Shared Autonomy Systems with Stochastic Operators

Clarissa Costen

St. John's College

University of Oxford

Supervised by: Prof Nick Hawes and Dr. Bruno Lacerda

August 2024

Abstract

As the autonomous capabilities of robots advance, they are becoming increasingly integrated into daily life. They serve a wide range of roles, from domestic cleaning aids and workplace assistants in warehouses to autonomous driving for cars. Despite these advancements, autonomous systems still exhibit limitations in executing high-risk or complex tasks. Shared autonomy systems address this by providing a framework where a human operator and an autonomous operator can take turns in controlling the robot.

The performance of human operators can be significantly affected by factors such as fatigue and attentiveness, and inadequate modelling of such factors can lead to unrealistic expectations and the assignment of tasks beyond the operator's capabilities, potentially endangering both the robot and surrounding individuals. Therefore, it is crucial to incorporate a rich model of human operators to ensure the selection of the most suitable operator for a given task. This thesis explores the design of shared autonomy systems with a particular emphasis on modelling the complexities associated with human operators.

The shared autonomy system is often conceptualized as a sequential decision-making problem, modelled using Markov decision processes, which assume a fully specified problem. However, this assumption is not always valid in shared autonomy systems, where human factors can affect the operator's behavior. These human factors are difficult to measure directly and must be inferred through observations. Such scenarios are characterized by Markov decision processes with epistemic uncertainty.

Markov decision processes with epistemic uncertainty use beliefs to account for hidden factors influencing transition dynamics. Previous research has either employed categorical distributions to represent these beliefs or used Bayesian particle filters to approximate continuous distributions. However, in problems with high-dimensional hidden spaces, particle-based approaches face computational limitations, as the number of particles needed to adequately cover the space increases exponentially with the number of dimensions.

There are two main contributions to this thesis: First, we propose shared autonomy frameworks for systems with uncertain operator behaviour, and we consider the following cases: We first examine a system with a single robot and multiple operators, each exhibiting a finite range of possible behaviours. The specific behaviour of an operator is unknown, so the agent maintains a categorical probability distribution over these behaviours. We investigate the exploration/exploitation trade-off and validate our framework using a computer game with simulated operators.

We also explore a multi-robot shared autonomy system, where the human operator's behaviour is represented as a continuous range. In this scenario, the human's assistance is distributed among multiple robots. The state and action spaces expand exponentially with the number of robots, making it impractical to solve the joint model. Therefore, we propose a system that explicitly rewards the reduction in variance of the posterior distribution over space of possible behaviours.

For our second contribution, we propose a subtype of Markov decision process with epistemic uncertainty, which we call a hidden parameter polynomial Markov decision process. Our formulation considers a continuous space for the latent parameters influencing the transition dynamics. We demonstrate how to maintain a closed-form belief distribution over these latent

parameters and update this belief after new observations. Traditionally, a particle-based approach would be used for belief representation. However, our method results in faster solution times and eliminates the need to tune the number of particles required to represent the belief.

Contents

Acknowledgements

I would like to thank the past and present members of GOALS, who made the group such an amazing thing to be a part of: Michael, Matthew, Alex St, Branton, An, Alex R, Carl, Michal, Roland, Alex Sc. Special thanks to Charlie and Marc, for helping me through my first year of my DPhil - I would have been truly lost without your help. And thank you to Anna and Lara, for listening to me complain, giving me advice and cheering me on. I have loved the time we spent together, and I hope there's many more.

Thank you to Nick, for giving me the opportunity (many years ago) to see what GOALS is like. You have helped me grow and I have learnt so much from you and your storytelling. The warm atmosphere that GOALS has is a testament to you as its leader. Bruno, you have been a constant support from day 1, answering every silly question I ever had. I'm so sorry for all of my writing I subjected you to, and you're the backbone of GOALS.

Thank you to Julia, Matthew and Oliver, you opened your hearts and home to me six years ago, and I'm forever grateful to you all. Thank you to the Costens - Angela, Michael, Tim, Clare, Jake and Matthew, for answering every small question with a lecture, growing curiosity in me. A big thank you to Fumie and Nick Costen. You have instilled in me academic rigour, resilience, and belief that I can tackle any problem - I could not achieved any of this without this firm foundation. Finally, thank you to Dan, for supporting throughout my DPhil, helping me make tough decisions, and giving me energy to keep going - and I am excited to see what the future brings to us.

Chapter 1

Introduction

Robotics has been widely used in industry to perform repetitive tasks, but robots have struggled to complete more dynamic tasks that require collaboration with humans. Robots are commonly used in manufacturing industries such as in the automobile and electronics industries (Grau et al., 2021). Their precision and consistency (FANUC, 2024) allows them to perform tasks that humans may struggle to do, making them ideal for factory work, such as on assembly lines. In these industries, robots operate in controlled environments and are programmed to perform a predefined sequence of tasks. However, as robots are pre-programmed, they cannot adapt to changes in the environment during tasks. This makes working in environments with humans more challenging, as humans may act unpredictably, and this requires the robots to safely adapt to these changes.

To enable robots to operate in more dynamic environments, the robots must be able to reason over how their actions may affect the human in the environment, making research in human-robot interaction (HRI) an increasingly significant research area. HRI includes applications such as self-driving cars (Wray et al., 2017), where the autonomous driver must anticipate the actions of humans in the environment and act accordingly. However, some HRI systems have unrealistic expectations on the human, such as in Duchetto et al. (2018), where an autonomous robot assumes that the human in the environment will

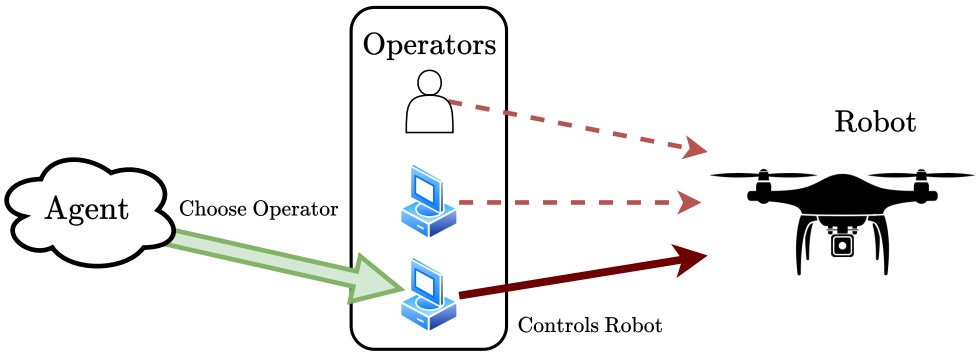


Figure 1.1: Example Shared Autonomy System

always help them and act optimally. This assumption that humans act perfectly does not hold true in the real world, where their performance is influenced by factors such as their skill level and fatigue (Dawson and Reid, 1997). Therefore, in this thesis, we focus on how to create a more realistic model of human behaviour.

In this thesis, we focus on shared autonomy systems, which are an HRI system where the control of a robot or system is shared between multiple operators. These systems can have applications in semi-autonomous vehicles, or remote-controlled robots, where the robot can be operated by humans or autonomy. Shared autonomy systems can support human operators of cars and robots by reducing their workload, by giving the control to autonomous operators in low-risk environments. The shared autonomy systems are also able to improve the performance of a system, by handing over control to the most suitable operator in any given scenario. Typically, the set of operators includes humans and autonomous operators, and the system's agent determines which operator should be in control. A diagram of the shared autonomy system is shown in Figure 1.1. The system's agent can be a human, a set of rules, or an autonomous agent.

When the agent is a human, they often also act as an operator. The human will both control the system, and determine if they would like to hand over control to an autonomous operator. An example of this is a remotely controlled robot navigating through a power

plant (Ishikawa and Suzuki, 1997). The human operator/agent monitors the autonomous operator driving the robot through a user interface, and takes over when an unexpected event occurs. However, this relies on the human to identify high-risk tasks and then complete the manual operation successfully. As mentioned by Casper and Murphy (2003), a significant limitation of such setups is the human's endurance throughout the mission; fatigue degrades the human's performance. Therefore, a rule-based or autonomous agent can reduce the human's workload by eliminating the need for constant monitoring of the autonomous operator.

Rule-based agents switch the operator when certain conditions are met, such as the duration since the human last interacted with the controller for a robot (Crandall and Goodrich, 2001). While these approaches reduce the attention needed by the human, the rule-based agent cannot guarantee that the best operator is in control in any given state. For example, the approach proposed by Crandall and Goodrich (2001) does not consider how to regain the attention of a human that has not interacted with the controller for a long period of time. This may result in the autonomous operator attempting a high-risk task and risking failure, as the time since the human has interacted with the controller has exceeded some pre-defined threshold. In contrast, an autonomous agent determines the optimal operator based on the state of the robot, and could consider the risks of the autonomous operator attempting the high-risk task.

The autonomous operator typically chooses the optimal operator by explicitly modelling the operators so that the agent has a representation of how each operator would act in any given state. This means that they can plan according to the advantages and disadvantages of each operator, thereby better utilising their skills. Planning also allows the agents to reason over longer time horizons, which is helpful for robot missions where actions taken now affect the robot's future state. To illustrate the importance of planning for longer time horizons, consider the problem shown in Figure 1.2, where there are two paths: one with a medium difficulty task at the start followed by an easy route to the

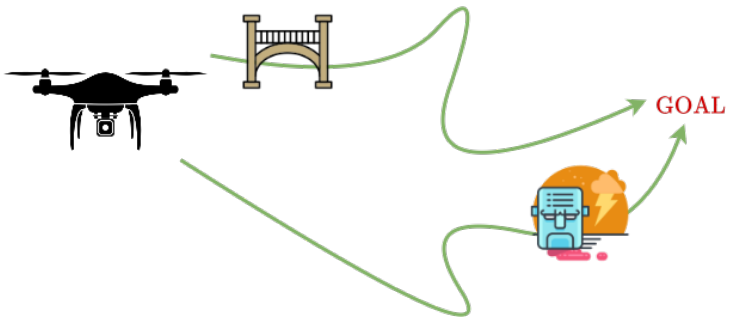


Figure 1.2: Example Sequential Decision-Making Problem

goal, and the other with an easy start but a high difficulty task at the end. If the agent acts greedily (selecting actions that only consider the rewards for the next time-step), the robot will enter the second path and then will have to attempt the high difficulty task. On the other hand, if the agent reasons over a longer time horizon, it considers the rewards the robot may gain over the entire trajectory. This would result in choosing the first path. Therefore, it is useful to frame the shared autonomy system as a sequential decision-making problem, where the agent explicitly considers how decisions made in the current time-step may affect future states.

Sequential decision-making problems are typically modelled using Markov decision processes (MDPs) (Mausam and Kolobov, 2012), and we consider frameworks based on MDPs throughout this thesis. MDPs describe how an action may cause a state to stochastically evolve, where the transition probability depends only on the current state. The autonomous agent's objective is to take the action that maximises the expected total reward over a horizon. The optimal action is determined by solving the MDP. However, there may be uncertainty about which state the robot is in or the transition dynamics may not be fully known. We collectively refer to these problems as MDPs with epistemic uncertainty. In such problems, the optimal action depends on the underlying true state or transition dynamics, which can only be inferred through observations. Therefore, the agents must consider how to balance actions taken to learn about the latent factors affecting the transition dynamics (commonly referred to as exploration) and actions taken to

maximise the expected total reward collected over the mission (referred to as exploitation).

To reason over MDPs with epistemic uncertainty, we can maintain a belief over the true value of the latent factors, such as the true state, or the true transition probability. For discrete latent factors, the belief is represented by a categorical distribution. However, for continuous latent factors, the literature typically uses a sample-based approach, where the continuous space is sampled, and a belief is maintained over these sampled values. The belief's probability distribution describes the likelihood that the sampled value is the true value governing the environment's dynamics. When there is a high dependency between the factor value and the transition probabilities, small changes in the value can result in vastly different transition dynamics. In such domains, if the true value is not among the samples, the belief distribution will never converge. Additionally, as the dimensions of the latent factor space increase, the number of samples needed to cover the space increases exponentially. To address this, this thesis explores how we can maintain a closed-form distribution for the belief over a continuous factor space.

There is extensive research aimed at improving algorithms to solve MDPs with epistemic uncertainty, focusing on finding the optimal balance between exploration and exploitation. For MDPs with smaller state spaces (such as a couple of thousand states), the optimal action can be found by solving the MDP fully. However, for MDPs with larger state spaces (e.g. domains where time is factored into the state, domains with multiple agents coordinating with each other), optimal actions are typically found using sample-based methods. These sample-based methods provide solutions that converge towards the optimal solution as the number of samples increases. They also save computational time by avoiding exploration in low-reachability regions of the state space. In this thesis, we use sample-based methods to determine the best action for MDPs with epistemic uncertainty.

1.1 Contributions

In this thesis, we aim to address the following key questions:

Q1. How can we model humans in a sequential decision-making problem for shared autonomy systems?

We approach this problem by considering the range of factors that may affect a human operator's performance. We start by modelling the case where the human's performance is influenced by a set of discrete factors (e.g., tired, not tired). For each factor, we model the human's performance using a Markov chain. The factor affecting the human may evolve during the mission, so we explicitly model the transition dynamics of the factors for two cases: when the human is controlling the robot and when they are not. This distinction is crucial because interactions with the robot may influence how the human behaves in later states. For example, a human operator may become more confident after gaining experience, resulting in better performance in subsequent tasks.

As these factors are difficult to observe directly, we model them through a latent state space. We combine these elements into a single Markovian model to describe the shared autonomy system. The optimal operator is determined by applying a sample-based solver to the model, which produces a policy that maps the current belief over the latent states to the operator for the next time step.

We also consider the case where the human's performance is affected by a set of continuous factors. We assume these factors remain static during the mission and are unknown. When a single human operates a single robot, the exploration/exploitation trade-off is straightforward to compute because any benefit from exploring the human's latent factors directly benefits the shared autonomy system.

However, this becomes more complex in the multi-robot case. In a multi-robot shared autonomy system, any exploration done by one robot could be exploited by another. If the multi-robot shared autonomy system can be described using a joint model and solved,

this trade-off is implicitly optimized. However, due to computational limits, centralized approaches are infeasible for problems with large numbers of robots. Therefore, we propose a framework for a decentralized multi-robot shared autonomy system for humans with uncertain capabilities.

Q2. How can we maintain an efficient belief distribution over the hidden attributes of an MDP while maintaining a continuous latent space?

To compute the optimal policy given the observations that an agent has made, the agent maintains a belief over the latent parameters. Previous approaches have used sample-based belief representations, where the continuous latent space is discretised using samples. However, to adequately cover the latent space, the number of samples required scales exponentially with the latent space dimensions. This can become computationally expensive and may suffer from particle deprivation. Therefore, in this thesis, we consider how we can maintain a belief distribution over a continuous latent space without using sample-based methods.

We present an MDP where the transition dynamics are described by polynomial functions of latent parameters, and we show that the belief is a closed-form conjugate distribution. We call this a hidden parameter polynomial MDP, and consider the case where the latent parameters are independent of each other (where N parameters' parameter space is defined by an N-dimensional hyper-cube), and when the latent parameters are dependent on each other (where the N parameters have a condition that they must sum to one, such that they form a N-simplex over their parameter space). For both cases, we consider how the belief can be updated, and we present the closed-form posterior belief distribution over the latent parameter space after k observations. We also consider how to approximate the posterior belief distribution to a lower order polynomial, as the order of the belief grows with the number of observations.

1.2 Thesis Outline

Chapter 2 provides a preliminary discussion on MDPs, and on how MDPs are adapted to consider epistemic uncertainty. Chapter 3 builds on the models introduced in Chapter 2 by conducting a literature review on the application of MDPs in human-robot interaction. This review highlights how MDPs have been used to model decision-making problems in shared autonomy systems.

The subsequent chapters correspond to papers I have written during my DPhil, and the papers are ordered as follows:

- Chapter 4: Clarissa Costen, Marc Rigter, Bruno Lacerda, Nick Hawes (2022). Shared Autonomy Systems with Stochastic Operators. *International Joint Conference on Artificial Intelligence*

- Chapter 5: Clarissa Costen, Marc Rigter, Bruno Lacerda, Nick Hawes (2023). Planning with Hidden Polynomial MDPs. *AAAI Conference on Artificial Intelligence*

- Chapter 6: Clarissa Costen, Bruno Lacerda, Nick Hawes (2024) Planning for Uncertain Operators with Continuous Dynamics in Shared Autonomy Systems. *Under Review at International Joint Conference on Artificial Intelligence*

- Chapter 7: Clarissa Costen, Anna Gautier, Nick Hawes, Bruno Lacerda (2024) Multi-Robot Allocation of Assistance from a Shared Uncertain Operator. *International Conference on Autonomous Agents and Multi-agent Systems*

Chapter 4 considers shared autonomy systems where human performance may evolve during a mission, and the internal states of the human operator are treated as discrete variables. This paper highlights the importance of explicitly modelling (instead of assuming full observability) the hidden nature of the human's internal state in shared autonomy systems. However, the factors affecting human performance are typically a continuous

spectrum, which cannot be fully represented by discrete variables. In scenarios with continuous hidden factors, we must consider how to maintain a belief over the hidden space. However, conventional methods (such as particle filters), struggle to represent beliefs over high-dimension continuous spaces.

Therefore, in Chapter 5, we introduce a formulation of Bayes-Adaptive MDP that allows for a closed-form belief distribution over a continuous latent parameter space. This provides computational advantages in comparison to conventional particle-based belief representations. While this paper does not focus explicitly on shared autonomy systems, the framework can be used to represent shared autonomy systems where the human performance is influenced by continuous but static factors.

However, many continuous factors affecting human performance (such as tiredness) may change during the mission. In Chapter 6, we propose a framework for shared autonomy systems where the factors affecting human performance are hidden, continuous and dynamic. Unlike the static case shown in Chapter 5, we are unable to maintain a closed-form distribution. Furthermore, the transition dynamics in the hidden state space are difficult to capture, as these transitions are inherently hidden. To address this, we discretise the hidden state space and introduce latent parameters to represent the unknown transition probabilities. This framework allows the agent to learn the hidden transition dynamics while maintaining a belief over the dynamic hidden state space.

Finally, Chapter 7 builds on the framework proposed in Chapter 5 to consider multi-robot shared autonomy systems. We consider a scenario where the performance of the human is uncertain, and the agent must efficiently allocate the human's help to the robot that will benefit most from the help, while reasoning over the human's actual capabilities. We propose a bidding system that explicitly rewards the reduction in variance over the human's capabilities, motivating the robots to learn about the human.

A summary of the four chapters and their respective topics is provided in Table 1.1. For clarity, the table is repeated at the beginning of each chapter, with the relevant paper

Table 1.1: Summary of the papers and their features

Chapters	4	5	6	7
Shared Autonomy (SA)	Yes	No	Yes	Yes
SA style	N operators controls one robot	N/A	One human and one autonomous agent controls a robot	one human and K robots, where the human can take control of any robot
Dynamic	Yes	No	Yes	No
Continuous	No	Yes	Yes	Yes
Closed-form	Yes	Yes	Yes	Yes

highlighted.

Chapter 2

Preliminaries - Modelling Epistemic Uncertainty

Mausam and Kolobov (2012) describe the key characteristics that are modelled by a Markov Decision Process (MDP) as uncertain domain dynamics and sequential decision-making. Uncertain domain dynamics can be a result of stochasticity in the environment, such as uncertainty over the outcome when a robot is driving, or random events that can occur in the environment, such as a door being open or closed. Sequential decision-making is the process of selecting a sequence of actions to maximise some objective, such as maximising the expected cumulative reward. This concept is crucial in scenarios where the action taken in the current state can significantly influence future outcomes. For example, a robot's current actions, such as navigating around an obstacle or choosing a path, can affect its ability to achieve long-term goals like efficient task completion or battery conservation. Therefore, sequential decision-making allows for strategic planning that anticipates the future states, improving the overall performance of autonomous systems.

However, MDPs assume full observability of the state, and that the transition probabilities between states are known. These assumptions may not hold in practice, as we may not be able to directly observe the state due to noisy sensors, or the transition probabili-

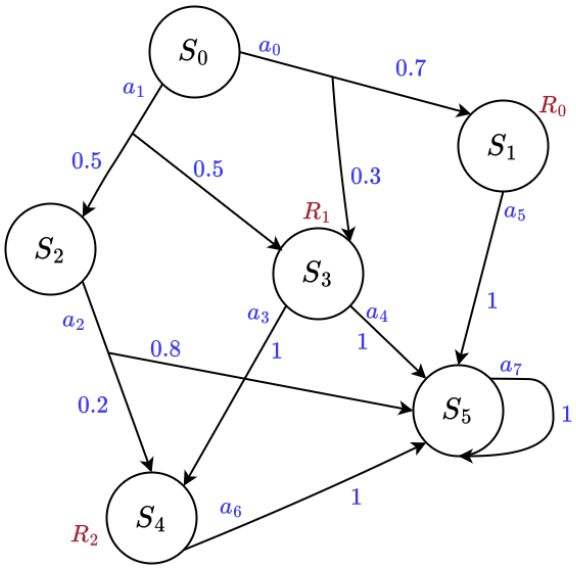


Figure 2.1: Example of MDP. The blue shows the actions, and the red shows the rewards.

ties may not be fully known due to hidden factors that affect them. Therefore, there have been extensions to the MDP model to incorporate these features, such as partially observable MDPs (POMDPs) and uncertain MDPs. In this section, we explore the different models that have been proposed to model uncertainty under the MDP framework.

2.1 Markov Decision Process

An MDP is defined by the tuple $M = \langle S, \overline{s}, A, T, R, \gamma \rangle$, where:

- S is the finite set of states;
- $\overline{s}$ is the initial state of the system;
- A is the finite set of actions;
- $T : S \times A \times S \to [0, 1]$ is the transition function, which gives the probability of transitioning from state s to state s' after taking action a;
- $R : S \times A \to \mathbb{R}$ is the reward function;

- $\gamma \in [0, 1]$ is the discount factor.

The general objective of an MDP is to maximise the expected cumulative discounted reward (Puterman, 1994). While the classical definition of an MDP considers an infinite horizon with a discount factor, the MDP can also be formulated as a stochastic shortest path (SSP) problem. In a SSP MDP, there is no discount factor, and instead there is a set of goal states $G \subset S$ that the agent is trying to reach. The reward function becomes a cost function ($C : S \times A \to \mathbb{R}$), and the agent's aim becomes to minimise the expected total cost to reach a goal state. For the purpose of simplicity, we consider MDPs with infinite horizon and reward functions in this chapter.

A deterministic and Markovian policy π maps between states and actions, $\pi : S \to A$. The value function $V^{\pi^\star}(s)$ is the expected cumulative discounted reward when starting in state s and following the optimal policy $\pi^\star$:

$$V^{\pi^\star}(s) = \max_{a \in A} \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{\pi^\star}(s') \right], \tag{2.1}$$

while the Q-value of a policy $\pi^\star$ is defined by the expected cumulative discounted reward given a state-action pair (s, a), and expressed as:

$$Q(s, a) = R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V^{\pi^\star}(s'). \tag{2.2}$$

To find the optimal policy $\pi^\star$, there are a wide range of solution methods that can be used, and this is a research field in itself. In this chapter, we focus on Value Iteration, Policy Iteration, and Monte-Carlo tree search (MCTS), as these are most commonly used in the literature we reference in Chapter 3.

2.1.1 Value Iteration and Policy Iteration

Value iteration computes the optimal value function within an ϵ bound for all states in the MDP by iteratively updating the value function using the previous iteration's value functions and the Bellman backup (Mausam and Kolobov, 2012). To implement value iteration, we begin by initiating a value function for all states: $V^0(s) = 0 \ \forall s \in S$. After $n-1$ updates, the nth iteration of the value function is updated by:

$$V^n(s) = \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^{n-1}(s') \right] \ \forall \ s \in S. \tag{2.3}$$

Once the value functions have converged within an ϵ bound (i.e. $|V^n(s) - V^{n-1}(s)| \leq \epsilon \ \forall s \in S$), the optimal policy can be found through selecting the action that maximises the expected cumulative reward for each state, following Equation 2.4.

$$\pi^\star(s) = \operatorname*{argmax}_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T(s,a,s') V^n(s') \right] \tag{2.4}$$

Another common method for solving MDPs is Policy Iteration, which iteratively updates the policy instead of the value function. Starting with an initial random policy π_0, we iteratively update the policy by evaluating the Q-values. Then, we update the policy by selecting the action that maximises the Q-value for each state. The policy is updated until it converges to the optimal policy. Both value iteration and policy iteration require us to consider the full set of states to evaluate the policy. However, unlike value iteration, convergence is defined when the optimal action for a state does not change between iterations. Furthermore, the policy iteration method cannot guarantee convergence if the horizon is infinite Puterman (1994).

2.1.2 UCT Algorithm

Both value iteration and policy iteration methods compute the policy for the entire state space, which can be computationally expensive for large state spaces and unnecessary for states with low reachability probability. In problems with large state spaces, approaches that approximate the optimal policy though a sample-based approach can be used, such as the UCT algorithm (Kocsis and Szepesvári, 2006). The UCT algorithm is a Monte-Carlo Tree Search (MCTS) algorithm that uses a tree structure to store the value functions of states and actions. The tree is made up of decision nodes, which are defined by the state of the environment, and chance nodes, which are defined by the actions the agent can take given the state they are in.

The UCT algorithm uses the Upper Confidence Bound (UCB) to select the actions in the tree. There are 4 stages in a UCT algorithm: selection, expansion, rollout, and back-propagation.

Selection - At the start of a trial, the algorithm begins at the root node of the tree, and selects actions based on:

$$a = \underset{a \in A}{\operatorname{argmax}} \left[Q(s,a) + c\sqrt{\frac{\log N(s)}{N(s,a)}} \right], \tag{2.5}$$

where $N(s)$ is the number of times state s has been visited, $N(s,a)$ is the number of times action a has been selected in state s, and c is a constant that controls the exploration-exploitation trade-off. c must be tuned to get the correct balance - too high and the algorithm will not value the rewards from taking actions, too low and the algorithm will not explore enough to find the optimal policy. For a given state/decision node, if $\forall a \in A\ N(s,a) = 0$, a random action is chosen, and if $\exists a \in A\ N(s,a) = 0$, an unexplored action is randomly chosen. After the action is chosen, this action is simulated, and a new state is visited. If this new state has been visited before, we repeat the action selection

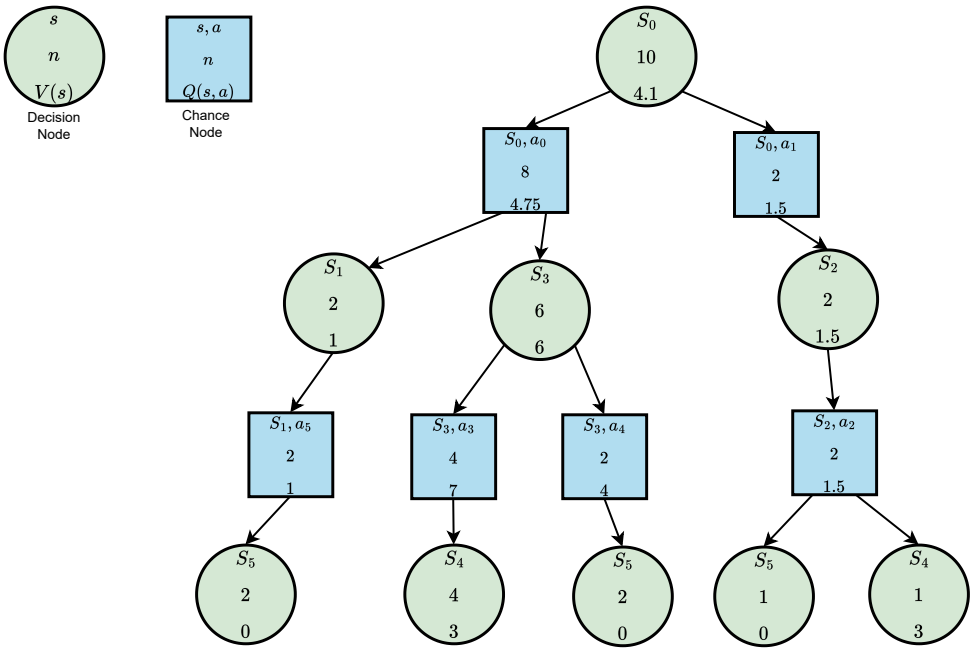


Figure 2.2: Monte-Carlo Tree Search

process from the new state's decision node.

Expansion - If the selected action has not been visited before, a child "chance" node is created for the action, and the action is simulated to get a new state and reward. The new state is added to the tree as a leaf node.

Rollout - If the new state is not a terminal state, a rollout policy is used to simulate the rest of the trial until a terminal state is reached, or a maximum number of steps has been reached. The rollout policy can be a random policy, or a heuristic policy that is used to guide the agent.

Back-propagation - The reward from the trial is back-propagated up the tree to update the value functions of the states and actions that were visited in the trial. The counts of the states and actions are also updated.

As the number of trials increases, the value functions of the states and Q-value of the state-action pairs converge towards their true values. Then, the optimal policy can be found by selecting the action that maximises the value function at each decision node. The UCT algorithm is particularly useful for problems with large state spaces, as it does

not require the full state space to be evaluated. Therefore, states that are unlikely to be reached are not fully explored, reducing the computational load.

However, unlike value iteration and policy iteration, there is not a clear convergence criteria for the algorithm, and there are no formal guarantees that the optimal policy has been reached. The UCT algorithm is commonly used as an online planning algorithm, where the agent will plan every time they take an action. This is useful for scenarios where unlikely events occur, and an offline planner may not have fully explored the tree in low probability regions.

2.2 Partially Observable MDPs

The MDP model assumes that the agent can fully observe the state of the environment, but this may not always be true. A classical example of partial observability is the tiger problem. In this problem, there are two doors, one leading to a tiger, and the other leading to a reward (Cass et al., 1995). The agent must choose a door to open, but if they choose the door with the tiger, there will be a large cost. As the agent begins with no prior knowledge of where the tiger is, the agent has the option to listen for the tiger for a small cost. This action provides an observation about the likely location of the tiger. However, the listening action is imperfect, and subject to the following stochastic outcomes:

- If the tiger is behind the left door, there is a probability of 0.85 that the agent will hear the tiger on the left and a probability of 0.15 that the agent will hear the tiger on the right.

- If the tiger is behind the right door, there is a probability of 0.85 that the agent will hear the tiger on the right and a probability of 0.15 that the agent will hear the tiger on the left.

The goal for the agent is to maximise the expected cumulative reward by choosing a sequence of actions (listening or opening a door) that both considers the immediate costs (listening) and the future potential outcomes (opening a door). For the agent to be able to reason over the possible locations of the tiger, the agent must maintain a **belief** over the possible states of the environment, which we define as $b(s) \in [0, 1]$ where $\sum_{s \in S} b(s) = 1$. The agent will start with an initial belief, $\bar{b}$, to represent their initial knowledge of the environment, and this will be updated as the agents make observations.

