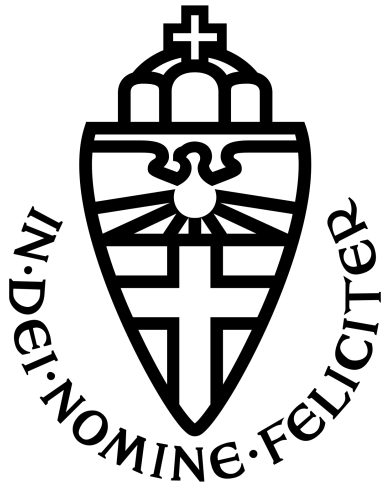
Active Measuring in Uncertain Environments

Master Thesis

Author: Merlijn Krale

Daily supervision: Dr. Thiago D. Simão

Assessors: Dr. Nils Jansen & Dr. Jana Tumova



Department of Software Science,
Radboud University Nijmegen.
August 2023

Abstract

Partial observability and model uncertainty are common problems in sequential decision-making. First, we consider *partial observability* in *active measure* contexts, which provide direct control over when and how the agents may gather information. Second, we employ *model uncertainty* to represent imprecise transition functions. We extend Markov decision processes (MDPs) to *uncertain action-contingent noiselessly observable MDPs* (uACNO-MDPs) that capture both types of uncertainty. We present an active-measure heuristic to solve uACNO-MDPs efficiently. We show how model uncertainty can, counterintuitively, lead the optimal policy to take fewer measurements and propose a method to counteract this behavior while only incurring a bounded additional cost. We empirically compare our methods to a number of baselines and show their superior scalability and performance.

Contents

1 Introduction

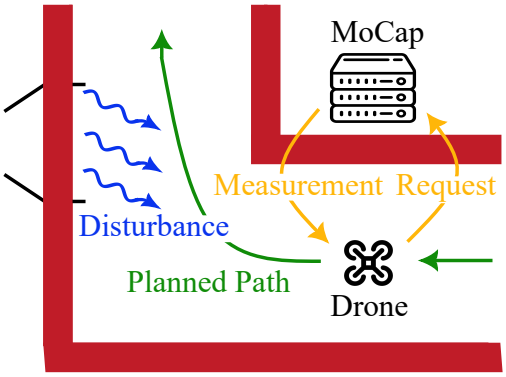


Figure 1: Visualisation of our motivating example. A drone has to plan a path through a corridor with (wind) disturbances, for which it can request measurements from a motion capture (MoCap) system.

1.1 Motivating Example

Suppose you are the manager of a warehouse that makes use of flying drones to move wares from one spot to another. The drones have the equipment and software required for low-level motion control, but to save on weight, the drones have no sensors for measuring their absolute velocity and speed directly. These quantities can be predicted from their low-level sensor data, but this might lead to inaccuracies. Alternatively, the drones can query an external motion capture system to directly measure their position and velocity. However, this external system has a limited capacity, so ideally, drones only ask for a measurement if their own predictions contain too much uncertainty for safe planning.

In one section of the warehouse, drones need to fly through a corridor with a corner, as visualized in Figure 1. The corridor has big open windows, which leads to hard-to-predict wind disturbances that can alter the path of the drone. How do we make sure the drones fly through this hallway as safely as possible without overloading the motion capture system with measurement requests?

1.2 Uncertain Active Measuring

The motivating example given above can be described as an *uncertain active measuring* problem. In this context, *uncertainty* refers to a problem where the dynamics of the system are not fully known but can only be approximated. Such problems may occur in settings with hard-to-predict disturbances to otherwise fully known dynamics (such as above or in Jiang et al. [2022]). Alternatively, settings where model dynamics need to be inferred from finite data often include similar model uncertainty in the form of confidence bounds [Xiao et al., 2012, Suilen et al., 2022].

On the other hand, *active measure* problems describe settings where agents have direct control over their observations, these have some related cost. Although not much research on active measure models exists, they could be used for robotics problems with expensive-to-use sensors (such as above or in Yin et al. [2020]), or predictive maintenance problems where the cost of an inspection needs to be weighed against the risk of failures [Jimenez-Roa et al., 2022].

Both uncertain and active measurement models have previously been studied separately, but no prior works exist on specifically combining the two. Instead, uncertain active measuring environments could be modeled using frameworks for more general *uncertain partially observable* settings, which include any setting in which agents do not have full information about their current state. However, although some methods exist to solve these environments, they either scale poorly or are particularly ill-suited for active-measure environments.

1.3 Problem Statement

This thesis focuses on uncertain active measuring environments, with the following goals:

1. **Modelling** : define a framework to model systems with uncertainty in both the system dynamics and current state, where the latter can be resolved through measurements.
2. **Model solving**: find a method of solving such models, i.e. find strategies that perform well while taking into account the cost of measuring for all possible system dynamics.

1.4 Overview

The remainder of this section gives a high-level overview of how we aim to achieve these goals in this thesis, with headings corresponding to the different sections:

Background: Modeling and Solving Uncertainty For our first goal, we fortunately find that methods for modeling both system- and state uncertainty already exist. The first type is captured by *Uncertain Markov decision processes* (uMDPs), an extension of generic Markov decision processes where the transition probabilities are replaced by uncertainty sets. The second type can be captured by *partially observable Markov decision processes* (POMDPs), in which we assume the exact state is unknown but can be inferred from current and previous *observations*. Of particular interest for us is a subclass of POMDPs, known as *action-contigent noiselessly observable Markov decision processes* (ACNO-MDPs), in which we only receive observations when taking a specific measuring action. Lastly, *uncertain POMDPs* (uPOMDPs) combine the model uncertainty from uMDPs and partial observability of POMDPs. We discuss all these frameworks, as well as methods of solving them, with a focus on details relevant to our own use case.

uACNO-MDPs For our first goal of modeling uncertainty with measuring actions, we define *uncertain ACNO-MDPs* (uACNO-MDPs) using definitions and properties of the frameworks introduced in the background section. Through a number of examples, we intuitively show a number of interesting properties uACNO-MDPs possess, which we compare with those of (more general) *uncertain POMDPs*. To make uACNO-MDPs more concrete, we implement a simplified version of the motivating example from this section as an uACNO-MDP.

Act-then-measure in uACNO-MDPs For our second goal of model solving, we start our search by looking for *robust* policies, which are policies that optimize for the worst-case transition probabilities within the given uncertainty set. Based on work from my research internship, we introduce *robust act-then-measure* (RATM) as an algorithm to compute approximate policies for uACNO-MDPs. This method makes the assumption that for choosing non-measuring actions, state uncertainty only needs to be considered for the current step. We consider how this assumption translates to an uncertain setting, then introduce *uncertain measuring value* to determine when to take measurements.

Measurement Leniency One downside of our method of finding robust strategies is that they can be very conservative in the number of measurements they take. Even though this behavior is optimal in the worst case, we show that making more measurements might increase performance for other probabilities within the uncertainty set with only limited cost in the worst case. In an effort to find policies that do not have this drawback, we define the concept of *measurement leniency*. Intuitively, measurement-lenient policies allow for non-robust measurements to be made if this could yield better performance in another (average-case) version of the model.

Empirical Analysis To evaluate our algorithms, we test the performance of both the fully robust and measurement-lenient versions of our algorithm on a number of test environments against a number of baselines. For measurement leniency, we consider a number of different variants which measure according to an optimistic, pessimistic, or average-case version of the uncertain model. To start, we use toy-example environments to show that our robust algorithms are able to correctly determine worst-case transition probabilities in a partially observable setting, as well as the measuring behavior of our measurement-lenient algorithms. Lastly, we test scalability in the larger uACNO-MDP environment inspired by our motivating example

Discussion and Conclusion To conclude, we discuss a number of limitations of our work, summarize our findings, and mention possible future research directions.

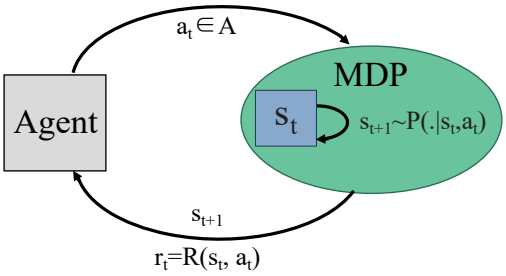


Figure 2: Visualisation of interactions with an MDP environment.

2 Background: Modeling & Solving Uncertainty

This section contains an overview of a number of modeling frameworks, which we will use as a basis for the uACNO-MDP framework defined in the next section. The goal is to give an intuitive understanding of both the workings and use case for these models, as well as provide more detailed background knowledge on topics that will be relevant in later sections. References to more in-depth analyses are provided for each framework. Throughout this thesis, we will only consider *infinite-horizon* and *discounted* models unless explicitly specified otherwise.

Basic Notation

Here, let us briefly define some basic notation as used throughout this thesis. We denote $\Delta(X)$ as the set of probability distributions over elements of set X. Given a function $F\colon X \to \Delta(Y)$ and elements $x \in X, y \in Y$, $F(\cdot|x)$ denotes the conditional probability distribution over Y given x, $F(y|x)$ the probability of element y given x, and $y \sim F(x)$ an element y randomly sampled from $F(x)$. We denote the $\delta_{x,y}$ as the Kronecker delta function, which returns 1 if x and y are equal and 0 otherwise. For notational convenience we overload all function $F\colon \Delta(X) \to Y$ with $F\colon X \to Y$ such that $F(x) = F\big(\{\delta_{x,x'}|x' \in X\}\big)$. For sequential decision-making processes, we denote x_t as the value of variable x at time-step t.

2.1 Markov Decision Processes

Markov decision processes (MDPs) are a commonly used framework for modeling sequential decision-making problems [Puterman, 1994]. They are defined as follows:

Definition 1. A *Markov decision process* (MDP) is given by a tuple $M = (S, A, P, R, \gamma)$, with:

- S the set of *states*;
- A the set of *actions*;
- $P : S \times A \to \Delta(S)$ the *transition function*;
- $R : S \times A \to \mathbb{R}$ the *reward function*: and
- $\gamma \in [0, 1]$ the *discount factor*.

Following Kaelbling et al. [1998], interactions of an agent with an MDP can be visualized as in Figure 2. Starting in some state s_0, at each time-step t, the agent chooses to take an action $a_t \in A$. The environment then changes its current state $s_t \in S$ according to $s_{t+1} \sim P(\cdot|s_t, a_t)$, and returns this new state and a reward $r_t = R(s_t, a_t)$ to the agent. The goal of the agent is to maximize its *expected discounted returns*, given as $\mathbb{E}[\sum_t \gamma^t r_t]$.

Solving MDPs

We notice that MDPs are *Markovian*, meaning that probabilities of events at any time $t = \tau$ only depend on properties at time $t = \tau - 1$. Thus, to maximize expected returns, agents only need to consider the current state of the MDP, which means optimal behavior can be captured by a *memoryless policy* (or *strategy*) of the form:

$$\pi : S \to \Delta(A). \tag{1}$$

For any policy π, we define a *value function* $V(\pi, s)$ as the expected return for following π from state s, which can recursively be defined as follows:

$$V(\pi, s) = R(s, \pi(s)) + \gamma \sum_{s' \in S} P(s'|s, \pi(s))V(\pi, s') \tag{2}$$

With the value function as a way of determining how 'good' a given policy is, we define an *optimal policy* π^* as a policy with the highest possible value for the initial state. One way of finding such a policy is through *value iteration*, as outlined in Algorithm 1. This method uses dynamic programming to approximate both the optimal value function V^* and policy π^* iteratively until some convergence condition is met (e.g., the value function changes with less than some ϵ at each iteration for all states) [Kochenderfer et al., 2015].

Algorithm 1 VALUE ITERATION

Initialise V^*
while Convergence condition not met **do**
  for $s \in S$ **do**
    $V^*(s) = \max_{a \in A}[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a)V^*(s')]$
    $\pi^*(s) = \arg\max_{a \in A}[R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a)V^*(s')]$
return V^*, π^*

As an alternative to value iteration, *Q-learning* [Watkins and Dayan, 1992] iteratively approximates the *Q-value function* $Q : S \times A \to \mathbb{R}$, which gives the optimal expected return after a given state-action pair. An implementation is shown in Algorithm 2. Instead of iteratively updating all states (such as in value iteration), Q-learning instead traverses the environment via its best estimation of π^* and updates only the states it visits. Despite this, like value iteration, Q-learning still provably converges to optimal policies [Watkins and Dayan, 1992]. Moreover, we notice Q-learning is *model-free*, meaning it does not explicitly keep track of the model but bases its updates solely on the current state and reward. Because of this, it can be used in *reinforcement learning* (RL) settings where the model dynamics are unknown.

Algorithm 2 Q-LEARNING

Initialise Q, pick *learning rate* α;
while Q not converged **do**
  $t = 0$;
  while episode not done **do**
    $\pi(s_t) = \arg\max_{a \in A} Q(s_t, a)$;
    Pick $a_t = \pi(s_t)$ with probability $1-\epsilon$, and a random action $a_t \in A$ otherwise;
    Perform action a_t, receive r_t, s_{t+1};
    $Q(s_t, a_t) \leftarrow (1 - \alpha)Q(s_t, a_t) + \alpha\big(r_t + \gamma Q(s_{t+1}, \pi(s_{t+1}))\big)$;
return V^*, π^*

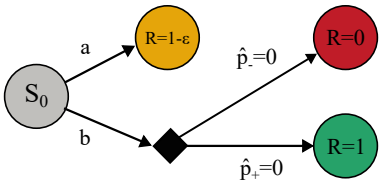


Figure 3: An example of an approximated MDP where the optimal behavior is 'risky.' We assume $\epsilon > 0$, and $\hat{p}_-, \hat{p}_+$ the transition probabilities that approximate some real probabilities p_-, p_+. For the approximated transition probabilities, action b is optimal, yielding an expected return of 1 instead of $1-\epsilon$. However, this behavior is suboptimal for even small errors in $\hat{p}_-$, while gaining only an extra return ϵ.

