.

[22] Richard S Sutton, Andrew G Barto, et al. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[23] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. Soft actor-critic algorithms and applications. arxiv 2018. *arXiv preprint arXiv:1812.05905*, 1812.

[24] Eitan Altman. Constrained markov decision processes with total cost criteria: Lagrangian approach and dual linear program. volume 48, pages 387–417. Springer, 1998.

[25] Nicolas Carrara, Edouard Leurent, Romain Laroche, Tanguy Urvoy, Odalric-Ambrym Maillard, and Olivier Pietquin. Budgeted reinforcement learning in continuous state space. *Advances in neural information processing systems*, 32, 2019.

[26] Daniel Augusto De Melo Moreira, Karina Valdivia Delgado, and Leliane Nunes de Barros. Risk-sensitive markov decision process with limited budget. In *2017 Brazilian Conference on Intelligent Systems (BRACIS)*, pages 109–114, 2017.

[27] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.

[28] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. Risk-constrained reinforcement learning with percentile risk criteria. *Journal of Machine Learning Research*, 18(167):1–51, 2018.

[29] Yinlam Chow, Aviv Tamar, Shie Mannor, and Marco Pavone. Risk-sensitive and robust decision-making: a cvar optimization approach. *Advances in neural information processing systems*, 28, 2015.

[30] Qisong Yang, Thiago D Simão, Simon H Tindemans, and Matthijs TJ Spaan. Wcsac: Worst-case soft actor critic for safety-constrained reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10639–10646, 2021.

[31] Jingjing Wang, Yanjing Sun, Bowen Wang, and Toshimitsu Ushio. Mission-aware uav deployment for post-disaster scenarios: A worst-case sac-based approach. *IEEE Transactions on Vehicular Technology*, 73(2):2712–2727, 2023.

[32] Daishi Harada. Reinforcement learning with time. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, page 577–582. AAAI Press, 1997.

[33] Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

[34] Panteha Naderian, Gabriel Loaiza-Ganem, Harry J. Braviner, Anthony L. Caterini, Jesse C. Cresswell, Tong Li, and Animesh Garg. C-learning: Horizon-aware cumulative accessibility estimation. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021.

[35] Matthew Thomas Jackson, Chris Lu, Louis Kirsch, Robert Tjarko Lange, Shimon Whiteson, and Jakob Nicolaus Foerster. Discovering temporally-aware reinforcement learning algorithms. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

[36] Chris Reinke. Time adaptive reinforcement learning. *CoRR*, abs/2004.08600, 2020.

[37] Qian Lin, Bo Tang, Zifan Wu, Chao Yu, Shangqin Mao, Qianlong Xie, Xingxing Wang, and Dong Wang. Safe offline reinforcement learning with real-time budget constraints. In *International Conference on Machine Learning*, pages 21127–21152. PMLR, 2023.

[38] Yinsen Jia and Boyuan Chen. Time-aware policy learning for adaptive and punctual robot control. *CoRR*, abs/2511.07654, 2025.

[39] Johannes Schneider and Christoph Carnehl. On risk and time pressure: When to think and when to do. *Available at SSRN 3490935*, 2020.

[40] Youtian Guo, Qichao Zhang, Junjie Wang, and Shasha Liu. Hierarchical reinforcement learning-based policy switching towards multi-scenarios autonomous driving. In *2021 international joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2021.

[41] Brendan Tidd, Akansel Cosgun, Jürgen Leitner, and Nicolas Hudson. Learning when to switch: Composing controllers to traverse a sequence of terrain artifacts. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*, pages 5144–5150. IEEE, 2021.

[42] Shubham Pateria, Budhitama Subagdja, Ah-hwee Tan, and Chai Quek. Hierarchical reinforcement learning: A comprehensive survey. *ACM Computing Surveys (CSUR)*, 54(5):1–35, 2021.

[43] Richard S. Sutton, Doina Precup, and Satinder Singh. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1):181–211, 1999.

[44] Martin Klissarov, Akhil Bagaria, Ziyan Luo, George Dimitri Konidaris, Doina Precup, and Marlos C. Machado. Discovering temporal structure: An overview of hierarchical reinforcement learning. *CoRR*, abs/2506.14045, 2025.

[45] Xiying Li and Chi-Guhn Lee. δv-learning: an adaptive reinforcement learning algorithm for the optimal stopping problem. *Expert Systems with Applications*, 231:120702, 2023.

[46] Zijun Zhang. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, pages 1–2, 2018.