Formally, we define a partially observable MDP (POMDP) as the tuple $P = \langle S, \bar{b}, A, O, T, R, \Omega, \gamma \rangle$, where:

- S, A, T, R, γ is the same as presented for an MDP in Section 2.1;
- $\bar{b}$ is the initial belief of the system;
- Ω is the finite set of observations;
- $O : S \times A \times \Omega \to [0, 1]$ is the probabilistic observation function, which gives the probability of observing o after the agent took action a which resulted in state s';

For the tiger problem, the state space describes the door the tiger is behind, $s \in [\textit{left-door}, \textit{right-door}]$. The action the human can take is either open a door, or listen to make an observation: $a \in [\textit{open-left}, \textit{open-right}, \textit{listen}]$, and when they make an observation, it is either $o \in [\text{left}, \text{right}]$. There is a small cost associated with listening ($R(s, \textit{listen}) = -1 \ \forall s \in S$), while there is a large cost associated with opening the door the tiger is behind ($R(\textit{left-door}, \textit{open-left}) = R(\textit{right-door}, \textit{open-right}) = -100$). The agent will maintain a belief $b(s)$ over which door the tiger is behind, and they will use the belief to choose which door to open.

If an agent has a belief $b_t(s)$ over the states of the environment at time t, and takes action $a \in A$ and receives an observation of $\omega \in \Omega$, the belief can be updated using Bayes'

rule:

$$b_{t+1}(s') = \frac{O(s', a, \omega) \sum_{s \in S} T(s, a, s') b_t(s)}{\sum_{s' \in S} O(s', a, \omega) \sum_{s \in S} T(s, a, s') b_t(s)}. \tag{2.6}$$

The agent will use Equation 2.6 every time they listen out for the tiger to update their belief based on their observations. However, the agent must balance the benefits of listening (learning more information) with the cost associated with listening. To do this, we must get a policy mapping the current belief to the optimal action. We examine the methods used to get a policy for a POMDP.

2.2.1 Belief MDP

A POMDP can be framed as an MDP by factoring in the belief into the state space, resulting in a belief MDP (BMDP) (Kaelbling et al., 1998). The belief MDP is defined by the tuple $M = \langle B, \bar{b}, A, T, R \rangle$, where:

- B is the set of belief states;
- $\bar{b}$ is the initial belief of the system;
- A is the finite set of actions;
- $\tau : B \times A \times B \to [0, 1]$ is the transition function, where the probabilities are:

$$\tau(b, a, b') = \sum_{\omega \in \Omega} P(b'|a, b, \omega) \sum_{s' \in S} \sum_{s \in S} O(s', a, \omega) \cdot b(s) \cdot T(s, a, s') \tag{2.7}$$

  where $P(b'|a, b, \omega) = 1$ if:

$$b'(s') = \frac{O(s', a, \omega) \sum_{s \in S} b(s) \cdot T(s, a, s')}{\sum_{s'' \in S} \sum_{s \in S} O(s'', a, \omega) \cdot b(s) \cdot T(s, a, s'')}, \tag{2.8}$$

  and $P(b'|a, b, \omega) = 0$ otherwise;

- $R : B \times A \to \mathbb{R}$ is the reward function, and found by:

$$R(b, a) = \sum_{s \in S} b(s) \cdot R(s, a). \tag{2.9}$$

For shorter time horizons, the POMDP can be solved by generating the belief MDP, and solved using the methods described in Section 2.1. This is feasible as, as the set of possible beliefs the agent could have is finite and limited. However, at larger time horizons, the belief MDP becomes computationally expensive to generate, as we must calculate all the possible beliefs and transition probabilities the agent may experience, and these grow exponentially with the horizon.

2.2.2 Point-Based Value Iteration

When we build a belief MDP, we only consider the reachable belief states from the initial belief, but in practice, the belief space is a continuous space, defined over an $|S|$-dimensional simplex. For a given policy π, the value function when an agent has a belief defined by:

$$V^{\pi}(b) = \sum_{s \in S} b(s) \cdot V^{\pi}(s). \tag{2.10}$$

This can be expressed by defining a vector $\boldsymbol{\alpha}_{\pi} = \langle V^{\pi}(s_1), \ldots, V^{\pi}(s_n) \rangle$ as the value functions of the states, and thus the value function for a belief vector $\boldsymbol{b} = \langle b(s_1), \ldots, b(s_n) \rangle$ when following policy π can be written as $V^{\pi}(b) = \boldsymbol{b} \cdot \boldsymbol{\alpha}_{\pi}$. The alpha vector can therefore be represented as a plane in the belief space. For a finite state and action space, there is a finite set of possible policies Π the agent can follow. The optimal policy $\pi^{\star}$ for a given belief b is the policy that maximises the value function for the belief:

$$\pi^{\star}(b) = \underset{\pi \in \Pi}{\operatorname{argmax}}\, \boldsymbol{b} \cdot \boldsymbol{\alpha}_{\pi}. \tag{2.11}$$

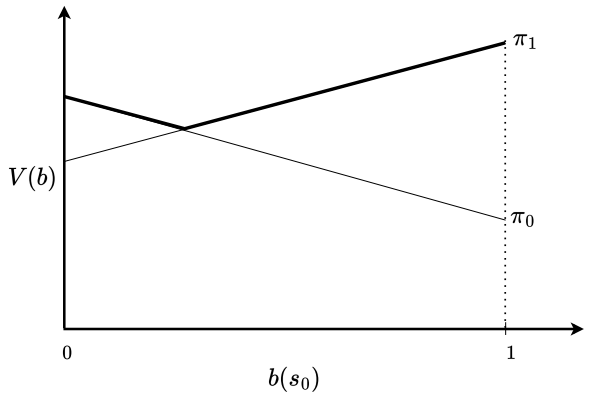


Figure 2.3: Alpha vectors for two policies, π_0 and π_1 in the belief space. The optimal policy for a given belief is shown in bold

The alpha vectors and the optimal policy can be visualised in the belief space for a two state problem, as shown in Fig. 2.3, where the bold upper line represents the optimal policy.

However, in practice the policy space is too large to compute the value function for all possible policies, especially for larger state spaces. Pineau et al. (2003) proposed the point-based value iteration (PBVI) algorithm to approximate the value function for the belief space by sampling a set of sparse belief points, and updating the optimal alpha vectors for the belief points. The alpha vectors are then used to approximate the value function for the belief space by using the piece-wise linear interpolation of the alpha vectors.

The PBVI algorithm starts by initialising an alpha-vector for each belief point. The value function is updated using the Bellman backup operator, where the nth iteration of the value function is updated by:

$$V_n(b) = \max_{a \in A} \left[\sum_{s \in S} R(s,a)b(s) + \gamma \sum_{o \in O} \max_{\alpha_{n-1} \in V_{n-1}} \sum_{s \in S} \sum_{s' \in S} T(s,a,s')\Omega(s',a,o)\alpha_{n-1}(s')b(s) \right]. \quad (2.12)$$

The updates are repeated until the value function converges within an ϵ bound.

The set of beliefs that are used to approximate the value function are expanded iteratively with the value iteration step, and we start with the set of beliefs $B = \overline{b}$. For each belief $b \in B$, we generate the set of successor beliefs $b_{a_0}, b_{a_1}, \ldots$ by simulating the next observation for each action $a \in A$, and using Equation 2.6. Any new belief b_{a_i} that is in B is discarded, and any new belief less than a distance l away from B is discarded too. Of the remaining new beliefs, the furthest away from B is added to B. Therefore, B at most doubles in size per iteration.

2.2.3 POMCP

While PBVI reduces the number of belief points required to compute a policy in comparison to the belief MDP, the algorithm still requires the belief to be explicitly calculated using Equation 2.6. This is computationally expensive for large state spaces, where PBVI requires a large number of belief points to adequately cover the belief space. Silver and Veness (2010) proposed a Monte Carlo Tree Search (MCTS) algorithm for POMDPs, called POMCP, which approximates the belief state using sampling methods. They alter the UCT algorithm for MDPs (as shown in Section 2.1.2) to work with POMDPs, by augmenting the decision nodes to store the history of action observations instead of the state.

Like the UCT algorithm, POMCP builds a search tree through trials. A decision node of the search tree is defined by the history of actions and observations made to reach the node, and the chance nodes are defined by the actions that can be taken from the decision node. At the start of a trial, the algorithm starts at the root node of the tree, and samples a state s from the initial belief, $\overline{b}$. An action a is chosen according to the UCB formula, as shown in Equation 2.5, and a new state s' and an observation o is generated by sampling $T(s, a, \star)$ and $\Omega(s', a, \star)$ respectively. The action a and observation o is used to generate a history $h = ao$, and this history defines the leaf node of the tree. This

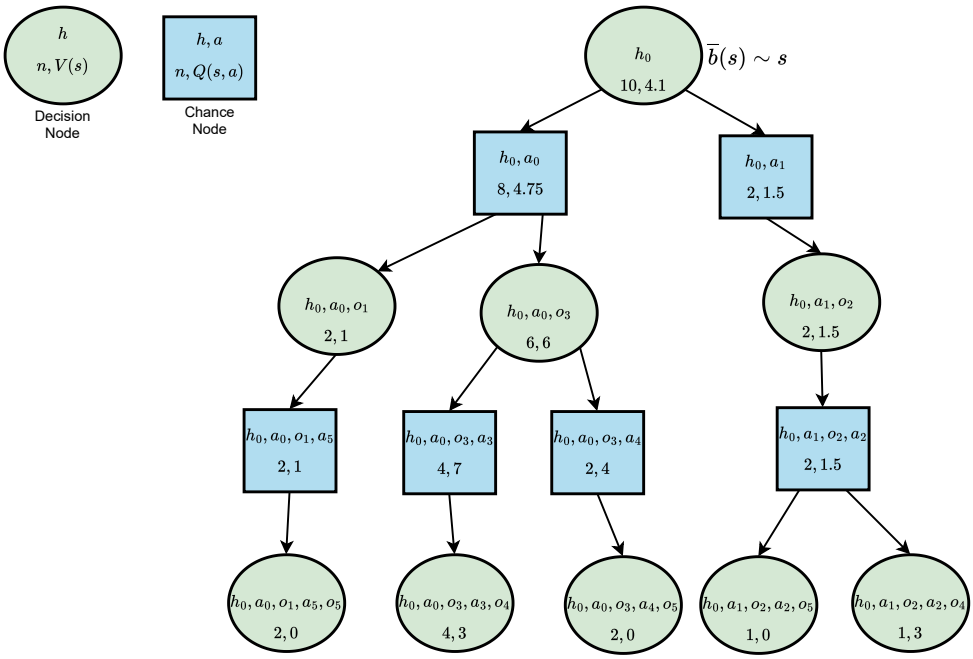


Figure 2.4: POMCP algorithm

simulation process is repeated until an unexplored leaf node is reached, at which point we follow the expansion, roll-out and back-propagation steps as described in Section 2.1.2.

A decision node in the POMCP algorithm stores the number of times the node has been visited, the value function of the node, and the state of the simulation when the node was visited. As the number of trials approaches infinity, the distribution of the states that visited the decision node with history h will converge to the belief distribution of $b(s|h)$. Therefore, POMCP approximates the belief of a given history through sampling, removing the need to explicitly calculate the belief as in PBVI. Once the POMCP algorithm has converged, the agent executes the action that maximises the value function in the environment. This results in an observation, and the agent can then update the belief using Equation 2.6, and the process is repeated.

2.3 Uncertain MDPs

In POMDPs, we considered problems where the state is not directly observable, and we must maintain a belief over the true state. However, there may be problems where the transition probabilities between states are not fully known.

We define an MDP where the transition probabilities are not fully known as an uncertain MDP (Rigter et al., 2021), represented by the tuple $\langle S, \overline{s}, A, \Theta, P_0(\theta), T_{\theta \in \Theta}, R, \gamma \rangle$, where:

- $S, \overline{s}, A, R, \gamma$ are as defined in Section 2.1;
- $\theta \in \Theta$ is the set of latent parameters that govern the transition probabilities, where θ is a vector over the latent parameter space;
- $P_0(\theta)$ is the initial belief over the latent parameters;
- $T_{\theta \in \Theta} : S \times A \times S \to [0, 1]$ is the transition function, which gives the probability of transitioning from state s to state s' after taking action a given the latent parameter θ.

There are multiple approaches to solving this problem. One approach is to plan under the assumption that the 'world' is an adversarial one - it will always change to give you the worst value function in any given state (Suilen et al., 2024). This robust approach is useful in domains where failure has a high cost, such as injury to humans in the domain. However, for less critical problems, the robust approach can result in overly conservative policies. Therefore, in this thesis, we focus on methods in which we maximise the expected rewards.

To mamixise the expected rewards collected over a run, the agent must maintain a belief over the latent parameters, and this belief is updated using the history of actions and observations, $h_t = s_0 a_0 \cdots a_{t-1} s_t$, using Bayes' rule. Similar to the Belief MDP built for

a POMDP in Section 2.2.1, we can build a Bayes-Adaptive MDP (BAMDP) that factors the history (and thus implicitly the belief) into the state space for an uncertain MDP with a finite horizon. A BAMDP (Duff, 2002) is defined by the tuple $\langle S^+, \overline{s^+}, A, T^+, R^+, \gamma \rangle$, where:

- $S^+ = S \times \mathcal{H}$, where $\mathcal{H}$ is the set of all histories;
- $\overline{s^+} = (\overline{s}, \overline{h})$, where $\overline{h}$ is the empty history;
- $T^+((s, h_t), a, (s', h_t a s')) = \int_{\theta \in \Theta} T_\theta(s, a, s') P_t(\theta) d\theta$ is the transition function, where $P_t(\theta) \propto P(h_t|\theta) P_0(\theta)$;
- $R^+(s^+, a) = R(s, a)$ is the reward function;
- A, γ are as defined in Section 2.1.

We typically observe two main implementations of BAMDPs; firstly where there is no prior or knowledge over any transition probabilities, and secondly where the agent is in a set of known MDPs, and must learn which MDP they are in through observations.

For the first case, the latent parameter space is $\theta_{s,a,s'} \in \Theta \, \forall s \in S \, \forall a \in A \, \forall s' \in S$, where each transition probability is a latent parameter. The latent parameters with the same initial state and action must follow $\sum_{s' \in S} \theta_{s,a,s'} = 1$, as the transition probabilities must add to 1. Each transition function is therefore parameterised over a n-dimensional simplex, where n is the number of possible states the agent can reach through the transition. The agent must maintain a belief over the latent parameters, and the belief for a single transition function can be represented as a Dirichlet distribution,

$$Dir(\theta_{s,a,s_1}, \theta_{s,a,s_2}, \cdots \theta_{s,a,s_n}; \alpha_{s,a,s_1} + 1, \alpha_{s,a,s_2} + 1, \cdots \alpha_{s,a,s_n} + 1), \tag{2.13}$$

where α_{s,a,s_i} is the number of times the agent has observed the transition from state s to state s_i after taking action a. Therefore, after the agent has made an observation, the

agent can update their belief over the transition probabilities by updating the count in the Dirichlet distribution.

For the second case where the agent is reasoning over a set of known MDPs, the latent parameter values are the set of MDPs the agent is reasoning over. To maintain a belief, the agent typically samples a set of MDPs from the initial belief, and approximates the belief by updating the probabilities of being in the sampled MDP based on the observations made. This problem becomes equivalent to a POMDP with a hidden fixed state, as the true MDP the agent is in can be considered the hidden state, and the observable states in the uncertain MDP are equivalent to the observations in the POMDP. However, this method does not scale well as the space of possible MDPs the agent could be in expands, and may not converge if there are no samples in the region of the true MDP.

Solving a BAMDP with a goal to maximise the expected cumulative reward is equivalent to solving the original uncertain MDP, with the exploration required to learn the latent parameters optimally balanced with the exploitation of the known parameters to gain rewards. The optimal policy for a BAMDP is the one that maximises the value function defined as:

$$\begin{aligned} V^{+}(s, h_t) &= \max_{a \in A} \left[R(s,a) + \gamma \sum_{s' \in S} T^{+}((s,h_t), a, (s', h_t \cdot as'))V^{+}(s', h_t \cdot as') \right]. \\ &= \max_{a \in A} \left[R(s,a) + \gamma \int_{\theta \in \Theta} \sum_{s' \in S} P(\theta | h_t) T_{\theta}(s,a,s') V^{+}(s', h_t \cdot as') \right]. \end{aligned} \tag{2.14}$$

However, much like the belief MDP, solving the BAMDP via value iteration becomes computationally expensive as the history space grows exponentially with the number of actions taken. Therefore, BAMDPs are typically solved with a sampling-based solver, such as the BAMCP algorithm (Guez et al., 2012).

2.3.1 BAMCP algorithm

The BAMCP algorithm is an extension of the POMCP algorithm, where the algorithm approximates the belief over the latent parameters using sampling methods. The leaf nodes in the BAMCP algorithm are defined by $h = s_0 a_0, \cdots a_{n-1} s_n$, the history of actions and states visited to reach the node, and the child nodes are defined by the actions that can be taken from the leaf node. In the POMCP algorithm, a trial starts at the root node by sampling a state s from the initial belief over the hidden states. Similarly in the BAMCP algorithm, a trial starts at the root node by sampling $\theta_i \sim P_0(\theta)$, a latent parameter that defines the transition functions. The algorithm simulates a step in the tree by sampling the transition function T_{θ_i}, and the history is updated with the action and state visited. The new history corresponds to a new decision node, and the transition dynamic is governed by θ_i This is repeated until an unexplored leaf node is reached, at which point the expansion, rollout and back-propagation steps are followed as described in Section 2.1.2. After a trial, a new θ_j is sampled from $P_0(\Theta)$, and the tree is explored using transition dynamics governed by θ_j. Once the algorithm has reached time-out, the agent executes the action with the highest Q-value. This results in a new state, and the prior distribution is updated using the methods outlined in Section 2.3.

Chapter 3

Modelling Humans for Sequential Decision Making

In this thesis, we are interested in modelling human behaviour in shared autonomy systems, which require capturing human interactions with robots or autonomous agents. These models allows agents to reason over human actions, improving collaboration and the overall performance of the system.

However, simple representations of human behaviour may result in the agent taking sub-optimal decisions. For example, Rosenthal and Veloso (2012) considers a robot navigating around an office, and needs to find a human to help them. The agent uses a prior probability distribution to predict the location of a human, and navigates the robot to the room with the highest likelihood of finding a human. However, as the human's current location is not explicitly modelled, the robot cannot reason over the human's true location upon seeing an empty room. Therefore, we focus on exploring how humans in shared autonomy systems can be modelled as sequential decision-making problem to achieve more adaptive planning.

When an agent is planning for a sequential decision-making problem in a domain with humans, the human's behaviour is explicitly considered and modelled. Chapter 2 explored

the different models used to describe sequential-decision making problems, and in this chapter, we consider how these models have been applied to describe human behaviour. We also explore the literature on how these models have been extended to describe and learn about complex human behaviours, such as how their behaviour changes over time.

3.1 MDPs

Markov Decision Processes (MDPs, outlined in section 2.1) can be used to describe the stochastic behaviour of a human in a domain. This could be how humans act in a shared autonomy system controlling a vehicle (Basich et al., 2020; van Wyk et al., 2020), or how they act in an environment shared with a robot (Junges et al., 2018; McGhan et al., 2015). Basich et al. (2020) aims to reduce the interventions required by the human by learning through interactions to anticipate the human's decisions. While they use the action space of an MDP to model the range of possible decisions the human can make, they assume that the human operator is perfect, and their transition dynamics do not model the possibility that the human may end in a fail state.

3.1.1 Modelling imperfect humans

However, the human may not always be perfect, and in certain domains, the agent must be able to reason over the probability of a human failing to complete a task. For example, a human operator of a robot may not be aware of obstacles in the environment, while the autonomy can better use signals from sensors to avoid them. Cubuktepe et al. (2021) uses the stochastic transition functions in an MDP to describe a human unaware of the obstacles when controlling a wheelchair, and the agent combines the human's action choice with the autonomy's action choice to reach the goal. This method of implementing shared autonomy is called blended autonomy. The authors use blended autonomy to maintain safety while the human is controlling the wheelchair, and this approach allows the human

to direct the wheelchair while avoiding collisions with obstacles.

Dahiya et al. (2022) also explicitly considers the probability of the human failing at a task, and uses this to plan how to allocate human help in a multi-agent setting. The authors consider a set of K semi-autonomous robots completing navigation tasks, where each robot has to complete N tasks to reach its goal. The human can help by taking over control of the robot and completing a task. The shared autonomy system for the kth robot is described by an MDP, where the state state space is defined by the task number the robot is on, $n^k \in [1, N]$, and the robot's internal state, $\{normal, fault\}$, which signifies if the robot is stuck. The action space determines whether the robot is operating autonomously ($a^k = 0$) or teleoperated ($a^k = 1$), and the action choice changes the probability of the robot transitioning into a *fault* state. The agent for the kth robot must reason over the probability of completing tasks successfully both for the robot and the human before asking for human help, by bidding to the centralised controller which determines which robot the human teleoperates. These frameworks allow the agent to make more realistic assumptions about a human's abilities, and allocate the human help to tasks that will be the most beneficial to the system.

MDPs modelling human behaviour are sometimes built using data collected from human demonstrations, where the transition probabilities are learnt through a frequentist approach. Charles et al. (2018) used 85 hours of humans playing a game to build an MDP that described the human's performance in game play, and used this in a shared autonomy system for a game that could be played by a human or an autonomous controller. However, the authors noted that the human player's behaviour could be altered due to a range of factors, and a single model would not be sufficient to accurately describe the human's behaviour.

3.1.2 Using Multiple Models to Describe Humans

Wu et al. (2017); Feng et al. (2016); Fu and Topcu (2016); Mahmud et al. (2023) attempt to address this by using multiple MDPs to describe a human's behaviour, where one MDP corresponds to one setting of the factors that affects their performance. Feng et al. (2016) considers the effect of tiredness on human performance in a shared autonomy system, in a domain of a drone completing a series of tasks, where some are high-risk. The human is modelled by two MDPs, one where the human is *normal* (M_{norm}), and another for when a human is *tired* (M_{tired}). Both MDPs' state space defines the environment the drone is in,$s_E \in S_E$, and the transition functions describe the respective probabilities of successfully completing a task when the human is *normal* and *tired*. The MDPs describing the human has one action per state, a = Human. M_{norm} and M_{tired} are combined to create a single MDP M_{human} describing the human, where the state space is defined by the environment the drone is in (S_E), the human's internal state {*normal*, *tired*}, and a counter k that keeps track of the number of actions the human has done. Feng et al. (2016) assumes that the human will become tired after a threshold of T steps, and this is formulated by a deterministic transition function:

$$\text{If } k = T \text{ then } T_{human}((s_E, normal, k), a, (s_E, tired, k+1)) = 1 \ \forall \ s_E \in S_E. \tag{3.1}$$

This human MDP is joined with an MDP describing the autonomous controller, and the agent plans over the joint MDP using value iteration. The optimal policy for this problem will ration the human help over the mission, and thus able to request a *normal* human to complete high-risk tasks that appear later in the mission.

Wu et al. (2017) also use multiple MDPs to model a human's behaviour, where each MDP corresponds to a different fatigue and trust level. The human and a robot work along-side each other on a production line, and the agent must consider how the human's trust in the robot affects their performance. The human's internal trust level is modelled

using an MDP M_t, where the state space S_t is the finite set of human trust levels, and the action space $A^t = A^r \cup \{repair\}$ is the union of the set of actions the robot can do, and a repair action that is done by humans when the robot is in a fail state. The transition functions define the probability of the human's trust level altering given the robot's actions. Similarly, the human's fatigue process is modelled using an MDP M_f, where the state space S_f is the set of fatigue levels, and the action space is $A^f = A^r \cup A^h \cup \{repair\}$ the union of the actions the human and the robot could do. The human models M_t and M_f are factored into the task model M_w and the robot model M_r to created a single MDP $\mathcal{M}$ that describes the entire production line, where a state is the tuple $s = (s^w, s^r, s^t, s^w)$, the state of the task, the state of the robot and the human's trust and fatigue levels. Therefore, by solving $\mathcal{M}$, the agent explicitly considers how to improve the human's trust in the robot, while also reasoning over the tasks the robot should complete to reduce the human's workload and the expected cumulative reward from completing the tasks successfully.

While these models allow the agents to reason over the effect the robots may have on the human, and plan accordingly, they still assume that the agent can directly observe the factors that affect the human's performance. However, factors such as fatigue, trust and skill level are difficult to measure. In the next section, we consider how partially observable models have been used to describe the latent factors that affect human performance.

3.2 POMDPs

3.2.1 Hidden Physical attributes

Partial observability can be used to express attributes of a human that cannot be directly induced, such as their physical and behavioural (e.g. skill level) attributes. An agent may need to maintain a belief over the physical attributes of a human due to sensor

inaccuracies, such as the lidar used to measure their location, or when they are partially obscured by other objects in the domain (Schörner et al., 2019).

Wray et al. (2017) use the state space of a POMDP to describe the physical location and speed of people in the environment for an autonomous car. The people in the environment can be pedestrians, or other cars driven by people. Each type of human (e.g. pedestrian, vehicle) is modelled by a POMDP, where the state space is their location relative to the car, while their action space is $\{stop, go\}$, and the autonomous car can make observations of the human through its sensors. These POMDPs are solved offline to get a policy prior to execution, by solving the belief MDP through value iteration. Then, during a mission, any number of pedestrians and vehicles may be instantiated depending on the domain. The agent maintains a belief over the humans, and assumes they act according to their offline policy. By maintaining a belief over the humans' location, the autonomy can consider the range of possible locations the human could be in, and avoid collisions with the human.

Jean-Baptiste et al. (2015) use a POMDP to maintain a belief over an impaired human's history-of-actions for an assistive agent. They demonstrate their system for a stroke survivor making a cup of tea, where the agent must prompt the human if they make an error during the tea-making process. However, the camera sensors used to track the human have a probability of not correctly registering the human's actions. The human's progress through the task is modelled using a POMDP, by keeping track of their actions, and whether they have made an error. The state space is $S = A_u \times S_d$, where A_u is the current action the user is taking (E.g. "Fill kettle", "Boil water", "Add teabag"), and $s_d \in S_d$ is the history of the actions taken by the human. The action space $A = A_u \times E$ is a cross product of A_u, the set of actions the user can take, and $E = \{True, False\}$, the error space. The error and the action determines what prompts the agent should present, where if $E = True$, the agent will inform the user that they have made a mistake, and the action they should take instead. The history-of-actions allows the agent to keep track

whether the human is completing a task in the correct order, and if not, gives prompts to help the human complete the task. To solve the POMDP, the authors used a Monte-Carlo based solver.

These papers demonstrate how real-world constraints in hardware can be overcome through modelling, and as sensors improve, the uncertainty over the human's physical attributes can be reduced. However, a human's internal attributes (such as tiredness, skill or trust level) are not directly observable, and cannot be inferred through better sensors. We can model the human's internal attributes through the state space of a POMDP, and we can infer the human's internal state through observations of the human's actions.

3.2.2 Hidden Human Goals

POMDPs are a common way to model uncertainty over the human's goal (Fern et al., 2014; Javdani et al., 2015; Karami et al., 2009). In both Fern et al. (2014) and Javdani et al. (2015), the robot is trying to assist the human in the environment, but has no initial knowledge over the human's goal. In Fern et al. (2014), the agent updates their belief over the set of possible goals by observing how the human interacts with the environment. The environment is described using an MDP, where W is the state space, and there are two action spaces - A for the action space of the human, and A' for the action space of the assisting agent. There is uncertainty over the goal g the human has chosen, where $g \in G$ and $G \subset W$. The environment MDP and this uncertainty is combined to create a POMDP for the agent to reason over, which Fern et al. (2014) name the *assistant POMDP*. In the assistant POMDP, the state space is $W \times G$, the action space is A', and the observations the agent can make is $W \times A$, because the environment is directly observable, and the agent can observe the action the human took.

The authors note that solving a POMDP directly can be computationally expensive in large state spaces. They instead propose a heuristic method, where prior to execution

they compute the value functions for the MDPs where the goals are known, so a MDP with goal g would have a Q-value of $Q_g(w, a)$. During the mission, the belief $b(g)$ over the goal is maintained using Bayes' rule. Then, the assistant robot takes the action that maximises:

$$Q(w, a) = \underset{a \in A'}{\operatorname{argmax}} \sum_{g \in G} Q_g(w, a) \cdot b(g). \tag{3.2}$$

This method will act greedily, as unlike the POMCP algorithm, the value of exploration will not be implicitly incorporated into the value function.

Javdani et al. (2015) also considers uncertainty over the human's goal, but in a shared autonomy context. The authors consider a robotic arm controlled by a human and autonomy, where there is uncertainty over which object the human wants to pick up. The inputs from the human directing the robot is used to update the belief, and the autonomy takes over the control as the uncertainty decreases. In Fern et al. (2014) and Javdani et al. (2015), the goals of the humans are static throughout execution, and the agent does not consider how the human's goal may change over time. However, POMDPs can also be used to describe dynamic and hidden attributes of the humans. In the next section, we explore how POMDPs are used to consider the change in the human's internal state.

3.2.3 Hidden Human Behaviours

Knowledge about the human internal states, such as their attentiveness, can be critical in missions such as handing over control to a human in a semi-autonomous vehicle (Wray et al., 2016). The human's internal state can be observed indirectly through interactions, such as asking them to do a task or by using eye-tracking software (Petric and Kovacic, 2020; Lam and Sastry, 2014). The state can also be altered by the agent (Wray et al., 2016), such as through sending audible alerts, which has a probability of improving their attentiveness.

Wray et al. (2016) models the transfer of control as a POMDP, where the human's internal state is unknown, and the agent has a limited time window to transfer control safely. The state space is defined by the current human state (distracted, aware, etc.), the time remaining for the transfer, the previous action taken by the agent, and the time since the last action. The action space for the agent is a set of messages $\mathcal{M}$ that the agent can use to get the attention of the human, from a blinker light to an alarm noise. However, constant prompts can limit human satisfaction with the system, and may result in the human dis-engaging or ignoring the alerts. This is described by the reward function, which negatively rewards sending out messages that are more disruptive, but heavily penalises ending in a mission abortion, and even more heavily a mission failure. The agent can make observations and update their belief over the human's internal state through noisy sensors, such as face or eye-trackers. The agent aims to minimise the total cost collected during the transfer of control, and uses point-based value iteration (Section 2.2.2) to solve the POMDP. By solving the POMDP, the agent can optimally balance the exploration required to learn the human's internal state, and the exploitation of the information to take actions to reduce the overall cost.

3.3 Learning over time

As the agent interacts with humans, the information collected can be used to improve the models the agent has of the human, and of the autonomy. (Nanavati et al., 2021) learns about the intrinsic willingness of humans to help the robot in an office environment. This willingness is defined by a continuous bounded parameter, and the robot maintains a belief over this value for each person during the mission. By modelling the problem as a BAMDP (Section 2.3), the robot can learn the humans' willingness to help, while also maximising the reward collected. The human's current busyness b, the frequency of the robot asking for help ρ_{ask}, and whether they helped in the previous timestep defines

the state space for the BAMDP. Human UID's helpfulness is described by the latent parameter h_{UID}, and the latent helpfulness, their busyness, and the frequency of the robot asking are used to generate the transition probabilities of the human helping:

$$P(\textit{human-helps}|h_{UID}, b, \rho_{ask}) = \frac{1}{\exp\left(-(h_{UID+c_1+c_2 \cdot b+c_3 \cdot b \cdot \rho_{ask}})\right)} \tag{3.3}$$

where c_1, c_2 and c_3 are learnt prior to execution. This BAMDP framework is useful in domains where people around the robot change every mission, so for each mission the agent can learn the latent parameters for each new person.