2.2 Model Uncertainty

Markov decision processes are an often-used framework to model real-life situations but have the downside that they require full knowledge of the modeled system dynamics, i.e. an accurate transition function. In reality, though, the dynamics of a system might only be known approximately. Errors in the approximations can easily lead to policies that take unnecessary risks, as can be seen by the example in Figure 3.

Uncertain Markov decision processes (uMDPs) [Nilim and Ghaoui, 2005] are a framework to express uncertainty in the transition function explicitly, dealing with the problem outlined above. They are defined as follows:

Definition 2. An *uncertain Markov decision processes* (uMDP) is given by a tuple $M_u = (S, A, \mathcal{U}, R, \mathcal{P}, \gamma)$, with:

- S, A, R, γ as defined for generic MDPs;
- $\mathcal{U}$ the *uncertainty set*: and
- $\mathcal{P} : S \times A \times S \to \mathcal{U}$ the *uncertain transition function*.

Interactions with a uMDP can be viewed as a two-player (adversarial) game between an agent and 'nature.' Starting in the initial state s_0, at each timestep t, the agent chooses an action a_t according to its policy π. Then, nature chooses transition probabilities $P(\cdot|s_t, a_t)$ such that $\forall s' : P(s'|s_t, a_t) \in \mathcal{P}(s'|s_t, a_t)$ and $\sum_{s'} P(s'|s_t, a_t) = 1$. After this, the model transitions as it would in a generic MDP: the environment transitions to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and the agent observes s_{t+1} and receives reward r_t. As for MDPs, the goal of the agent is to maximize their expected discounted return.

We note that our definition as a two-player game implicitly assumes nature can choose any transition probability independently of past or future choices. Such a model is referred to as *((s, a)-)rectangular* [Wiesemann et al., 2013] and *dynamic* [Nilim and Ghaoui, 2005], with the former meaning probabilities for different state-action pairs are independent, and the latter that probabilities for the same state-action pair at different times are independent. This assumption is not always realistic for real-life applications but massively decreases the complexity of the model and is thus common in literature. Likewise, we'll assume both s,a-rectangular and dynamic models for all upcoming model definitions.

Robustness

The behavior of nature is commonly assumed to be *adversarial*, meaning its goal is to minimize the agent's expected return. For infinite-horizon rectangular uMDPs, this behavior simplifies our problem to solving a generic MDP that captures the worst case, which we refer to as the *worst-case* MDP [1] which we denote as RMDP. We can define the worst-case value function V_R and transition function P_R for this MDP as follows:

$$V_R^*(s) = \max_{a \in A} R(s, a) + \gamma \sum_{s' \in S} P_R(s'|s, a) V_R^*(s'), \tag{3}$$

$$P_R(s'|s, a) = \arg \inf_{p \in \mathcal{P}(s'|,a,s)} pV_R(s') - \sum_{s'' \in S/s'} P_R(s''|s, a) V_R(s''), \text{ s.t. } \sum_{s'' \in S} P_R(s''|s, a) = 1 \tag{4}$$

We notice finding V_R^* requires solving a nested optimization problem, i.e. maximizing a over the minimum of the uncertainty set. To keep this computationally tractable, we often add restrictions on the uncertainty set that allow us to solve this optimization problem more efficiently.

[1] In some sources, the term 'robust MDP' is used as a synonym for the full uncertain MDP instead.

Interval and Confidence MDPs

One commonly-used subset of uMDPs are *Interval MDPs* (IMDPs) [Nilim and Ghaoui, 2005], which are defined as:

Definition 3. An *interval Markov decision process* (IMDP) is a uMDP with an uncertainty set consisting only of *intervals*. As such, an IMDP is given by a tuple $M_I = (S, A, \mathcal{I}, \mathcal{P}, R, \gamma)$, with:

- S, A, R, γ as defined for generic MDPs;
- $\mathcal{I} : \{[p_{\text{min}}, p_{\text{max}}] | 0 \leq p_{\text{min}} \leq p_{\text{max}} \leq 1\}$ the set of *intervals* [2]: and
- $\mathcal{P} : S \times A \times S \to \mathcal{I}$ the *uncertain transition function*.

For notational convenience, we further define $p_{\text{min}}(s'|s,a), p_{\text{max}}(s'|s,a)$ as the bounds on the interval $\mathcal{P}(s'|s,a)$.

To model real-life problems, interval uncertainty sets are often sufficient. In particular, they straightforwardly represent confidence intervals, which can arise both in RL-settings [Suilen et al., 2022], or in other settings where models are based on sampling [Xiao et al., 2012, Jiang et al., 2022]. Moreover, the interval structure of IMDPs allows us to find the robust transition function of IMDPs more quickly than for general uMDPs. To do this, we simply order transitions from worst-case to best-case, then maximize worst-case probabilities and minimize best-case probabilities. More formally, an implementation of value iteration using this idea is given in Algorithm 3.

Algorithm 3 VALUE ITERATION FOR IMDPS

Initialise V^*;
while V^* not converged **do**
  Re-order states such that $V(s_0) \leq V(s_1) ... \leq V(s_n)$
  for $s \in S$ **do**
    for $a \in A$ **do**
      Set $i = 0$
      $p_{\text{total}} = \sum_{s' \in S} p_{\text{min}}(s'|s,a)$
      while $p_{\text{tot}} < 1$ **do**
        $P_R(s,a,s_i) = \min\{p_{\text{max}}(s'|s,a), 1 - p_{\text{tot}}\}$
        $p_{\text{tot}} = p_{\text{tot}} + \big(p_{\text{max}}(s'|s,a) - p_{\text{min}}(s'|s,a)\big)$
        $i = i + 1$
      $\forall j \geq i : P_R(s_j|s,a) = p_{\text{min}}(s_j|s,a)$
    $V^*(s) = \max_a [R(s,a) + \gamma \sum_{s' \in S} P_R(s'|s,a) V^*(s')]$
    $\pi^*(s) = \arg\max_a [R(s,a) + \gamma \sum_{s' \in S} P_R(s'|s,a) V^*(s')]$
return V^*, π^*

In addition to modeling settings with real model uncertainty, uMDPs can also be used as a way to compute *risk-averse* policies for fully known models. In particular, Osogami [2012] proves that the problem of optimizing for a certain class of risk measures can be interpreted as a robustness problem. For this, they introduce a class of uMDPs that have uncertainty sets such that any probability is at most increased by a factor $\frac{1}{\alpha}$, where α is a parameter related to the risk measure. We'll refer to these as *confidence MDPs*, and formally define them as follows:

Definition 4. A *confidence Markov decision process* is given by a tuple $M_c = (S, A, P, \alpha, R, \gamma)$, with:

- S, A, P, R, γ as defined for generic MDPs: and
- $\alpha \in [0, 1]$ the *confidence level*;

For any confidence Markov decision process M_c, its *uncertain transition function* is defined as follows:

$$\mathcal{P}(s'|s,a) = [0, p], \text{ with } p = \min\{\frac{1}{\alpha} P(s'|s,a), 1\} \tag{5}$$

For the purpose of this thesis, confidence MDPs are used as an intuitive way to add uncertainty to generic MDPs, which will prove useful for our empirical analysis.

[2] Other sources sometimes add the assumption that intervals are *non-vanishing*, meaning either $p_{\text{min}} = p_{\text{max}} = 0$ or $p_{\text{min}} > 0$. However, we will allow vanishing intervals throughout this thesis.

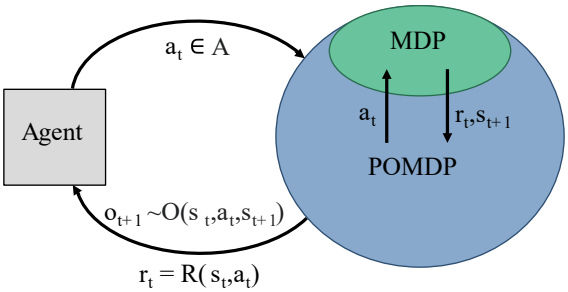


Figure 4: Visualisation of interactions with a POMDP environment.

2.3 Partial Observability

When using an MDP framework, we assume agents have full knowledge of the state of their environment. However, in many real-life applications, agents do not have full knowledge but can only infer the real state of the environment from partial or noisy observations.

Partially observable MDPs (POMDPs) [Kaelbling et al., 1998] are an extension of the MDP framework, which expresses the agents limited observation capabilities explicitly. They are defined as follows:

Definition 5. A *partially observable Markov decision process* (POMDP) is given by the tuple $\mathcal{M}_P = (S, A, P, \Omega, O, R, \gamma)$, with:

- S, A, P, R, γ as defined for generic MDPs;
- Ω the set of *observations*: and
- $O : S \times A \times S \to \Delta(\Omega)$ the *observation function*.

Interactions with POMDPs are visualized in Figure 4. They follow a similar pattern as that of generic MDPs, with agents choosing an action a_t for each timestep t. However, instead of receiving full state information after each step, the agent now receives an observation o_t according to the observation function $O(s_t, a_t, s_{t+1})$. This means agents do not know, but can only infer the real state of the model.

Belief states

In contrast to generic (u)MDPs, optimal policies for POMDPs generally require *memory*, meaning they need to remember past interactions. The most intuitive way to represent these interactions is with a *history* h_t, defined as:

$$h_t = (a_0, o_0, a_1, o_1, ...a_t, o_t). \tag{6}$$

Unfortunately, the size of this representation grows with time, meaning it is often impractical to use for policy computation. Alternatively, a *belief state* b_t gives the probability distribution over possible current states, which we can define recursively as follows:

$$b_t(s') = \begin{cases} \delta_{s',s_0} & \text{if } t = 0, \\ \Pr(s'|b_{t-1}, a_{t-1}, o_{t-1}) & \text{otherwise.} \end{cases} \tag{7}$$

Applying Bayes' rule to this equation, we may write a *belief update function* to calculate the next belief b_{t+1} as follows:

$$b_{t+1}(s') = \frac{O(s', o_t) \sum_{s \in S} b_t(s) P(s'|s, a_t)}{\sum_{s'' \in S} O(s'', o_t) \sum_{s \in S} b_t(s) P(s''|s, a_t)}, \tag{8}$$

where the numerator gives the probability of reaching state s' and observing o_t after taking action a_t in b_t, while the denominator is a 'normalization factor' which gives the total chance of observing o_t after this belief-action pair. Beliefs are considered a *sufficient statistic* [Kaelbling et al., 1998], meaning they summarize the full history h_t well enough for optimal policy computation.

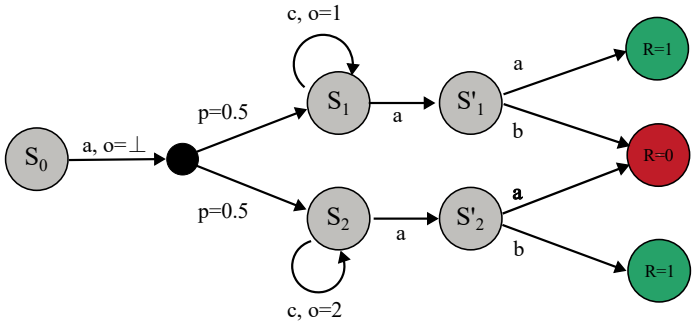


Figure 5: A simple example of a POMDP where Q_{MDP} takes a sub-optimal actions. After taking action a in s_0, action c is optimal (for any $\gamma > 0.5$), since this gives us an observation with which we can determine our current state. However, with the Q_{MDP} assumption of full observability for future states, the value of being able to distinguish s'_1 and s'_2 is ignored, in which case action c is suboptimal as compared to action a.

Solving POMDPs

Although exact methods of solving POMDPs exist [Sondik, 1978], the problem of finding exact solutions is known to be PSPACE-complete in general and thus infeasibly hard for practical applications [Papadimitriou and Tsitsiklis, 1987]. Thus, POMDPs are often solved approximately instead. Many different approximation methods for POMDPs exist, including Monte Carlo methods (such as *POMCP*, Silver and Veness [2010]), neural networks (such as *Deep Recurrent Q-learning Networks*, Hausknecht and Stone [2015]), or point-based belief methods [Shani et al., 2013].

Another such approximate method is Q_{MDP}. Introduced by Littman et al. [1995], the idea of this method is to approximate the Q-values for a given belief using those of the underlying MDP, as follows:

$$Q(b,a) \approx \sum_{s \in S} b(s) Q_{\text{MDP}}(s,a) = \sum_{s \in S} b(s) R(s,a) + \gamma \sum_{s' \in S} P(s'|s,a) V_{\text{MDP}}(s'), \tag{9}$$

where $Q_{\text{MDP}}, V_{\text{MDP}}$ are the optimal Q-value and generic value functions of the underlying MDP, or equivalently, the POMDP where observation function O always returns the current state. The Q_{MDP} algorithm, then, starts by precomputing the value function for the underlying MDP, then for each step computes its current belief and takes the best action according to Equation (9). An implementation is shown in Algorithm 4.

Algorithm 4 Q-MDP

Precompute Q_{MDP}
Intitialize $b_0 = \delta(s, s_0), t = 0$
while episode not done **do**
  $a_t \leftarrow \arg\max_{a \in A} \sum_{s \in S} b_t(s) Q_{\text{MDP}}(s,a)$
  Perform action a_t, receive r_t, o_t
  Compute b_{t+1} using (b_t, a_t, o_t) in Equation (8)
  $t = t + 1$
return $\sum_t \gamma^t r_t$

The main advantage of Q_{MDP} over other approximation methods is its speed: MDPs can generally be solved very efficiently, and after that choosing actions with the Q_{MDP} method is very quick. However, since the method essentially disregards state uncertainty for future steps, it never takes actions that give extra information through their observations if these actions would be sub-optimal (or have no effect) in a fully observable setting. An example of such a setting is shown in Figure 5.

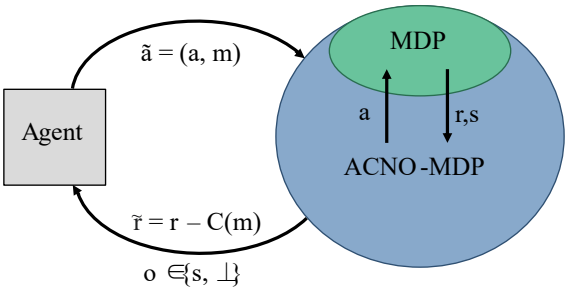


Figure 6: Visualisation of interactions with an ACNO-MDP environment.

2.4 ACNO-MDPs

POMDPs are a powerful framework for modeling state uncertainty but can be hard to solve. For this reason, researchers sometimes focus on subsets of POMDPs with properties that make policy computation more tractable.

Action contingent noiselessly observable MDPs (ACNO-MDPs) [Bellinger et al., 2021, Nam et al., 2021] are a subset of POMDPs specifically designed for *active measure* problems as defined in the introduction, defined as follows:

Definition 6. An *action-contingent noiselessly-observable Markov decision process* (ACNO-MDP) is given by the tuple $\mathcal{M}_{\mathcal{A}} = (S, \tilde{A}, P, R, c, \gamma)$, with:

- (S, P, R, γ) as for generic MDPs;
- $\tilde{A} = A \times M = A \times \{0, 1\}$ the set of *action pairs*: and
- $c \in \mathbb{R}$ the *measurement cost*.

Futhermore, we define an observation set $\Omega : S \cup \{\perp\}$ and observation function $O : S \times M \to \Omega$, such that:

$$O(s, m) = \begin{cases} \perp & \text{if } m = 0 \\ s & \text{if } m = 1. \end{cases} \tag{10}$$

Lastly, we define the measuring cost function $C : M \to \mathbb{R}$, such that:

$$C(m) = \begin{cases} 0 & \text{if } m = 0 \\ c & \text{if } m = 1. \end{cases} \tag{11}$$

Agent-environment interactions for ACNO-MDPs are similar to those of generic POMDPs and are visualized in Figure 6. Starting in some initial state s_0, for each time-step t, the agent chooses to execute an action pair $\langle a_t, m_t \rangle$, consisting of *control action* a_t and *measuring action* m_t. The environment then transitions to a new state $s_{t+1} \sim P(\cdot|s, a)$, which we note is independent of the measuring action m_t. Lastly, the environment returns observation $o_t = O(s_{t+1}, m)$, as well as both a return r_t and *measuring cost* $c_t = C(m)$. We combine the latter two into a *scalarized reward* $\hat{r}_t = r_t - c_t = R(s_t, a_t) - C(m_t)$. The goal of the agent is to maximize the expected discounted scalarized returns $\mathbb{E}[\gamma^t \hat{r}_t]$.

ACNO-MDPs nicely represent the measuring behavior described in our motivating example. However, we point out a number of assumptions that may not hold in many real-life applications:

- *Complete measurements*: a measurement returns full state information. In reality, certain measurements might only give partial state information, e.g. a speedometer might measure speed but not position.
- *Noiseless measurements*: the observations are always error-free. In reality, measurements may contain sporadic errors or only give uncertain state information.
- *Predictably costly*: measuring has a constant cost that can be compared directly to gained returns. In reality, measuring costs may fluctuate depending on state, time, or the number of previous measurements, or the total costs might be constrained.

ACNO-MPDs are a particularly interesting framework for RL settings, where we assume the model dynamics are unknown but can be learned through interactions. In these settings, learning a model through taking some kind of

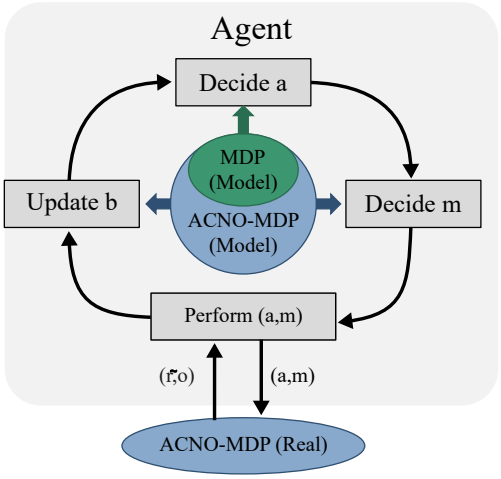


Figure 7: Visualisation of the control-loop for solving ACNO-MDPs using the *act-then-measure* heuristic.

measurement might have costs that need to be weighed against expected returns. Thus, most previous research has focussed on such RL applications [Bellinger et al., 2021, Nam et al., 2021].

The Act-then-measure Heuristic & Measuring Value

The *act-then-measure* (ATM) heuristic is a method I proposed in my research internship (published as [Krale et al., 2023]) for finding approximately optimal policies for ACNO-MDPs by making use of the explicit distinction between measuring and non-measuring actions. Inspired by the Q_{MDP} method for POMDPs mentioned earlier, the ATM heuristic can be stated as follows:

> When choosing control actions, assume all (state) uncertainty will be resolved in the next state(s).

Like for Q_{MDP}, this assumption suggests control actions can be chosen using Equation (9), which can be done very efficiently. Moreover, the main downside of Q_{MDP} does not apply in the same way here since control actions do not influence observations directly. Thus, situations like Figure 5 will not occur.

To determine when to measure, we use *measuring value*, defined as the difference in expected returns between measuring and not measuring:

$$\text{MV}(b,a) = Q_{\text{ATM}}(b,\langle a,1\rangle) - Q_{\text{ATM}}(b,\langle a,0\rangle), \tag{12}$$

where Q_{ATM} is the Q-value of a belief-action pair assuming future control actions are chosen according to Equation (9). We choose to measure only if doing so gives us higher expected returns, giving us the following measuring condition:

$$m_{\text{MV}}(b,a) = \begin{cases} 1 & \text{if } MV(b,a) \geq 0 \\ 0 & \text{otherwise .} \end{cases} \tag{13}$$

Combing our methods of choosing control- and measuring actions, we define a planning algorithm for following the ATM heuristic, as shown in Algorithm 5 and visualized in Figure 7:

Algorithm 5 ATM PLANNER(P, Q, s_0)

Intitialize $b_0 = \delta(s, s_0), t = 0$
while episode not done **do**
  Pick control action a_t using b_t in Equation (9)
  Pick m_t using b_t and a_t in Equation (12)
  Execute $\tilde{a}_t = \langle a_t, m_t \rangle$, receive reward $\tilde{r}_t$ and observation o_t
  Compute b_{t+1} using b_t, a_t, o_t in Equation (8)
return $\sum_t \gamma^t \tilde{r}_t$

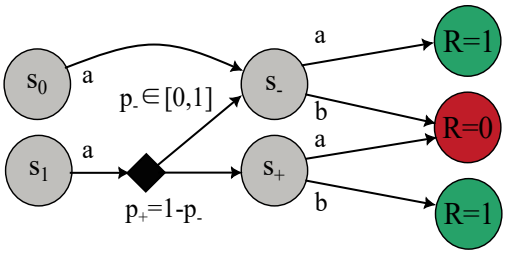


Figure 8: An example uPOMDP which shows worst-case transitions are generally belief-dependent.

2.5 Combining Uncertainties

Uncertain partially observable MDPs are a framework that combines the partial observability of POMDPs with the model uncertainty of uMDPs. They are defined as follows:

Definition 7. An *uncertaint partially observable Markov decision process* (uPOMDP) is given by a tuple $\mathcal{M}_u = (S, A, \mathcal{U}, \mathcal{P}, \Omega, O, R, \gamma)$, with:

- (S, A, R, γ) as for generic MDPs;
- $\mathcal{U}, \mathcal{P}$ as for uMDPs: and
- Ω, O as for POMDPs.

Agent-model interactions can be defined as before, where we assume partial observability only for the agent, meaning nature has full observability of both the current state and the agent's current belief.

As for POMDPs, optimal policies for uPOMDPs generally rely on past interactions. However, these cannot be summarized with only a probability distribution over states, but must instead incorporate all possible choices of transition functions by nature. Both Rasouli and Saghafian [2018] and Bovy [2023] tackle this problem, but their solutions scale too poorly for our aims. However, if we focus only on the worst-case choices of nature (i.e., we optimize for robustness), then Osogami [2015] shows we can construct *robust beliefs* according to the worst-case dynamics, which do function as sufficient statistics and information states. The rest of this section will thus focus on this problem.

Dynamic uPOMDPs

For robustness in uPOMDPs, we start by noticing that in contrast to generic (infinite-horizon) uMDPs, there is a difference between dynamic and non-dynamic (or *static*) models. We recall that dynamic models assume nature can choose transition probabilities independently of its previous choices, while in a static model, it must always choose the same probabilities for the same state-action pair. Intuitively, in uPOMDPs, agents may act differently in the same state depending on their current belief, which means that worst-case transitions for a state may be different as well.

As an example, consider the model of Figure 8, where an agent currently has a belief over states s_0 and s_1. For its next step, the expected reward is minimized if the reward for action a is equal to that of action b, which is achieved if $b'(s_-){=}0.5+\epsilon, b'(s_+){=}0.5{-}\epsilon$. However, depending on the current belief b, this is achieved by different transition probabilities p_-, p_+. If we consider s_1 separately (i.e. $b(s_1){=}1$), the worst-case is given by $p_-{=}0.5{+}\epsilon$. If instead the probability is divided equally between s_0 and s_1 (i.e. $b(s_0){=}0.5, b(s_1){=}0.5$), the worst-case is given by $p_-{=}\epsilon$.

The above example shows we cannot represent our robust model with a value- and transition function of the forms $V_R^* : S \to \mathbb{R}$ and $P_R : S \times A \to \Delta(S)$, but are forced to make the transition function *belief-dependent* with forms $V_R^* : S \times \Delta(S) \to \mathbb{R}$ and $P_R : S \times \Delta(S) \times A \to \Delta(S)$.

Solving uPOMDPs

In principle, optimal policies for uPOMDPs can be computed with a variant of value iteration over all possible beliefs, as proposed in Osogami [2015]. In practice, however, even for simple uncertainty sets, such value iteration quickly becomes intractable. For instance, notice that the simple method of Algorithm 3 would not work for interval POMDPs as shown by the above example, meaning a more expensive solving method is required. For better scalability, Cubuktepe et al. [2021] instead compute (non-optimal) policies as finite state machines and use model checking to verify their performance lies above a specified bound. This method scales to bigger environments, but its limited memory makes it ill-suited for environments with sparse observations, such as the active measure environments considered here.

3 uACNO-MDPs

In this section, we introduce uACNO-MDPs as the combination of the uMDP and ACNO-MDP frameworks explained above. We start by giving a formal definition, then highlight a few noteworthy properties of uACNO-MDPs, and finally revisit the motivating example of the introduction as a concrete uACNO-MDP.

3.1 Definition

For completeness, let us fully write out the definition of our new framework as follows:

Definition 8. An *uncertaint action-contingent noiselessly observable Markov decision process* (uACNO-MDP) is given by a tuple $\mathcal{M} = (S, \tilde{A} = A \times M, \mathcal{U}, \mathcal{P}, R, c, \gamma)$, defined as follow:

- S the set of *states*;
- A the set of *control actions*;
- $M = \{0, 1\}$ the set of *measurement actions*;
- $\tilde{A} = A \times M$ the set of *action pairs*;
- $\mathcal{U}$ the *uncertainty set*;
- $\mathcal{P} : S \times A \times S \to \mathcal{U}$ the *uncertain transition function*;
- $R : S \times A \to \mathbb{R}$ the *reward function*[3];
- $c \in \mathbb{R}$ the *measurement cost*: and
- $\gamma \in \langle 0, 1]$ the *discount factor*.

Futhermore, we define an observation set $\Omega : S \cup \{\perp\}$ and observation function $O : S \times M \to \Omega$, such that:

$$O(s, m) = \begin{cases} \perp & \text{if } m = 0 \\ s & \text{if } m = 1. \end{cases} \tag{14}$$

Lastly, we define the measuring cost function $C : M \to \mathbb{R}$, such that:

$$C(m) = \begin{cases} 0 & \text{if } m = 0 \\ c & \text{if } m = 1. \end{cases} \tag{15}$$

Agent-environment interactions for uACNO-MDPs can be viewed as a two-player game between the agent and 'nature'. Starting from some initial state s_0, for each time-step t, the agent chooses an action pair $\tilde{a}_t = \langle a_t, m_t \rangle$ to execute according to some policy π.states, meaning policies are of the form $\pi(b_t)$. Based on the chosen action pair, the current state, and the agent's current belief, nature picks a transition function $P(\cdot|s_t, \tilde{a}_t)$, subject to the constraint that $\forall s' : P(s'|s_t, \tilde{a}_t) \in \mathcal{P}(s'|s_t, a_t)$ and $\sum_{s' \in S} P(s'|s_t, \tilde{a}_t) = 1$. The environment then transition to a new state $s_{t+1} \sim P(\cdot|s_t, a_t)$, and returns to the agent a scalarized reward $\tilde{r}_t = r_t - C(m_t)$ and observation $o_t = O(\cdot|s_{t+1}, m_t)$.

The goal of the agent is to compute a policy π with the highest expected total discounted scalarized reward. For this thesis, we'll assume these policies are belief-based, meaning they are of the form $\pi : \Delta(S) \to \tilde{A}$. We note that our definition of agent-environment interactions implicitly assumes a dynamic and rectangular system and full observability for nature, which are common assumptions for uPOMDPs.