Nanavati et al. (2021) solves the BAMDP by simplifying the model into a stationary POMDP. This is done by discretising the latent parameter space into 20 states, and framing the latent parameter as a hidden state factor that does not change during a mission. Then, the POMDP is solved by using the POMCP alogrithm (Section 2.2.3). However, this approach reduces the set of possible human behaviours the agent can observe, and thus risks the agent not being able to accurately capture how a person is behaving.

When a robot is doing the same task repeatedly, the robot can use the experiences gained over the missions to improve their performance. Rigter et al. (2020) considers a shared autonomy system as a multi-arm bandit problem. At each episode, the agent must choose from a finite set of arms, where each arm has an unknown reward distribution and each arm corresponds to an action in an MDP. In the shared autonomy system, the set of arms/actions is comprised of the set of autonomous drivers the agent can use, and the human driver. The agent aims to minimise the overall cost collected over all episodes, where costs are accumulated by failing to complete a task, and by asking the human to take over. The outcomes of an episode can be used in two ways: to update the performance predictions for the autonomous agent used, and, if the outcome is successful, the episode is stored into a buffer that is used to train an autonomous controller using behavioural cloning. Therefore, the demonstrations done by the human improves the performance

of the autonomous agent, and thus the amount of intervention required by a human will decrease over time. However, the agent assumes that the human will always be successful, and given there is only a single model for the human, this formulation does not allow for us to consider the range of possible human behaviours.

Chapter 4

Shared Autonomy Systems with Stochastic Operators

In this chapter, we examine how to model a shared autonomy system where there are multiple operators and we are uncertain on how they may behave. We first consider how to model such an operator. Given the different factors that may affect an operator's performance, we explicitly consider different *modes* they could be in. For each possible mode, we use a Markov chain (MC) to model their expected performance given the environment they are in. We also consider how the operator may transition between different modes depending on whether they are participating in the system or not. We use MCs to model how the modes change, where the set of possible modes defines the state space. These models are combined to describe how the operator acts when they are active in the shared autonomy system, versus when they are not.

Secondly, we must consider how to choose the optimal operator in a shared autonomy system given the uncertainty over how the operators may behave. We model the shared autonomy system as a mixed observablility MDP (MOMDP), where the state space is split into hidden and observable state spaces. The observable state space represents the environmental factors affecting the system that are directly observable, while the hidden

state space is used to represent the mode each operator is in. The action space allows the agent to choose which operator should control the shared autonomy system in the next timestep, and the transition function is informed by the MC modelling the operator. The agent maintains a discrete belief distribution over the hidden state space, and uses the current belief to choose the optimal operator. We solve the MOMDP by adapting the POMCP algorithm. In the POMCP algorithm, the decision node is defined by the history of actions and observations made to reach the current state. We extend this to include the current observable state, so a decision node is defined by the sequence of observable state, action and observations made to reach the current state.

The framework proposed can be used for any number of operators, and we do not specify if an operator is a human or autonomous. Therefore, we can also extend this model to consider how unobservable factors can affect autonomous operators too. For example, in a domain where there is shared autonomy over a remotely controlled robot, weather conditions, such as slipperiness, may affect the performance of an autonomous operator. The shared autonomy system can also be implemented with a domain of computer games too, as shown in the experiments section of the paper.

Table 4.1: Summary of the papers and their features

Chapters	4	5	6	7
SA	Yes	No	Yes	Yes
SA style	N operators controls one robot	N/A	One human and one autonomous agent controls a robot	one human and K robots, where the human can take control of any robot
Dynamic	Yes	No	Yes	No
Continuous	No	Yes	Yes	Yes
Closed-form	Yes	Yes	Yes	Yes

Shared Autonomy Systems with Stochastic Operator Models

Clarissa Costen , **Marc Rigter** , **Bruno Lacerda** and **Nick Hawes**
Oxford Robotics Institute, University of Oxford, UK
{clarissa, mrigter, bruno, nickh}@robots.ox.ac.uk

Abstract

We consider shared autonomy systems where multiple operators (AI and human), can interact with the environment, e.g. by controlling a robot. The decision problem for the shared autonomy system is to select which operator takes control at each timestep, such that a reward specifying the intended system behaviour is maximised. The performance of the human operator is influenced by unobserved factors, such as fatigue or skill level. Therefore, the system must reason over stochastic models of operator performance. We present a framework for stochastic operators in shared autonomy systems (SO-SAS), where we represent operators using rich, partially observable models. We formalise SO-SAS as a mixed-observability Markov decision process, where environment states are fully observable and internal operator states are hidden. We test SO-SAS on a simulated domain and a computer game, empirically showing it results in better performance compared to traditional formulations of shared autonomy systems.

1 Introduction

As automation and robots become prevalent in our lives, designing systems that enable co-operation between humans and autonomy becomes necessary. In this work, we are interested in a shared autonomy (SA) setting, where a set of *operators* interact with the environment by taking control of a system (such as a robot) in order to contribute towards achieving a common goal. In the most general case the set of available operators may include both human teleoperators and autonomous controllers. An SA framework is responsible for deciding which operator should be in charge in order to manage progress towards the goal.

To enable this decision making, current SA approaches tend to use simplified models of their operators. For example, in Basich *et al.*, [2020] an autonomous controller drives a car until safety is compromised, at which point the human operator must take over. The authors make the assumption that the human driver is always perfect, therefore the SA system can always switch to the human operator to maintain safety. Cubuktepe *et al.*, [2021] consider a scenario where a human operator and an autonomous system share control of a wheelchair. The human in this setting is not fully aware of the risks present in the environment, but the autonomous system is assumed to have full visibility of the risks, and thus is able to always take appropriate action choices. Their model of the human operator makes the simplifying assumption that human performance does not evolve over time, e.g. due to learning or tiredness. This static human operator assumption is relaxed in Feng *et al.*, [2016], who present a SA system in which the human performance level might change due to underlying factors (fatigue). However, their model is simplified by the assumption that the onset of fatigue is deterministic, with the human transitioning from an alert to a tired state after a fixed number of timesteps. As each person has an individual level of tolerance to fatigue, it is unrealistic to assume that all operators are fatigued after the same number of steps.

In this paper, we relax the simplifying assumptions described above by introducing *performance profiles* that model the behaviour of the operators available to an SA system. These performance profiles depend on a set of partially observable state factors that evolve stochastically as the SA system acts. This leads us to the stochastic operators in shared autonomy systems (SO-SAS) model, which maintains a belief over the performance profile state for each operator. The observations gained from the interactions of the different operators with the environment are used to update the belief on which performance profile state each operator is in. To avoid the need for defining complicated observation functions, belief updates in SO-SAS are performed by observing environment state transitions.

Our main contribution is the SO-SAS framework, a novel formalisation of an SA system as a mixed-observability Markov decision process (MOMDP). To the best of our knowledge, this is the first SA framework that considers operator performance to be stochastic, dynamic and not directly observable by the SA decision-making framework. We conduct an empirical evaluation of SO-SAS on a simulated domain, and on a computer game, with comparison to existing modelling approaches. We show that our formulation results in better performance in comparison to using the simpler models (e.g. assuming a perfect human or static performance) typically used in previous work.

2 Related Work

When an autonomous system works in the same space as humans [Wu *et al.*, 2017], or shares control over a robot with a human [Feng *et al.*, 2016], they must be able to reason over potential human interaction. These human-aware autonomous systems typically determine the next action by attempting to maximise the rewards collected over the future timesteps. The rewards are defined by the objective of the system, such as completing tasks successfully [Wu *et al.*, 2017; Feng *et al.*, 2016] or negative rewards associated with the cost of human intervention [Rigter *et al.*, 2020; Basich *et al.*, 2020].

Approaches motivated by reducing the amount of human intervention needed typically do not explicitly model the human. Instead, they consider humans as a resource, such as asking them to do tasks the robot cannot do [Rosenthal and Veloso, 2012], or as an example to replicate [Duchetto *et al.*, 2018]. This means that the autonomous system often models the human help as an action choice, which guarantees a deterministic transition to the target state. However, this makes an implicit assumption that humans act perfectly. An SA system that does not consider the weaknesses of the human operator is vulnerable to making sub-optimal choices where the autonomy outperforms the human operator.

The autonomous systems that attempt to create a realistic model of the humans in the environment must consider the probabilities associated with humans making mistakes [Wu *et al.*, 2017; Jean-Baptiste *et al.*, 2015; Charles *et al.*, 2018]. These systems use Markov chains (MCs) to model the stochastic behaviour of humans, such as their likelihood to miss a piece of litter in a picking exercise [Junges *et al.*, 2018]. [Wu *et al.*, 2017] uses a MC to model the impact of the human's fatigue level and their trust in the autonomy on their productivity. This allows them to describe the changes to human behaviour due to internal factors. However, these approaches assume the human state to be fully observable, which is unrealistic in many scenarios. Partially observable Markov decision processes (POMDPs) are used to model humans when the state of the human is not fully observable. Examples of the hidden state space of the human can be their location and speed [Wray *et al.*, 2017], or the goal they are aiming to fulfill [Jean-Baptiste *et al.*, 2015]. We also consider the human state to be hidden, but do so in an SA context. [Javdani *et al.*, 2015] and [Fern *et al.*, 2014] use a POMDP to describe a SA system, where the human's goal is hidden. The human operator's behaviour is dependent on the unknown goal. The agent updates their belief over the true goal by observing the human's actions, and attempts to assist the human. While these papers consider how hidden human variables affect their behaviour in a SA system, unlike our system, they do not consider dynamic hidden human variables that change value during a run, or how the system is affected by human error.

We are interested in modelling the uncertainty in the behaviour of the operator in an SA system. [Wray *et al.*, 2016] considers a similar problem to ours in the domain of semi-autonomous cars. However, they only consider the uncertainty in the engagement level of the human operator during the transfer of control, and otherwise assume humans are perfect drivers. The agent in their problem uses actions such as beeping to gain observations of the human operator. In contrast, SO-SAS infers the state of the human operator by observing the outcomes of their interactions with the environment. This avoids the need to design complicated observation functions for the sensors tracking the human operators.

3 Preliminaries

We propose a new formulation based on a mixed-observability Markov decision processes (MOMDPs) [Ong *et al.*, 2010] to formalise SO-SAS. In a MOMDP, there is an explicit partition of the observable and hidden parts of the state, which allows for better scalability than the corresponding POMDP model.

Definition 1 (MOMDP)**.** A MOMDP is defined by the tuple $\langle S_o, S_h, s_0, b_0, A, \Omega, T, O, R, \gamma \rangle$, where S_o is the set of the observable state factors; S_h is the set of the hidden state factors - the state space of the MOMDP is $S = S_o \times S_h$, and a single state is defined by (s_o, s_h); s_0 is the initial observable state; b_0 is the initial belief distribution of the agent. A belief distribution $b_i(s_h)$ gives the probability of the agent being in a hidden state s_h at time step i; A is the finite set of actions; Ω is the finite set of observations; $T : S \times A \times S \to [0, 1]$ is the transition function, where $T((s_o, s_h), a, (s'_o, s'_h))$ gives the probability of moving to state (s'_o, s'_h) given that action a was taken in state (s_o, s_h); $O : S \times A \times \Omega \to [0, 1]$ is the observation function, where $O((s'_o, s'_h), a, o)$ gives the probability of observing o given action a was taken and the MOMDP transitioned to state (s'_o, s'_h); $R : S \times A \to \mathbb{R}$ is the reward function; and γ is the discount factor.

An MDP is a MOMDP where $S_h = \emptyset$, $\Omega = \emptyset$, and the belief b_0 and observation function O are undefined. We denote MDPs as tuples $\langle S_o, s_0, A, T, R, \gamma \rangle$. An MC is an MDP without action choices. We denote the transition function for an MC as $T : S \times S \to [0, 1]$.

We consider the problem of, given a MOMDP, finding the policy π^* that maximises the infinite-horizon discounted cumulative reward.

4 SA System Modelled With MOMDPs

4.1 Problem Formulation

We consider a set of environment states S^E representing some process we wish to control, along with an initial environment state s^E_0. There are N operators in the SA system that can take control of the process and change its state. The SA agent chooses, at each timestep, which operator will take control of the process. Given a reward function $R^E : S^E \times S^E \to \mathbb{R}$ over environment state transitions, the agent aims to maximise the infinite-horizon discounted cumulative reward collected.

4.2 General Approach

The agent is required to predict which model best describes each operator's behaviour, which evolves stochastically. The agent starts with the initial environment state and an initial belief over how the operators will behave. It then uses this

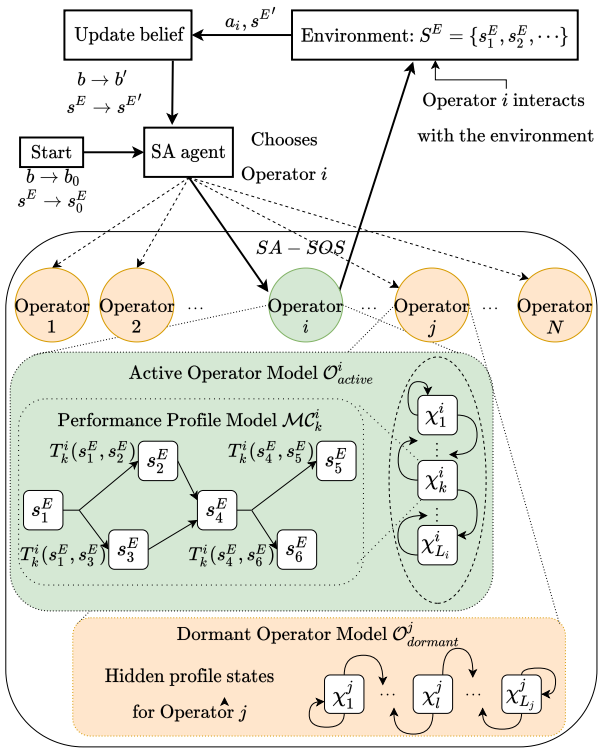


Figure 1: Diagram of the SO-SAS framework.

information to choose an operator. The chosen operator takes control of the system, causing the environment to evolve to a new state. The SA agent then updates its belief by observing the new environment state, and uses it to choose the operator that will take control next. The framework we design for an SA system is shown in Fig. 1. We will formally define each component in the remainder of this section.

Example 1 (UAV surveillance)**.** *We consider an adaptation of the surveillance problem proposed in Feng et al., [2016]. An unmanned aerial vehicle (UAV) is tasked with taking photos at four waypoints. At each waypoint, the UAV will take photos until a "good" photo is taken. The system is comprised of a human operator and autonomous operation. The human might be in a alert state or might be tired, and this influences their ability to take good photos. The aim is to take good photos at all the waypoints while minimising the amount of petrol used. To do so, the SA system should select human control when the human is in an alert state and autonomous control when the human is tired.*

The surveillance problem described above has an environment state space $S^E = S^{Nodes} \times GP_1 \times GP_2 \times GP_3 \times GP_4$. *The set of nodes the UAV could be at is* S^{Nodes}. $GP_i = \{0, 1\}$ *is a binary flag of whether a good photo has been successfully taken at waypoint* W_i, $i \in \{1, 2, 3, 4\}$. *The initial environment state is* $s_0^E = (s^{Nodes} : W_1, GP_1 : 0, GP_2 : 0, GP_3 : 0, GP_4 : 0)$. *If the UAV remains at the same waypoint to take another photo, there is a reward of* -20. *If the UAV transitions from one node to another, there is a reward of* -60. *The states where* $(GP_1 : 1, GP_2 : 1, GP_3 : 1, GP_4 : 1)$ *are absorbing, with zero reward. These rewards reflect the petrol costs incurred by the UAV and the goal state of the mission.*

4.3 Operator Model

Operator $i \in \{1, \ldots, N\}$ can exhibit L_i patterns of behaviour. We model this using *performance profile states*.

Definition 2 (Performance Profile States)**.** The set of performance profile states for operator i is defined as $X^i = \{\chi_1^i, \chi_2^i \cdots \chi_{L_i}^i\}$.

At each timestep the operator is in one performance profile state χ_j^i, and this information is hidden from the SA agent.

Example 2. *In the UAV suveillance problem, the human operator has two performance profile states,* $X^{human} = \{\chi_{alert}^{human}, \chi_{tired}^{human}\}$. *The autonomous operator has one performance profile state* $X^{auto} = \{\chi_1^{auto}\}$.

The behaviour of the ith operator in the jth performance profile state is described by a *performance profile model*. This describes how the ith operator changes the environment state when they are in control and in χ_j^i.

Definition 3 (Performance Profile Model)**.** The performance profile model for performance profile state $\chi_j^i \in X^i$ is an MC $\mathcal{MC}_j^i = \langle S^E, s_0^E, T_j^i \rangle$, where $T_j^i(s^E, s^{E'})$ is the probability of the environment state transitioning from s^E to $s^{E'}$, given that the ith operator is in control and in performance profile state χ_j^i. We assume that the performance profile model accurately represents the operator's behaviour.

Example 3. $\mathcal{MC}_{alert}^{human}$ *and* $\mathcal{MC}_{tired}^{human}$ *are the performance profile models for the human operator, and* $\mathcal{MC}_1^{auto}$ *is the performance profile model for the autonomous operator.* $\mathcal{MC}_{alert}^{human}$, $\mathcal{MC}_{tired}^{human}$ *and* $\mathcal{MC}_1^{auto}$ *have a probability of taking a "good" photo at any waypoint of* 0.9, 0.3 *and* 0.6, *respectively. The transition probabilities in the performance profile models were based on the example given in Feng et al., [2016]. To build a performance profile for a real UAV system, we would gather performance data from the running system, and use this data to build the models for autonomous and human operation.*

At each timestep, the performance profile state of an operator may change. This change is dependent on whether an operator is interacting with the environment or not. Therefore, we model the ith operator interacting with the environment with the *active operator model*, $\mathcal{O}_{active}^i$. When the ith operator is not interacting with the environment, they are described by the *dormant operator model*, $\mathcal{O}_{dormant}^i$.

Definition 4 (Active Operator Model)**.** The active operator model for the ith operator is defined as the MC $\mathcal{O}_{active}^i = \langle S^i, s_0^i, T_{active}^i \rangle$, where $S^i = S^E \times X_i$ is the state space for the ith operator; $s_0^i = (s_0^E, \chi_0^i)$ is the initial state; $T_{active}^i : S^i \times S^i \to [0, 1]$ is the transition function, defined as:

$$T_{active}^i((s^E, \chi^i), (s^{E'}, \chi^{i'})) = \quad T_j^i(s^E, s^{E'}) \cdot P(\chi^{i'} | s^E, \chi^i). \tag{1}$$

$P(\chi^{i'} | s^E, \chi^i)$ is the probability of the ith operator's performance profile state transitioning from χ^i to $\chi^{i'}$, given they are in control and the system is in environment state s^E.

When the ith operator is not interacting with the environment, the change in the performance profile states is described relying only on its their own profile states.

Definition 5 (Dormant Operator Model)**.** The dormant operator model for the ith operator is defined as the MC $\mathcal{O}^i_{dormant} = \langle X^i, \chi^i_0, T^i_{dormant} \rangle$, where $T^i_{dormant}(\chi^i, \chi^{i'})$ is the probability of the ith operator's performance profile state transitioning from χ^i to $\chi^{i'}$, when they are not in control of the system.

Example 4. *We define the average number of tasks done before a human gets tired to be n_f. We represent this using the human's active operator model, $\mathcal{O}^{human}_{active}$, where we set*

$$P(\chi^{human}_{tired}|s^E, \chi^{human}_{alert}) = 1 - (\frac{1}{2})^{\frac{1}{n_f}},$$
$$P(\chi^{human}_{alert}|s^E, \chi^{human}_{alert}) = (\frac{1}{2})^{\frac{1}{n_f}}, \quad (2)$$
$$P(\chi^{human}_{tired}|s^E, \chi^{human}_{tired}) = 1.$$

As [Feng et al.*, 2016] did not consider the human operator recovering from the tired state, the dormant operator model for the human only contains self-loop transitions. Formally, $T^{human}_{dormant}(\chi^{human}, \chi^{human'}) = 1$ if $\chi^{human} = \chi^{human'}$, and is 0 otherwise. The value of n_f can be estimated from studies measuring fatigue [Dawson and Reid, 1997].*

4.4 The SO-SAS MOMDP

In SO-SAS, the SA system has N operators that are uncertain and dynamic. We present SO-SAS as a MOMDP, where *the hidden states correspond to the N operators' performance profile states*. The dormant and active operator models for the N operators are used to define the transition functions in the SO-SAS MOMDP. We define the observation space to be the environment states, which are assumed to be fully observable. The observation function is a deterministic Dirac delta function, which returns one if the environment state and the observation are the same, and zero otherwise. Formally:

Definition 6 (SO-SAS)**.** The SO-SAS MOMDP is defined by the tuple $\langle S_o, S_h, s_0, b_0, A, \Omega, T, O, R, \gamma \rangle$, where $S_o = S^E$ is the environment state space; $S_h = X = X^1 \times \cdots \times X^N$ is the hidden performance profile state space for the N operators; $s_0 = s^E_0$ is the initial environment state; b_0 is the initial belief distribution over the performance profile state space X; $A = \{a_1, a_2 \cdots a_N\}$, where a_i denotes the SA agent choosing operator i to take control of the system; $\Omega = S^E$, i.e. the set of observations is the observable state space; $T : (S^E \times X) \times A \times (S^E \times X) \to [0, 1]$ is the transition function. Using the active and dormant operator models for the N operators, we calculate the transition probability for the MOMDP as:

$$T((s^E, \chi^1, \chi^2, \cdots, \chi^N), a_i, (s^{E'}, \chi^{1'}, \chi^{2'}, \cdots, \chi^{N'})) =$$
$$T^i_{active}((s^E, \chi^i), (s^{E'}, \chi^{i'})) \times \prod_{\substack{k=1 \\ k \neq i}}^{N} T^k_{dormant}(\chi^k, \chi^{k'}); \quad (3)$$

$O : (S^E \times X) \times A \times \Omega \to [0, 1]$ is the observation function, which we define as:

$$O((s^E, \chi^1, \cdots \chi^N), a, o) = \begin{cases} 1 & \text{if } o = s^E \\ 0 & \text{otherwise;} \end{cases} \quad (4)$$

$R : (S^E \times X) \times A \to \mathbb{R}$ is the reward function obtained from the reward over the environment transitions:

$$R((s^E, \chi), a_i) = \sum_{(s', \chi') \in S^E \times X} T((s^E, \chi), a_i, (s^{E'}, \chi')) R^E(s^E, s^{E'}); \quad (5)$$

and $\gamma \in (0, 1]$ is the discount factor.

Example 5. *The SO-SAS MOMDP for the AUV surveillance problem has $A = \{a_{human}, a_{auto}\}$, allowing the SA agent to choose the controller of the UAV. As there is only one profile state for the autonomous operator, the hidden state space can be simplified to $S_h = X^{human}$. The SA agent initially believes the human to be in the alert profile state, i.e. $b_0(\chi^{human}_{alert}) = 1.0$ and $b_0(\chi^{human}_{tired}) = 0.0$. The discount factor is $\gamma = 1$.*

In the SO-SAS MOMDP, the observation space is simply the environment state. This removes the need to design observation functions for the hidden operator profile states, which can be very complex and problem specific, possibly requiring sensors close to the operator. Observing the transitions between the environment states enables us to refine the belief over the hidden profile states, due to the different models for each performance profile.

5 Experiments

In this section, we apply and evaluate SO-SAS on a simulated domain and a real computer game. We compare SO-SAS to MDP formulations presented in existing work. We also compare robustness to inaccurate modelling for SO-SAS and MDP policies. The MDPs presented in the this section are solved exactly using value iteration. The MOMDPs are solved approximately using a straightforward adaptation of the sampling-based search algorithm POMCP [Silver and Veness, 2010], which extends the definition of history to also include the observable part of the states visited until the current state.

5.1 UAV Surveillance

Domain Description

We applied our approach to the *UAV domain* from Feng *et al.*, [2016]. The UAV flies around a map and takes photos at the waypoints until it has obtained good photos at all four waypoints. There is a human operator and an autonomous operator that can control the UAV, with their models following Examples 1 to 5. In the UAV domain, the average number of tasks done by a human before they enter the tired state is defined as n^d_f.

Methods Compared

We evaluate three types of policies.

SO-SAS policy: The policy found by solving the SO-SAS for each instance using POMCP.

Deterministic Human MDP policy: This makes the assumptions made in Feng *et al.*, [2016]: the performance profile is observable, and the human has a deterministic transition to the tired profile state after n^m_f steps.

Stochastic Human MDP policy: The human is modelled to have a stochastic transition to the tired state. However, unlike

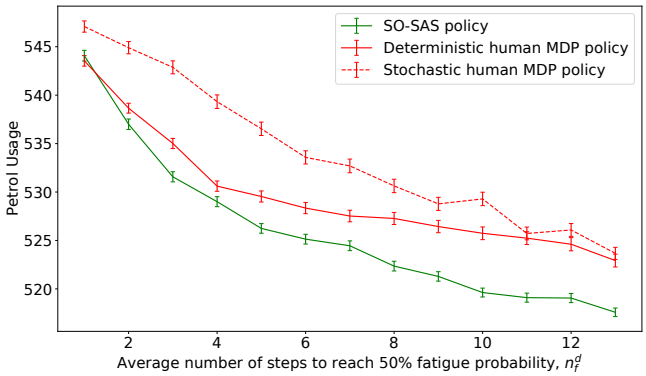


Figure 2: Average petrol usage when following the SO-SAS policy, the deterministic human MDP policy from Feng *et al.*, [2016] and the stochastic human MDP policy on the *UAV domain* over 2000 simulations.

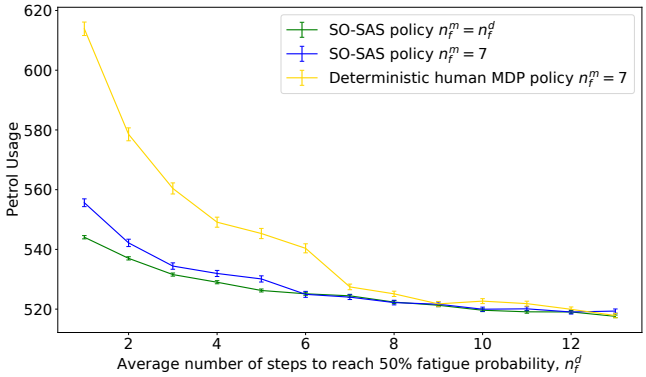


Figure 3: Average Petrol Usage when following the SO-SAS and the deterministic human MDP policy from Feng *et al.*, [2016] where $n_f^m = 7$ in the *UAV domain* over 2000 simulations.

SO-SAS the agent does not hold a belief over the (unobservable) profile state of the human, and therefore samples this state from the model after each action. Sampling is performed according to the probabilities described in Equation 2, and the environment state transitions are not considered.

Results

We applied the these three approaches to the *UAV domain* with a correct model, i.e. with the average number of steps used in the model, n_f^m, set to match the value used in the evaluation environment, n_f^d. To solve the SO-SAS MOMDP, we ran POMCP for 3×10^6 trials. The results are shown in Fig. 2 and show that SO-SAS is able to achieve lower costs than the MDP-based approaches. This is because many times the MDP policies make decisions assuming the human operator to be tired when they are not, and vice-versa. In contrast, by maintaining a belief based on the outcomes of their actions, the SO-SAS policy is able to more accurately predict the human operator profile state, and choose who controls the UAV next accordingly. Given its poor performance, we will not consider the stochastic human MDP policy further in the experiments in order to present clearer plots.

In Fig. 2, the model is assumed to correspond to the domain (i.e. $n_f^m = n_f^d$). However, in practice a human model may not be accurate, e.g. because it is designed to generalise over a range of operators, rather match the current operator. To test the robustness of these policies to inaccurate modelling, we evaluate the SO-SAS policy and the deterministic human MDP policy on the *UAV domain* where the n_f^d value and n_f^m do not align. We use the policies found by solving models with $n_f^m = 7$, and apply the policies to the *UAV domain* where n_f^d ranges from 1 to 13. The results are shown in Fig. 3, where we also plot the value of the SO-SAS policy associated with the correct n_f^m in order to provide a lower bound on the petrol usage for each n_f^d. When $n_f^d < 7$, the petrol used when following the deterministic human MDP policy is much higher than following the SO-SAS policy, because it assumes the human is not tired, but it is likely they are. In contrast, the SO-SAS policy updates the belief distribution to reflect the high failure rate. This allows it to stop granting control to the human operator when they underperform. This allows the performance of the SO-SAS policy to remain closer to optimal.

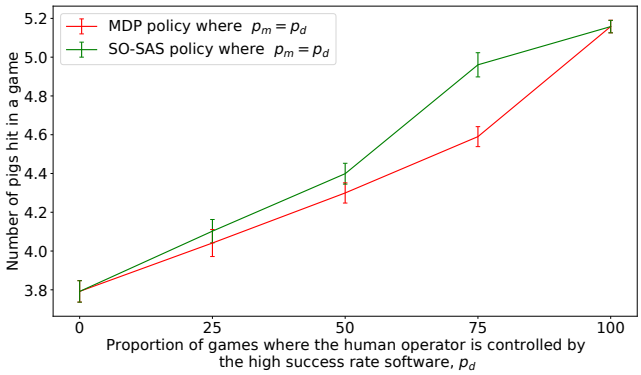


Figure 4: The average number of birds hit in a game when following the SO-SAS policy and the MDP policy, where the domain and the model's p values are aligned ($p_d = p_m$).

5.2 Angry Birds

Domain Description

We considered the game Angry Birds (AB) as an SA system, where two operators can be used to play the game. AB is a puzzle game, where the player uses a catapult to hit pigs hidden in a structure. The player has a fixed number of birds to catapult, and the aim is to maximise the number of pigs hit. Birds are shot sequentially, and each shot is taken by a single player.

Our AB domain has two players, a human and an autonomous operator. We simulate the human operator using two pieces of software, that can play the game through the human interface (i.e. click and drag objects in the game). The two pieces of software have different success rates, to represent human players with low and high skill levels. The software with the lower success rate randomly chooses a pig to shoot at and ignores the presence of obstacles. The software with the higher success rate was developed by Datalabs [Borovička *et al.*, 2014] and won the Angry Birds AI

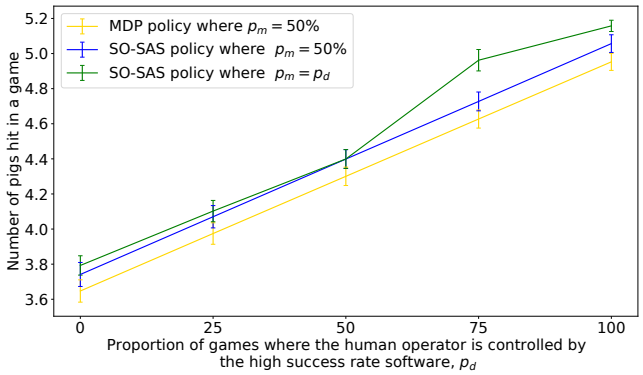


Figure 5: The average number of birds hit in a game when following the SO-SAS policy where the domain and the model's p values are aligned ($p_d = p_m$) and the SO-SAS and MDP policy where the model's p value is $p_d = 50\%$.

competition [Jochen Renz, 2021] in 2013 and 2014. The human is simulated by the high success rate software in $p_d\%$ of games, and the low success software in $(100 - p_d)\%$ of games, where $p_d \in [0, 100]$. The software controlling the human operator does not change during a game. At the start of a game, the agent does not have any information on the current human player's skill level.

The autonomous operator is controlled by a third piece of software that considers all the possible shots to hit the pigs and chooses an unblocked trajectory at random. If there are no clear shots possible, the autonomous operator will choose a random pig to shoot at. Our AB SA evaluation considers a single level with five birds and six pigs.

Methods Compared

We compared the following Angry Birds strategies.

SO-SAS policy: The SO-SAS for the AB game has an environment state space defined by the number of birds and pigs in the environment, and a binary flag of whether the game has ended. At the end of the game, the agent receives a reward equal to the number of pigs hit. There is an additional reward of $+10$ when all of the pigs are hit. At each shot, the agent chooses an action from the set $A = \{a_{auto}, a_{human}\}$.

We construct the performance profile models for the human and the autonomous operator by recording the results of 150 games played by each operator in a known profile state. The transition probabilities for each performance profile model are constructed from the state transition frequencies. These transition probabilities for each automated player are used to generate the associated performance profiles. The human is assumed to have a $p_m\%$ probability of having a high skill level, and a $(100 - p_m)\%$ probability of having a low skill level, where $p^m \in [0, 100]$. The performance profile states for the human operator is $X^h = \{\chi^h_{low}, \chi^h_{high}\}$, and the initial belief is $b_0(\chi^h_{low}) = p_m\%, b_0(\chi^h_{high}) = (100 - p_m)\%$. The SO-SAS model is solved by running POMCP for 3×10^5 trials.

MDP policy: This policy is created from an MDP based on the SA AB domain. As the typical SA formulation does not consider multiple profiles for the human operator, the human operator is represented by a single model. We create the single model using $p_m\%$ of the data from the results of the high skill level human operator, and $(100 - p_m)\%$ of the data from the low skill level human operator.

The MDP agent will choose the operator for the next step based on their model for the default and the mixed human operator. The mixed human operator is represented by a single performance profile, obtained by mixing the two models for the greedy and random performance profiles.

Results

The results from applying the SO-SAS and the MDP policies where $p_m = p_d$ on the domain where p_d ranges from 0% to 100% are shown in Fig. 4. The SO-SAS and MDP policies where $p_m = p_d$ are shown to have similar results in cases where p_d is close to 0 or 100%. In these cases, the agent is certain about the ability of the human operator, so the effect of maintaining a belief distribution over the ability of the human operator is minimal. When p_d is closer to 50%, as there is high uncertainty in the ability of the human, the SO-SAS policy will vary depending on the behaviours exhibited by the human operator. For example, if the human operator misses an easy shot, the agent updates their belief in the human operator having a low ability level. This alters whether the human operator is chosen to take the next shot, and results in the SO-SAS policy outperforming the MDP policy when $p_m = 25, 50, 75\%$.

However, much like the UAV domain, our modelling assumptions may not be accurate. Therefore we tested the robustness of the SO-SAS and MDP policies to inaccurate modelling by evaluating the policies created with $p_m = 50\%$ in the domain where p_d varies between 0 and 100%. This is shown in Fig. 5. The SO-SAS policy outperforms the MDP policy where they model $p_m = 50\%$ in all domains where p_d ranges from 0 to 100%. However, we were unable to observe a significant difference in performance between the SO-SAS policies where $p_m = p_d$ and $p_m = 50\%$.

These results again demonstrate the strength of the SO-SAS approach, compared to an MDP approach, when the performance profile of the operator is not directly observable – an assumption which is likely to hold in reality. The robustness results also demonstrate the importance of the online estimation of operator performance in SA applications. Not only should an SA system not assume that operator performance is observable, but it should also not treat its modelling assumptions as accurate, since human performance cannot be predicted with total accuracy.

6 Conclusion

We have presented a framework for stochastic, dynamic and partially observable operators in a SA system, SO-SAS, using MOMDPs. We compared our SO-SAS policy to MDP policies, and found that our SO-SAS policy significantly outperforms MDP policies.

Acknowledgments

This work was supported by the Defence Science and Technology Laboratory, the EPSRC Programme Grant 'From

Sensing to Collaboration' (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. This document is an overview of UK MOD's Defence Science and Technology Laboratory (DSTL) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy.

References

[Basich *et al.*, 2020] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Learning to Optimize Autonomy in Competence-Aware Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pages 123–131, Richland, SC, May 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[Borovička *et al.*, 2014] Tomáš Borovička, Radim Špetlík, and Karel Rymeš. DataLab Birds - Angry Birds AI, August 2014.

[Charles *et al.*, 2018] Jack-Antoine Charles, Caroline P. C. Chanel, Corentin Chauffaut, Pascal Chauvin, and Nicolas Drougard. Human-Agent Interaction Model Learning based on Crowdsourcing. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, HAI '18, pages 20–28, New York, NY, USA, December 2018. Association for Computing Machinery.

[Cubuktepe *et al.*, 2021] Murat Cubuktepe, Nils Jansen, Mohammed Alshiekh, and Ufuk Topcu. Synthesis of Provably Correct Autonomy Protocols for Shared Control. *IEEE Transactions on Automatic Control*, 66(7):3251–3258, July 2021. Conference Name: IEEE Transactions on Automatic Control.

[Dawson and Reid, 1997] Drew Dawson and Kathryn Reid. Fatigue, alcohol and performance impairment. *Nature*, 388(6639):235–235, July 1997. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 6639 Primary_atype: Research Publisher: Nature Publishing Group.

[Duchetto *et al.*, 2018] Francesco Duchetto, Ayse Kucukyilmaz, Luca Iocchi, and Marc Hanheide. Do Not Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation From Human Demonstrations. *IEEE Robotics and Automation Letters*, PP:1–1, July 2018.

[Feng *et al.*, 2016] L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu. Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems. *IEEE Transactions on Automation Science and Engineering*, 13(2):450–462, April 2016. Conference Name: IEEE Transactions on Automation Science and Engineering.

[Fern *et al.*, 2014] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50:71–104, May 2014.

[Javdani *et al.*, 2015] Shervin Javdani, Siddhartha S. Srinivasa, and J. Andrew Bagnell. Shared Autonomy via Hindsight Optimization. *arXiv:1503.07619 [cs]*, April 2015. arXiv: 1503.07619.

[Jean-Baptiste *et al.*, 2015] Emilie M. D. Jean-Baptiste, Pia Rotshtein, and Martin Russell. POMDP Based Action Planning and Human Error Detection. In Richard Chbeir, Yannis Manolopoulos, Ilias Maglogiannis, and Reda Alhajj, editors, *Artificial Intelligence Applications and Innovations*, IFIP Advances in Information and Communication Technology, pages 250–265, Cham, 2015. Springer International Publishing.

[Jochen Renz, 2021] Jochen Renz. AI Birds.org - Angry Birds AI Competition, 2021.

[Junges *et al.*, 2018] Sebastian Junges, Nils Jansen, Joost-Pieter Katoen, Ufuk Topcu, Ruohan Zhang, and Mary Hayhoe. Model Checking for Safe Navigation Among Humans. In Annabelle McIver and Andras Horvath, editors, *Quantitative Evaluation of Systems*, Lecture Notes in Computer Science, pages 207–222, Cham, 2018. Springer International Publishing.

[Ong *et al.*, 2010] Sylvie C. W. Ong, Shao Wei Png, David Hsu, and Wee Sun Lee. Planning under Uncertainty for Robotic Tasks with Mixed Observability. *The International Journal of Robotics Research*, 29(8):1053–1068, July 2010. Publisher: SAGE Publications Ltd STM.

[Rigter *et al.*, 2020] Marc Rigter, Bruno Lacerda, and Nick Hawes. A Framework for Learning From Demonstration With Minimal Human Effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, April 2020.

[Rosenthal and Veloso, 2012] Stephanie Rosenthal and Manuela Veloso. Mobile robot planning to seek help with spatially-situated tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 2067–2073, Toronto, Ontario, Canada, July 2012. AAAI Press.

[Silver and Veness, 2010] David Silver and Joel Veness. Monte-Carlo planning in large POMDPs. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, pages 2164–2172, Red Hook, NY, USA, December 2010. Curran Associates Inc.

[Wray *et al.*, 2016] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 517–523, New York, New York, USA, July 2016. AAAI Press.

[Wray *et al.*, 2017] Kyle Hollins Wray, Stefan J. Witwicki, and Shlomo Zilberstein. Online Decision-Making for Scalable Autonomous Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4768–4774, Melbourne, Australia, August 2017. International Joint Conferences on Artificial Intelligence Organization.

[Wu *et al.*, 2017] Bo Wu, Bin Hu, and Hai Lin. Toward efficient manufacturing systems: A trust based human robot collaboration. In *2017 American Control Conference (ACC)*, pages 1536–1541, May 2017. ISSN: 2378-5861.

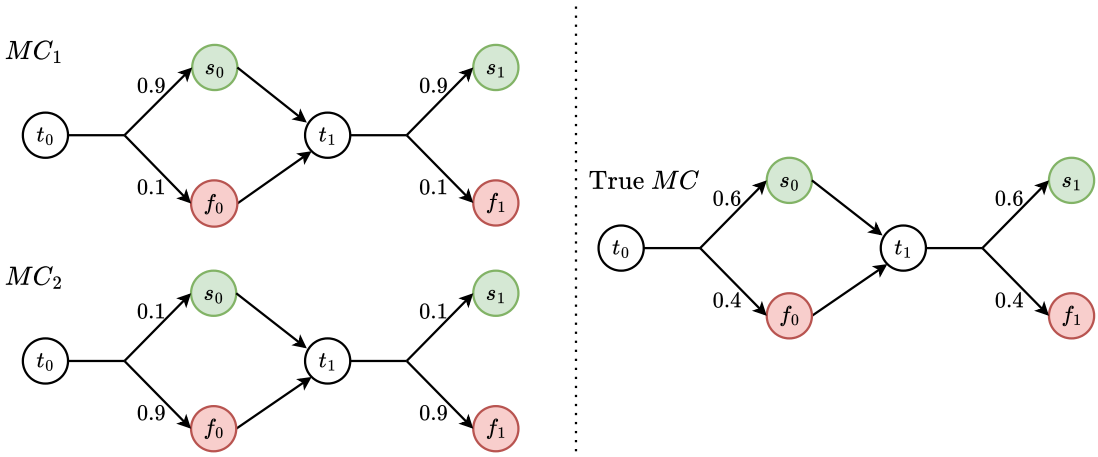


Figure 4.1: Example Problem

4.1 Limitations and Future Work

One of the main limitations in this work is the assumption that the operator is in a single *performance profile* state. In reality, the factors that affect operator performance are on a continuous scale, and their performance should be described as a combination of existing models. However, in the SO-SAS framework, the agent assumes the operator is in a "true" state and updates the belief accordingly. This may result in a policy that prematurely stops taking exploratory actions, as they assume that the belief over the operator's state has collapsed.

To illustrate this point, consider a human operator whose true behaviour is modelled by a MC shown in Figure 4.1. However, the agents assume the human operator acts as one of two MCs: MC_1 and MC_2, as shown in Figure 4.1. In this scenario, the human acts as a linear combination of the two MCs, where the transition function for the human is defined by:

$$T_{human}(s, a, s') = T_1(s, a, s') * 0.625 + T_2(s, a, s') * 0.375. \tag{4.1}$$

Suppose the agent begins with a uniform prior over the human's true MC, where $B(1) =$

$B(2) = 0.5$. If the agent starts out at t_0, the probability the human ends up in a fail state is 0.4. However, if the agent reaches f_1, the belief over the human's true state is updated using Bayes' rule and becomes $B(1) = 0.1, B(2) = 0.9$. Therefore, the agent believes that there is a 0.82 probability of entering into the fail state in the next task. This results in a policy that never selects the human to do a task again after observing a single action. Therefore, we must consider our confidence we have in our belief, and explicitly model the human as a linear combination of MDPs.

Another limitation is the paper only considers simulated human profiles. While the angry birds experiments use real software that can play the game in real time, we are using the software to represent how a human may play the game. To further this work, collecting real human data and applying our shared autonomy framework would add to the motivation of using multiple models to represent a single operator.

Chapter 5

Planning with Hidden Polynomial MDPs

As mentioned in the limitations of Chapter 4, the problem with using MOMDPs to describe a shared autonomy system is that the factors that affect the operators are typically continuous. To handle this, we can consider a continuous set of MDPs that could be used to describe an operator. The agent does not have a prior on which MDP the operator is in, but maintains a belief over the continuous set of MDPs, and updates the belief based on their observations of the operator. This framework can be described as an uncertain MDP and thus be converted into a BAMDP, as defined in Section 2.3. BAMDPs have two main implementations: 1. there are no priors over the transition probabilities, and 2. the transition dynamics are governed by an unknown MDP from a set of known MDPs.

We cannot use the first type of BAMDP to describe our operator, as we have prior knowledge of the range of behaviours they could exhibit. For example, if we had tasks i and j in our sequence of tasks that are similar, we could express this in our continuous set of MDPs, as the two tasks would have a similar probability of success. Therefore, if the operator was successful on task i, this would update the belief of which MDP the operator is represented by, and thus the belief that they would be successful on task j. However,

in the setting where there is no prior over the transition probabilities, the belief over the success probabilities of task i and j will be represented by two independent Dirichlet distributions. Therefore, there is no method to pass the information gained from task i to inform the success probabilities for task j.

The second type of BAMDP is implemented by sampling a set of MDPs from the continuous set, and maintaining a discrete belief over the sampled set. However, sample-based methods are slower to update the belief, and as the belief must be updated for every sample. Furthermore, as the dimensions of the continuous space of possible MDPs increases linearly, the number of samples required to cover the belief space increases exponentially. This means that problems with high dimensionality cannot be efficiently solved using sample based methods.

We therefore propose a framework for uncertain MDPs where the transition functions are defined by a polynomial function where the variables are the continuous latent parameters. This approach allows us to use information learnt in earlier tasks to inform the agents for decisions in later tasks that require a similar skill set. Our framework does not rely on sample-based belief representations, and we show that we can maintain a closed-form distribution for the belief over the latent parameter space. This framework can be used in describing problems other than shared autonomy systems. where the transition dynamics are governed by latent parameters, such as a glider being pushed by unknown currents.

Table 5.1: Summary of the papers and their features

Chapters	4	5	6	7
SA	Yes	No	Yes	Yes
SA style	N operators controls one robot	N/A	One human and one autonomous agent controls a robot	one human and K robots, where the human can take control of any robot
Dynamic	Yes	No	Yes	No
Continuous	No	Yes	Yes	Yes
Closed-form	Yes	Yes	Yes	Yes

The Thirty-Seventh AAAI Conference on Artificial Intelligence (AAAI-23)

Planning with Hidden Parameter Polynomial MDPs

Clarissa Costen, Marc Rigter, Bruno Lacerda, Nick Hawes

Oxford Robotics Institute, University of Oxford
{clarissa, mrigter, bruno, nickh}@robots.ox.ac.uk

Abstract

For many applications of Markov Decision Processes (MDPs), the transition function cannot be specified exactly. Bayes-Adaptive MDPs (BAMDPs) extend MDPs to consider transition probabilities governed by latent parameters. To act optimally in BAMDPs, one must maintain a belief distribution over the latent parameters. Typically, this distribution is described by a set of sample (particle) MDPs, and associated weights which represent the likelihood of a sample MDP being the true underlying MDP. However, as the number of dimensions of the latent parameter space increases, the number of sample MDPs required to sufficiently represent the belief distribution grows exponentially. Thus, maintaining an accurate belief in the form of a set of sample MDPs over complex latent spaces is computationally intensive, which in turn affects the performance of planning for these models. In this paper, we propose an alternative approach for maintaining the belief over the latent parameters. We consider a class of BAMDPs where the transition probabilities can be expressed in closed form as a polynomial of the latent parameters, and outline a method to maintain a closed-form belief distribution for the latent parameters which results in an accurate belief representation. Furthermore, the closed-form representation does away with the need to tune the number of sample MDPs required to represent the belief. We evaluate two domains and empirically show that the polynomial, closed-form, belief representation results in better plans than a sampling-based belief representation.

Introduction

Markov Decision Processes (MDPs) (Puterman 1994) are a common framework for sequential decision-making under uncertainty. In this paper, we consider an agent acting in an environment where the parameters governing the underlying MDP model are unknown. In such environments, the agent must reason over the uncertainty in the parameters, and balance exploration of the environment with gathering reward. An example of this is a robot planning over a domain with unknown weather conditions. As the robot interacts with the environment, the observations made by the robot can be used by it to plan for future steps. Bayes-Adaptive MDPs (BAMDPs) (Duff 2002) are used to model such problems in a way that allows for optimally trading off exploration and exploitation.

In BAMDPs, latent parameters are used to represent unknown dynamics that influence the transition probabilities. Therefore, the state space of the original MDP is augmented by a posterior distribution which represents the belief over the true value of the latent parameters. As transitions are observed, the posterior distribution over the latent parameters is refined. By reasoning about the information currently known about the MDP in the form of the posterior, BAMDPs provide an elegant and optimal solution to the exploration-exploitation trade-off. Specifically, solving the BAMDP optimally results in the optimal behaviour in expectation under the initial belief. One challenge in developing algorithms for BAMDPs is the need to accurately maintain the posterior. There are several methods to do so, depending on the class of BAMDP needed to be solved.

In a BAMDP where every transition probability is independent, each transition probability is a latent parameter. The posterior distribution over the latent parameters is represented by a set of Dirichlet distributions, where each distribution represents the transition function from a state-action pair. This allows us to maintain an exact closed-form representation of the posterior distribution. However, for many problems, the transitions cannot be assumed to be independent of each other. In such problems, the posterior distribution of the BAMDP can be approximated by a discrete set of *particles*, i.e. a set of sampled MDPs plus a categorical distribution over the samples (Guez et al. 2014). This method relies on the MDP samples providing adequate coverage of the space of possible parameters. As the number of latent parameters increases, so does the dimensionality of the belief distribution over the parameters. Thus, the number of samples must increase exponentially to represent the posterior over the belief accurately. This causes scalability issues, which in turn limits the ability to effectively plan for BAMDPs with realistic latent spaces.

In this paper, we present a new class of BAMDPs, hidden parameter polynomial MDPs (HP^2-MDP). In a HP^2-MDP, the transition probabilities are defined by polynomial functions of the latent parameters. MDPs where the transitions are described by polynomial functions are traditionally used in probabilistic model checking for parameter synthesis, where the variables of the polynomials are assumed to be known and controllable (Junges et al. 2018). We re-frame these models such that the variables represent latent aspects of the environment that can be estimated during execution. This allows

for an exact closed-form representation of the belief over the latent variables, whilst also enabling the specification of dependencies between transitions. The closed-form belief representation can accurately deal with high-dimensional latent parameter spaces, resulting in better action selection. Furthermore, our method removes the need for hand-tuning the number of samples used to represent the belief.

We empirically show, in two domains, that incorporating the belief representation of HP^2-MDP in typical BAMDP solvers, such as Bayes-adaptive Monte-Carlo planning (BAMCP) (Guez, Silver, and Dayan 2012), yields improvement in planning performance. The first domain is a benchmark problem proposed in the original BAMCP paper. The second, more realistic, domain uses data from forecast ocean currents to simulate a problem where an ocean glider is planning a route to a goal location.

Related Work

MDPs with Hidden Factors Markov decision processes (MDPs) have been used in planning under uncertainty (Mausam and Kolobov 2012) to help agents make optimal action choices. When MDPs are used to describe a system, we assume that the state is fully observable to the agent. However, this assumption cannot be satisfied in cases where the sensors used to determine the state are unreliable (Kaelbling, Littman, and Cassandra 1998; Hsiao, Kaelbling, and Lozano-Perez 2007), or the states are dependent on unknown factors, such as a human's likelihood to assist a robot (Nanavati et al. 2021; Costen et al. 2022). Partially Observable MDPs (POMDPs) (Kaelbling, Littman, and Cassandra 1998), and specialisations thereof are typically used to describe such systems.

In POMDPs, the state space is hidden from the agent. The agent receives stochastic observations, which are used to maintain a belief distribution over the hidden state space. The posterior over the belief can be updated to reflect hidden state transitions, which can be used in scenarios such as tracking the location of a human through observations made from an unreliable sensor in an autonomous driving setting (Wray, Witwicki, and Zilberstein 2017). The belief distribution is updated by reasoning over the hidden state transitions and the stochastic observations. This can be difficult, as the hidden state is dynamic, and thus hard to keep track of through stochastic observations.

Hidden Goal MDPs (Fern et al. 2014) and POMDP-lite (Chen et al. 2016) attempt to simplify the problem by splitting the state space into a static hidden state space and a stochastic observable state space. The hidden state is unknown at the start of the run, and the agent updates their belief over the hidden states as observable state transitions occur. As the hidden state is fixed, the agent does not have to reason over hidden state transitions, reducing the complexity. Each hidden state corresponds to an MDP, and the hidden goal MDPs and POMDP-lites can be framed as a problem where the agent is planning over a set of MDPs, where the true MDP is unknown.

Similarly, robust MDPs (Tamar, Mannor, and Xu 2014) and contextual MDPs (Hallak, Di Castro, and Mannor 2015) represents an MDP governed by uncertain parameters, which are a member of a known set. While the robust MDP solvers attempt to plan for the worst-case scenario, contextual MDP solvers aim to minimise the regret; the gap between the cumulative reward collected by the agent, and the cumulative reward the agent would have collected if they knew the true parameters (Rigter, Lacerda, and Hawes 2021a). However, this assumes that the system can be represented as a discrete set of MDPs.

Doshi-Velez and Konidaris (2016) present a Hidden Parameter MDP, where the transitions are functions governed by latent parameters. By approximating the transition functions to Gaussian processes, they outline a method of maintaining a belief over the latent parameters.

Bayes-Adaptive MDPs BAMDPs allow the agent to reason over a continuous set of MDPs the system might be in (Duff 2002; Guez, Silver, and Dayan 2012). This is done by parameterising the transition probabilities with a set of continuous parameters, which can more accurately reflect a real system. However, many approaches discretise the parameter space (Nanavati et al. 2021) to reduce the complexity of the problem. In other approaches where transition probabilities are assumed to be independent, every transition probability is represented as a latent parameter (Rigter, Lacerda, and Hawes 2021b; Grover, Basu, and Dimitrakakis 2020).

In general BAMDPs, the belief over the latent parameters is typically approximated through sampling-based methods (Guez et al. 2014). This allows us to solve BAMDPs where there is dependence between transitions. In cases where the transitions are assumed to be independent, an exact form of the belief can be maintained. In such BAMDPs, the probability distribution over the parameters is represented by Dirichlet distributions. As transitions occur, these Dirichlet distributions are updated. While we can maintain an exact form of the belief distribution for such BAMDPs, we are unable to describe more complex relationships between the transition probabilities.

Polynomial MDPs In polynomial Markov Chains (MCs) and MDPs (Winkler et al. 2019), the transition function is expressed as a polynomial function of a set of parameters. This enables adding dependencies between transitions.These polynomial MDPs have been widely used in the verification community to express spaces of possible behaviour (Hahn, Han, and Zhang 2011a). They have been used in probabilistic model checking to provide formal guarantees on the reachability of goals over the parameter space (Junges et al. 2018; Hahn, Han, and Zhang 2011b). In the verification problem, we check whether a known sub-space of the parameter space for the polynomial MDP can satisfy a given specification. Other works consider the parameter synthesis problem (Junges et al. 2019), where they aim to partition the parameter space into regions where the specification is satisfied or not. Both the verification and synthesis problems assume that the true values of the parameters are known, or within a given bound. In this work, we re-frame the polynomial MDP as a BAMDP, where the parameters are hidden, and we must maintain and reason over a posterior probability distribution over their true value.

Preliminaries

In this paper, we consider stochastic shortest path (SSP) MDPs (Mausam and Kolobov 2012), where the objective is to minimise the expected cumulative cost to reach a set of goal states. This is done to match the domains used for empirical evaluation. However, the approach presented here is independent of the objective and can be applied to reward maximisation problems straightforwardly. Thus, we will refer to the model simply as an MDP.

When an MDP's transition dynamics is governed by a set of latent parameters, Θ, a BAMDP can be used to represent the world. Consider a MDP with an uncertain transition function $\mathcal{M} = \langle S, s_0, A, \{T_\theta\}_{\theta\in\Theta}, C, G\rangle$, where S is a set of states; $s_0 \in S$ is the initial state; A is a set of actions; $\{T_\theta\}_{\theta\in\Theta}$ is a set of possible transition functions, governed by a set of latent parameters in Θ; $C : S \times A \to \mathbb{R}_{\geq 0}$ is a cost function; and $G \subset S$ is a set of goal states. The parameter space is $\Theta = \mathbb{R}^N$, i.e. the latent parameters are N-dimensional vectors, and θ defines a possible transition function $T_\theta : S \times A \times S \to [0, 1]$ which gives the probability of transitioning to s' given that a was executed at s and the parameter values are θ, i.e., $T_\theta(s, a, s') = P(s' \mid s, a, \theta)$. We assume a prior distribution $P_0(\theta)$ over the parameter space. The agent maintains a posterior distribution, or belief, over Θ at each timestep t, $P_t(\theta)$. This is determined by the observed history, $h_t = s_0 a_0 s_1 a_1 \cdots a_{t-1} s_t$ using Bayes' rule, $P_t(\theta) = P(\theta|h_t) \propto P(h_t|\theta)P_0(\theta)$. A BAMDP factors the history into the state space, and explicitly models how the transition probabilities change with the history of the agent.

Definition 1 (BAMDP) *A BAMDP is defined by the tuple, $\langle S^+, s_0^+, A, T^+, C^+, G^+\rangle$, where:*

- $S^+ = S \times \mathcal{H}$*, where $\mathcal{H}$ is the set of all histories;*
- $s_0^+ = (s_0, h_0)$*, where $h_0 = s_0$ is the initial history;*
- A *is the set of actions;*
- $T^+((s, h_t), a, (s', h_t a s')) = \int_{\theta\in\Theta} T_\theta(s, a, s')P_t(\theta)d\theta$ *is the transition function, where $P_t(\theta) \propto P(h_t|\theta)P_0(\theta)$;*
- $C^+((s, h_t), a) = C(s, a)$ *is the cost function;*
- $G^+ = \{(s, h) \in S^+ \mid s \in G\}$ *is the set of goal states.*

Although the state-history pairs in S^+ are redundant because the current state can be extracted from the history, we use the (s, h) notation for clarity as in (Guez, Silver, and Dayan 2012). The goal of the BAMDP is defined as minimising the expected cumulative cost to reach a state in G^+. Therefore, setting $V^{\pi^*}(s, h_t) = 0$ for all $(s, h_t) \in G^+$, the optimal policy selects the action that minimises the value function according to:

$$V^{\pi^*}(s, h_t) = \arg\min_{a\in A} C(s, a) + \int_{\theta\in\Theta} \sum_{s'\in S} P(\theta|h_t)T_\theta(s, a, s') \cdot V^{\pi^*}(s', h_t a s')d\theta. \quad (1)$$

To solve the BAMDP via value iteration, the posterior distribution $P(\theta|h_t)$ for every history $h_t \in \mathcal{H}$ must be computed, which quickly becomes computationally infeasible. Therefore, BAMDPs are typically solved using a sample-based solver, such as BAMCP (Guez, Silver, and Dayan 2012).

Algorithm 1: Bayes-Adaptive Monte Carlo Planning

1: **Function**{Run}$\{P_0(\theta), s_0, \langle S, A, \{T_\theta\}_{\theta\in\Theta}, C, G\rangle\}$
2: $t \leftarrow 0$
3: $h_0 \leftarrow s_0$
4: **while** $s_t \notin G$ **do**
5: $\quad a_t \leftarrow$ `Search(`$P_t(\theta), s_t$`)`
6: $\quad s_{t+1} \leftarrow$ execute action a_t and observe outcome.
7: $\quad P_{t+1}(\theta) \leftarrow$ `UpdateBelief(`$P_t(\theta), a_t, s_{t+1}$`)`
8: $\quad t \leftarrow t + 1$
9: **end while**
10: **Function**{Search}$\{P_t(\theta), s_t\}$
11: **repeat**
12: $\quad \theta_{sample} \sim P_t(\theta)$.
13: $\quad$ `Simulate(`$s_t, T_{\theta_{sample}}$`)`
14: **until** `Timeout()`
15: **return** $\arg\max_a Q(s_t, a)$

The BAMCP algorithm is adapted from Monte-Carlo Tree Search (MCTS) (Kocsis and Szepesvári 2006), and is based on running trials from the root node to estimate the values of the available actions. Each trial simulates a run through the model and records the cost collected. The average cost over the trials is used to estimate the value of each state-action pair. As the number of trials increases, the estimated values converges towards the true value. The BAMCP algorithm is outlined in Algorithm 1. It is an online algorithm where a search procedure (line 5) is interleaved with action execution and updating of the belief over the parameters given the observed action outcomes (line 6 and 7). The search procedure is based on sampling a possible parameter instantiation from the belief distribution, and then simulating a run according to the corresponding transition function, repeating this process until a user-defined timeout (lines 11–14). A key part of BAMCP is maintaining and updating the belief given the online observations.

We now identify the two classes of BAMDPs considered more often. For these classes, the posterior over the belief is maintained using distinct methods.

BAMDP-L The simplest form of a BAMDP, presented in (Duff 2002; Ross et al. 2011), considers the case where the transition functions are independent. We refer to this as BAMDP-L, since the latent parameters are the local transition probabilities, and there are no global parameter dependencies. The posterior for a BAMDP-L can be represented one Dirichlet distribution per transition. These distributions are updated as transition outcomes are observed, and are used for the sampling step in line 12 of Algorithm 1. Thus, for a BAMDP-L, we can maintain a closed-form distribution for the posterior distribution in the BAMCP algorithm. However, the strong assumptions over transition independence make this model inapplicable to many relevant problem domains.

BAMDP-G In BAMDP-Gs, the transition probabilities are functions of global latent parameters. In this case, it is often necessary to approximate the belief. A typical method for this is particle-based, where the posterior over the belief is described by a set of M sample MDPs (Guez et al. 2014).

Each MDP has a weight describing the likelihood of it being the true description of the domain. At the start of a run, we sample M times from the prior belief $P_0(\theta)$ to get a set of parameter instantiations, $U = \{\theta^1, \cdots, \theta^M\}$. These sampled parameter values are used to generate a set of MDPs, $U_{MDP} = \{\mathcal{M}^1, \cdots, \mathcal{M}^M\}$, with the transition function of $\mathcal{M}^k$ equal to T_{θ^k}. Each MDP $\mathcal{M}^k$ is a particle with an associated weight function $w_k : \mathcal{H} \to \mathbb{R}_{>0}$ which represents how probable $\mathcal{M}^k$ is, given an observed history. We initialise $w_k(h_0) = \frac{1}{M} \forall k \in \{1, \cdots, M\}$. These weights are then updated using Bayes' rule as the agent observes state-action transitions, via

$$w_k(has') \propto T_{\theta^k}(s, a, s') \cdot w_k(h). \tag{2}$$

However, the accuracy of the representation using particles is dependent on M, as this affects how well the parameter space is covered: as the number of parameters increases, the value of M required to accurately represent the belief grows exponentially. In this work, we tackle this issue by proposing a class for which we can maintain a closed-form representation of the belief for a set of global latent parameters.