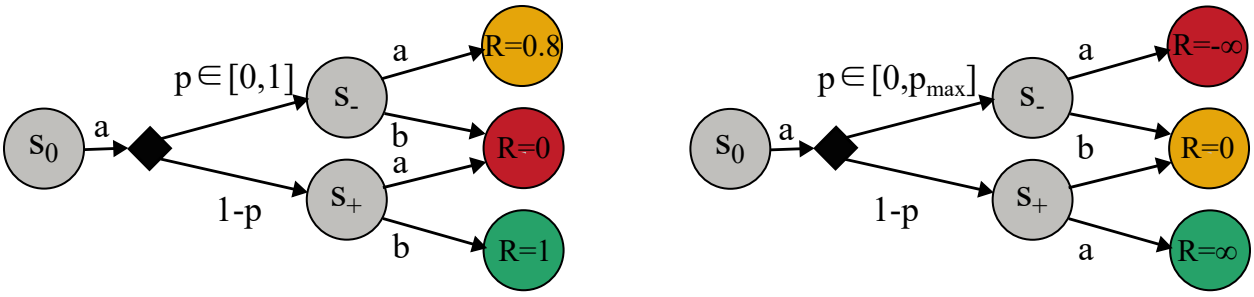
For the case of an adversarial nature, we denote P_R and V_R as the worst-case transition- and value function, which (as for uPOMDPs) are both belief-dependent. To write these out fully, we start by defining $b'_R(b, \tilde{a})$ as an expected distribution over states in the next step when taking action pair $\tilde{a}$ in belief b, given by:

$$b'_R(s'|b, \tilde{a}) = \sum_{s \in S} b(s) P_R(s'|s, b, \tilde{a}). \tag{16}$$

Using this notation, we define the robust value function V^*_R as follows:

$$V^*_R(b) = \max_{\tilde{a} = \langle a, m \rangle \in \tilde{A}} \inf_{P_R(\cdot|s,b,\tilde{a}) \in \mathcal{P}(\cdot|s,a)} \sum_{s \in S} b(s) \Big(R(s, a) - C(m) + \gamma \sum_{s' \in s} P_R\big(s'|s, b, \tilde{a}\big) V^*_R\big(b'_R(b, \tilde{a})\big) \Big), \tag{17}$$

[3]For notational convenience, we overload this function for action-measurement pairs as $R(s, \langle a, m \rangle) = R(s, a)$.



(a) The *a-b* environment.

(b) The *lucky-unlucky* environment.

Figure 9: Two example environments to show properties of uACNO-MDPs.

where we note b'_R is dependent on P_R and thus affected by the minimization. For completeness, we also define the robust transition function P_R as follows:

$$P_R(s'|s,b,\tilde{a}) = \arg \inf_{P_R(s'|s,b,\tilde{a}) \in \mathcal{P}(s'|s,a)} V^*_R\big(b'_R(b,'\,\tilde{a})\big)$$
$$\text{s.t.} \sum_{s'' \in S} P_R(s''|s,b,\tilde{a}) = 1 \tag{18}$$

3.2 Properties of uACNO-MDPs

Here, we highlight a number of interesting properties of uACNO-MDPs that follow from our definition. For this, we use two examples (Figures 9a and 9b), both of which will also be used in our empirical analysis.

3.2.1 Transition Functions Depend on Measurements

In Equation (18), we have defined our robust transition function to be dependent on the complete action pair $\tilde{a}$ rather than on only control actions a. This dependency was not present for generic ACNO-MDPs, where the transition function was assumed to be independent of measurements.

Let us first show an example of why this dependency must hold. Consider the uACNO-MDP shown in Figure 9a, which we'll refer to as the *a-b* environment. This environment has three states: an initial state s_0, and two next states s_- and s_+, which have different optimal actions a and b. Furthermore, the reward for taking the optimal action in s_- is slightly lower than that in s_+.

Now, let us determine the worst-case probabilities of p for both a measuring and non-measuring action pair. When measuring, s_- has a lower expected value than s_+, meaning the worst-case transition has $p = 1$. However, when not measured, having this deterministic transition means the agent can safely pick action a and receive a reward 0.8. Clearly, the expected return can be made much lower if p is closer to 0.5 instead [4]. Thus, we find the worst-case transition function must be dependent on the measuring action chosen.

To generalize our example, we may state that for fully observable transitions (such as when measuring), worst-case transitions simply maximize worst-case outcomes, leading to low-variance transitions. However, for partially observable transitions (such as when not measuring), high-variance transitions are often worse since these make it harder to take optimal actions. For uACNO-MDPs, we summarize this observation as follows:

> **Observation 1.** *In the worst case, transitions are generally low variance when measuring but may be high variance when not measuring.*

3.2.2 High Uncertainty Discourages Measuring

Similarly to how Observation 1 shows a relation between measuring and the variance of a transition, we also observe an interesting relation between measuring and the uncertainty of a transition.

[4] More precisely, the worst-case is achieved for $p_- = 0.56$, which yields expected returns 0.36.

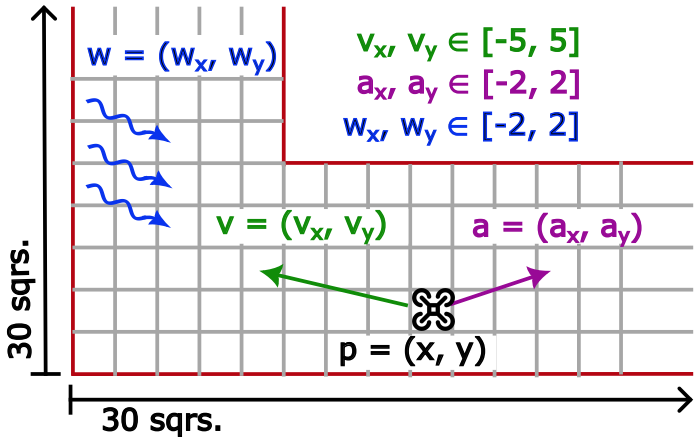


Figure 10: Visualisation of our motivating example. A drone has to plan a path through a corridor with (wind) disturbances, for which it can request measurements from a motion capture (MoCap) system.

Consider the uACNO-MDP shown in Figure 9b, which we will refer to as the *lucky-unlucky* environment. Like before, this environment has three states s_0, s_-, and s_+, but we now interpret these as a lucky and unlucky state, which are otherwise the same. In both states, taking action 'safe' b leads to a neutral reward of 0, while taking 'risky' action a gives a positive reward in s_+ and a negative reward in s_-.

Again, we make our problem concrete by setting $p_{\max} = 1$, and are now interested in measuring behavior for different uncertainty intervals, i.e., values for $p_{\max}$. We notice the expected returns for s_- are strictly lower than those of s_+, meaning the worst-case environment is always given by maximizing p, regardless of whether the agent measures. Consider first $p_{\max} = 1$, then the transition function is deterministic, and measuring is sub-optimal. However, we notice that for any $p_{\max} > 0$ and finite c, measuring would be optimal since this gives a chance of achieving infinite returns. Counterintuitively, we thus find that for larger model uncertainties, optimal robust policies might take fewer measurements to alleviate uncertainty. We may summarize this finding as follows:

> **Observation 2.** *In the worst case, higher uncertainty does not necessarily make measuring more valuable.*

We note that this property may occur even for finite returns and non-zero probabilities, although the effect is less extreme. For example, suppose the best- and worst-case rewards in Figure 9b are given by ± 1 instead of $\pm\infty$. Then, the expected value when measuring is given by $p - c$, meaning measuring is optimal for $(1 - p) \geq c$. Thus, for any $p_{\max} < c$, measuring will be suboptimal, even though it would yield positive returns for any $(1 - p_{\max}) \geq c$.

The behavior described here results from optimal robust policies (over) optimizing for the worst case. The higher the model uncertainty, the more likely worst-case outcomes become, and for these, it is generally less valuable to take measurements. This property makes sense in contexts where the environment is adversarial, such as for security applications. However, in many contexts, the environment is only assumed to be adversarial to guarantee good performance over the entire uncertainty set. In such cases, it seems undesirable that being *more* conservative about the uncertainty of the model could lead to policies that take *fewer* measurements. We will return to this behavior in Section 5.

3.3 Example: a Drone in a Corridor

As a slight tangent, this subsection describes a concrete implementation of the motivating example from the introduction as an uACNO-MDP. This environment will be used in Section 6 to test our methods, meaning this subsection serves both as a concrete example of an uACNO-MDP as well as a complete description of a testing environment. The environment is visualized in Figure 10.

Discretized Dynamics

To start, let us define a simple generic framework to represent continuous movement using discrete variables [5]. Firstly, we represent the drone's position as coordinates in a grid, meaning the drone's position can be expressed as $p = \langle x, y \rangle$, with $x, y \in \mathbb{Z}$ grid coordinates. For simplicity, we'll assume independence in dynamics between these two directions. With this assumption, we define velocity as $v = \langle v_x, v_y \rangle \in \mathbb{Z}^2$, which represents the number of grid cells moved each

timestep. Lastly, we define acceleration $a = \langle a_x, a_y \rangle \in \mathbb{Z}^2$ and perturbations $w = \langle w_x, w_y \rangle \in \mathbb{Z}^2$, which influence the change in velocity for each time-step. The (discrete-time) dynamics of our system are described as follows:

$$\begin{aligned} x_t &= x_{t-1} + \lfloor \frac{v_{x,t-1} + v_{x,t}}{2} \rfloor, \quad v_{x,t} = v_{x,t-1} + a_{x,t} + w_{x,t}, \\ y_t &= y_{t-1} + \lfloor \frac{v_{y,t-1} + v_{y,t}}{2} \rfloor, \quad v_{y,t} = v_{y,t-1} + a_{y,t} + w_{y,t}. \end{aligned} \tag{19}$$

For the positions variables $i \in \{x, y\}$, we note that $\lfloor \frac{v_{i,t-1}+v_{i,t}}{2} \rfloor$ represents the average velocity during timestep t, which could be a fraction and thus needs to be rounded.

This simple framework can be used to express an MDP with states of form $s = \langle p, v \rangle = \langle x, y, v_x, v_y \rangle$, actions of form $a = \langle a_x, a_y \rangle$, any reward function R, and a transition function P_W following Equation (19) for a given probability distribution $W : \Delta(\mathbb{Z})$ over perturbations w.

A Drone in a Corridor

To make these general dynamics concrete for our environment, we first represent our corridor by two overlapping rectangles of 6×30 squares [6], with all values outside this area expressed as one sink-state s_{sink}. Next, we restrict velocities to be within the range $[-v_{\max}, v_{\max}] = [-5, 5]$, with any value outside this range set to the closest valid value. Similarly, we restrict accelerations to be within the range $[-a_{\max}, a_{\max}] = [-2, 2]$, giving us a finite action space. We choose a probability distribution W based loosely on a Gaussian:

$$w_{i \in \{x,y\},t} = \begin{cases} 0 & \text{with probability } 0.68, \\ \pm 1 & \text{with probabilities } 0.14, \\ \pm 2 & \text{with probabilities } 0.02. \end{cases} \tag{20}$$

We define an initial state $s_0 = \langle x{=}29, y{=}2, v_x{=}v_y{=}0 \rangle$ corresponding to a motionless drone at one end of the corridor. Lastly, we define a simple reward function that yields one only if a goal area at the other end of the corridor is reached:

$$R(s, a) = \begin{cases} 1 & \text{if } y > 27 \text{ and } s \neq s_{\text{sink}}, \\ 0 & \text{otherwise.} \end{cases} \tag{21}$$

Episodes end either if $s = s_{\text{sink}}$, representing a crash, or the goal area is reached. The full MDP has a state space S of size $|S| = 26.245$ and action space A of size $|A| = 25$, with each state-action pair having up to 25 successor states.

From MDP to uACNO-MDP

To turn our environment into an uACNO-MDP, we start by defining a *confidence MDP* (Section 2.2) with confidence level α, using the above MDP as its base. This environment has an uncertain transition function $\mathcal{P}$ with maximum transition probabilities equal to those in the generic MDP times a factor $1/\alpha$, capped at 1. Put another way, both independent directions follow a conditional probability distribution given as:

$$\forall i \in \{x, y\} : \Pr(i_t | , i_{t-1}, v_{i,t-1}, a_{i,t-1}) = \begin{cases} [0, \max(\frac{0.02}{\alpha}, 1)] & \text{if } i_t = i_{t-1} + \lfloor v_{i,t-1} + a_{i,t-1} \rfloor \\ [0, \max(\frac{0.14}{\alpha}, 1)] & \text{if } i_t = i_{t-1} + \lfloor v_{i,t-1} + a_{i,t-1} \pm 1 \rfloor \\ [0, \max(\frac{0.68}{\alpha}, 1)] & \text{if } i_t = i_{t-1} + \lfloor v_{i,t-1} + a_{i,t-1} \pm 2 \rfloor \end{cases} \tag{22}$$

To turn this into a uACNO-MDP, we simply add a measuring cost c, and all components are defined.

[5] Since this example is only used illustratively, we will use a simple and easy-to-understand representation rather than a more robust abstraction method. For more formal methods, see e.g. Badings et al. [2023].

[6] To be more concrete, we can imagine these grid cells have length $l \approx 20cm$, in which case we have a $1.2m$ wide corridor. Further assuming times-steps of $1s$, we get speeds which increment by $0.2m/s$ up to a maximum of $1m/s$, and accelerations up to a maximum of $0.4m/s^2$.

4 Act-then-measure in uACNO-MDPs

In this section, we discuss how to translate the *act-then-measure* (ATM) heuristic from Krale et al. [2023], as discussed in Section 2.4, to the uACNO-MDP framework. As we will see later, this section requires us to additionally make assumptions about measuring actions to make computations tractable. Thus, we define the *robust act-then-measure* heuristic as follows:

The robust act-then-measure (RATM) heuristic:

1. chooses control-actions assuming that all (state) uncertainty will be resolved *in the next* states (after one time-step); and
2. chooses measuring-actions and updates beliefs assuming that all (state) uncertainty will be resolved *after the next* states (after two time-steps).

As in the standard setting, the first point of the heuristic allows us to *pick control actions independently from measuring actions*, since measurements only affect state uncertainty. Thus, our high-level strategy is given by Algorithm 6. The remainder of this section will explain in detail how to perform each step in this algorithm.