HP^2-MDP

We consider an MDP where the transition probabilities can be expressed by a polynomial of a set of global parameters, which are unknown to the agent.

Definition 2 (HP^2-MDP) *A hidden parameter polynomial MDP (HP^2-MDP) is defined by the tuple $\langle S, s_0, A, \Theta, P_0(\theta), T_\Theta, C, G \rangle$, where:*

- *S, s_0 and A are the set of states, initial state and set of actions, respectively;*
- *$\Theta \subseteq [0,1]^N$ is the parameter space of a set of N global parameters. We denote elements of Θ as $\theta = (\theta_1, \cdots, \theta_N)$;*
- *$P_0(\theta)$ is the prior belief over Θ. $P_0(\theta) \in Pol(\Theta)$, i.e., it is a polynomial function over Θ;*
- *$T_\Theta : S \times A \times S \to Pol(\Theta)$ is the (polynomial) set of possible transition functions;*
- *$C : S \times A \to \mathbb{R}$ is the cost function and $G \subset S$ is the set of goal states.*

To be well-formed, the transition functions of a HP^2-MDP must be valid probability distributions over the set of states, i.e., for all $s \in S$ and $a \in A$ that can be executed in s:

$$\sum_{s' \in S} T_\Theta(s, a, s')(\theta) = 1 \; \forall \theta \in \Theta. \tag{3}$$

While the BAMDP has a parameter space of $\Theta = \mathbb{R}^N$, in the HP^2-MDP, the parameter space is restricted to $\Theta \subseteq [0,1]^N$. This change in parameter space is required to maintain a closed-form distribution, to confine the possible parameter values to a fixed range. In the case of a parameter with a range outside of $[0,1]$, the parameter can be scaled to fit the above constraint. Thus, in HP^2-MDPs, we consider a bounded parameter set, for which Equation 3 imposes validity constraints. We cover the case where a) the parameters are independent of each other, and b) when the parameters follow $\sum_{i=1}^N \theta_i = 1$. For each case, we will provide an example illustrating when these constraints would be used. Other linear constraints generated by Equation 3 can be computed by combining the methods used in the cases outlined above. We give an example of this by showing that BAMCP-L is a type of HPP-MDP.

Independent Global Parameters

We consider a HP^2-MDP $\mathcal{M}$, where the latent parameters $\theta = (\theta_1, \ldots, \theta_N)$ are independent. We start with an example of such a case.

Example 1 *Consider an underwater glider planning a route from a start location to a finish location in the sea. The longitudinal and latitudinal velocity of the water at each location is known, but the effect of these on the glider's movement is unknown. We assume a linear relationship between the water's speed and the effect on the glider, and the effect of the longitudinal and latitudinal water velocity are respectively expressed by θ_h and θ_v. The effect of the longitudinal and latitudinal water velocity is independent, where the longitudinal water velocity will only affect the glider's movement in the east-west direction, and the latitudinal water velocity will only affect the glider in the north-south direction. For example, a point in the sea may have a longitudinal water velocity of $-0.3ms^{-1}$, and a latitudinal water velocity of $0.6ms^{-1}$. If the agent chose the action to travel west, the outcome probabilities would become the following: $P_{stationary} = 0.3\theta_h \cdot (1 - 0.6\theta_v)$, $P_{west} = (1 - 0.3\theta_h) \cdot (1 - 0.6\theta_v)$, $P_{north} = 0.3\theta_h \cdot 0.6\theta_v$ and $P_{north-west} = (1 - 0.3\theta_h) \cdot 0.6\theta_v$. The sum of the probability of these 4 outcomes is equal to 1 regardless of the values of θ_h and θ_v, thus both parameters can independently range from 0 to 1.*

Closed-Form Belief Update The posterior belief after transitioning from state-action pair s, a to state s' follows Bayes' rule:

$$P_{t+1}(\theta \mid s_t, a_t, s_{t+1}) = \frac{P_t(\theta) T_\Theta(s_t, a_t, s_{t+1})(\theta)}{\int_{\theta' \in \Theta} P_t(\theta') T_\Theta(s_t, a_t, s_{t+1})(\theta') d\theta'}. \tag{4}$$

Assuming the prior belief, $P_t(\theta)$, to be polynomial, we have that $P_t(\theta) T_\Theta(s_t, a_t, s_{t+1})(\theta)$ is a polynomial. Thus, we can consider its expansion into J terms, $\sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i^{a_i^j}$, where β_j is the coefficient of the j-th term, and a_i^j is the order of θ_i in the j-th term. In the independent global parameter case, the valid parameter space is a hyper-cube, where each parameter is in $[0,1]$. Therefore, the denominator can be expressed as:

$$\int_0^1 \cdots \int_0^1 \sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i'^{a_i^j} d\theta' = \sum_{j=1}^J \beta_j \prod_{i=1}^N \frac{1}{a_i^j + 1}. \tag{5}$$

From this, the posterior belief can be written as:

$$P_{t+1}(\theta) = \frac{\sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i{}^{a_i^j}}{\sum_{j=1}^J \beta_j \prod_{i=1}^N (a_i^j + 1)^{-1}} \tag{6}$$

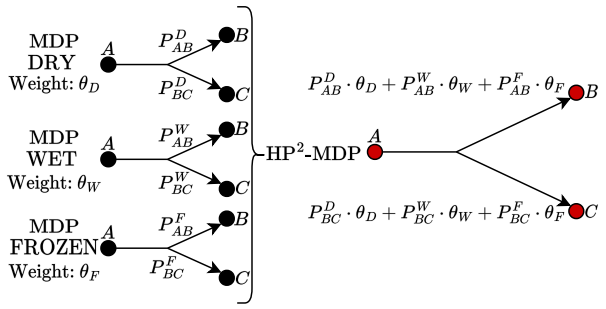


Figure 1: HP^2-MDP example with dependent parameters.

Since $P_0(\theta)$ is polynomial by definition, a simple induction argument yields that, for independent parameter sets, $P_{t+1}(\theta)$ is a polynomial of the form stated in Equation 6.

Global Parameters Bound by the Standard Simplex

In some scenarios, there will be dependencies between the latent parameters. The example below illustrates this.

Example 2 *Consider a robot planning a route over a domain with unknown weather conditions. We have three MDPs, shown on the left in Figure 1, respectively describing the motion of the robot in dry, wet and frozen weather conditions. The underlying true MDP is a combination of the three MDPs, depending on the wetness and the temperature of the domain. Each MDP contributes a different proportion to the true MDP, which is represented by weights θ_D, θ_W and θ_F. In the true MDP, the transition probability from state A to state B would be $P^T_{AB} = P^D_{AB} \cdot \theta_D + P^W_{AB} \cdot \theta_W + P^F_{AB} \cdot \theta_F$. The resulting HP^2-MDP is shown on the right side of Figure 1. To ensure that the transition function of the HP^2-MDP is well-formed, the latent parameters must satisfy the validity constraint $\theta_D + \theta_W + \theta_F = 1$, i.e., the latent parameters must lie in the standard 2-simplex.*

Closed-Form Belief Update When the parameters in $\mathcal{M}$ are of the form $\Theta = \{(\theta_1, \ldots, \theta_n) \in [0,1] \mid \sum_{i=1}^N \theta_i = 1\}$, i.e., Θ is the standard $(N-1)$-simplex, the denominator in Equation 4 is a closed line integral around the standard $(N-1)$-simplex. This is known as Dirichlet's integral on the simplex, which can be expressed as (cf. Equation (2.1) in (Gupta and Richards 2001)):

$$\oint_{\theta \in \Theta} \prod_{i=1}^N \theta_i^{a_i - 1} d\theta = \frac{\prod_{i=1}^N \Gamma(a_i)}{\Gamma\left(\sum_{i=1}^N a_i\right)}, \tag{7}$$

where $\Gamma(k) = (k-1)!$ is the gamma function. Therefore, the denominator of Equation 4 can be expressed as:

$$\oint_{\theta \in \Theta} \sum_{j=1}^J \beta_j \prod_{i=1}^N {\theta'_i}^{a_i^j} d\theta' = \sum_{j=1}^J \beta_j \cdot \frac{\prod_{i=1}^N \Gamma(a_i^j + 1)}{\Gamma\left(\sum_{i=1}^N (a_i^j + 1)\right)}. \tag{8}$$

Thus, the posterior belief over the latent parameter is:

$$P_{t+1}(\theta) = \frac{\sum_{j=1}^J \beta_j \prod_{i=1}^N \theta_i^{a_i^j}}{\sum_{j=1}^J \beta_j \cdot \frac{\prod_{i=1}^N \Gamma(a_i^j+1)}{\Gamma\left(\sum_{i=1}^N (a_i^j+1)\right)}}. \tag{9}$$

Since the Γ function is polynomial, again a simple induction argument yields that, for dependent parameter sets bounded by the standard simplex, $P_{t+1}(\theta)$ is a polynomial of the form stated in Equation 9.

BAMDP-L: A Special Case of HP^2-MDP

Other linear relationships between the global parameters can be expressed by applying a combination of Equations 5 and 7. For example, consider the case where every transition probability is independent, and thus each state-action-state transition probability is defined as a latent parameter, i.e., $\theta = \{\theta_{sas'} \in [0,1] \mid (s,a,s') \in S \times A \times S\}$, with $T_\Theta(s,a,s')(\theta) = \theta_{sas'}$. The latent parameter set Θ must satisfy the validity constraint, thus $\Theta = \{\theta \mid \sum_{s' \in S} \theta_{sas'} = 1 \text{ for all } (s,a) \in S \times A\}$. When the prior belief distribution over Θ is uniform, this is equivalent to BAMDP-L. Therefore, BAMDP-L can be seen as a special case of HP^2-MDP.

To see this, we first consider the posterior belief. Using Equation 4 and the uniform prior, the posterior belief can be expressed as:

$$\begin{aligned} P_t(\theta) &= \frac{P_t(\theta) T_\Theta(s_{t-1}, a_{t-1}, s_t)(\theta)}{\int_{\theta' \in \Theta} P_{t-1}(\theta') T_\Theta(s_{t-1}, a_{t-1}, s_t)(\theta') d\theta'} \\ &= \frac{P_0(\theta) \prod_{t'=0}^{t-1} T_\Theta(s_{t'}, a_{t'}, s_{t'+1}))(\theta)}{\int_{\theta' \in \Theta} P_t(\theta') \prod_{t'=0}^{t-1} T_\Theta(s_{t'}, a_{t'}, s_{t'+1}) d\theta'} \\ &= \frac{\prod_{t'=0}^{t-1} \theta_{s_{t'} a_{t'} s_{t'+1}}}{\int_{\theta' \in \Theta} \prod_{t'=0}^{t-1} \theta'_{s_{t'} a_{t'} s_{t'+1}} d\theta'} = \frac{\prod_{\substack{(s,a,s') \in \\ S \times A \times S}} \theta_{sas'}{}^{\alpha^t_{sas'}}}{\int_{\theta' \in \Theta} \prod_{\substack{(s,a,s') \in \\ S \times A \times S}} \theta'_{sas'}{}^{\alpha^t_{sas'}} d\theta'}, \end{aligned} \tag{10}$$

where $\alpha^t_{sas'}$ is the number of times the transition from state-action sa to new state s' has occurred until time t. The denominator of Equation 10 can be simplified to:

$$\int_{\theta' \in \Theta} \prod_{\substack{(s,a,s') \in \\ S \times A \times S}} \theta'_{sas'}{}^{\alpha^t_{sas'}} d\theta' = \prod_{\substack{(s,a) \in \\ S \times A}} \oint_{\theta_{sa.} \in \Theta_{sa}} \prod_{s' \in S} \theta_{sas'}^{\alpha^t_{sas'}} d\theta_{sa.}, \tag{11}$$

where for every state-action pair (s,a), we integrate over the standard simplex defined by $\Theta_{sa} = \{\theta_{sa.} \in [0,1]^{|S|} \mid \sum_{s' \in S} \theta_{sas'} = 1\}$, with $\theta_{sa.} = \{\theta_{sas'} \in [0,1] \mid s' \in S\}$. We can use Equation 7 to express the denominator of the

belief distribution as:

$$\prod_{\substack{(s,a)\in\\ S\times A}} \oint_{\theta_{sa.}\in\Theta_{sa}} \prod_{s'\in S} \theta_{sas'}^{\alpha^t_{sas'}} d\theta_{sa.} = \prod_{\substack{(s,a)\in\\ S\times A}} \frac{\prod_{s'\in S}\Gamma(\alpha^t_{sas'}+1)}{\Gamma\left(\sum_{s'\in S}(\alpha^t_{sas'}+1)\right)}. \tag{12}$$

Thus, the posterior belief over the latent parameter is:

$$P_t(\theta) = \frac{\prod_{(s,a,s')\in S\times A\times S} \theta_{sas'}{}^{\alpha^t_{sas'}}}{\prod_{(s,a)\in S\times A} \frac{\prod_{s'\in S}\Gamma(\alpha^t_{sas'}+1)}{\Gamma\left(\sum_{s'\in S}(\alpha^t_{sas'}+1)\right)}}. \tag{13}$$

This is equivalent to the posterior for a Dirichlet distribution used to solve a BAMDP-L.

Sampling the Belief Distribution The closed form expression shown in Equation 6 and 9 allows us to maintain an exact representation of the belief distribution for HP^2-MDPs where the transition probabilities are expressed as a polynomial function of the global parameters. To solve a HP^2-MDP, we can use the BAMCP algorithm, where we sample from the closed-form belief distribution at the root node instead of sampling from a weighted set of sampled MDPs. In HP^2-MDP, the number of parameters defines the number of dimensions of the space we sample from. At lower dimensions, we analytically compute the cumulative density function (CDF) of the posterior belief and sample from it to produce a possible parameter instantiation. At higher dimensions, we use a Markov Chain Monte-Carlo (MCMC) (Chib and Greenberg 1995).

Belief Distribution Dimension Reduction As the steps taken by the agent increase, the number of terms in the polynomial representing the belief increases as well. For problems with large horizons, the belief distribution may gain enough terms such that the MCMC algorithm becomes time-consuming. To minimise the time taken to sample the belief while maintaining an accurate representation, we apply dimension reduction to such problems.

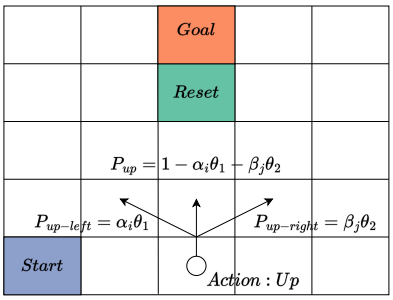
After a fixed number of steps, we use ordinary least-squares to fit our belief distribution to a polynomial of a lower degree, which is then used to represent the belief posterior. This enables the use HP^2-MDP to represent domains with large state spaces and sample from the belief accurately, whilst bounding the size of its polynomial representation.

Experiments

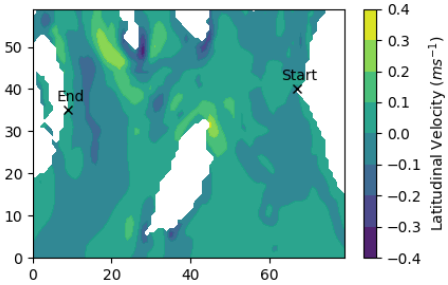
We use BAMCP (Algorithm 1) and apply HP^2-MDP, BAMDP-G and a particle-filter-based belief representation to two domains, showing empirically that HP^2-MDP results in better performance.

Algorithms

We apply the following algorithms for updating the belief: **HP^2-MDP**, as presented in Definition 2. We used dimension reduction on the second domain. **BAMCP-G**, as presented in Guez et al. (2014), which uses the approach described in



(a) Grid5 problem with an example transition. α_i and β_j are known to the agent.



(b) Latitudinal water velocity of the ocean for SeaGrid. The white regions represent land.

Figure 2: Domains used in experiments.

BAMDP-G to maintain the belief. **BAMCP-PF**, where we add noise into the samples when we update the posterior distribution. Similar to BAMCP-G, we sample the initial belief to generate the initial set of particles, $U = \{\mathcal{M}^1, \cdots, \mathcal{M}^M\}$. We assign a weight of $w_k(h_0) = \frac{1}{M}$ to particle $\mathcal{M}^k$. When an action-state pair has been observed, the weights are updated using Equation 2. Then, we use low-variance sampling to select M samples from the updated distribution. We add a small perturbation ($\mathcal{E} \sim \mathcal{N}(0, 0.01)$) to the sample values to allow the posterior distribution to explore the latent parameter space. The new sampled M particles are given the same weight of $\frac{1}{M}$. Between each step, the algorithms sample the belief and run simulations for time $\mathcal{T}$.

Domains

Grid5 Guez, Silver, and Dayan (2012) considered a 5×5 grid, where the only reward is located at the goal, directly next to a reset square. If the agent enters the reset square, they are sent to the start. The agent can take actions to navigate to neighbouring squares, and each action has a small probability of failure. The authors use Dirichlet distributions to model the unknown state transition probabilities, i.e., the problem is modelled as a BAMDP-L. Here, we modify it to introduce dependencies between the transitions.

We consider a 5×5 grid, as shown in Figure 2a. The transition probabilities of actions taken from the square at (i, j) are determined by global latent parameters $\theta = (\theta_1, \theta_2)$, and local known parameters $\{\alpha_i, \beta_j\}$. Every step has a cost of 1, and the run ends at the goal square. θ_1 and θ_2 both are in the range $[0, 1]$ and are independent from each other. The prior distribution over the latent parameters is uniform.

Algorithm	HP^2-MDP	BAMCP-G					BAMCP-PF				
M	–	100	200	300	400	500	100	200	300	400	500
Failure Rate	**0.0**	1.0	1.0	0.85	0.85	0.9	0.74	0.8	0.64	0.74	0.77
Mean Cost	**17.6± 0.435**	–	–	58.7± 3.64	50.3± 3.53	37.5± 3.32	49.6± 2.64	55.2± 2.85	48.8± 2.52	44.1± 2.73	41.8± 2.71

Table 1: Results from 100 simulation conducted in the SeaGrid domain. The bold values indicate the best performance.

SeaGrid We consider the problem outlined in Example 1, where a glider is planning a route from a start to a finish location. The domain is a 17×13 grid. The forecast for the longitudinal and latitudinal velocity of the water is known ahead of time, using data from Copernicus (2022). An example of the vertical velocity is shown in Figure 2b. The influence the horizontal and vertical water velocity will have on the motion of the glider is described by the two latent parameters, $\theta = (\theta_h, \theta_v)$. The prior over the θ is uniform. The agent has an action choice of the cardinal directions and the choice to do nothing. The choice of doing nothing may result in the glider being pushed along by the water.

Results

Grid5 We tested the HP^2-MDP without dimension reduction, BAMCP-G and BAMCP-PF algorithms on the Grid5 problem, and we set $\mathcal{T}$ to 20 seconds. We also varied the size of the set of samples used to represent the posterior distribution, M, for BAMCP-G and -PF. Each algorithm was tested on the domain 100 times, with the underlying "true" parameters for each test sampled uniformly. We ran the simulations on Linux OS with an Intel Core i9-11900H processor and 16GB of RAM.

The results are in Figure 3. The HP^2-MDP algorithm does not rely on sets of samples, thus the results do not vary with M. The BAMCP-G and -PF algorithms are shown to perform worse at low values of M, as the particles are sparsely distributed over the latent parameter space. As the value of M increases, the posterior distribution is better represented, and thus able to return actions closer to the optimal strategy. However, at high values of M, the time taken to sample the posterior distribution increases such that the number of trials that can be completed until timeout is reduced. This leads to the algorithm returning an action without converging to the optimal policy, and thus higher costs.

We found that the BAMCP algorithms with particle-based belief updates lead to higher costs than the HP^2-MDP closed-form updates, where we varied M between 10 and 200. BAMCP-PF performed marginally better than the BAMCP-G, as the particles were able to explore the parameter space. The BAMCP-PF generally resulted in higher costs than the HP^2-MDP algorithm, except in the case where $M = 150$, where the particle filter and the HP^2-MDP algorithm performed comparably. Both the BAMCP-G and -PF experience high variability in performance as M is altered. To get the best performance from these algorithms, the optimal value of M must be explored. In comparison, as HP^2-MDP is using an exact form of the posterior distribution, there is no need no fine-tuning needed.

SeaGrid We compared HP^2-MDP to BAMCP-PF and BAMCP-G with $M = 100, 200, 300, 400$ and 500. We applied dimension reduction to HP^2-MDP, where the highest degree of the polynomial was fixed to 12. We increased the number of particles used, as the transition probabilities were more sensitive to variation in the parameter values. This meant a slight deviation from the true parameter values would result in significantly different transition probabilities. Therefore, sample-based methods would struggle to fully capture the true MDP without large values of M. We set $\mathcal{T}$ to 90 seconds and ran 100 simulations for each algorithm, again with the underlying "true" parameters sampled uniformly. We ran the trials on CentOS with an Intel Xeon Platinum 8268 CPU at 2.90 GHz Processor, with 16GB of RAM. We applied a limit of 75 steps before halting the simulation due to memory constraints for BAMCP-PF with $M = 500$. This is reflected in the failure rate, shown in Table 1.

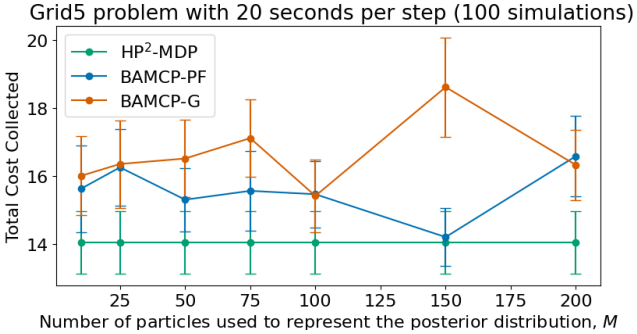


Figure 3: Costs collected in the Grid5 domain

We found that HP^2-MDP significantly outperforms the particle-based approaches. We attribute this to the size of the state space and the high dependence of the transition probabilities on the latent parameters. This meant that for low values of M, the samples were insufficient to cover the latent parameter space, resulting in poor performance. This was particularly pronounced in BAMCP-G, where the samples were fixed at the start of a trial, and thus the performance was heavily dependent on the existence of a sample similar to the underlying MDP.

Conclusion

We have presented a new class for BAMDPs, HP^2-MDP, where the transitions are expressed as polynomial functions of the latent parameters. HP^2-MDP allows the posterior distribution over the latent parameters to be maintained in a closed-form and exact representation. We have shown, in two domains, that planning with the HP^2-MDP closed-form posterior distribution substantially outperforms the commonly used approach of planning with a particle-based representation of the posterior distribution.

Acknowledgments

This work was supported by the Defence Science and Technology Laboratory, the EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. This document is an overview of UK MOD's Defence Science and Technology Laboratory (DSTL) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy.

References

Chen, M.; Frazzoli, E.; Hsu, D.; and Lee, W. S. 2016. POMDP-lite for robust robot planning under uncertainty. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 5427–5433.

Chib, S.; and Greenberg, E. 1995. Understanding the Metropolis-Hastings Algorithm. *The American Statistician*, 49(4): 327–335. Publisher: [American Statistical Association, Taylor & Francis, Ltd.].

Copernicus, M. S. 2022. Data | Copernicus Marine.

Costen, C.; Rigter, M.; Lacerda, B.; and Hawes, N. 2022. Shared Autonomy Systems with Stochastic Operator Models. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, 4614–4620. Vienna, Austria: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-1-956792-00-3.

Doshi-Velez, F.; and Konidaris, G. 2016. Hidden parameter markov decision processes: a semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, 1432–1440. New York, New York, USA: AAAI Press. ISBN 978-1-57735-770-4.

Duff, M. O. 2002. *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Process*. Ph.D. thesis, University of Massachusetts.

Fern, A.; Natarajan, S.; Judah, K.; and Tadepalli, P. 2014. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50: 71–104.

Grover, D.; Basu, D.; and Dimitrakakis, C. 2020. Bayesian Reinforcement Learning via Deep, Sparse Sampling. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, 3036–3045. PMLR. ISSN: 2640-3498.

Guez, A.; Heess, N.; Silver, D.; and Dayan, P. 2014. Bayes-Adaptive Simulation-based Search with Value Function Approximation. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc.

Guez, A.; Silver, D.; and Dayan, P. 2012. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.

Gupta, R. D.; and Richards, D. S. P. 2001. The History of the Dirichlet and Liouville Distributions. *International Statistical Review / Revue Internationale de Statistique*, 69(3): 433–446. Publisher: [Wiley, International Statistical Institute (ISI)].

Hahn, E. M.; Han, T.; and Zhang, L. 2011a. Synthesis for PCTL in Parametric Markov Decision Processes. *NASA Formal Methods - Third International Symposium, NFM 2011, Pasadena, CA, USA, April 18-20, 2011. Proceedings*, 146–161.

Hahn, E. M.; Han, T.; and Zhang, L. 2011b. Synthesis for PCTL in Parametric Markov Decision Processes. In Bobaru, M.; Havelund, K.; Holzmann, G. J.; and Joshi, R., eds., *NASA Formal Methods*, volume 6617, 146–161. Berlin, Heidelberg: Springer Berlin Heidelberg. ISBN 978-3-642-20397-8 978-3-642-20398-5. Series Title: Lecture Notes in Computer Science.

Hallak, A.; Di Castro, D.; and Mannor, S. 2015. Contextual Markov Decision Processes. Technical Report arXiv:1502.02259, arXiv. ArXiv:1502.02259 [cs, stat] type: article.

Hsiao, K.; Kaelbling, L. P.; and Lozano-Perez, T. 2007. Grasping pomdps. In *Proceedings 2007 IEEE International Conference on Robotics and Automation*, 4685–4692. IEEE.

Junges, S.; Abraham, E.; Hensel, C.; Jansen, N.; Katoen, J.-P.; Quatmann, T.; and Volk, M. 2019. Parameter Synthesis for Markov Models. Technical Report arXiv:1903.07993, arXiv. ArXiv:1903.07993 [cs] type: article.

Junges, S.; Jansen, N.; Katoen, J.-P.; Topcu, U.; Zhang, R.; and Hayhoe, M. 2018. Model Checking for Safe Navigation Among Humans. In McIver, A.; and Horvath, A., eds., *Quantitative Evaluation of Systems*, Lecture Notes in Computer Science, 207–222. Cham: Springer International Publishing. ISBN 978-3-319-99154-2.

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

Kocsis, L.; and Szepesvári, C. 2006. Bandit Based Monte-Carlo Planning. In Fürnkranz, J.; Scheffer, T.; and Spiliopoulou, M., eds., *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, 282–293. Berlin, Heidelberg: Springer. ISBN 978-3-540-46056-5.

Mausam; and Kolobov, A. 2012. Planning with Markov Decision Processes: An AI Perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1): 1–210.

Nanavati, A.; Mavrogiannis, C.; Weatherwax, K.; Takayama, L.; Cakmak, M.; and Srinivasa, S. 2021. Modeling Human Helpfulness with Individual and Contextual Factors for Robot Planning. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation. ISBN 978-0-9923747-7-8.

Puterman, M. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Ltd.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021a. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 11930–11938. Number: 13.

Rigter, M.; Lacerda, B.; and Hawes, N. 2021b. Risk-Averse Bayes-Adaptive Reinforcement Learning. In *Advances in Neural Information Processing Systems*, volume 34, 1142–1154. Curran Associates, Inc.

Ross, S.; Pineau, J.; Chaib-draa, B.; and Kreitmann, P. 2011. A Bayesian Approach for Learning and Planning in Partially Observable Markov Decision Processes. *The Journal of Machine Learning Research*, 12(null): 1729–1770.

Tamar, A.; Mannor, S.; and Xu, H. 2014. Scaling Up Robust MDPs using Function Approximation. In *Proceedings of the 31st International Conference on Machine Learning*, 181–189. PMLR. ISSN: 1938-7228.

Winkler, T.; Junges, S.; Pérez, G. A.; and Katoen, J.-P. 2019. On the Complexity of Reachability in Parametric Markov Decision Processes. *arXiv:1904.01503 [cs]*. ArXiv: 1904.01503.

Wray, K. H.; Witwicki, S. J.; and Zilberstein, S. 2017. Online Decision-Making for Scalable Autonomous Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, 4768–4774. Melbourne, Australia: International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3.

-

5.1 Further detail

For the interest of space, we did not include how the MCMC algorithm works, and how the belief distribution dimension reduction is done. We outline the algorithms used to implement these features here.

5.1.1 MCMC algorithm

We use a MCMC algorithm to generate samples from a closed-form posterior distribution with high parameter space dimensions. In a MCMC, a random sample Θ_1 is generated, and the value of the posterior distribution $P(\Theta_1)$ at this point is calculated. This is added to the set of samples. Then, a new random sample Θ_2 is generated, and has a posterior distribution value of $P(\Theta_2)$. If $P(\Theta_2) > P(\Theta_1)$, so the second sample has a higher likelihood of being sampled, and Θ_2 is added to the set of samples. If $P(\Theta_2) < P(\Theta_1)$, Θ_2 is accepted with a probability of $\frac{P(\Theta_2)}{P(\Theta_1)}$, otherwise, the previous sample, Θ_1 is added to the set of samples. The full algorithm is shown in Algorithm 1.

5.1.2 Belief Distribution Dimension Reduction

The dimension reduction is necessary to reduce the complexity of the belief functions for problems with long horizons, and a larger number of observations. We outline how the dimension reduction works in practice for the case where Θ is a one dimensional space for simplicity.

Algorithm 1 Markov Chain Monte-Carlo

Function{MCMC}{$P(\Theta)$}
$\Theta_1 \sim$ `Random(N)`
$\Theta_{i-1} \leftarrow \Theta_1$
$Samples \leftarrow [\Theta_1]$
repeat
  $\Theta_i \sim$ `Random(N)`
  $ratio = P(\Theta_i)/P(\Theta_{i-1})$
  if $ratio > 1$ **then**
    $Samples.Append(\Theta_i)$
    $\Theta_{i-1} \leftarrow \Theta_i$
  else if $ratio >$ `Random()` **then**
    $Samples.Append(\Theta_i)$
    $\Theta_{i-1} \leftarrow \Theta_i$
  else
    $Samples.Append(\Theta_{i-1})$
    $\Theta_i \leftarrow \Theta_{i-1}$
  end if
until `Timeout()`
return *Samples*
EndFunction

The posterior distribution $P_t(\theta)$ at time t with the order J can be expressed as:

$$P_t(\theta) = P(\theta) = (1, \theta, \theta^2, \cdots \theta^J) \cdot \begin{pmatrix} \beta_0 \\ \beta \\ \vdots \\ \beta_J \end{pmatrix} \doteq \overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J}, \tag{5.1}$$

where $\overrightarrow{B}_J$ is the vector of coefficients for the polynomial $P_t(\theta)$.