Algorithm 6 ROBUST ATM PLANNER

Pre-compute P_{RMDP} and Q_{RMDP}
Initialise $b_0(s) = \delta(s, s_0)$
while episode not completed **do**
  Pick control action a_t ▷ Equation (23)
  Pick corresponding measuring action m_t ▷ Equation (25)
  Execute $\tilde{a}_t = \langle a_t, m_t \rangle$
  Receive reward $\tilde{r}_t$ and observation o_t
  Determine next worst-case belief state b_{t+1} ▷ Equation (16)
return $\sum_t \gamma^t \tilde{r}_t$

4.1 Choosing Control Actions

Similar to the Q_{MDP} heuristic [Littman et al., 1995], the RATM heuristic means that returns of control actions can be approximated by those of the underlying RMDP:

$$Q_{\text{RATM}}(b, a) = \sum_{s \in S} b(s) Q_{\text{RMDP}}(s, a) \approx Q_{\text{R}}(b, a), \tag{23}$$

where Q_{RATM} denotes the approximate expected value when following the RATM heuristic, and Q_{RMDP} and Q_{R} denote optimal expected values for the uACNO-MDP and its underlying uMDP, respectively. As discussed in Section 2.2, uMDPs can be solved very efficiently as long as their uncertainty set is convex. Thus, our heuristic allows us to quickly compute control actions, especially in comparison to methods that fully consider partial observability.

4.2 Computing Measuring Value

Next, we need a method to pick measurement actions. For this, we define the *robust measuring value* MV_{RATM} as the difference in expected value between measuring and non-measuring actions:

$$\text{MV}_{\text{RATM}}(b, a) = Q_{\text{RATM}}(b, \langle a, 1 \rangle) - Q_{\text{RATM}}(b, \langle a, 0 \rangle). \tag{24}$$

Measuring is optimal for positive measuring values only, which yields the following measuring condition:

$$m_{\text{RATM}}(b, a) = \begin{cases} 1 & \text{if } \text{MV}_{\text{RATM}}(b, a) \geq 0; \\ 0 & \text{otherwise,} \end{cases} \tag{25}$$

To compute MV_{RATM}, we need expressions for $Q_{\text{RATM}}(b, \langle a, 1 \rangle)$ and $Q_{\text{RATM}}(b, \langle a, 0 \rangle)$. We note that the RATM heuristic means that for both, Equation (23) can be applied to all beliefs *after* the next one. Thus, the Q-value when measuring is given as:

$$Q_{\text{RATM}}(b, \langle a, 1 \rangle) = R(b, a) - c + \gamma \sum_s b(s) \Big[\sum_{s'} P_{\text{R}}(s' | s, \langle a, 1 \rangle, b) \max_a Q_{\text{RMDP}}(s', a) \Big], \tag{26}$$

with $R(b,a)=\sum_s b(s)R(s,a)$. Here, we decide the next actions for each state separately, which we can only do if we take a measurement. When not measuring, we must instead pick an optimal action considering all possible next states:

$$Q_{\text{RATM}}(b,\langle a,0\rangle) = R(b,a) + \gamma \sum_s b(s)\Big[\max_a \sum_{s'} P_{\text{R}}(s'|s,\langle a,0\rangle,b)Q_{\text{RMDP}}(s',a)\Big]. \tag{27}$$

Combining both equations, we write the robust measuring value of Equation (24) as follows:

$$\text{MV}_{\text{RATM}}(b,a) = -c + \max_{a'\in A}\gamma \sum_{s\in S} b(s)\Big[Q_{\text{RMDP}}(s,a) - \sum_{s'\in S} P_{\text{R}}(s'|s,\langle a,0\rangle,b)Q_{\text{RMDP}}(s',a')\Big]. \tag{28}$$

We notice that this equation contains only belief-independent and thus pre-computable quantities, with the exception of the transition function P_{R}. This function is equal to P_{RMDP} when measuring and otherwise given as:

$$P_{\text{R}}(s'|s,\langle a,0\rangle,b) = \max_{a_b\in A}\min_{P(\cdot|s,a)\in\mathcal{P}(s,a,\cdot)} \sum_s b(s)\Big[\sum_{s'\in S} P(s'|s,a)Q_{\text{RMDP}}(s',a_b)\Big]. \tag{29}$$

This equation includes a (non-trivial) optimization problem, which has to be computed at every step and scales with the size of the current belief. In our experiments (Section 6), we find these computations are tractable but take up the majority of the (online) runtime. With all quantities defined, we have fully described all steps in Algorithm 6. Alternatively, we may define this algorithm as a policy:

$$\pi_{\text{RATM}}(b) = \langle \max_{a\in A} Q_{\text{R}}\big(b,a\big), m_{\text{RATM}}\big(b, \max_{a\in A} Q_{\text{R}}(b,a)\big)\rangle. \tag{30}$$

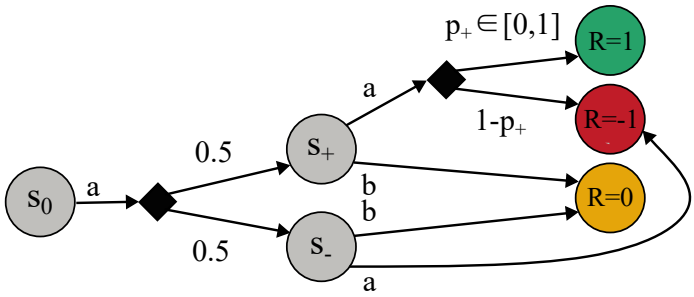


Figure 11: An example uACNO-MDP where using an optimistic measuring value in measurement-lenient policies is sub-optimal.

5 Measurement Leniency

5.1 Defining Measurement Leniency

In this section, we propose a new concept called *measurement leniency* as a solution for the (overly) conservative measuring behavior of robust policies described in Observation 2. Intuitively, measurement leniency means agents take control actions robustly but allow extra measurements to be made according to non-robust metrics (such as to optimize average expected returns)[7]. Since the cost of extra measurements is bounded and predictable, measurement leniency might give sufficient robustness guarantees for many real-life applications while allowing less conservative behavior. Formally, we define measurement leniency as follows:

Definition 9. Let $\mathcal{M}$ be an uACNO-MDP, and π_R a robust policy for $\mathcal{M}$. A *measurement-lenient* policy π_{ML} is a policy with the following properties:

1. $\forall b, \pi_R(b) = \langle a, 1\rangle \implies \pi_{\text{ML}}(b) = \langle a, 1\rangle$
2. $\forall b, \pi_R(b) = \langle a, 0\rangle \implies \pi_{\text{ML}}b) = \langle a, m\rangle$

Using this definition, we define our new objective as finding the measurement-lenient policy which has the highest expected return for some ACNO-MDP $\mathcal{M}_{\text{ML}}$. Here, $\mathcal{M}_{\text{ML}}$ can be viewed as a less conservative representation of the environment. For example, if a probability distribution over transition functions is known, $\mathcal{M}_{\text{ML}}$ could represent the most-likely behavior. We formalize optimal measurement-lenient policies as follows:

Definition 10. Let $\mathcal{M}$ be an uACNO-MDP with a robust policy π_R, and Π_{ML} the corresponding set of measurement-lenient policies. Furthermore, let $\mathcal{M}_{\text{ML}}$ be an ACNO-MDP with the same state- and action space as $\mathcal{M}$. The *optimal measurement-lenient* policy π^*_{ML} with respect to $\mathcal{M}_{\text{ML}}$ is defined as follows:

$$\pi^*_{\text{ML}} = \arg\max_{\pi_{\text{ML}} \in \Pi_{\text{ML}}} \mathbb{E}_{\pi_{\text{ML}}, \mathcal{M}_{\text{ML}}} \sum_t \gamma^t \tilde{r}_t \tag{31}$$

5.2 Computing Measurement-Lenient policies

By construction, control actions of measurement-lenient policies are equal to those in fully robust policies, meaning we can use Equation (23) to compute them. For readability, we denote these as $a_R(b_R) = \max_{a \in A} Q_R(b_R, a)$.

For measurement choices, measurement-lenient policies should maximize expected values in $\mathcal{M}_{\text{ML}}$. To compute this, we need to keep track of the current belief according to the dynamics of $\mathcal{M}_{\text{ML}}$, which we'll denote as b_{ML}. Correspondingly, we denote $b'_{\text{ML}}(b, \tilde{a})$ as the belief after taking action $\tilde{a}$ in belief b However, simply using this belief to compute generic measuring value for $\mathcal{M}_{\text{ML}}$ does not yield the correct behavior since this does not correctly take into account that future control actions will be based on $\mathcal{M}_R$ instead.

To show this, consider Figure 11, and assume we choose our measuring actions based on the best-case ACNO-MDP where $p_+ = 1$. In this environment, the optimal actions for states s_+ and s_- are b and a respectively, which using Equation (12) tells us measuring is optimal for $c \leq 0.5$. However, for a measurement-lenient policy, control actions are

[7] This behavior intuitively matches with our ATM heuristic, where control actions and measurements are already chosen separately. However, this section will give a general definition of measurement leniency with no assumptions about the original policy.

chosen according to the worst-case where $p_+ = 0$. In that environment, both s_+ and s_- have the same optimal control action a, meaning measuring has no effect. Thus, considering only the measuring value of $\mathcal{M}_{\text{ML}}$ for measurement-lenient policies would lead to sub-optimal measuring behavior.

Instead of only using the measuring value of $\mathcal{M}_{\text{ML}}$, measurement-lenient policies should thus take into account that future control actions will be chosen according to $\mathcal{M}$. To express this, we start by defining the Q-value function for following a measurement-lenient policy in $\mathcal{M}_{\text{ML}}$ as follows:

$$Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, m\rangle) = R(b_{\text{ML}}, a) - C(m) + \gamma \max_{m' \in M} Q_{\text{ML}}\Big(b'_{\text{ML}}(b_{\text{ML}}, \tilde{a}), b'_R(b_R, \tilde{a}), \langle a_R(b'_R), m'\rangle\Big) \tag{32}$$

As previously, we are interested in finding the measuring value for this policy, which is given as:

$$\text{MV}_{\text{ML}}(b_{\text{ML}}, b_R, a) = Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, 1\rangle) - Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, 0\rangle) \tag{33}$$

Writing this out explicitly, we first note that after a measurement, b'_R and b'_{ML} are always equal to the observation that has been made. Thus, the Q-value when measuring can be expressed as follows:

$$Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, 1\rangle) = R(b_{\text{ML}}, a) - c + \gamma \sum_{s \in S} b'_{\text{ML}}(s|b_{\text{ML}}, \tilde{a}) \max_{m' \in M} Q_{\text{ML}}(s, s, \langle a_R(s), m'\rangle) \tag{34}$$

Unfortunately, there's no way to simplify our expression for non-measuring actions without making further assumptions about the policy choosing control actions. One solution is to make (strong) assumptions about the robust policy used. For example, the assumption that Q_{ML} is linear in both b_{ML} and b_R, would yield the following:

$$Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, 0\rangle) = R(b_{\text{ML}}, a) + \gamma \max_{m' \in M} \sum_{s \in S} \sum_{s' \in S} b'_{\text{ML}}(s|b_{\text{ML}}, \tilde{a}) b'_R(s'|b_R, \tilde{a}) Q_{\text{ML}}(s, s', \langle a_R(s'), m'\rangle). \tag{35}$$

Even with this strong assumption, however, the number of distinct Q-values that need to be computed grows quadratically with the number of states, which makes computations expensive for larger environments. As an alternative, we thus propose to approximate our Q-values using the *optional act-then-measure heuristic* from Section 4 for $\mathcal{M}_{\text{ML}}$. We'll state this as follows:

> **Robust control approximation**: When choosing measurement-lenient measuring actions, assume all (state) uncertainty will be resolved after the next state.

Using this approximation, we can simplify our equations by replacing future Q-values with those of the fully observable variant of the environment. We denote this Q-value as Q_{CRMDP} and rewrite Equation (32) as follows:

$$Q_{\text{ML}}(b_{\text{ML}}, b_R, \langle a, m\rangle) \approx R(b_{\text{ML}}, a) - C(m) + \gamma \sum_{s' \in S} Q_{\text{CRMDP}}\Big(s', a_R\big(b'_R(b_R, \langle a, m\rangle)\big)\Big) \tag{36}$$

Interestingly, we find this expression requires no optimization since future measurements have no direct effect on the approximate Q-values used. Using this expression, we simplify our equation for measurement-lenient measuring value as follows:

$$\text{MV}_{\text{ML}}(b_C, b_R, a) \approx -C(m) + \gamma \max_{a_b \in A} \sum_{s \in S} \max_{a \in A} Q_{\text{CRMDP}}(s', a) - Q_{\text{CRMDP}}(s', a_b) \tag{37}$$

For our measurement-lenient policy, our measuring condition should require that *both* MV_{ML} and MV_R are greater than 0. Thus, the measuring condition for a measurement-lenient policy looks as follows:

$$m_{\text{ML}}(b_{CR}, b_R, a) = \begin{cases} 1 & \text{if } \text{MV}_{CR}(b_{CR}, a) \geq 0 \\ & \text{or } \text{MV}_R(b_R, a) \geq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{38}$$

5.3 Performance Regret of Measurement-Lenient Policies

One obvious downside of using measurement-lenient policies is their worst-case performance loss as compared to their robust counterparts. However, we can show that this performance loss is bounded. Intuitively, this bound follows from the fact that measurement-lenient policies only take extra measurements, and these can only decrease the total expected returns by c. Thus, the total performance loss of measurement-lenient policies is bounded by the expected number of measurements times the cost of measuring. We formally state this as follows:

Theorem 1. *Given an uACNO-MDP $\mathcal{M}$ and a set of belief states $\mathcal{B}$. Let π be any policy such that $b \in \mathcal{B} \implies \exists a : \pi(b) = \langle a, 0 \rangle$, and π' a policy defined as follows:*

$$\pi'(b) = \begin{cases} \langle a, 1 \rangle & \text{if } b \in \mathcal{B} \text{ and } \pi(b) = \langle a, 0 \rangle \\ \pi(b) & \text{otherwise} \end{cases} \tag{39}$$

Furthermore, let $N_{\mathcal{B}}(\pi, b_0)$ denote a (possibly infinite) upper limit for the expected number of visits of any belief in $b \in B$ when following a policy π from any (initial) belief b_0, defined as follows:

$$N_{\mathcal{B}}(\pi, b_0) = \begin{cases} \lceil \sum_{s \in S} b'_R(s|b_0, \pi(b_0)) N_{\mathcal{B}}(\pi, s) \rceil + 1 & \text{if } b_0 \in \mathcal{B} \text{ and } \exists a : \pi(b_0) = \langle a, 1 \rangle \\ \lceil \sum_{s \in S} b'_R(s|b_0, \pi(b_0)) N_{\mathcal{B}}(\pi, s) \rceil & \text{if } b_0 \notin \mathcal{B} \text{ and } \exists a : \pi(b_0) = \langle a, 1 \rangle \\ N_{\mathcal{B}}(\pi, b'_R(b_0, \pi(b_0))) + 1 & \text{if } b_0 \in \mathcal{B} \text{ and } \exists a : \pi(b_0) = \langle a, 0 \rangle \\ N_{\mathcal{B}}(\pi, b'_R(b_0, \pi(b_0))) & \text{if } b_0 \notin \mathcal{B} \text{ and } \exists a : \pi(b_0) = \langle a, 0 \rangle. \end{cases} \tag{40}$$

Then, the following holds:

$$\forall b : V(\pi, b) - V(\pi', b) \leq \sum_{n=0}^{N_{\mathcal{B}}(\pi', b)} \gamma^n c. \tag{41}$$

Corollary 1. *Since measurement-lenient policies can be defined from their robust counterparts using Equation* (39)*, their performance loss follows the same bound.*

Corollary 2. *For any environment where $\mathcal{B}$ or $N_{\mathcal{B}}$ are not known, we can use the following over-estimation:*

$$\forall b : V(\pi, b) - V(\pi', b) \leq \sum_{n=0}^{\infty} \gamma^n c \tag{42}$$

Proof. Let V^H and $N^H_{\mathcal{B}} \leq H$ denote V and $N_{\mathcal{B}}$ for some horizon H. Then, the following is equivalent to our theorem:

$$\forall b : \lim_{H \to \infty} V^H(\pi, b) - V^H(\pi', b) \leq \sum_{n=0}^{N^H_{\mathcal{B}}(\pi', b)} \gamma^n c \tag{43}$$

We show this equation holds via induction over H. As a base case, we note the equation trivially holds for $H = 1$, in which case the difference is simply given by c.

For $H > 1$, we first note that the equation trivially holds for beliefs where both policies pick the same action pair. Thus, we only need to prove the equation holds for beliefs $b \in \mathcal{B}$, which are the only beliefs where both policies pick different measurements. For any such belief b, denote the control action picked by both policies as a, then the difference in finite-horizon value functions is given as:

$$\begin{aligned} V^H(\pi, b) - V^H(\pi', b) &= \Big(R(b, a) + \gamma V^{H-1}(\pi, b'_R\big(b, \langle a, 0 \rangle\big) \Big) - \Big(R(b, a) - c + \gamma V^{H-1}(\pi', b'_R\big(b, \langle a, 1 \rangle\big) \Big) \\ &= c + \gamma \Big(V^{H-1}\big(\pi, b'_R(b, \langle a, 0 \rangle)\big) - V^{H-1}(\pi', b'_R\big(b, \langle a, 1 \rangle\big) \Big) \end{aligned} \tag{44}$$

To simplify this, we use the following general inequality:

$$\forall b, a, \pi : V(\pi, b'_R(b, \langle a, 0 \rangle)) \leq V(\pi, b'_R(b, \langle a, 1 \rangle)) \tag{45}$$

Using this, we replace $b'_R(b, \langle a, 0 \rangle)$ with $b'_R(b, \langle a, 1 \rangle)$ in the first value function of Equation (44). Moreover, we can now re-write our value function as a sum over states, as follows:

$$\begin{aligned} V^H(\pi, b) - V^H(\pi', b) &\leq c + \gamma \sum_{s \in S} b'_R(s'|b, \langle a, 1\rangle)\Big(V^{H-1}(\pi, s) - V^{H-1}(\pi', s)\Big) \\ &\leq c + \gamma\Big(\sum_{s \in S} b'_R(s'|b, \langle a, 1\rangle) \sum_{n=0}^{N_{\mathcal{B}}^H(\pi', s)} \gamma^n c\Big), \end{aligned} \tag{46}$$

where we obtain the second line by using our induction hypothesis. Next, we try finding an upper bound for the bracketed term. From the definition of $N_{\mathcal{B}}$, we obtain the following constraint for measuring for beliefs in $\mathcal{B}$;

$$\sum_{s \in S} b'_R(s|b, \pi(b)) N_{\mathcal{B}}^H(\pi, s) \leq N_{\mathcal{B}}^H(\pi, b) - 1. \tag{47}$$

For each state s, we notice that the contribution to this constraint grows quicker with $N_{\mathcal{B}}^H(\pi, s)$ than its contribution to the bracketed term in Equation (46). Thus, it achieves its maximum when $N_{\mathcal{B}}^H(\pi, s)$ is equal for all next states, or more precisely when $\forall s : N_{\mathcal{B}}^N(\pi', s) = N_{\mathcal{B}}^N(\pi', b) - 1$. We use this as an upper bound to Equation (46), in which case the sum over all states can be left out to give the following:

$$\begin{aligned} V^H(\pi, b) - V^H(\pi', b) &\leq c + \gamma \sum_{n=0}^{N_{\mathcal{B}}^N(\pi', b)-1} \gamma^n c \\ &\leq \sum_{n=0}^{N_{\mathcal{B}}^N(\pi', b)} \gamma^n c. \end{aligned} \tag{48}$$

This proves our induction step for H and thus the theorem. □

6 Empirical Analysis

In this section, we give an empirical analysis of our proposed methods on a number of environments. We start by describing the setup of our experiments, then show the results and highlight some key conclusions. All code, results, and plots can be found at `https://github.com/MKrale/ATM`.

The code used for the analysis was written in Python and makes use of the OpenAI Gym interface for all environments [Brockman et al., 2016], with a wrapper to add ACNO-MDP functionality. From these environments, we pre-learn the transition- and value functions using a slightly altered version of the method of Araya-López et al. [2011]. As a side effect of this implementation, our code can be run on any tabular environment implementing the Gym interface with only slight alterations. For efficiency, we also pre-compute P_{RMDP} and Q_{RMDP} using a variant of Algorithm 3, and re-use these for all algorithms.

6.1 Algorithms

In our testing, we include the following algorithms:

- ATM: the generic ATM planner of Krale et al. [2023].
- RATM: the fully robust ATM algorithm as described in Section 4.
- MLATM: the measurement-lenient variant of the ATM algorithm, as described in Section 5,

Both ATM and MLATM use a generic ACNO-MDP for planning, which we vary in our experiments. In particular, we define a *pessimistic, optimistic* and *average* ACNO-MDP, as follows:

- *Pessimistic* (pes): the worst-case environment within $\mathcal{M}$ assuming full observability, i.e. $\mathcal{M}_{\text{RMDP}}$.
- *Average* (avg): the environment where each transition probability is the average of the highest and lowest valid probability in $\mathcal{M}$. Its transition function is defined such that[8]:

$$\forall s, s', a : \mathcal{P}(s'|s,a) = [p_{\text{min}}, p_{\text{max}}] \rightarrow P(s'|s,a) = (p_{\text{min}} + p_{\text{max}})/2.$$

- *Optimistic* (opt): the best-case environment within $\mathcal{M}$ assuming full observability. We define it in a similar fashion as the robust transition function (Equations (3) and (4)), but we choose a transition function that maximizes returns instead of minimizing them:

$$P_{\text{opt}}(s'|s,a) = \arg \max_{p \in \mathcal{P}(s'|s,a)} pV_{\text{opt}}(s') - \sum_{s'' \in S/s'} P_{\text{opt}}(s''|s,a)V_{\text{opt}}(s''), \text{s.t. } \sum_{s'} P(s'|s,a) = 1$$

$$\text{with } V_{\text{opt}}(s) = \max_{a \in A} R(s,a) + \gamma \sum_{s' \in S} P_{\text{opt}}(s'|s,a)V_{\text{opt}}(s').$$

We will consider all three variants as $\mathcal{M}_{\text{ML}}$ for MLATM, giving us the algorithms MLATM-*pes*, MLATM-*avg* and MLATM-*opt*. For ATM, we only consider the first two as planning environments, giving us ATM-*pes* and ATM-*avg*.

6.2 Experiments

For our algorithms, we distinguish two goals that we want to test for:

1. **Robust performance:** performance for the *worst-case* model within the uncertainty set.
2. **General performance:** performance for *any* model within the uncertainty set.

In this section, we first run experiments on two small toy environments to isolate the behavior of the algorithms related to both goals. Next, we also consider both types of performances in the larger and more realistic drone environment.

6.2.1 Robust Performance: Considering Partial Observability

For the first research question, we run all algorithms on the *a-b* environment (Figure 9a) with $p_{\text{min}} = 0, p_{\text{max}} = 1, R_{-} = 0.8$, and different measuring costs. As mentioned in Section 3.2.1, the worst-case transition function in this

[8]This transition function is not generally valid, but we can easily verify it is for all environments used here.

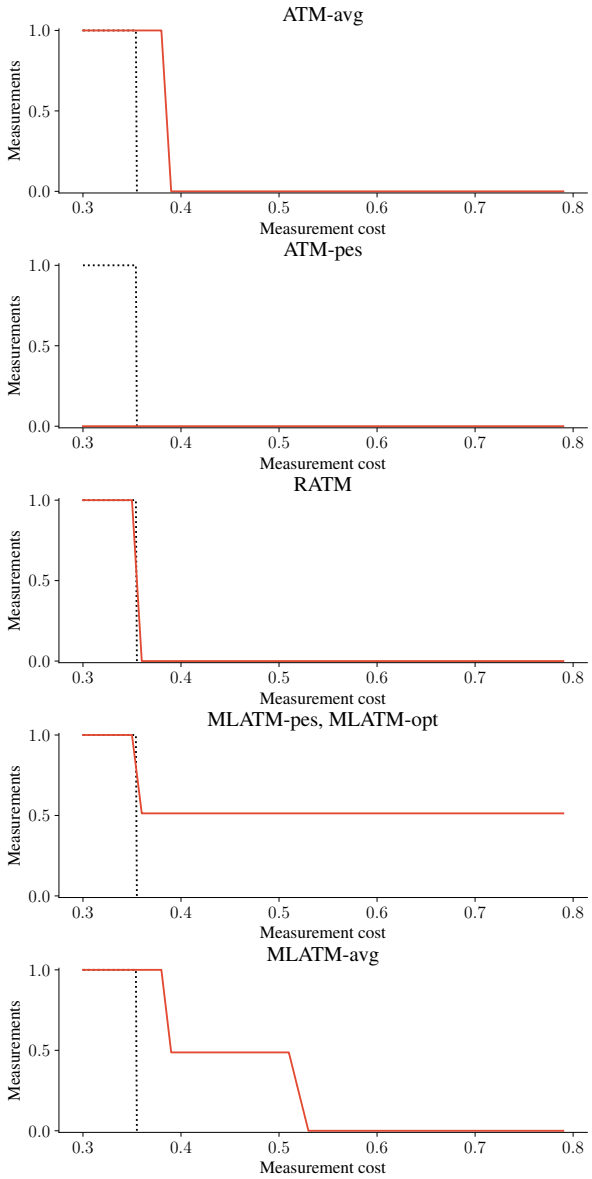


Figure 12: Measuring behaviour in the *a-b* environment against measuring cost c, for different algorithms. The dotted line shows optimal measuring behavior.

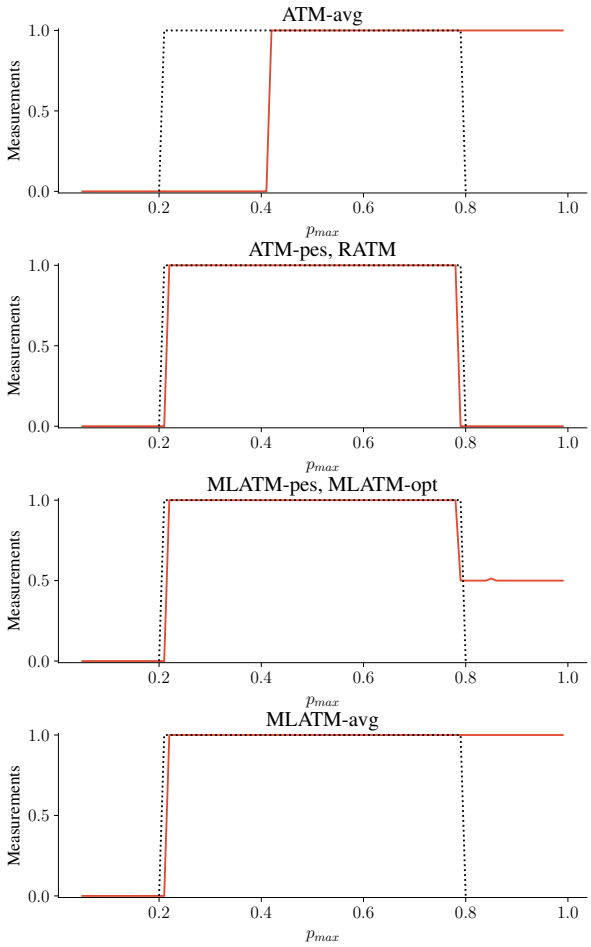


Figure 13: Measuring behaviour in the *lucky-unlucky* environment against probability p_{-}, for different algorithms. The dotted line shows optimal measuring behavior.

environment is measurement-dependent. Thus, we use it to show the difference in measuring behavior between the algorithms that do consider this dependency and those that do not.

As shown in Appendix A, we can algebraically show the measuring value (Equation (24)) is given by $\text{MV}_{\text{RATM}} = 0.3\bar{5}$.

Figure 12 shows the measuring behaviour of the different algorithms on the *a-b* environment. As expected, we find that both ATM-*avg* and ATM-*pes* are not able to correctly predict the transition function and thus show suboptimal measuring behavior. More precisely, we find ATM-*pes* never measures since it predicts the worst-case transition function is deterministic, while the ATM-*avg* uses average transition probabilities with no correlation to the worst case at all. In contrast, RATM does use the correct worst-case transition function and thus shows optimal behavior.