We aim to fit this function to a polynomial with an order J', where $J > J'$. Using ordinary least squares (OLS) formula, the fitted polynomial can be expressed as:

$$P_t(\theta) = \overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J} = \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}} + \epsilon(\theta), \tag{5.2}$$

where $\epsilon(\theta)$ is the error term. In OLS, we aim to minimise the magnitude of the error over

the space of the parameters, and we can express the magnitude of the error as:

$$\begin{aligned}\epsilon^2 &= \int_0^1 \left(\overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J} - \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}}\right)^2 d\theta \\ &= \int_0^1 \left(\overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J} - \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}}\right)^{\top} \cdot \left(\overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J} - \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}}\right) d\theta \\ &= \int_0^1 \overrightarrow{B_J}^{\top} \cdot \overrightarrow{\theta}_J \cdot \overrightarrow{\theta}_J^{\top} \cdot \overrightarrow{B_J} - 2\overrightarrow{B}_J^{\top} \cdot \overrightarrow{\theta}_J \cdot \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}} + \overrightarrow{B_{J'}}^{\top} \cdot \overrightarrow{\theta}_{J'} \cdot \overrightarrow{\theta}_{J'}^{\top} \cdot \overrightarrow{B_{J'}} d\theta\end{aligned} \tag{5.3}$$

To find the vector of coefficients that minimises the error, we take the derivative of the error with respect to the vector of coefficients $\overrightarrow{B_{J'}}$ and set it to zero.

$$\frac{d\epsilon^2}{d\overrightarrow{B_{J'}}} = -2\overrightarrow{B}_J^{\top} \int_0^1 \overrightarrow{\theta}_J \cdot \overrightarrow{\theta}_{J'}^{\top} d\theta + 2\overrightarrow{B_{J'}}^{\top} \int_0^1 \overrightarrow{\theta}_{J'} \overrightarrow{\theta}_{J'}^{\top} d\theta = 0. \tag{5.4}$$

We can define the matrix $\Phi_{p,q} = \int_0^1 \overrightarrow{\theta}_p \cdot \overrightarrow{\theta}_q^{\top} d\theta$, where:

$$\begin{aligned}\Phi_1^{p\times q} \doteq \int_0^1 \overrightarrow{\theta}_p \cdot \overrightarrow{\theta}_q^{\top} d\theta &= \int_0^1 \begin{pmatrix} 1 \\ \theta \\ \vdots \\ \theta^p \end{pmatrix} \cdot (1, \theta, \cdots, \theta^q) d\theta \\ &= \int_0^1 \begin{pmatrix} 1 & \theta & \cdots & \theta^q \\ \theta & \theta^2 & \cdots & \theta^{q+1} \\ \vdots & & \vdots & \\ \theta^p & \theta^{p+1} & \cdots & \theta^{p+q} \end{pmatrix} d\theta = \begin{pmatrix} 1 & \frac{1}{2} & \cdots & \frac{1}{q+1} \\ \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{q+2} \\ \vdots & & \vdots & \\ \frac{1}{p+1} & \frac{1}{p+2} & \cdots & \frac{1}{p+q+1} \end{pmatrix}\end{aligned} \tag{5.5}$$

We can use this matrix to simplify Equation 5.4 to:

$$\begin{aligned}\overrightarrow{B}_J^{\top} \cdot \Phi_{J,J'} &= \overrightarrow{B}_{J'}^{\top} \cdot \Phi_{J',J'} \\ \overrightarrow{B}_{J'} &= \overrightarrow{B}_J \cdot \Phi_{J,J'} \cdot \Phi_{J',J'}^{-1}\end{aligned} \tag{5.6}$$

The new co-efficient vector can be used to represent the posterior distribution at time t

with the order J'.

5.2 Using HPP-MDPs to represent shared autonomy systems

While my paper does not explicitly mention shared autonomy systems, the framework was originally considered to represent the continuous space of behaviours an operator can exhibit. Consider a human operator that could be in any linear combination of N MDPs, such that the transition dynamics are governed by $T_{human}(s, a, s') = \sum_{i \in N} T_i(s, a, s') \cdot \theta_i$, where θ_i is the transition function from MDP i, $\sum_{i \in N} \theta_i = 1$, and $a \in A_{human}$ is the set of actions the human can do in the environment. Given that this human operator is in a shared autonomy system with an autonomous operator, whose transition dynamics are governed by $T_{auto}(s, a, s')$, we can define the shared autonomy system as the HPP-MDP $\mathcal{M}$. $\mathcal{M}$ is defined as $\langle S, s_0, A, \Theta, P_\Theta, T_\Theta, C, G \rangle$, where S is the set of states, s_0 is the initial state, $A = A_{human} \cup A_{auto}$ is the union of the set of actions the human can take, and the set of actions the autonomous operator can take. Θ is the space of latent parameters, defined over the N-simplex, and $P_0(\theta)$ is the initial prior over the parameter space. The transition function is $T_\Theta(s, a, s') = T_{human}(s, a, s') \forall a \in A_{human}$ and $T_\Theta(s, a, s') = T_{auto}(s, a, s') \forall a \in A_{auto}$. $\mathcal{M}$ can then be solved as laid out in the paper, and the optimal policy can be used to inform the agent which operator should be in control, and what action they should take in the environment.

5.3 Limitations

There are two main limitations to the HPP-MDP framework proposed in the paper. The first is that the framework assumes that the transition dynamics of the environment can be represented by a polynomial function. While some functions may have transition dynamics

that may be approximated using polynomial functions, piece-wise functions may not be approximated using polynomials. For example, exponential transition functions can be approximated with polynomial functions through Taylor series expansion. However, piece-wise transition functions that significantly change between the different regions cannot be approximated sufficiently using polynomial functions.

The second is that the underlying dynamics is assumed to be stationary during execution. This assumption may be unrealistic in domains with a longer horizon, such as where the dynamics describe the weather conditions. Here, the dynamics may change stochastically over time, so the planning done by the agent at the beginning of the mission may not be optimal by the end of the mission. For the shared autonomy system, the actions the agent chooses may influence the operator's behaviour, and thus change the transition dynamics for the operator. An example of this is where the agent chooses the human operator to perform a task, and this contributes to their tiredness. This will result in a degradation of the operator's performance over time. However, it is difficult to model how a continuous parameter changes over time, while maintaining a closed-form belief distribution over the parameter space.

Chapter 6

Planning for Uncertain Operators with Continuous Dynamics in Shared Autonomy Systems

While the HPP-MDP allows us to express the continuous range of possible human behaviour in a shared autonomy system, the framework does not allow us to represent how human behaviour may change during a mission. In this paper, we look at how to represent the hidden, continuous and dynamic nature of human behaviour in shared autonomy systems. We use a continuous hidden state MOMDP (cont-MOMDP) for the framework, where the continuous hidden state space describes the possible human behaviour, and the discrete observable state space describes the domain. However, unlike the HPP-MDP, we are unable to maintain a closed-form probability distribution over the hidden state space. This is because the transition functions in the cont-MOMDP are discontinuous, and we use a particle filter to maintain the belief over the hidden state space instead.

However, the cont-MOMDP formulation assumes that the transition dynamics of the human's internal state is known prior to execution. This assumption may not be feasible in practice, as the transitions are inherently hidden, and thus difficult to observe. We

therefore present a framework using Bayes-Adaptive MOMDPs (BA-MOMDP), where the agent learns the transition dynamics of the human's internal state. We achieve this by discretising the human hidden state space, and parameterise the internal transition probabilities.

We compare the two frameworks on the UAV domain presented in Chapter 4. We consider two cases, where the agent in the cont-MOMDP framework knows and does not know the true hidden human transition dynamics. We found that the cont-MOMDP outperforms the BA-MOMDP when the hidden dynamics are known, while the BA-MOMDP performs better when the hidden dynamics are unknown.

We submitted this paper to IJCAI 2025 and the paper is currently under review.

Table 6.1: Summary of the papers and their features

Chapters	4	5	6	7
SA	Yes	No	Yes	Yes
SA style	N operators controls one robot	N/A	One human and one autonomous agent controls a robot	one human and K robots, where the human can take control of any robot
Dynamic	Yes	No	Yes	No
Continuous	No	Yes	Yes	Yes
Closed-form	Yes	Yes	Yes	Yes

Planning for Uncertain Operators with Continuous Dynamics in Shared Autonomy Systems

Clarissa Costen[1], **Bruno Lacerda**[1], **Nick Hawes**[1]
[1]Oxford Robotics Institute, University of Oxford
{clarissa, bruno, nickh}@robots.ox.ac.uk

Abstract

Shared autonomy systems must effectively model and reason over human behaviour, which is inherently irrational. Furthermore, humans do not have a single way of acting. Their behaviour is also variable, and is influenced by factors that may change during a run. Theese factors (e.g. tiredness, expertise) are difficult to observe directly, and tend to evolve along a continuous spectrum, rather than discretely moving between states (e.g. tired, not tired; novice, expert). In this paper, we attempt to represent and exploit the hidden, continuous and dynamic nature of human behaviour in the context of shared autonomy systems. We present a framework for shared autonomy systems using a continuous-state mixed observability Markov decision process (MOMDP) to capture these three features of human behaviour. The continuous-state MOMDP has a hidden state space, and we use this to represent the space of possible human behaviour. However, the evolution of a human's internal state cannot be observed directly, and thus the transition dynamics for human behaviour is difficult to know exactly. We therefore propose a framework using mixed-observability Bayes-adaptive Mixed Observability MDP (BA-MOMDP) for shared autonomy systems where the transition dynamics for a specific human are learnt as that human interacts with the shared autonomy system. We evaluate and compare the two frameworks, and consider the scenario where the assumptions made on the continuous-state MOMDP are different to the ground truth.

1 Introduction

As the capabilities of artificial intelligence systems advance, autonomous robots are increasingly integrated into our lives, leading to more frequent human-robot interactions. One notable form of such interactions is found in shared autonomy systems (SAS), where the control of a robot is divided between humans and autonomous controllers. Generally in these systems, autonomous controllers handle simple and low-risk tasks, while humans manage higher-risk tasks. The agent of the SAS determines the appropriate operator for each task, relying on the models of the operators' performance to make informed decisions.

However, many existing SAS assume that humans are perfect controllers [Rigter *et al.*, 2020]. This assumption can be problematic, as the SAS agent does not consider human limitations in its decision-making, potentially assigning tasks beyond their capabilities. To address the possibility of human error, Basich *et al.* [2021] and Cubuktepe *et al.* use Markov decision processes (MDPs) to model human behaviour. This approach frames the problem as a sequential decision-making process, allowing the agent to consider how the current choice of operator affects future states. However, this framework assumes uniform human behaviour, and does not consider the latent, continuous, and stochastic factors (such as skill level and fatigue [Wu and Durfee, 2010]) that affect human performance. To model more complex human behaviour, two main approaches are used. The first approach employs partially observable MDPs (POMDPs) [Costen *et al.*, 2022], where a hidden discrete state space represents the latent and stochastic nature of factors influencing human behaviour. The discrete states space of the POMDP (which describes the stochastic nature) implicitly assumes that a human changes discretely between internal states (e.g., from rested to tired or from novice to expert). This often means that the belief over the human internal state does not converge to a specific state as more observations are performed, since the probability of having a discrete state that actually exactly matches the human is almost surely zero. This can lead to unstable behaviour by the agent, who does not fully consider the capabilities of the human.

The second approach uses Bayes-Adaptive MDPs (BAMDPs) [Costen *et al.*, 2024], where the latent parameter space represents a set of continuous and static factors. The continuous latent space allows the agent to consider the full spectrum of possible behaviour, and determine the best policy given their behaviour. However, if the human changes behaviour (e.g. gets tired), the agent will no longer be able to make the optimal action choice, as their belief of the human is incorrect. Although each approach models different aspects of human behaviour, neither can fully capture both the continuous and stochastic nature of these latent factors.

In this paper, we attempt to model both the continuous and stochastic nature of human behaviour. Our first contribution is the introduction of a continuous mixed-observability MDP

framework for shared autonomy systems. In this framework, the hidden continuous state space represents factors that influence human performance. Maintaining a belief representation in a closed-form is challenging because the transition function (which maps from one continuous space to another) is discontinuous. To address this discontinuity, we employ a particle filter to maintain a belief over the hidden continuous state space.

However, determining the transition dynamics of a human's internal state is often difficult. For example, consider a continuous 'tiredness' factor in $[0, 1]$ that affects performance. Although it is straightforward to identify humans at the extremes (0 being not tired and 1 being very tired), it is challenging to observe and quantify intermediate levels of tiredness and how they evolve over time. Our second contribution addresses this issue by proposing a mixed-observability BAMDP framework for shared autonomy. In this framework, a discrete hidden state space exists within the BAMDP, and we account for uncertainty in the transition dynamics between hidden states.

We apply this framework to a ground-truth mixed-observability MDP and compare the performance of the frameworks under known and unknown human internal transition dynamics.

2 Related Work

Shared Autonomy SAS allows the control of a robot to be shared between multiple operators, typically a mix of human and autonomous operators. These systems aim to reduce the workload on human operators by transferring control to autonomous systems, and they also increase the robustness of autonomous systems by enabling them to hand over control to humans. These autonomous operators can perform tasks to minimise human interventions [Rigter *et al.*, 2020; Duchetto *et al.*, 2018] or take control when human operators are unable to make informed decisions [Cubuktepe *et al.*, 2021]. However, many approaches that seek to minimize human intervention assume perfect human performance [Basich *et al.*, 2020; Duchetto *et al.*, 2018; Rigter *et al.*, 2020]. This assumption is unrealistic, as human operators may not control the robot flawlessly, which can cause the robot to enter a failure state [Dahiya *et al.*, 2022], or they may lack the necessary skills to complete a task successfully [Charles *et al.*, 2018].

To better model the stochastic nature of human behaviour, humans are represented through transition functions in an MDP [Puterman, 1994]. The reward function in an MDP is defined by task completion success [Wu *et al.*, 2017] or penalties from human intervention [Basich *et al.*, 2020]. The SAS agent aims to maximise the expected total reward over a mission and solves the MDP to determine whether the human or the autonomous operator should control the system for the next task. However, human behaviour is influenced by factors such as fatigue and trust levels [Wu and Durfee, 2010], resulting in variability in their performance. [Wu and Durfee, 2010; Feng *et al.*, 2016] and [Mahmud *et al.*, 2023] employ multiple MDP models to describe a single human operator in a shared autonomy system. [Feng *et al.*, 2016] consider the impact of tiredness on a human operator, using two MDPs to represent behaviour when the operator is normal and when tired. The agent monitors the human's internal state by tracking the number of tasks completed. Once this count exceeds a threshold, the agent assumes that the human is in a tired state.

Factors influencing human behaviour range from the operator's goal selection for the robot [Fern *et al.*, 2014; Javdani *et al.*, 2015] to their attentiveness to the system [Wray *et al.*, 2016]. These factors are often difficult to measure or observe directly. To address this, we must consider how to model MDP problems with hidden factors.

Modelling Shared Autonomy Systems with Hidden Factors In sequential decision-making problems, agents often lack access to all the information necessary for optimal decision-making. In this paper, we focus on two characteristics of hidden information: whether it changes during a mission (stochasticity) and whether it consists of discrete or continuous values.

If the unobservable factors govern the transition dynamics and can change during a mission, the agent must maintain a belief over this hidden and evolving factor. Partially observable MDPs (POMDPs) [Spaan and Vlassis, 2004] and mixed observability MDPs (MOMDPs) [Araya-López *et al.*, 2010] provide frameworks for such problems, where there is a discrete set of hidden states evolving stochastically. Wray *et al.* use POMDPs and Costen *et al.* use MOMDPs to describe shared autonomy systems. They maintain a belief over a human's internal state, and the belief is updated through observations and interactions, such as eye-tracking data [Petric and Kovacic, 2020]. However, the POMDP approach assumes that the set of hidden factors is discrete, which is not realistic for modelling human factors, which are typically not binary.

If an agent has a continuous set of potential MDPs that they could be in (but does not know which one), the hidden information is static and continuous. An example of such an problem is if the agent has no prior knowledge of the transition probabilities between states, and these probabilities can vary continuously from 0 to 1. BAMDPs [Duff, 2002] address this problem by describing a situation in which a continuous set of latent parameters governs the transition functions. In this case, the latent parameter remains static, and the agent maintains a belief over this parameter space.

If the agent lacks a prior over transition probabilities in an MDP, they can maintain a continuous closed-form belief distribution over latent parameters [Duff, 2002]. In problems where transition probabilities can be represented by polynomial functions with latent parameters as variables, we can also maintain a closed-form belief distribution [Costen *et al.*, 2023]. Typically, when latent continuous parameters correspond to a continuous set of MDPs, the agent uses a sampled set of MDPs to represent the belief [Guez *et al.*, 2014].

BAMDPs are used to represent the continuous nature of factors that affect humans in shared autonomy systems [Nanavati *et al.*, 2021; Costen *et al.*, 2024]. Nanavati *et al.* use latent parameters in the BAMDP to describe the unknown "helpfulness" of a human, while Costen *et al.* use latent parameters to represent the skill level of a human operator. In both papers, the agent learns about the latent parameters through interaction with the human, enabling better decision-making based on the updated belief.

However, these approaches make unrealistic assumptions

that latent parameters remain static, implying fixed levels of helpfulness and skill. If the latent continuous parameters change during a mission, this can be modelled by continuous POMDPs [Sunberg and Kochenderfer, 2018; Ross *et al.*, 2008]. In such problems, the hidden state space is continuous and evolves over the mission. Sunberg and Kochenderfer and Ross *et al.* use particle filters to maintain a posterior distribution over the true state of the system. These particle filters are initialized by sampling the initial belief and then stochastically evolving the particles according to predefined transition functions. We apply this continuous POMDP model to create the first shared autonomy systems to describe the continuous, dynamic, and hidden nature of factors affecting human performance.

3 Preliminaries

MDPs are used in sequential decision-making problems, where the decisions made in the current state may affect the outcome of future states. They are commonly used to describe the behaviour of operators (human and autonomous) in shared autonomy domains.

Definition 1 (Markov Decision Process)**.** *A Markov decision process (MDP) is defined by the tuple $M = \langle S, \overline{s}, A, T, R, \gamma \rangle$, where:*

- *S and $\overline{s} \in S$ are the set of states and initial state, respectively;*
- *A is the set of actions;*
- *$T : S \times A \times S \to [0, 1]$, where $T(s, a, s)$ is the probability of transitioning to state s' after taking action a in state s;*
- *$R : S \times A \to \mathbb{R}$ is the reward function;*
- *$\gamma \in (0, 1]$ is the discount factor.*

The goal of the agent is to take the action that maximises the expected cumulative reward.

MDPs assume that the current state is visible to the agent. However, when considering human behaviour, the factors that affect the outcome of actions may be hidden from the agent. Furthermore, the discrete state space does not allow us to capture the continuous nature of human behaviour. In this paper, we use a continuous-state mixed-observability MDP to describe the hidden, stochastic and continuous features of the range of behaviours from the human operator.

Definition 2 (Cont-MOMDP)**.** *A continuous-hidden-state mixed observability MDP (cont-MOMDP) is defined as the tuple $\mathcal{M}_C = \langle S_o, S_h, \overline{s}_o, b_0, A, \Omega, T, O, R, \gamma \rangle$, where:*

- *S_o is the discrete set of observable sub-states; and S_h is the continuous set of hidden sub-states. A single state is defined by a pair of an observable and a hidden sub-state, $(s_o, s_h) \in S_o \times S_h$;*
- *$\overline{s}_o \in S_o$ is the initial observable sub-state;*
- *$b_0 : S_h \to \mathcal{R}$ is the initial probability density function of the belief distribution over the hidden sub-state space. Thus:*

$$\int_{s_h \in S_h} b_0(s_h) ds_h = 1; \tag{1}$$

- *Ω is the discrete set of observations;*
- *$T : S_o \times S_h \times A \times S_o \times S_h \to [0, 1]$ is the probabilistic transition function;*
- *$O : S_o \times S_h \times A \times \Omega \to [0, 1]$ is the observation function, where $O(s_o, s_h, a, \omega)$ is the probability of observing ω after taking action a, and entering state (s_o, s_h);*
- *A, R and γ are as defined for a MDP.*

In a cont-MOMDP, while there is uncertainty over the hidden state factors, the transition dynamics are assumed to be known. When there is uncertainty over the transition dynamics but the state is observable, the problem can be described as an uncertain MDP. In an uncertain MDP, a latent parameter $\lambda \in \Lambda$ governs the transition probabilities, $T_\lambda(s, a, s')$. The agent has a prior $P_0(\lambda)$ over the parameter space Λ, and this distribution is updated using the history. For example, at time t, given the agent has observed a history of $h_t = s_0 a_0 \dots a_{t-1} s_t$, the posterior distribution is $P_t(\lambda) \propto P_0(\lambda) P(h_t|\lambda)$, updated using Bayes rule. To solve an uncertain MDP, we frame the problem as a BAMDP, where the history is factored into the state space, and the expected transition dynamics evolves with the history.

Definition 3 (Bayes-Adaptive MDP)**.** *A Bayes-adaptive MDP (BAMDP) is defined by the tuple $\langle S^+, \overline{s^+}, A, \Lambda, P_0(\lambda), T^+, R, \gamma \rangle$, where:*

- *$S^+ = S \times \mathcal{H}$, where S is the discrete set of states, and $\mathcal{H}$ is the set of possible histories;*
- *$\overline{s^+} = (\overline{s}, \overline{h})$, where $\overline{s}$ is the initial state, and $\overline{h}$ is the empty history;*
- *$\lambda \in \Lambda$ is the continuous set of latent parameters that define the transition dynamics;*
- *$P_0(\lambda)$ is the prior over Λ;*
- *$T^+((s, h_t), a, (s', h_t a s')) = \int_{\lambda \in \Lambda} T_\lambda(s, a, s') P_t(\lambda) d\lambda$ is the transition function, where $P_t(\lambda) \propto P_0(\lambda) P(h_t|\lambda)$;*
- *R, A, γ are as defined for an MDP.*

If there is no prior over the transition function in the BAMDP, every possible transition probability corresponds to a latent parameter, where $T(s, a, s') = \lambda^a_{s,s'} \forall s, s' \in S, \forall a \in A$. We use Dirichlet distributions to maintain a belief over the latent parameters [Duff, 2002]. The Dirichlet distribution is a probability distribution over the parameters of multinomial distributions [Ross *et al.*, 2007]. For example, consider a state-action pair (s, a), where there is no prior over the transition probabilities. For the n possible successor states, the number of transitions observed to the ith successor state is defined as ϕ^a_{s,s_i}. The posterior distribution over the transition probabilities is defined by the Dirichlet distribution $\text{Dir}(\phi^a_{s,s_1}, \dots, \phi^a_{s,s_n})$. The Dirichlet distribution is updated as new transitions are observed, and the expected transition probability for the state s and action a to result in state $s_i \forall i \in \{1, \dots, n\}$ is given by:

$$\mathbb{E}[\lambda^a_{s,s'}] = \frac{\phi^a_{s,s'} + 1}{\sum_{s''} \phi^a_{s,s''} + 1}. \tag{2}$$

As the number of observations increases, the variance of the Dirichlet distribution reduces, representing a narrowing of the uncertainty over the transition probabilities.

4 Modelling Humans

We consider a shared autonomy system where the human operator's behaviour is dynamic and the set of possible behaviours exist in a continuous range. An example of such a factor is tiredness, as it evolves during a run (caused by asking the human to complete difficult tasks), and humans become progressively more tired, rather than suddenly transitioning from "not tired" to "tired". In this section, we consider a human model that can express K different behaviours, and the human is able to exhibit multiple behaviours in the same mission (e.g. "tired" and "inattentive"). We start by building a model of how the human operator performs in the shared autonomy system.

First, we model each human behaviour as an MDP, resulting in K MDPs, $M_1, M_2, \dots M_K$.

Definition 4 (Single human behaviour)**.** *The MDP $M_i \forall i \in [1, \dots, K]$ describes the ith behaviour of a human in the SA system. All MDPs share the same state space S_o and A_h, which describes the domain the robot is operating in and the actions available to the human operator respectively. For the ith MDP, M_i, the transition function is defined by $T_i : S_o \times A_h \times S_o \to [0, 1]$, describing the probability of a transition given that a human only exhibits the ith behaviour. The MDPs also share the same reward function R and discount factor γ.*

Next, we consider how a human that is not exhibiting a single behaviour would act. We use $\Theta = \{(\theta_1, \dots \theta_K) \in [0, 1]^K \mid \sum_{i=1}^{K} \theta_i = 1\}$ to describe the space of possible behaviours, where θ_i defines the contribution of that behaviour. The space of K behaviours form a $(K-1)$ simplex, and we linearly scale the transition probabilities according to the values of $\theta_i \forall i \in [1, \dots, K]$. Therefore, the transition function to describe the probability the human transitioning to state s'_o from state s_o doing action a when their behaviour is described as $\overrightarrow{\theta} = (\theta_1, \dots, \theta_K) \in \Theta$ is:

$$T(s_o, a, s'_o | \overrightarrow{\theta}) = \sum_{i=1}^{K} T_i(s_o, a, s'_o) \cdot \theta_i. \tag{3}$$

We also consider how $\overrightarrow{\theta}$ evolves during an episode. The parameter θ_i evolves depending on the action and the state of the human. Given that the human is in state s and does action a, we express the stochastic change as a Gaussian distribution on the subspace $\sum_i \theta_i = 0$.

$$\Delta \overrightarrow{\theta} \sim \mathcal{N}(\mu_{s_o,a}, \Sigma) \tag{4}$$

$$\tag{5}$$

The successive value of $\theta_{i'}$ is bounded in the range of $[0, 1]$, so the new value of $\theta_i \forall i \in [1, \dots K]$ is

$$\theta'_i = \min(\max(\theta_i + \Delta\theta_i, 0), 1). \tag{6}$$

However, the discontinuous transition can result in a new value of $\overrightarrow{\theta}'$ where $\sum_{i=1}^{K} \theta'_i \neq 1$. In such states, we normalise the vector, so the new vector is $\overrightarrow{\theta}'' = \overrightarrow{\theta}' / (\sum_{i=1}^{K} \theta'_i)$.

This can also be expressed as:

$$\begin{aligned} T(\overrightarrow{\theta}, \overrightarrow{\theta'} | s_o, a) = {} & \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{((\overrightarrow{\theta}' - \overrightarrow{\theta}) - \mu_{s,a})^2}{2\sigma^2}} \\ & + \int_0^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{((\overrightarrow{\theta}''(\alpha) - \overrightarrow{\theta}) - \mu_{s,a})^2}{2\sigma^2}} d\alpha \end{aligned} \tag{7}$$

where $\overrightarrow{\theta}''(\alpha) = \arg\min_{\theta''}(\alpha \cdot \overrightarrow{\theta}' - \overrightarrow{\theta}'')$ and $\sum_{i=1}^{K} \theta''_i = 1$. The first half of Equation 7 describes the probability of transition given that there was no normalisation, and the second half considers the set of points that would be normalised to $\overrightarrow{\theta}'$.

Definition 5 (Human Cont-MOMDP)**.** *We describe the human operator's behaviour in a shared autonomy system using a cont-MOMDP $\mathcal{M}_{C_h} = \langle S_o, S_h, \overline{s}_o, b_0, A_h, \Omega, T_h, O, R, \gamma \rangle$, where:*

- *$S_h = \Theta$ is the space of possible human behaviour;*
- *$\omega \in \Omega = S_o$, the set of observations is the set of observable states;*
- *$b_0(\overrightarrow{\theta})$ is the prior over the hidden state space;*
- *The transition function dynamics is given by*

$$T_h(s_o, \overrightarrow{\theta}, a, s'_o, \overrightarrow{\theta}') = T(s_o, a, s'_o | \overrightarrow{\theta}) \cdot T(\overrightarrow{\theta}, \overrightarrow{\theta}' | s_o, a); \tag{8}$$

- *The observation function is given by*

$$O(s_o, \overrightarrow{\theta}, a, \omega) = \begin{cases} 1 & \text{if } s_o = \omega \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

- *R and γ are the reward function and discount factor of the MDPs $[M_1, \dots M_K]$.*

When the system is controlled by an autonomous operator, we can describe their performance as an MDP, $M_{auto} = \langle S_o, \overline{s}_o, A_{auto}, T_{auto}, R, \gamma \rangle$, where S_o is the state space of the robot and A_{auto} is the set of actions available to the autonomy. The transition function $T_{auto} : S_o \times A \times S_o \to [0, 1]$ describes the probability that the robot will transition to state s'_o from state s_o through action a. The reward and discount factor γ is the same as in $\mathcal{M}_{C_h}$.

We combine the human model $\mathcal{M}_{C_h}$ and the autonomy model M_{auto} to create a unified model of the shared autonomy system.

Definition 6 (SAS Cont-MOMDP)**.** *The shared autonomy system (SAS) is described as a cont-MOMDP, $\mathcal{M}_{C_{SA}} = \langle S_o, S_h, \overline{s}_o, b_0, A, \Omega, T, O, R, \gamma \rangle$, where the state spaces, the initial belief distribution, the observation and reward functions are the same as $\mathcal{M}_{C_h}$. The action space $A = A_{auto} \cup A_h$ is the union of set of actions available to the autonomous operator and human, and the transition function dynamics is given by:*

$$T(s_o, \overrightarrow{\theta}, a, s'_o, \overrightarrow{\theta}') = \begin{cases} T_{auto}(s_o, a, s'_o) & \text{if } a \in A_{auto} \\ T_h(s_o, \overrightarrow{\theta}, a, s'_o, \overrightarrow{\theta}') & \text{if } a \in A_h \end{cases} \tag{10}$$

The prior, the observation function, the reward function and the discount factor γ are the same as those in $\mathcal{M}_{C_h}$.

As the agent observes the human interacting with the robot, it maintains a belief distribution over the human's internal state, and uses this to inform the SAS agent's decision making. The aim of the SAS is to maximise the infinite horizon discounted cumulative reward collected. By fully solving Cont-MOMDP, the agent can optimally balance the exploration required to learn about the human operator while exploiting the knowledge it has to maximise the reward collected.

We are unable to maintain a closed-form belief distribution, as the transition function (as shown in Equation 6) is discontinuous. Therefore, to maintain a belief distribution over the hidden continuous states, we use a particle filter. As the number of particles N used to maintain the belief distribution increases, the particle filter converges to the true belief distribution over the hidden state space. However, particle-based methods can become computationally expensive as the number of particles increases.

5 Modelling Unknown Human Behaviour

The shared autonomy system introduced in the previous section assumes that the transition dynamics for the hidden state space is known. However, in practice, it is difficult to know how the human's internal state evolves, as we are not able to observe this directly. Furthermore, the rates of change the human may experience is unique to the individual. In this section, we consider how we can learn the internal state's hidden transition dynamics of the human through observations.

5.1 BA-MOMDP

BAMDPs allows us to express problems where there is uncertainty over the transition probabilities between states by parametrising the model with latent continuous parameters $\overrightarrow{\theta}$. However, in our scenario, there is uncertainty over the human's hidden state, as well as uncertainty over the transition probabilities between each hidden state. To solve this problem, we construct a Bayes-adaptive MOMDP (BA-MOMDP) from the SAS Cont-MOMDP.