As expected, the measurement-lenient policies do not follow optimal measuring behavior but instead keep making measurements even after this becomes suboptimal. Surprisingly, however, the algorithms show *inconsistent* measuring behavior, i.e. measure only for 50% of episodes, at certain measuring costs. To understand why this happens, we first notice that when not measuring in the first step, the expected returns for taking action a or b in the second step are identical. In such cases, we have implemented our algorithm to randomly picks one of the two actions to calculate MV. However, in $\mathcal{M}_{\text{ML}}$, these actions may not have the same value, which might give a higher measuring value for one action than the other. In the pessimistic variant, for example, state p_- is always maximized, meaning action MLATM-*pes* will measure only if action b is chosen, and similarly for MLATM-*opt* and MLATM-*avg*. This causes sporadic measuring, which explains the behavior seen in our results. If undesirable, this behavior could be prevented by choosing control actions based on $\mathcal{M}_{\text{ML}}$ in case of ties.

From these results, we take the following findings:

> **Result 1.** In small environments where robustness depends on the measuring action, RATM outperforms non-robust algorithms, while our measurement-lenient algorithms perform extra (sub-optimal) measurements as expected.

6.2.2 General Performance: Effect of Uncertainty on Measuring

To test the measuring behavior of our algorithms, we run all algorithms on the *lucky-unlucky* environment (Figure 9b), with $R_+ = -R_- = 1, p_{\text{min}} = 0$ and $c = 0.2$. As mentioned in Section 3.2.1, this environment has the counterintuitive property that for large uncertainty, measuring becomes sub-optimal. More precisely, we can show algebraically that for our particular setup, measuring is optimal for $0.2 \leq p_{\text{min}} < 0.8$ (details are given in Appendix A). Thus, we use the environment to show the measuring behavior of our algorithms for such uncertainty sets.

Figure 9b shows the number of measurements taken by each algorithm for different (planning) values of p_{min}. We see that apart from ATM-*avg*, start measuring after $p_- = 0.2$, as is optimal, and both ATM-*pes* and RATM stop measuring at the optimal value of $p_- = 0.8$. However, all three measurement-lenient algorithms keep measuring after this point. As in the *a-b* environment, MLATM-*opt* and MLATM-*pes* measure only when the algorithm chooses the sub-optimal control action in their planning environment, while MLATM-*avg* always measures. From this, we take the following finding:

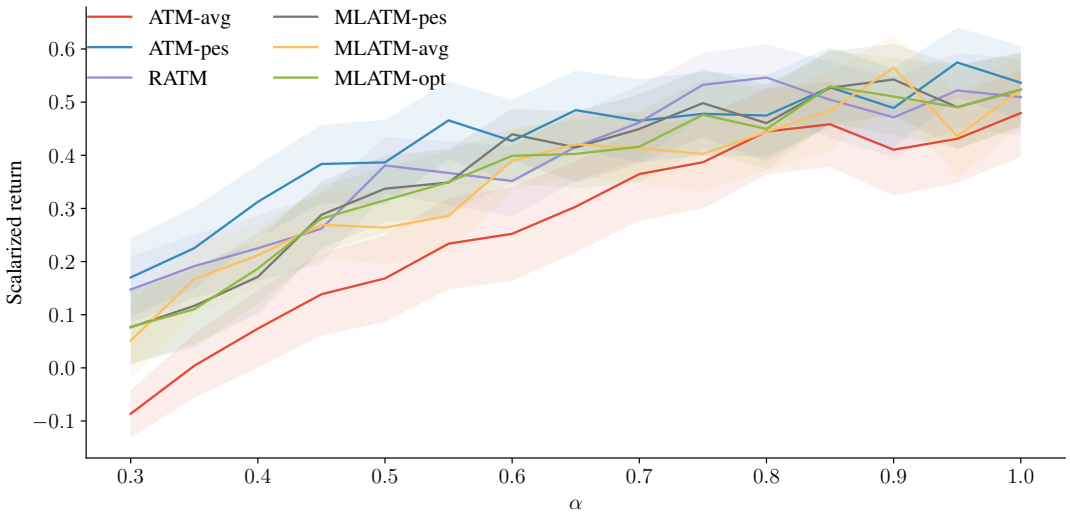
> **Result 2.** In small environments, measurement-leniency (partially) alleviates the problem of conservative measuring as described in Section 3.2.2 by making more measurements than optimal.

6.2.3 Robust Performance: Drone Environment

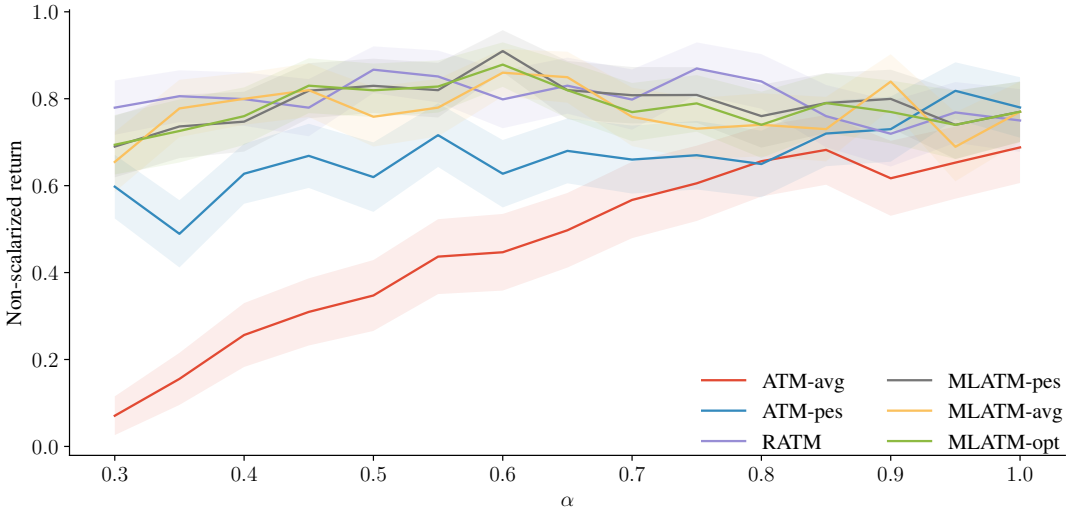
To test the robust performance on a more realistic environment, we run all algorithms on our custom drone environment, as defined in Section 3.3. We add model uncertainty with the use of the confidence MDP framework (Section 2.2), which means our uncertainty is parametrized by the confidence level α. We recall that intuitively, a confidence level α means the uncertainty set allows any probability to be at most a factor $\frac{1}{\alpha}$ larger than for the base transition function.

Although we would ideally test our algorithms on a fully adversarial environment, in practice obtaining such an environment is computationally intractable. For this reason, we instead evaluate our algorithms on the worst-case MDP variants of these environments, i.e. the environments with a worst-case transition function assuming full observability. This means that measurement-dependent worst-case transitions (such as in the *a-b* environment) never occur, and ATM-*pes* should give optimal performance.

Figure 14 shows the scalarized and non-scalarized returns of the different algorithms at different confidence levels. As expected, we see in Figure 14a that ATM-*pes* performs slightly better than all other algorithms on average. However,

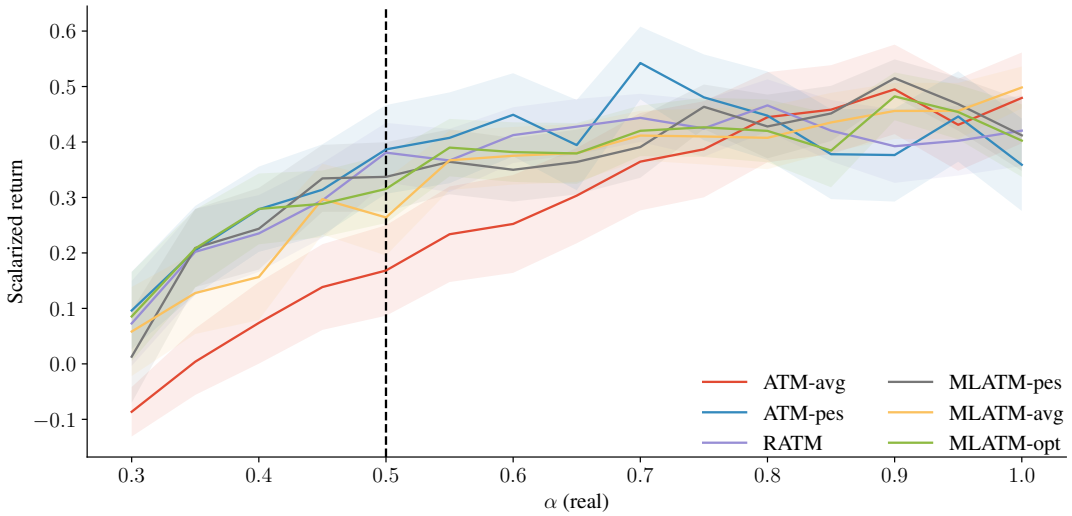


(a) Average scalarized returns.

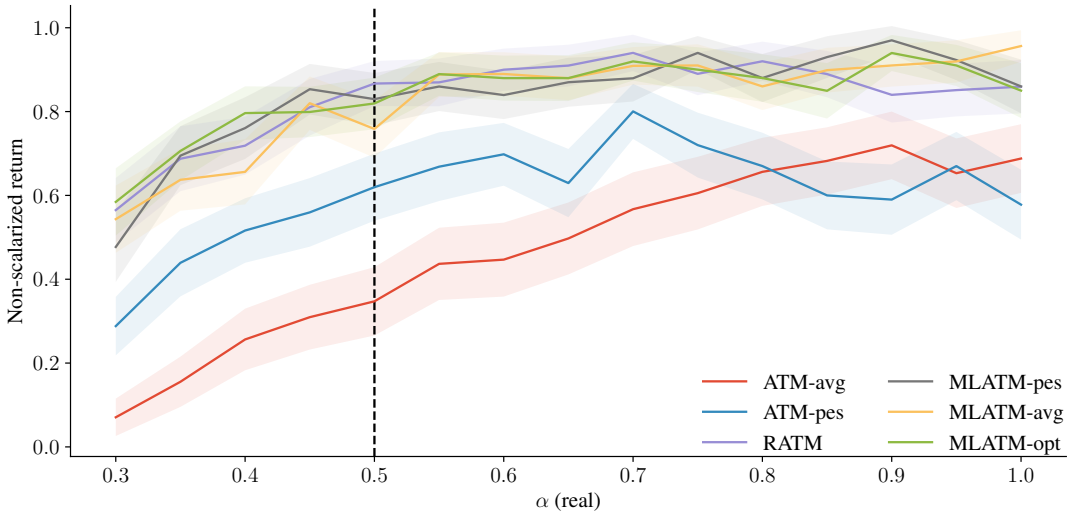


(b) Average non-scalarized returns.

Figure 14: Scalarized and non-scalarized return in the drone environment, with $c = 0.05$, for different algorithms at different uncertainties. Uncertainty is parametrized by confidence levels α and decreases left-to-right. Lines show the average of 100 runs, and shaded areas show 95% confidence intervals.



(a) Average scalarized returns.



(b) Average non-scalarized returns.

Figure 15: Scalarized and non-scalarized return in the drone environment, with $c = 0.05$, for different algorithms. The algorithms plan with uncertainty parametrized by $\alpha_p = 0.5$, but dynamics are in reality given by a worst-case MDP environment parametrized by α. Thus, real uncertainty decreases from left to right, while planned uncertainty stays constant. Lines show the average of 100 runs, and shaded areas show 95% confidence intervals.

the difference with the robust and measurement-lenient algorithms is small, particularly for larger confidence levels. In contrast, ATM-*avg* performs worse than all other algorithms for all confidence levels.

Interestingly, Figure 14b shows that the (control) robust algorithms do significantly outperform both baselines in terms of non-scalarized returns. This suggests that these algorithms take more measurements than the baselines but are able to use these measurements to reach the goal state more often. This difference is not relevant in settings where the scalarized return is the only relevant metric to optimize but could be relevant in others. We summarize this as follows:

Result 3. In large RMDP environments, the *u(c)*ATM algorithms outperform our non-robust baseline and outperform both baselines in terms of non-scalarized returns.

6.2.4 General Performance: Drone Environment

To test the generalisability on a large environment, we re-run all algorithms on the drone environment (Section 3.3) but make the algorithms plan on a different algorithm than the one they are deployed on. More precisely, the algorithms

plan on an environment with uncertainty parametrized by $\alpha_p = 0.5$, while we deploy the algorithms on the environment with $\alpha \in [0.3, 1]$. This way, we simulate the performance of our algorithm in settings where the uncertainty of a real environment is over- or underapproximated in its model.

Figure 15 shows both scalarized and non-scalarized returns of the different algorithms. As for robust performance, we find our algorithms outperform ATM-*avg*, perform on par with the ATM-*pes*, and outperform both in terms of non-scalarized returns.

Surprisingly, we do not find a significant difference between the robust and measurement-lenient algorithms, even for large mismatches between the real and planning uncertainty. We can explain this finding by noting that the worst-case dynamics of the drone environment will generally increase the probabilities of large but unlikely perturbations. Since these probabilities are small to begin with, these probabilities stay relatively small even for the (relatively large) uncertainty considered here. In other words, this particular environment does not exhibit the property outlined in Observation 2, meaning the robust algorithm makes more measurements for higher uncertainties already, and measurement-lenientness has little to no effect on measuring behavior.

Result 4. In larger RMDP environments, the findings of Result 3 extend to settings where the uncertainty is incorrectly specified. For the tested environment, measurement-lenientness does not lead to better general performance.

7 Discussion

In this section, we discuss a number of limitations of the present work:

uACNO-MDPs have strong assumptions on measurement actions: Like generic ACNO-MDPs, uACNO-MDPs assume measurements are both complete and noiseless. However, most realistic environments have either partial measurements (such as inspections on only certain parts of the system), noisy measurements (such as imperfect sensors), or both. Thus, the methods proposed in this thesis would need to be generalized to incorporate these properties before they could be used for real-life applications.