Definition 7 (Bayes-adaptive MOMDP)**.** *The BA-MOMDP is defined by the tuple* $\mathcal{M}_B = \langle S_o, \tilde{\Theta}, A, \Lambda, T, \Omega, O, b_0, P_0(\lambda), R, \gamma \rangle$*, where:*

- S_o, A, R, γ *are as defined in an MDP;*
- $\tilde{\Theta}$ *is the set discrete of hidden states.* $\tilde{\Theta}$ *is generated by discretising the continuous parameter space* Θ *in the SAS Cont-MOMDP. This can be through random sampling or taking a uniform sample from the space of possible values for* θ*;*
- Λ *is the set of latent parameters governing the transition probabilities;*
- $T : S_o \times \tilde{\Theta} \times A \times S_o \times \tilde{\Theta} \to [0, 1]$ *is the transition function, where there is uncertainty over the transition probabilities;*
- Ω *is the set of observations;*
- $O : S_o \times \tilde{\Theta} \times A \times \Omega \to [0, 1]$ *is the observation function;*
- $b_0 : \tilde{\theta} \to [0, 1]$ *is the initial belief distribution over the hidden state space,* $\tilde{\Theta}$*;*
- $P_0(\lambda)$ *is the prior over the latent parameters;*

For a given SAS cont-MOMDP $\mathcal{M}_{C_{SA}}$, and we define the SAS BA-MOMDP as $\mathcal{M}_{B_{SA}}$. $\mathcal{M}_{C_{SA}}$ and $\mathcal{M}_{B_{SA}}$ have the same observable state space S_o, action space A, observations Ω, reward function R and decay rate γ. We sample the Θ space from $\mathcal{M}_{C_{SA}}$ to generate the discrete hidden states $\tilde{\Theta}$ in $\mathcal{M}_{B_{SA}}$. A comparison of the respective hidden state spaces

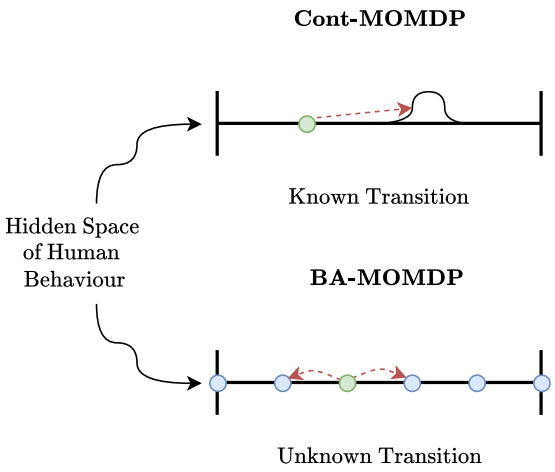


Figure 1: Comparison of the hidden space for $\mathcal{M}_{C_{SA}}$ and $\mathcal{M}_{B_{SA}}$

for $\mathcal{M}_{C_{SA}}$ and $\mathcal{M}_{B_{SA}}$ are shown in Figure 1. The transition probabilities for $\mathcal{M}_{B_h}$ can be expressed as:

$$T(s_o, \tilde{\theta}, a, s'_o, \tilde{\theta}') = T(s_o, a, s'_o|\tilde{\theta}) \cdot T(\tilde{\theta}, a, \tilde{\theta}') \tag{11}$$

$T(s_o, a, s'_o|\tilde{\theta})$ is the transition probability given the human had an internal state of $\tilde{\theta}$ in $\mathcal{M}_{C_{SA}}$. As $T(\tilde{\theta}, a, \tilde{\theta}')$ is unknown to the agent, the latent parameter $\lambda^a_{\tilde{\theta},\tilde{\theta}'}$ represents the transition probability. We use a Dirichlet distribution to describe the belief over the latent parameters $\lambda^a_{\tilde{\theta},\tilde{\theta}'} \forall \tilde{\theta}' \in \tilde{\Theta}$. The Dirichlet distribution is defined through a counter ϕ, which represents the number of times a transition has occurred (e.g. the transition $(\lambda^a_{\tilde{\theta},\tilde{\theta}'})$ has happened $\phi^a_{\tilde{\theta},\tilde{\theta}'}$ times). We maintain a joint belief over the hidden state space $\tilde{\Theta}$, and the latent parameter space Λ. The initial belief is $b_0(\tilde{\theta}, \vec{0}) = b_0(\tilde{\theta})$, as the initial count value for all $\phi^a_{\tilde{\theta},\tilde{\theta}'}$ is zero. As an action is taken, and a new observable state is reached, the belief over the hidden state space and the counter ϕ is updated.

Algorithm 1 Belief Update

Update Belief: $\{b(\tilde{\theta}, \phi), s_o, a, s'_o\}$
$b' \leftarrow \vec{0}$
for $(\tilde{\theta}, \phi, \tilde{\theta}') \in \tilde{\tilde{\theta}} \times \Phi \times \tilde{\Theta}$ **do**
$b'(\tilde{\theta}', \phi + \delta^a_{\tilde{\theta},\tilde{\theta}'}) += b(\tilde{\theta}, \phi) \cdot T_{\tilde{\theta}}(s_o, a, s'_o) \cdot \frac{\phi^a_{\tilde{\theta},\tilde{\theta}'}}{\sum_{\tilde{\theta}'' \in \Theta} \phi^a_{\tilde{\theta},\tilde{\theta}''}}$
end for
return Normalise b'

The belief space expands exponentially with the number of hidden states and the planning horizon. To mitigate this, we use approximate belief monitoring [Ross *et al.*, 2007] to maintain a belief distribution in polynomial time. At each time step, we use algorithm 6 to do an exact belief update, then we keep the K most probable states in b' and then re-normalise.

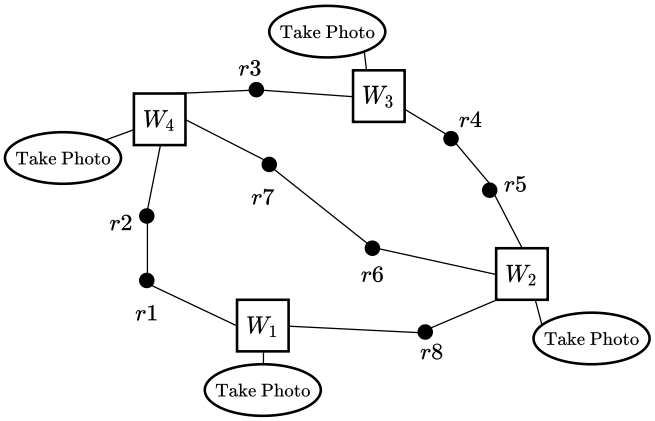


Figure 2: UAV Domain

6 Experiments

6.1 Domain

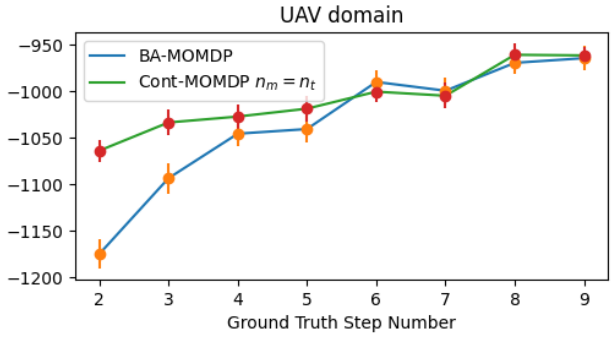
We evaluated the cont-MOMDP and the BA-MOMDP framework for shared autonomy systems in the domain presented by Feng *et al.*. The authors consider a domain where a unmanned aerial vehicle (UAV) is tasked with a monitoring mission, and the UAV must fly to multiple waypoints and capture a "good" photo at each waypoint. The UAV can be flown either by a human operator or an autonomous operator, and at any given waypoint, if a "good" photo is not taken, the UAV must repeat the photo-taking until a "good" photo is taken. The objective of the agent of this shared autonomy system is to determine the best operator for the next step, given that they aim to minimise the amount of petrol they use.

Feng *et al.* considers how the impact of fatigue affects human performance. They model two possible states for the human to be in, "normal" and "tired". They describe the behaviour of the human operator using MDPs M_{norm} and M_{tired}, where their main difference is the probabilities of taking a "good" photo. However, the authors only consider the dynamic characteristic of the human internal state, and assume full observability of the internal state. They model the change in internal state through a deterministic transition function, where the human will go from a "normal" to a "tired" state after the human has completed n tasks.

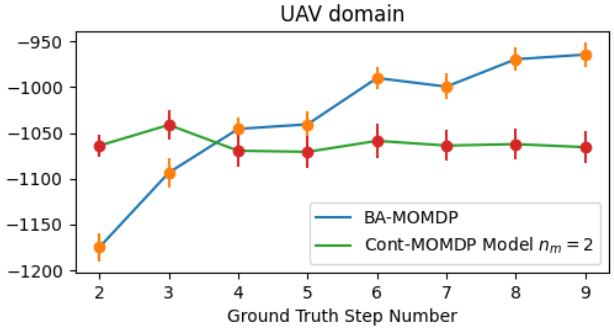
However, in reality, we cannot directly observe the human internal state directly, and it is more realistic to consider the human's tiredness level as a continuous spectrum. The human will travel along this spectrum as they accumulate the number of tasks they have done.

6.2 Human Model

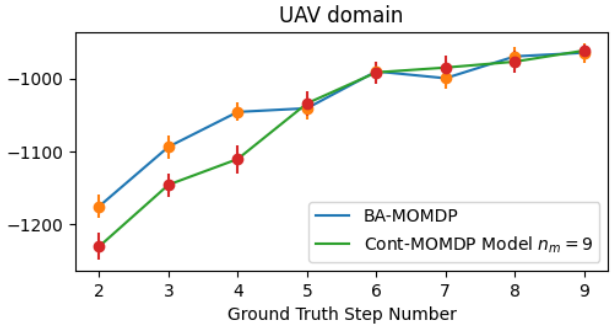
Therefore, we model our shared autonomy system as a cont-MOMDP, $\mathcal{CM}_{\mathcal{SA}} = \langle S_o, S_h, \overline{s}_o, b_0, A, \Omega, T, O, R, \gamma \rangle$, where the observable state space S_o describes the environment state the UAV is in (as shown in Figure 2), and Booleans describing whether a "good" photo has been taken at each waypoint. $S_h = [0, 1]$ is the continuous set of possible hidden states, describing the tiredness of the human. The initial observable state is $\overline{s}_o = W_1$, and the initial belief is $b_0(0) = 1$, since the agent knows that the agent is not tired at the start of a mission.



(a) Cont-MOMDP with $n_t = n_m$ and BAMDP model



(b) Cont-MOMDP with $n_t = 2$ and BAMDP model



(c) Cont-MOMDP with $n_t = 9$ and BAMDP model

Figure 3: Results for UAV domain

The set of actions is $A = A_h \cup A_{auto}$, the union of the set of human actions and the set of autonomous operator's actions. The transition functions for the hidden transition dynamics are as follows: If a human takes on average n tasks to becomes tired, we assume that for each action the human takes, the human's internal state evolves as

$$\Delta s_h \sim N(\frac{1}{n}, \sigma^2) \tag{12}$$

where the new state is $s'_h = min(max(s_h + \Delta s_h, 0), 1)$. The set of observations $\Omega = S_0$ is also the set of observable states, and the observation function O is as shown in Equation 9. The reward function R defines a negative reward of -60 for every state visited, and the mission terminates once a "good" photo has been taken at every waypoint. The decay rate $\gamma = 1$, as the mission length is finite.

However, the average number of steps taken before the

human reaches tired is individual to them and difficult to know ahead of the mission. We therefore use BA-MOMDP to learn the human internal transition dynamics during the mission. The BA-MOMDP is defined by the tuple $\mathcal{M}_{B_{SA}} = \langle S_o, \tilde{\Theta}, A, \Lambda, T, \Omega, O, b_0, P_0(\lambda), R, \gamma \rangle$, where $S_o, A, \Omega, b_0, R, \gamma$ are as presented in $\mathcal{M}_{C_{SA}}$. $\tilde{\Theta}$ is defined by 4 points taken from the continuous space of S_h, where $\tilde{\Theta} = [0, 0.\dot{3}, 0.\dot{6}, 1]$. The latent parameters $\lambda^a_{\tilde{\theta},\tilde{\theta}'} \in \Lambda$ define the unknown transition probabilities between the hidden states. We begin with no prior over the latent parameters, so $P_0(\lambda)$ is a uniform distribution.

6.3 Results

We solve the models of the shared autonomy system $\mathcal{M}_{C_{SA}}$ and $\mathcal{M}_{B_{SA}}$, to generate a history-dependent policy, and apply this to the ground-truth model, which is an instance of $\mathcal{M}_{C_{SA}}$. We vary the ground-truth model through n_t, the average number of steps taken by the human to become tired ranges from 2 to 9.

The model of the shared autonomy system, $\mathcal{M}_{C_{SA}}$, assumes that the average number of steps taken by the human is n_m. We consider three versions of $\mathcal{M}_{C_{SA}}$ to generate policies for the agent to follow: first, where $n_m = n_t$, so the agent knows the true value of n_t. The second version assumes $n_m = 2$, independent of the true value, n_t, and the third version assumes that $n_m = 9$. These models are chosen to demonstrate when the model makes incorrect assumptions about the hidden transition dynamics.

We only consider one model for $\mathcal{M}_{B_{SA}}$, as the model does not make any assumptions over the hidden transition dynamics, and learns n_t through observations instead.

We solved $\mathcal{M}_{C_{SA}}$ and $\mathcal{M}_{B_{SA}}$ with the POMCP algorithm and an altered BAMCP algorithm [Guez *et al.*, 2014], respectively. The BAMCP algorithm is modified to maintain belief over the hidden state and the hidden parameter, as shown in Algorithm 1. We ran $200,000$ trials per step in the mission for both models, and took the action with the highest expected cumulative reward. We ran 250 runs for each data point shown in the results.

First, we compared $\mathcal{M}_{C_{SA}}$ with $n_m = n_t$ and $\mathcal{M}_{B_{SA}}$, and the results are shown in Figure 3 (a). At lower values of n_t, $\mathcal{M}_{C_{SA}}$ outperforms $\mathcal{M}_{B_{SA}}$, since the change in the internal hidden state is rapid, so $\mathcal{M}_{B_{SA}}$ cannot adapt their policy sufficiently quickly. However, at higher values of n_t, as the rate of change for the internal state is slower, $\mathcal{M}_{B_{SA}}$ is able to learn the dynamics of the internal transition and perform similarly to $\mathcal{M}_{C_{SA}}$.

Secondly, we consider $\mathcal{M}_{C_{SA}}$ where $n_m = 2$. As shown in Figure 3 (b), when $n_t = 2, 3$, $\mathcal{M}_{C_{SA}}$ is able to outperform $\mathcal{BA}_{SA}$, as the rate of change is similar to or equal to the assumed value of n_m. However, if $n_t > 3$, $\mathcal{M}_{B_{SA}}$ outperforms $\mathcal{M}_{C_{SA}}$, since the Bayes-adaptive model learns the hidden transition probabilities, while $\mathcal{M}_{C_{SA}}$ will prematurely retire the human operator from the shared autonomy system, as they assume the human has become tired.

Finally, we consider $\mathcal{M}_{C_{SA}}$ where $n_m = 9$. If $n_t < 5$, $\mathcal{M}_{C_{SA}}$ assumes that the human will not get tired and will continue to ask the human to complete tasks even when the human will perform worse than the autonomous controller. Therefore, $\mathcal{M}_{B_{SA}}$ outperforms $\mathcal{M}_{C_{SA}}$ when $n_t < 5$, as $\mathcal{M}_{B_{SA}}$ is able to learn that the human is likely to be tired and stop asking them to complete tasks. When $n_t > 5$, $\mathcal{M}_{B_{SA}}$ and $\mathcal{M}_{C_{SA}}$ perform similarly, since the true rate of change of the internal state is similar to that assumed by $\mathcal{M}_{C_{SA}}$, so the ground-truth model and the assumed model are close enough.

7 Conclusion

We have presented two frameworks for shared autonomy systems, the cont-MOMDP and the BA-MOMDP. In the cont-MOMDP, we can express the human operator's behaviour with a hidden continuous space, and we assume that we know the transition dynamics from one continuous state to another. This model allows us to effectively track the human's evolving behaviour, and adapt our policy accordingly. The BA-MOMDP has a discrete hidden state space, but the framework allows the agent to learn the hidden transition dynamics, and exploit the information learned to maximise the total reward collected over a mission. This framework removes the need for the agent to make assumptions on the behaviour of the human, and learns the huamn's unique internal transition dynamics through observations.

We tested our two frameworks out on a UAV domain, and considered the case where the cont-MOMDP knew the true hidden dynamics, and when the cont-MOMDP assumed the wrong hidden dynamics. We found that the cont-MOMDP model performed better than the BA-MOMDP when the model is fully known. However, when the model is not known, BA-MOMDP outperforms cont-MOMDP, as it requires less knowledge, and the policy adapts as the agent learns the transition dynamics. In future works, we will extend this to consider domains with more complex hidden state spaces.

Acknowledgements

This work was supported by the Defence Science and Technology Laboratory, the EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. This document is an overview of UK MOD's Defence Science and Technology Laboratory (DSTL) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy.

References

[Araya-López *et al.*, 2010] M. Araya-López, V. Thomas, O. Buffet, and F. Charpillet. A Closer Look at MOMDPs. In *2010 22nd IEEE International Conference on Tools with Artificial Intelligence*, volume 2, pages 197–204, October 2010. ISSN: 2375-0197.

[Basich *et al.*, 2020] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. Learning to Optimize Autonomy in Competence-Aware Systems. In *Proceedings of the 19th International Conference on Autonomous Agents*

and MultiAgent Systems, AAMAS '20, pages 123–131, Richland, SC, May 2020. International Foundation for Autonomous Agents and Multiagent Systems.

[Basich *et al.*, 2021] Connor Basich, Justin Svegliato, Allyson Beach, Kyle H. Wray, Stefan Witwicki, and Shlomo Zilberstein. Improving Competence via Iterative State Space Refinement. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1865–1871, Prague, Czech Republic, September 2021. IEEE.

[Charles *et al.*, 2018] Jack-Antoine Charles, Caroline P. C. Chanel, Corentin Chauffaut, Pascal Chauvin, and Nicolas Drougard. Human-Agent Interaction Model Learning based on Crowdsourcing. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, HAI '18, pages 20–28, New York, NY, USA, December 2018. Association for Computing Machinery.

[Costen *et al.*, 2022] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Shared Autonomy Systems with Stochastic Operator Models. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*, pages 4614–4620, Vienna, Austria, July 2022. International Joint Conferences on Artificial Intelligence Organization.

[Costen *et al.*, 2023] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. Planning with Hidden Parameter Polynomial MDPs. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(10):11963–11971, June 2023. Number: 10.

[Costen *et al.*, 2024] Clarissa Costen, Anna Gautier, Nick Hawes, and Bruno Lacerda. Multi-Robot Allocation of Assistance from a Shared Uncertain Operator. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '24, pages 400–408, Richland, SC, May 2024. International Foundation for Autonomous Agents and Multiagent Systems.

[Cubuktepe *et al.*, 2021] Murat Cubuktepe, Nils Jansen, Mohammed Alshiekh, and Ufuk Topcu. Synthesis of Provably Correct Autonomy Protocols for Shared Control. *IEEE Transactions on Automatic Control*, 66(7):3251–3258, July 2021. Conference Name: IEEE Transactions on Automatic Control.

[Dahiya *et al.*, 2022] Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan, and Stephen L. Smith. Scalable Operator Allocation for Multi-Robot Assistance: A Restless Bandit Approach. *IEEE Transactions on Control of Network Systems*, 9:1397–1408, September 2022. arXiv:2111.06437 [cs, eess].

[Duchetto *et al.*, 2018] Francesco Duchetto, Ayse Kucukyilmaz, Luca Iocchi, and Marc Hanheide. Do Not Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation From Human Demonstrations. *IEEE Robotics and Automation Letters*, PP:1–1, July 2018.

[Duff, 2002] Michael O'Gordon Duff. *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Process*. PhD thesis, University of Massachusetts, February 2002.

[Feng *et al.*, 2016] Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu. Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems. *IEEE Transactions on Automation Science and Engineering*, 13(2):450–462, April 2016. Conference Name: IEEE Transactions on Automation Science and Engineering.

[Fern *et al.*, 2014] A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50:71–104, May 2014.

[Guez *et al.*, 2014] Arthur Guez, Nicolas Heess, David Silver, and Peter Dayan. Bayes-Adaptive Simulation-based Search with Value Function Approximation. In *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.

[Javdani *et al.*, 2015] Shervin Javdani, Siddhartha S. Srinivasa, and J. Andrew Bagnell. Shared Autonomy via Hindsight Optimization. *arXiv:1503.07619 [cs]*, April 2015. arXiv: 1503.07619.

[Mahmud *et al.*, 2023] Saaduddin Mahmud, Connor Basich, and Shlomo Zilberstein. Semi-Autonomous Systems with Contextual Competence Awareness. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 689–697, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems.

[Nanavati *et al.*, 2021] Amal Nanavati, Christoforos Mavrogiannis, Kevin Weatherwax, Leila Takayama, Maya Cakmak, and Siddhartha Srinivasa. Modeling Human Helpfulness with Individual and Contextual Factors for Robot Planning. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021.

[Petric and Kovacic, 2020] Frano Petric and Zdenko Kovacic. Design and Validation of MOMDP Models for Child–Robot Interaction Within Tasks of Robot-Assisted ASD Diagnostic Protocol. *International Journal of Social Robotics*, 12(2):371–388, May 2020.

[Puterman, 1994] Martin Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Ltd, 1994.

[Rigter *et al.*, 2020] Marc Rigter, Bruno Lacerda, and Nick Hawes. A Framework for Learning From Demonstration With Minimal Human Effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, April 2020.

[Ross *et al.*, 2007] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayes-Adaptive POMDPs. In *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2007.

[Ross *et al.*, 2008] Stephane Ross, Brahim Chaib-draa, and Joelle Pineau. Bayesian reinforcement learning in continuous POMDPs with application to robot navigation. In *2008 IEEE International Conference on Robotics and Automation*, pages 2845–2851, Pasadena, CA, USA, May 2008. IEEE.

[Spaan and Vlassis, 2004] M.T.J. Spaan and N. Vlassis. A point-based POMDP algorithm for robot planning. In *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA '04. 2004*, volume 3, pages 2399–2404 Vol.3, April 2004. ISSN: 1050-4729.

[Sunberg and Kochenderfer, 2018] Zachary Sunberg and Mykel Kochenderfer. Online Algorithms for POMDPs with Continuous State, Action, and Observation Spaces. *Proceedings of the International Conference on Automated Planning and Scheduling*, 28:259–263, June 2018.

[Wray *et al.*, 2016] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 517–523, New York, New York, USA, July 2016. AAAI Press.

[Wu and Durfee, 2010] Jianhui Wu and Edmund H. Durfee. Resource-Driven Mission-Phasing Techniques for Constrained Agents in Stochastic Environments. *Journal of Artificial Intelligence Research*, 38:415–473, July 2010. arXiv:1401.3845 [cs].

[Wu *et al.*, 2017] Bo Wu, Bin Hu, and Hai Lin. Toward efficient manufacturing systems: A trust based human robot collaboration. In *2017 American Control Conference (ACC)*, pages 1536–1541, May 2017. ISSN: 2378-5861.

6.1 Limitations

The cont-MOMDP framework presented in this paper is constrained by its ability to handle only low-dimensional hidden state spaces. This constraint is a result of the particle filter used to maintain the belief over the continuous hidden states, which suffers from the curse of dimensionality. While the framework proposed in Chapter 5 allowed for a closed-form belief representation, we were unable to replicate this for a dynamic continuous hidden state space. Any framework that is capable of maintaining a closed-form belief distribution over such a state space would need to impose restrictions on the evolution of the hidden state space (e.g. the state evolves following an exponential decay distribution), which is unrealistic for many scenarios.

Another significant limitation of the cont-MOMDP framework is the assumption that the human behaviour can be represented as linear combinations of K MDPs. While this approach allows us to represent a human that is not fully in any of the K MDPs, the approach presumes that a linear combination of K MDPs provides an accurate approximation of the human's behaviour. This may not hold true in practice. For example, a human who is partially tired may exhibit behaviours that corresponds to a fully tired state in certain scenarios while behaving normally in others. A richer model of the human, where we consider varying rates of internal state evolution, would allow for a more nuanced representation.

The BA-MOMDP framework was introduced to address the difficulty of knowing the transition dynamics in a hidden state space prior to execution. However, this framework's main limitation is the discretisation of the hidden state space, which was necessary to enable the agent to learn the hidden transition dynamics. The coarseness of the the discretisation (defined by the number of discrete states), must be tuned by the user prior to execution. Finer discretisation can better capture a broader range of human behaviours, but comes with increased computational complexity to maintain a belief over

a larger hidden state space.

Chapter 7

Multi-Robot Allocation of Assistance from a Shared Uncertain Operator

As the capabilities of autonomous operators increases, the amount of intervention required by the human operator decreases. To better utilise the human operator, we propose a multi-robot shared autonomy system that allocates the human assistance to the robot that requires it the most. Like the problem setting shown in Chapter 4, we consider the human operator to have multiple ways of acting. However, in this chapter, we consider the human operator's possible modes to be continuous, rather than discrete. We implement this by using our HPP-MDP (Chapter 5) to model the human operator's performance as a continuous distribution over the latent parameters.

The main difficulty of multi-robot shared autonomy system is the allocation of the human operator's assistance. Existing literature on multi-robot systems with constrained resources have focused on de-centralised methods, where each robot considers only their own tasks and rewards. This is because centralised approaches are not scalable, as the number of robot increases, the state and action spaces expand exponentially. However, in our setting with a uncertain human operator, the robots learn about the human operator's performance through observations, and this information alters the optimal actions for

the robots. A traditional de-centralised approach will not allow the robots share this information, and the robots will not be motivated to perform exploratory actions that may help other robots learn about the human's performance.

Therefore, we propose a de-centralised approach for the multi-robot shared autonomy system, where we explicitly reward robots for taking actions that reduce the variance of the belief over the human's performance. We are able to do this by using the HPP-MDP to maintain a closed-form posterior distribution over the human operator's performance. The closed-form posterior distribution allows us to efficiently calculate the variance of the belief, and we incorporate this into the BAMCP algorithm. This allows us to augment the reward function such that reducing the variance is explicitly rewarded, thus maintaining a exploration/exploitation when selecting the robot to receive human help.

Table 7.1: Summary of the papers and their features

Chapters	4	5	6	7
SA	Yes	No	Yes	Yes
SA style	N operators controls one robot	N/A	One human and one autonomous agent controls a robot	one human and K robots, where the human can take control of any robot
Dynamic	Yes	No	Yes	No
Continuous	No	Yes	Yes	Yes
Closed-form	Yes	Yes	Yes	Yes

Multi-Robot Allocation of Assistance from a Shared Uncertain Operator

Clarissa Costen
University of Oxford
Oxford, United Kingdom
clarissa@robots.ac.uk

Anna Gautier
KTH Royal Institute of Technology
Stockholm, Sweden
annagau@kth.se

Nick Hawes
University of Oxford
Oxford, United Kingdom
nickh@robots.ox.ac.uk

Bruno Lacerda
University of Oxford
Oxford, United Kingdom
bruno@robots.ox.ac.uk

ABSTRACT

Shared autonomy systems allow robots to either operate autonomously or request assistance from a human operator. In such settings, the human operator may exhibit sub-optimal behaviours, influenced by latent variables such as attention level or task proficiency. In this paper, we consider shared autonomy systems composed of multiple robots and one human. In this setting, we aim to synthesise a controller that selects, at each decision step, the actions to be taken by each robot and which (if any) robot the human operator should assist. To efficiently allocate the human operator to a robot at any given time, we propose a controller that reasons about the uncertainty over the latent variables impacting the human operator's performance. To ensure scalability, we use an online bidding system, where each robot plans while considering its belief over the human's performance, and bids according to the direct benefit of human assistance and how much information will be gained by the system about the human. We experiment on two domains, where we outperform approaches for allocation of human assistance that do not consider the human's latent variables, and show that the performance of the overall system increases when robots consider the information gained by requesting human assistance when bidding.

KEYWORDS

Multi-agent planning, Planning under Uncertainty, Planning with abstraction and generalization

ACM Reference Format:
Clarissa Costen, Anna Gautier, Nick Hawes, and Bruno Lacerda. 2024. Multi-Robot Allocation of Assistance from a Shared Uncertain Operator. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

1 INTRODUCTION

As autonomous robots become increasingly reliable and integrated into our lives [18, 21], they occasionally enter situations where they cannot complete their task without human intervention. Shared autonomy systems address this challenge by allowing the control of a robot to be shared between a human and an autonomous agent. As automation improves, the level of human supervision required for each robot decreases. In this context, we examine scenarios where a single human operator oversees multiple robots. Allocating human assistance optimally to these robots becomes a complex issue, particularly when multiple robots require assistance simultaneously. Moreover, the human operator may possess varying skills, making them more proficient at assisting with some tasks compared to others, but these skill differences may be unknown to the robots. In this paper, we introduce an efficient approach to allocate human assistance within multi-robot systems characterized by uncertainty surrounding the human operator's performance.

Shared autonomy systems reduce the workload on the human operator [2] by deciding when a human or autonomous controller should operate the robot. The system's controller determines the optimal operator (human or autonomy) by reasoning over its internal model of both the human and the autonomous robot. Typically, these internal models use Markov decision processes (MDPs) [11] to account for the stochastic nature of the operators. However, human behaviour is inherently diverse, making it challenging to predict human performance accurately before execution. To address this variability, Costen et al. [9], Nanavati et al. [22], Wray et al. [30] have utilized partially observable MDPs (POMDPs) and Bayes-adaptive MDPs (BAMDPs) to model the human operator, incorporating probability distributions to represent the human's true location or internal state. The papers that consider uncertainty over the human focus on systems involving one robot and one human. We extend this to systems with multiple robots and one human.

We define our system as a multi-robot shared autonomy system, where a human operator supervises multiple robots. The multi-robot shared autonomy system can be framed as a scenario where robots must share a constrained resource: the human operator. Multi-robot systems with constrained resources typically use decentralised approaches, as solving the joint model can become computationally infeasible. These approaches often assume fixed resource availability with known impacts on the robots [23], such

as battery charge, and typically provide offline solutions where the resources are pre-allocated to robots before deployment [32]. However, when there is uncertainty over the outcome of consuming a resource, decentralised approaches fail to provide policies that change as other robots observe the effect of a resource. Thus, decentralised approaches are not suited to our scenario, where the robots must adapt their decision-making as other robots learn about the human operator's abilities through observations.

We propose a bidding system to distribute uncertain human assistance efficiently, avoiding the need to solve a joint model. The human is described using a hidden parameter polynomial MDP (HPP-MDP), a class of uncertain MDP where the transition function is parameterised by a set of latent parameters. At every decision step, each robot uses BAMCP [17], an online tree-search algorithm, to plan for a combined human-robot HPP-MDP that models a single-robot shared autonomy system. This assumes that the robot can always access human assistance. If the optimal action requires human operation, then the robot submits a bid to the centralised controller. This bid is based on the difference in value between the human taking control and the robot executing autonomously. The centralised controller then awards human help to the highest bidder, with the other robots executing their optimal autonomous action. Then, the centralised belief distribution over the human operator's latent parameters is updated by considering the outcome of the human's action. This is then repeated until all robots have completed their tasks.