Scalability is limited by computations of P_R: As explained in Section 4, the uATM algorithm requires online computations of P_R (via Equation (18)) at each step. These computations are expensive since they require solving a non-convex optimization problem. Moreover, although the problem size does not explicitly scale with the size of the environment, it does scale with both the number of states with a non-zero occupancy probability and the number of state-action pairs, and the number of computations increases with the length of an episode. Combining these factors, this computation is the main computational bottleneck of the algorithm and prevents the algorithm to be used in environments larger than the ones used here. Future research could thus focus on finding efficient but accurate ways of approximating this function.

The testing environment is not fully adversarial: For the drone environment, we were not able to run the evaluation using the real robust transition function but instead approximated this with the robust transition function of the underlying uMDP. The reason for this is that finding the real robust transition function is intractably expensive, given it is belief-dependent. As far as we are aware, no prior work explicitly computes robust transition functions of uPOMDP, and works that compute them for policy computation [Osogami, 2015, Rasouli and Saghafian, 2018] are not tractable for the environment used here. We considered approximating the robust transition function in parallel with our algorithm and somehow incorporate this, but felt this was outside the scope of this thesis [9].

[8] Moreover, we have no reason to believe such a method would converge to the real robust transition function.

8 Conclusion

In this thesis, we define uACNO-MDPs as a framework to represent active measure environments with model uncertainty and analyze a number of properties of this framework. Next, we propose a variant on the act-then-measure heuristic of our previous work for uACNO-MDPs. We show that this heuristic allows for a tractable policy computation method and that following the heuristic has only a bounded effect on the expected returns. Next, we formally define *measurement leniency* as a way of categorizing policies that take more measurements than optimal but are otherwise fully robust, and show proof performance regret of following such policies is bounded. Lastly, we analyze both a fully robust and a number of measurement-lenient variants of our proposed method. We show their behavior follows theoretical expectations in a number of toy environments and compare performance against a number of naive baselines in a larger custom environment.

8.1 Future Research

We highlight a number of interesting directions for future research:

Generalisations to other active measure environments: As mentioned in the discussion, (*u*)ACNO-MDPs make the strong assumptions that measurements are complete and noiseless, which is often not valid in practice. Future research could define a more generic active measure framework with weaker assumptions on measurements and try to solve these in a similar fashion as (u)ACNO-MDPs. We note that the act-then-measure heuristic might prove even more useful in environments with multiple measurement actions since separating the decision-making process between control- and measuring actions would have a bigger effect.

The effect of $\mathcal{M}_{\text{ML}}$: Although we have tested a number of choices of $\mathcal{M}_{\text{ML}}$ in our experiments, the result of this choice in realistic environments is not clear. Moreover, many different choices of $\mathcal{M}_{\text{ML}}$ could be considered. In particular, one might consider constructing an environment that explicitly tries to maximize the number of additional measurements used, for example, by maximizing the measuring cost for future states. However, it is not trivial to calculate the transition function of such an environment, nor is it clear how such an environment would compare to the ones tested in this thesis.

Applying partial robustness in other contexts: The concept of measurement leniency, as defined in this thesis, is very specific to active measure environments. However, the idea of *partial robustness*, i.e. being fully robust in only a part of your decision-making process, might be applicable more broadly. For example, a robot might want to take actions related to certain expensive actuators robustly to prevent them from breaking, while actions unrelated to these parts do not require this extra care. For such cases, it could be explored if a partially robust policy could be defined that is less conservative overall but still has safety guarantees

Model-based reinforcement learning for ACNO-MDPs: In *model-based reinforcement learning* (RL), agents do not know the dynamics of the environment they are interacting with but try to learn these dynamics on the fly. In this process, agents need to take into account the uncertainty in their current model approximation. For fully observable settings, one known method is to express the environment as an interval MDP, where intervals get updated dynamically according to the observations made [Suilen et al., 2022]. Future research could explore how such a method could be used for reinforcement learning in ACNO-MDPs. This would require methods for both planning in uACNO-MDPs, as well as for updating intervals according to the history thus far. The algorithms described in this thesis could be used for the former requirement, where we particularly note that measurement-lenient policies could be used to incentivize exploratory measurements in early episodes.

Acknowledgments

To start, this thesis was a continuation of my research internship. The original topic of my internship was suggested to me by Dr. Thiago D. Simão, who also helped me a lot in understanding the general field of sequential decision-making and how to get information across to others. Both his and Dr. Nils Jansens' supervision throughout my research internship and the master thesis has been excellent, and without their help and feedback, this thesis would not exist.

Next, half a year of my master's was spent in the research group of Dr. Jana Tumova at KTH Stockholm. Even though I was only a master's student and my thesis topic did not fully align with her work, she made me feel welcomed and part of her group, which I'm very grateful for. From Janas' group, I'd particularly like to thank my office buddies Matti Vahs and Joris Verhagen, as well as Jose Manuel and Wei Wang, for all the good conversations during lunches and drinks. From RPL, I'd also like to thank Grace Hung and Alexander Sleat for setting up and inviting us to these drinks in the first place. Overall, everyone I met at KTH was lovely and made me feel at home, and I'm very grateful to everyone there for the great experience.

Lastly, the final months of my master's have been spent as a Ph.D. student in Nils Jansens' research group, where I again felt welcomed with open arms. In particular, I'd like to shout out Eline Bovy and Maris Galesloot, who both started in Nils' group at about the same time and thus went through the same process. Let's make it a fun 4 years!

References

Jesse Jiang, Ye Zhao, and Samuel Coogan. Safe learning for uncertainty-aware planning via interval MDP abstraction. *IEEE Control. Syst. Lett.*, 6:2641–2646, 2022.

Hua Xiao, Kai Yang, Xiaodong Wang, and Huai-Zong Shao. A robust MDP approach to secure power control in cognitive radio networks. In *ICC*, pages 4642–4647. IEEE, 2012.

Marnix Suilen, Thiago D. Simão, David Parker, and Nils Jansen. Robust anytime learning of Markov decision processes. In *NeurIPS*, 2022.

Haiyan Yin, Yingzhen Li, Sinno Jialin Pan, Cheng Zhang, and Sebastian Tschiatschek. Reinforcement learning with efficient active feature acquisition. *arXiv preprint arXiv:2011.00825*, 2020.

Lisandro A Jimenez-Roa, Tom Heskes, Tiedo Tinga, Hajo JA Molegraaf, and Mariëlle Stoelinga. Deterioration modeling of sewer pipes via discrete-time Markov chains: A large-scale case study in the netherlands. In *32nd European Safety and Reliability Conference, ESREL 2022: Understanding and Managing Risk and Reliability for a Sustainable Future*, pages 1299–1306, 2022.

Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. Wiley Series in Probability and Statistics. Wiley, 1994.

Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artif. Intell.*, 101(1-2):99–134, 1998.

M.J. Kochenderfer, C. Amato, G. Chowdhary, J.P. How, and H.J.D. Reynolds. *Decision Making Under Uncertainty: Theory and Application*. MIT Lincoln Laboratory Series. MIT Press, 2015. ISBN 9780262029254. URL `https://books.google.nl/books?id=hUBWCgAAQBAJ`.

Christopher J. C. H. Watkins and Peter Dayan. Technical note q-learning. *Mach. Learn.*, 8:279–292, 1992.

Arnab Nilim and Laurent El Ghaoui. Robust control of Markov decision processes with uncertain transition matrices. *Oper. Res.*, 53(5):780–798, 2005.

Wolfram Wiesemann, Daniel Kuhn, and Berç Rustem. Robust Markov decision processes. *Math. Oper. Res.*, 38(1): 153–183, 2013.

Takayuki Osogami. Robustness and risk-sensitivity in Markov decision processes. In *NIPS*, pages 233–241, 2012.

Edward J. Sondik. The optimal control of partially observable markov processes over the infinite horizon: Discounted costs. *Oper. Res.*, 26(2):282–304, 1978.

Christos H. Papadimitriou and John N. Tsitsiklis. The complexity of markov decision processes. *Math. Oper. Res.*, 12 (3):441–450, 1987.

David Silver and Joel Veness. Monte-carlo planning in large POMDPs. In *NIPS*, pages 2164–2172. Curran Associates, Inc., 2010.

Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable MDPs. In *AAAI Fall Symposia*, pages 29–37. AAAI Press, 2015.

Guy Shani, Joelle Pineau, and Robert Kaplow. A survey of point-based POMDP solvers. *Auton. Agents Multi Agent Syst.*, 27(1):1–51, 2013.

Michael L. Littman, Anthony R. Cassandra, and Leslie Pack Kaelbling. Learning policies for partially observable environments: Scaling up. In *ICML*, pages 362–370. Morgan Kaufmann, 1995.

Colin Bellinger, Rory Coles, Mark Crowley, and Isaac Tamblyn. Active measure reinforcement learning for observation cost minimization. In *Canadian Conference on AI*. Canadian Artificial Intelligence Association, 2021.

Hyunji Alex Nam, Scott L. Fleming, and Emma Brunskill. Reinforcement learning with state observation costs in action-contingent noiselessly observable Markov decision processes. In *NeurIPS*, pages 15650–15666, 2021.

Merlijn Krale, Thiago D. Simão, and Nils Jansen. Act-then-measure: Reinforcement learning for partially observable environments with active measuring. *CoRR*, abs/2303.08271, 2023.

Mohammad Rasouli and Soroush Saghafian. Robust partially observable Markov decision processes. *SSRN Electronic Journal*, 01 2018. doi: 10.2139/ssrn.3195310.

Eline Bovy. The underlying belief model of uncertain partially observable Markov decision processes. *Master Thesis*, 2023.

Takayuki Osogami. Robust partially observable Markov decision process. In Francis Bach and David Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pages 106–115, Lille, France, 07–09 Jul 2015. PMLR. URL `https://proceedings.mlr.press/v37/osogami15.html`.

Murat Cubuktepe, Nils Jansen, Sebastian Junges, Ahmadreza Marandi, Marnix Suilen, and Ufuk Topcu. Robust finite-state controllers for uncertain POMDPs. In *AAAI*, pages 11792–11800. AAAI Press, 2021.

Thom S. Badings, Licio Romao, Alessandro Abate, David Parker, Hasan A. Poonawala, Mariëlle Stoelinga, and Nils Jansen. Robust control for dynamical systems with non-gaussian noise via formal abstractions. *J. Artif. Intell. Res.*, 76:341–391, 2023.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

Mauricio Araya-López, Olivier Buffet, Vincent Thomas, and François Charpillet. Active learning of MDP models. In *EWRL*, volume 7188 of *Lecture Notes in Computer Science*, pages 42–53. Springer, 2011.

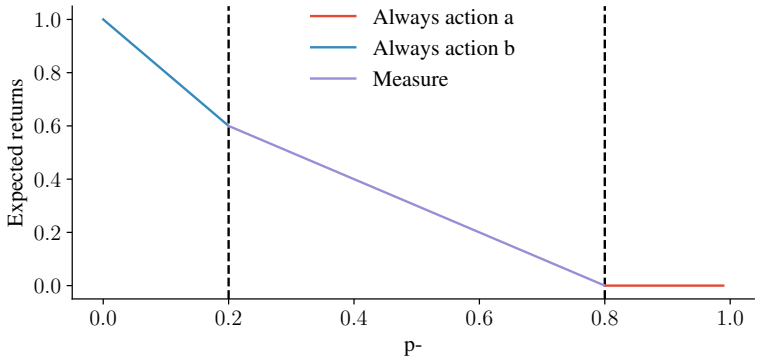


Figure 16: Expected returns for the optimal policy of the *lucky-unlucky* environment against p_-, for measuring cost $s = 0.2$. Colors represent different strategies.

A Optimal Behavior in example environments

Here, we explain and analyze optimal measuring behavior in the example environments mentioned in Section 3.

The *a-b* environment

To find the optimal policy for this environment, we first notice that in s_0, only one action is permitted, and measuring in the second step is never optimal. Thus, we only need to analyze the returns for measuring and not measuring in s_0 for each possible action in the second step. We show our analysis for $p_{\text{min}} = 0$ and $p_{\text{max}} = 1$, and note other probability intervals can be analyzed analogously.

When measuring, we notice that s_+ has a higher expected return than s_-. Thus, in this case, the worst-case transition function deterministically brings us to s_-, meaning $Q(s_0, \langle a, 0\rangle) = R_-$. When not measuring, the worst-case transition function should be such that the expected value for actions a and b is equal. These values are given by $p_- R_-$ and $1 - p_-$ respectively, meaning the worst-case transition probability is given by $p_- = 1/(1 + R_-)$ and the expected return is given by $Q(s_0, \langle a, 1\rangle) = R_-/(1 + R_-)$. Combining the two expected returns, we find measuring value is given by $\text{MV}_{\text{RATM}} = R_-(1 - \frac{1}{1+R_-}) - c$. For $R_- = 0.8$ (as used in Section 6), we find $p_- = 0.5\bar{6}$ when not measuring, giving a measuring value of $\text{MV}_{\text{RATM}} = 0.3\bar{5}$.

The *lucky-unlucky* environment

As for the *a-b* environment, we note the optimal policy for this environment never measures in the second step, meaning expected returns only depend on the measurement action in the first step and the control action in the second. Moreover, the expected returns for s_- are strictly lower than those of s_+, meaning the worst-case environment for any p is simply given by the ACNO-MDP with $p_- = p_{\text{max}}$. Depending on p_{max}, then, there are three optimal strategies for this environment. If p_{max} is sufficiently large or small, the state uncertainty is small, and measuring is not worth the cost. In these cases, we get strategies that never measure and take only action a or b, which lead to expected returns of $r = p_+ R_+ + p_- R_-$ and $r = 0$, respectively. Another strategy is to always measure when transitioning to s_+ or s_- and decide the next action based on our observation: take action a in s_+, and b in s_- This yields an expected return of $r = p_+ R_+ - c$. An optimal policy, then, simply chooses the strategy with the highest return according to p_{max} and c. For $c = 0.2$, Figure 16 shows the returns of this policy for different probabilities p_{max}.