Planning with single-robot models introduces challenges: robots cannot predict if the information they receive is useful to others, and other robots may make observations that alter the posterior distribution over the latent parameters, hindering long-term planning. This is because planning over single-robot shared autonomy model does not capture the long-term effects of the other robots in the true, joint shared-autonomy model. These issues lead to undervaluing explorative actions, which leads to sub-optimal human help allocation. To address this, we add an exploration bonus which considers the variance in the posterior distribution over the latent parameters. The HPP-MDP formulation of the shared autonomy system allows us to keep a closed-form representation of the posterior distribution over the human's performance, reducing the computational complexity of calculating the variance.

Our main contributions are the modelling of a multi-robot shared autonomy system with uncertain human performance as an uncertain MDP; the bidding system to allocate human help; and the explicit reward function motivating the reduction in the variance over the belief in human performance. To the best of our knowledge, this is the first approach proposed to solve multi-robot shared autonomy systems where there is uncertainty in the human operator's performance. We compare our approach to 1) solving the joint model using the BAMCP algorithm [17], and 2) a decentralized approach proposed by Dahiya et al. [12], which considers a multi-robot system where the human operators have known success rates. We empirically show that our approach outperforms the joint model under time constraints and the method proposed in Dahiya et al. [12] in two domains.

2 RELATED WORK

2.1 Shared Autonomy Systems

Shared autonomy systems are useful in settings where the workload on the human can be reduced by letting autonomy take control in low-risk tasks [27]. Common assumptions in many of these systems is that the human operator can operate the system at all times and they can act perfectly [3]. Some systems use the human operator as an example to learn from [14, 25], with the long-term objective of reducing the reliance on human operators over time.

However, human operators' capabilities are not always guaranteed and can be influenced by a variety of factors, such as fatigue, distraction, and ability [31]. The variance in the human operator's performance can be explicitly modelled using Markov models [8, 15, 20]. This allows the autonomous agent to consider the stochasticity in the human operator's behaviour when planning, such as the probability of the human failing a task. However, the factors that affect the human's performance are not always observable to the agent. Jean-Baptiste et al. [19] and [9] use POMDPs to model the uncertainty over the human operator's behaviour. The factors affecting the human operator (such as skill level or fatigue) are modelled as latent internal states, and the agent must maintain and reason over a belief distribution over the latent states. By solving models with partial observability fully, it is possible find the policy for the agent where exploration (learning about the human operator) and exploitation (using the information about the human to gain rewards) are optimally balanced. These papers only consider systems where there is one agent and one human operator. In contrast, we consider the setting where there are multiple agents and one human operator.

2.2 Multi-Agent Shared Autonomy Systems

As the amount of human intervention required per robot decreases, we can move to a system where a single human operator supervises multiple robots. This can be useful in search-and-rescue missions, as well as multi-robot systems in warehouses [26]. In these systems, using a single joint model to describe the multi-agent system can be intractable, as the state and action space grows exponentially with the number of agents. To solve the joint model, Rosenfeld et al. [26] restricts the horizon of the problem to only 1 or 2 decision steps. This short horizon allows the problem to be solved fully for the current and next action, but is unable to consider the effects of those actions into the future.

Swamy et al. [28] highlights the difficulty of the human appropriately choosing the best robot to help in scenarios with large numbers of robots. They focus on the human choosing a robot to help, while we consider the robots determining how useful human help would be. They suggest a method where the system learns the human's preferences for helping robots when there is a small number of robots. This learnt preference is generalised to fit problems with a larger number of robots, thus able to suggest the most appropriate robot for the human to help. However, this method assumes that the human always chooses the optimal robot to help initially, and does not consider the possibility that the human may choose the wrong robot.

Other approaches for allocating human assistance in multi-agent shared autonomy systems impose rules onto the system to reduce

the complexity of the problem. Cai et al. [7] assumes that the time taken by the human and the robot for every task is exactly known and that the human is always faster or equal to the robot. These assumptions reduce the problem to a deterministic scheduling problem.

Similar to our work, Dahiya et al. [12] considers a multi-agent system where the human operator can take over at an agent's request. They consider the possibility that the human operator may fail a task, and use an MDP to model the dynamics of the human and the autonomous agents. The agent with the highest benefit from human intervention is given human help, and this benefit is defined as the difference in the relative expected cost of the agent doing the task autonomously versus the human helping. These benefits are calculated offline before the system is deployed, and they assume that the probability of the human failing a task is known. Therefore, their system is unable to adapt if the human operator does not perform tasks as expected. We compare our approach to theirs in Section 6.

2.3 Multi-agent constrained resource problems

Systems with multiple agents have been modelled using multi-agent MDPs (MMDPs) [5], which have joint state and action spaces. The state space of an MMDP grows exponentially with the number of agents, making it computationally intractable to solve fully. Thus MMDPs are generally solved using decentralised methods when the transition probabilities for each agent are independent [4]. Our multi-agent shared autonomy system can be framed as a multi-agent system where the agents must share a constrained resource, the human operator with independent transition probabilities. In MMDPs with global resource constraints, each agent has its own set of transition and reward dynamics, but their actions are coupled by a global resource constraint [32], such as advertising space [6].

Both [23, 32] solve constrained resource multi-agent systems using a mixed integer linear program (MILP) to pre-allocate the resource offline to each agent at every decision step. While MILP-based solutions can guarantee that hard constraints are satisfied, they are not scalable to large problems as their complexity grows exponentially with the horizon. If the resource has a soft constraint where some constraint violations are permitted, we can use column generation algorithms for offline policy synthesis that guarantees the resource constraint to be satisfied in expectation [29, 33]. Column generation algorithms are more scalable than MILP, as they parallelise the computation by finding policies for each agent instead of solving the joint problem. Alternative approaches for soft constraints consider the risk of resource violations in the form of a chance-constraint [13] and a conditional value-at-risk constraint [16].

In contrast to [29], where the agents have partial observability of their state and a deterministic shared resource, we consider the case where the agents have full observability of their state, but they must reason over the uncertainty over the effect of the shared resource on the transition probabilities. This leads to a problem where the observation made by one agent may affect the optimal policy for another agent, because the other agents' observation may resolve some of the uncertainty in the transition function. Therefore, we cannot use decentralised methods such as column generation. In this paper, we propose an approach to address this problem of a constrained resource with an uncertain effect on the agent in a multi-agent system without solving the joint model.

3 PRELIMINARIES

MDPs have been widely used in planning problems to describe the stochastic behaviour of agents. MDPs allow us to reason over aleatoric uncertainty, such as the randomness in the success of the autonomous robot completing a task.

Definition 3.1 (MDP). A Markov Decision Process (MDP) is defined by the tuple $M = \langle S, \bar{s}, A, T, R, \gamma \rangle$, where:

- S is the set of states and $\bar{s}$ is the initial state;
- A is the set of actions;
- $T(s, a, s') = P(s' \mid s, a)$ is the probabilistic transition function;
- $R : S \times A \to \mathbb{R}$ is the reward function;
- γ is the discount factor,

where the goal of the agent is to maximise the cumulative discounted reward.

When a human is operating the robot, the probability of successful task completion is dependent on unobservable factors such as their skill level or fatigue. Uncertain MDPs can be used to describe worlds where there is uncertainty over the exact transition probabilities. An **uncertain MDP** is defined by the tuple $\mathcal{M} = \langle S, \bar{s}, A, \{T_\theta\}_{\theta \in \Theta}, R, \gamma \rangle$, where the transition dynamics $\{T_\theta\}_{\theta \in \Theta}$ are governed by a set of global latent parameters, θ, such that $T_\theta(s, a, s') = P(s' \mid s, a; \theta)$. The latent parameter $\theta \in \Theta$ is an $N-$dimensional vector, i.e. $\Theta = \mathbb{R}^N$. The agent has a prior distribution, $P_0(\theta)$ over the parameter space Θ. As the agent observes the outcome of actions, the posterior distribution over the parameters is updated using Bayes' rule, $P_t(\theta) \propto P_0(\theta) \cdot P(\theta \mid h_t)$, where $h_t = s_0 a_0 s_1 \cdots s_t$ is the history of the agent's actions and observations.

A challenge in using uncertain MDPs is how to update the posterior distribution over the latent parameters. In our setting, we are interested in capturing the change in variance of the posterior distribution as the outcome of the human operator's action is observed. To reduce the computational complexity of this, we use a hidden parameter polynomial MDP (HPP-MDP) [10], a model that uses polynomials to represent the uncertain transitions, to model the shared autonomy system. The HPP-MDP allows us to maintain a closed-form representation of the posterior distribution over the latent parameters, and thus allows us to efficiently calculate the variance of the posterior distribution.

Definition 3.2 (HPP-MDP). An HPP-MDP is defined by the tuple $\mathcal{M} = \langle S, \bar{s}, A, \Theta, P_0(\theta), T_\Theta, R, \gamma \rangle$, where:

- S and $\bar{s} \in S$, A, R and γ are as in the MDP definition;
- $\Theta = [0, 1]^N$ is the parameter space of a set of N latent parameters. We denote elements of Θ as $\theta = (\theta_1, \cdots, \theta_N)$;
- $P_0(\theta)$ is the prior distribution over θ, where $P_0(\theta) \in Pol(\theta)$ i.e. it is a polynomial function of θ;
- $T_\Theta : S \times A \times S \to Pol(\theta)$ is the polynomial set of possible transition functions.

To be well formed, the transition functions must be valid for all $\theta \in \Theta$, such that

$$\sum_{s' \in S} T_\theta(s, a, s') = 1, \forall s \in S, a \in A, \theta \in \Theta. \quad (1)$$

The optimal policy maps the history of the trajectory the action with the highest Q-value, which is defined as:

$$Q^*(s, h_t, a) = R(s, a) + \max_{a \in A} \int_{\theta \in \Theta} \sum_{s' \in S} T_\theta(s, a, s') P_t(\theta) \gamma V^*(s', h_t a s') d\theta \quad (2)$$

where $V^*(s, h_t) = \max_{a \in A} Q^*(s, h_t, a)$ is the optimal value function.

Uncertain MDPs can be solved using a Monte-Carlo search tree based algorithm, such as the BAMCP algorithm [17]. BAMCP is a Monte-Carlo tree search algorithm which builds a history-dependent tree by sampling the prior distribution $P_0(\theta)$ to generate a sample MDP. The sample MDP used to simulate a run, and the trajectory taken is used to update the history-dependent nodes. BAMCP estimates the Q-value of each node, and the estimates converge towards the optimal Q-value as the number of samples increases. The optimal policy will give the optimal trade-off between exploratory actions to reduce the uncertainty over the human operator's performance, and exploitative actions that minimise the expected cost of the system.

4 SHARED AUTONOMOUS SYSTEMS WITH UNCERTAIN HUMAN OPERATORS

We first consider a shared autonomy system with one human operator and one robot, and then extend this to consider multiple robots and one human operator. In our shared autonomy system, the controller determines when and which robot the human operator should asisst, and determines the best action the robots and human should take.

4.1 Autonomous Robot Model

Autonomous robot $i \in [K] = \{1, \dots K\}$ has an environment described by the set of states S_i, and an initial state of $\bar{s}_i$. The behaviour of autonomous robot i is known, so we can describe their behaviour with an MDP, $M_i = \langle S_i, \bar{s}_i, A_i, T_i, R_i, \gamma \rangle$. A_i is the set of possible actions in the autonomous robot's domain, such as cardinal directions. $R_i(s, a)$ defines the reward associated with autonomous robot i attempting action a in state s.

4.2 Single-Robot Shared Autonomy Systems

In a shared autonomy system with a single robot and a human operator, the controller of the system must determine whether the human operator or the autonomy should perform the next action, while reasoning over the uncertainty of the human operator's performance. The autonomous robot i is described by $M_i = \langle S_i, \bar{s}_i, A_i, T_i, R_i, \gamma \rangle$.

When the robot is operated by a human, we describe their performance with an HPP-MDP, $\mathcal{M}_{i,h} = \langle S_i, \bar{s}_i, A_{i,h}, P_0(\theta), T_{i,\Theta}, R_{i,h}, \gamma \rangle$. $A_{i,h}$ is the set of actions the human can take, and $P_0(\theta)$ is the controller's prior distribution over the human operator's latent parameters. The transition probability function $T_{i,\theta}(s, a, s')$ is a polynomial function of $\theta \in \Theta$, where Θ describes the space of all possible human performance. $R_{i,h}(s, a)$ is the reward function for the controller when a human attempts an action.

The aim of the controller of the system is to determine whether the human or the autonomy should operate the robot and the action they should take to maximise the expected cumulative reward of the system. As the controller does not know the true value of θ, the controller must balance learning about the human operator's performance by requesting the human to attempt actions that will inform the posterior distribution $P_t(\theta)$, and using the current posterior distribution to choose actions that will maximise the expected reward of the next action. We use a HPP-MDP to model the shared autonomy system.

Definition 4.1 (Single-Robot Shared Autonomy System). The single-robot shared autonomy system for robot i is described by the tuple $\mathcal{M}^i_{SA} = \langle S_i, \bar{s}_i, A^i_{SA}, P_0(\theta), T^i_{SA}, R^i_{SA}, \gamma \rangle$, where:

- The set of actions A_{SA} is the union of the set of actions by the robot and the human operator, $A^i_{SA} = A_i \cup A_{i,h}$;
- The transitions T^i_{SA} are defined by the following:

$$T^i_{SA}(s, a, s') = \begin{cases} T_{i,\theta}(s, a, s') & \text{if } a \in A_{i,h} \\ T_i(s, a, s') & \text{if } a \in A_i \end{cases} \quad (3)$$

  where the transition probabilities for the actions taken by the robot is fixed, while the transition probabilities for the actions taken by the human operator are governed by the history of the trajectory.
- The reward function R^i_{SA} is the union of the reward functions for the autonomous robot and the human operator, such that

$$R^i_{SA}(s, a) = \begin{cases} R_{i,h}(s, a) & \text{if } a \in A_{i,h} \\ R_i(s, a) & \text{if } a \in A_i. \end{cases} \quad (4)$$

4.3 Multi-Robot Shared Autonomy Systems

We consider the problem where there are K robots and one human operator, where there is uncertainty over how the human operator may perform. The controller must determine when to deploy human assistance and which robot to deploy the human to. Each robot has a separate set of tasks, so the set of robots operated by autonomy can be described as $\langle M_1, M_2, \dots, M_K \rangle$. Similarly, each robot can be operated by the human operator, so the set of robots operated by the human operator can be described as $\langle \mathcal{M}_{1,h}, \mathcal{M}_{2,h}, \dots, \mathcal{M}_{K,h} \rangle$. We model K robot shared autonomy system with a HPP-MDP.

Definition 4.2 (Multi-Robot Shared Autonomy System). A shared autonomy system with K robots is described by the tuple $\boldsymbol{\mathcal{M}}_{SA} = \langle \boldsymbol{S}_{SA}, \bar{\boldsymbol{s}}_{SA} \boldsymbol{A}_{SA}, P_0(\theta), T_{SA}, R_{SA}, \gamma \rangle$, where:

- $\boldsymbol{S}_{SA} = \bigtimes_{i=1}^{K} S_i$ is the state space;
- $\bar{\boldsymbol{s}}_{SA} = (\bar{s}_1, \bar{s}_2 \dots \bar{s}_K)$ is the initial state;
- The set of actions $\mathbf{A}_{SA}$ is defined as the subset of $(A_{1,h} \cup A_1) \times \cdots \times (A_{K,h} \cup A_K)$ for which there is at most one element corresponding to a human actions. More precisely, $(a_1, \cdots, a_K) \in \mathbf{A}_{SA}$ if and only if there is at most one j for which $a_j \in A_{i,h}$ and for all other $k \neq j$, $a_k \in A_k$. Thus, there is at most one action done by the human operator in the joint action.

- For $\boldsymbol{a} = (a_1, \cdots a_K) \in \mathbf{A}_{SA}$, define $\boldsymbol{a}|_h$ as the index of the human operator action in $\boldsymbol{a}$, or $\perp$ if there is no human operator action in $\boldsymbol{a}$. The joint transition function T_{SA} is defined as:

$$T_{SA}(\boldsymbol{s}, \boldsymbol{a}, \boldsymbol{s'}) = \begin{cases} \prod_{i \in [K]} T_i(s_i, a_i, s_i') & \text{if } \boldsymbol{a}|_h = \perp \\ T_{j,\theta}(s_j, a_j, s_j') \prod_{i \in [K]^{-j}} T_i(s_i, a_i, s_i') & \text{if } \boldsymbol{a}|_h = j \end{cases} \tag{5}$$

  where $[K]^{-j} = \{1, \ldots, j-1, j+1, \ldots, K\}$.
- The joint reward function $R_{SA}(\boldsymbol{s}, \boldsymbol{a})$ is the sum of the reward functions for each robot, $R_{SA}(\boldsymbol{s}, \boldsymbol{a}) = \sum_{i=1}^{K} R_i(s_i, a_i)$.

Solving the $\boldsymbol{\mathcal{M}}_{SA}$ will give the optimal policy that balances the robots' need for assistance and the need to learn about the human operator's performance.

5 BIDDING SYSTEM FOR SHARED UNCERTAIN OPERATOR

The shared autonomy system model with multiple robots, $\boldsymbol{\mathcal{M}}_{SA}$ has a state space $\mathbf{S}_{SA}$ that expands exponentially with the number of robots, K. Therefore, trying to solve this model directly is intractable. We instead propose a bidding system where each robot bids for the human operator's assistance. The bid describes how much the robot will benefit from the human operator helping them, given the robot's current state. We define this as the difference in the expected reward when the human operates the robot, and when the autonomy operates the robot. The bid for the ith robot in local state s with a history h_t is:

$$\text{Bid}_i(s) = \max_{a_h \in A_{i,h}} Q_{\mathcal{M}_{i,h}}(s, h_t, a_h) - \max_{a_i \in A_i} Q_{\mathcal{M}_{i,h}}(s, h_t, a_i), \tag{6}$$

where $Q_{\mathcal{M}_{i,h}}(s, h, a)$ is the Q-value defined in Equation 2 for the HPP-MDP $\mathcal{M}_{i,h}$. We will first outline the bidding system, then how the bids are generated.

5.1 Bidding System

In our bidding system, there is a centralised controller, and K robots, where for robot i, M_i and $\mathcal{M}_{i,h}$ respectively models the robot being operated by auto and the human. The robots maintain a shared prior distribution $P_0(\theta)$ over the human operator's latent parameter θ. Each robot generates a bid for human help, and the controller instructs the human to assist the robot with the highest positive bid. The human will act according to $\arg\max_{a_h \in A_{i,h}} Q(s, h_t, a_h)$, and the other robots will act according to $\arg\max_{a_i \in A_i} Q(s, h_t, a_i)$. The robot receiving human help will observe the outcome of the human's action, and updates the posterior distribution $P_t(\theta)$ with the information gained during the interaction. This posterior distribution is shared with the other robots. This is because the system is cooperative, and the robots choose to share the observations of the human operator with the other robots to prevent asymmetry in the information available to each robots. Such asymmetry in information may result in sub-optimal bids from robots that do not have all the information available to them. This process is repeated until all robots have finished their tasks. A diagram of the bidding process is shown in Figure 1.

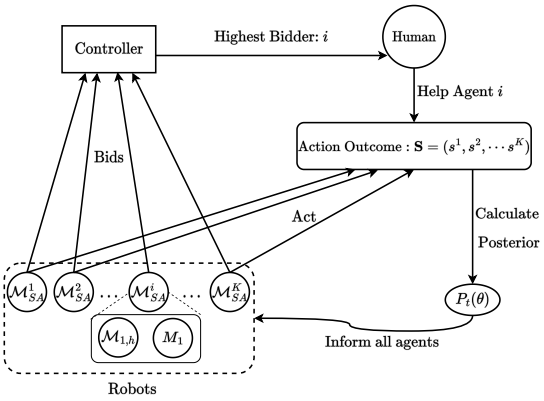


Figure 1: Diagram of the bidding process.

5.2 Bid Generation

For a given posterior distribution $P_t(\theta)$ over the latent variables, and a human operator described by $\mathcal{M}_h$, the robot M_i can formulate a shared autonomy system $\mathcal{M}^i_{SA}$. The robot assumes that they have access to human help at any time, and only they are able to observe the human and update the posterior distribution. This shared autonomy system can be solved using the BAMCP algorithm to get the history-dependent Q-values. These Q-values would only consider the expected rewards collected by the robot, and not the expected rewards collected by the other robots. We can use these Q-values to generate a bid for the human operator using Equation 6. However, this does not consider 1) how the observations made by the robot can help increase the rewards collected by other robots, and 2) how the posterior distribution may change during a run due to observations made by others. We address these problems by making the following modifications to the BAMCP algorithm.

5.2.1 Planning Horizon. In a typical BAMCP algorithm, a search tree is built by sampling the prior distribution to generate a sample MDP. The sample MDP is then simulated until reaching a goal state, and the trajectory taken is used to update the history-dependent nodes. This is defined as a single trial. As the number of samples increases, the distribution of the sample MDPs reaching nodes will converge to the posterior distribution over the latent variables given the node's history. However, information gained by other robots during a run can change the posterior distribution, making simulating until the end of a run unrealistic, as it cannot account for external changes in the distribution.

We therefore use a short planning horizon, as the runs with longer planning horizons do not accurately reflect the posterior distribution. This allows for a shorter run time. At the end of the planning horizon, we use an heuristic value function $V_i(s)$ to estimate the expected reward collected from the state. The heuristic value function $V_i(s)$, where the robot assumes no help from the human operator, is found by using value iteration [24] to solve M_i. This expected reward is then back-propagated through the nodes. This allows us to simulate the run with a short planning horizon to consider the benefits of asking for human help, while still considering the long-term aim of reaching a goal state.

5.2.2 Variance-Based Reward Bonus. When an robot has a short planning horizon, the Q-value of an action does not reflect the true value of an informative action, as the exploitation that can occur in later transitions is not considered. An informative action is a human action that may not directly result in rewards, but will inform the posterior distribution over the latent variables. Furthermore, in a multi-robot system, there may be actions that can benefit other robots, but not the robot that is taking the informative action. The robot only considers their own state-action space to compute the Q-values of these actions, and thus will not consider the expected value of the action for other robots. This leads to the robot undervaluing informative actions. To address this problem, we explicitly reward actions that reduce the variance of the belief distribution over the human operator's behaviour by altering the reward function of the BAMCP algorithm. The new history-dependent reward function for robot i is defined as:

$$R^i_{\text{new}}(s,h,a) = (1-\alpha)\cdot R^i_{SA}(s,a) + \alpha\cdot\left(\text{var}(\theta \mid h) - \int_{\theta\in\Theta}\sum_{s'\in S} T^i_{SA}(s,a,s')P_t(\theta)\text{var}(\theta \mid h,a,s')d\theta\right), \tag{7}$$

where α is the tuning parameter determining the weight of the variance term.

Calculating the variance using sample-based methods can be computationally expensive. The HPP-MDP reduces the computational complexity, as the posterior distribution over the latent variables is a polynomial function of θ. We can express the posterior distribution at a node with history h as:

$$P(\theta \mid h) = \sum_{j=1}^{J}\beta_j\prod_{i=1}^{N}\theta_i^{a_i^j}, \tag{8}$$

where β_j is the coefficient of the jth term of the polynomial function, and a_i^j is order of θ_i in the jth term of the polynomial function [10]. This can be used to calculate the variance of the posterior distribution at a node with history h as:

$$\begin{aligned}\text{var}(\theta \mid h) &= \sum_{k=1}^{N}\left[\int_{\theta\in\Theta} P(\theta \mid h)\cdot\theta_k^2 d\theta - \left(\int_{\theta\in\Theta} P(\theta \mid h)\cdot\theta_k d\theta\right)^2\right]\\ &= \sum_{k=1}^{N}\left\{\sum_{j=1}^{J}\beta_j\left(\prod_{i\neq k}\frac{1}{a_i^j+1}\right)\cdot\frac{1}{a_k^j+3} - \left[\sum_{j=1}^{J}\beta_j\left(\prod_{i\neq k}\frac{1}{a_i^j+1}\right)\cdot\frac{1}{a_k^j+2}\right]^2\right\}.\end{aligned} \tag{9}$$

The closed-form expression for the variance allows us to efficiently calculate the variance of the posterior distribution at each node in the search tree. We use this to calculate the new reward function in Equation 7 when running the BAMCP algorithm.

6 EXPERIMENTS

We compare the performance of our bidding system to the joint HPP-MDP approach and a baseline where the system assumes to know the human's behaviour, proposed by Dahiya et al. [12]. The methods are compared on two domains, a sequential task and a robot navigation task. We show empirically that our bidding system can produce a similar performance to the joint HPP-MDP approach under fixed time constraints. We outperform the approach proposed by Dahiya et al. [12] when there is uncertainty over the human's performance.

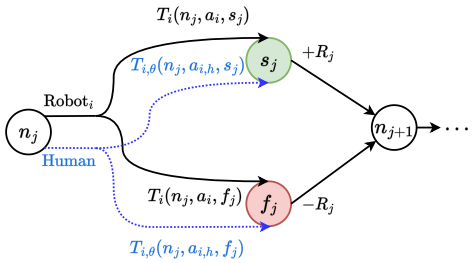


Figure 2: The jth task for robot i in the sequential task domain. The probability of robot i entering the success state is known to be $T_i(n_j, a_i, s_j)$. The probability of the human entering the success state is $T_{i,\theta}(n_j, a_{i,h}, s_j)$, a function of θ.

6.1 Sequential Tasks

We use this domain to directly compare joint HPP-MDP approach to our bidding system, and analyse the impact of α on the total rewards collected. We use a small domain with two robots, as the joint HPP-MDP approach cannot scale to larger domains.

6.1.1 Domain. We consider a domain with two robots, where they each have 10 tasks to complete. The tasks must be completed in a sequential order, and the task will either result in success or failure. This will either yield a positive or negative reward. The robot can attempt the task, or they can request help from the human operator. The probability of success for the robot for any given task is known, while the human operator's probability of success is expressed as a polynomial function over a latent parameter, θ. For example, the probability of success for task j for the human operator could be expressed as $T_\theta(n_j, a^h, s_j) = 0.4 + 0.3\cdot\theta$, where θ is unknown to the robot. An example of a task is shown in Figure 2. We configure the tasks such that 80% of robot 1's tasks have low reward and high uncertainty over the human's success rate ($T_\theta(n_j, a^h, s_j) = 0.1 + 0.9\cdot\theta$), while 20% of robot 2's tasks have high reward and high uncertainty over the human's success rate ($T_\theta(n_j, a^h, s_j) = 0.05 + 0.8\cdot\theta$). This configuration is designed to demonstrate how the information gained about the human by one robot can be beneficial to others, so acting independently does not result in overall higher rewards. The decay rate γ is 1.0, and at the end of the 10 tasks, the robot will enter a zero-reward absorbing state.

6.1.2 Algorithms. We apply the following methods on the sequential task domain: Our **bidding system** with a planning horizon of 2 steps, and $\alpha \in [0, 1]$. The **joint HPP-MDP** model is solved with a BAMCP-based solver.

6.1.3 Results. The joint HPP-MDP solver took on average 10.5 minutes to run 150, 000 trials per decision step. Our bidding system

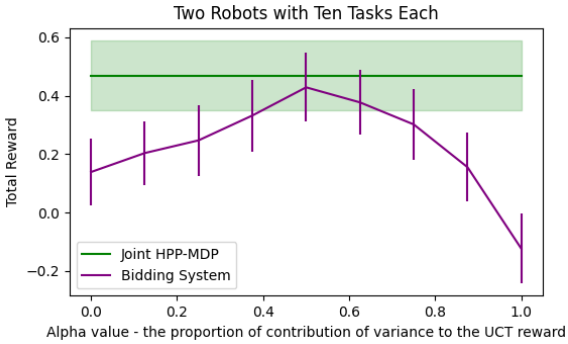


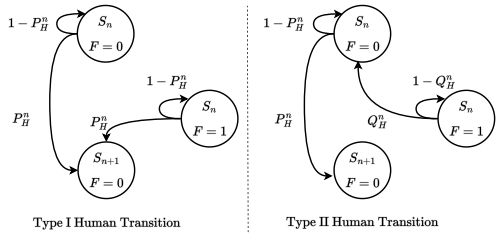
Figure 3: Results for bidding system and joint HPP-MDP solver on the sequential tasks domain with $k = 2$ robots and $N = 10$ tasks.

was run for 2 minutes per decision step. The results are shown in Figure 3. Our bidding system performs at an equivalent level to the joint HPP-MDP approach when α is between 0.4 and 0.6. As shown in Figure 3, the bidding system is sensitive to extreme values of α, where low values of α result in a lack of informative actions, while high values of α result in a lack of exploitative actions. As the number of tasks increases, the time taken to solve the joint HPP-MDP model increases, while the time taken to solve our bidding system remains constant. For example, we found that a three robot domain with 10 tasks took approximately 193 minutes to run 150, 000 steps, making it impractical to use the joint HPP-MDP model to allocate the human operator. Therefore, in problems where there are longer horizons or a higher number of robots, the joint HPP-MDP model may not be able to solve the problem in a reasonable amount of time, while our bidding system can still produce a competitive solution.

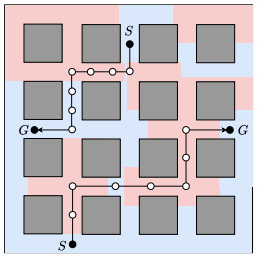
6.2 Robot Navigation

Dahiya et al. [12] proposes a framework for multi-robot shared autonomy, where the human operator can help the robot navigate to a goal. They used MDPs to model the robot and the human's behaviour, and each operator is modelled with a known probability of failure for each segment of the path to the goal. However, it is difficult to know how well the human can teleoperate the robot in a given environment. Therefore, a model with uncertainty over the human's performance may be able to better capture the true environment. We compare the performance of our bidding system approach to the approach proposed by Dahiya et al. [12] on a domain where the human's failure probabilities are unknown.

6.2.1 Domain. **Fixed Path Domain** Four robots are navigating in a city block-like randomly generated environment. The start and end location is also randomly generated, and the route planner finds a path. The path is split into 8 segments, where each segment is described as a single transition that can be done either by the robot or the human operator. In each segment, the autonomous robot can attempt the transition and may end in a fail state, where they cannot progress. Once in a fail state, a human must intervene to leave the fail state. There are two types of transitions possible, type-I and type-II transitions. In a type-I transition, the human operator



(a) Type I and II transitions for human operators. P_H^n is the probability of reaching the next state when the human controls the robot, and Q_H^n is the probability the human can recover the robot from a fail state in type II transition.



(b) Example Domain with two robots. The regions with type I transitions are coloured in blue, type II transitions are coloured in red.

Figure 4: Robot Navigation Domain

can go from a fail state to the end of the segment, while in a type-II transition, the human operator must return to the start of the segment before the autonomous robot can attempt the transition again. This is shown in Figure 4a. An example of the domain is shown in Figure 4b, where the type-I and II transition regions are shown in the blue and red areas. The decay rate for this domain was set to $\gamma = 1.0$, and the goal state was defined by a zero-reward absorbing state.

Generalized Domain Dahiya et al. [12]'s domain assumed the path was known to the robot, and the route was split into 8 segments independent of the start and end location. We generalize this domain to a five-by-five grid, where k robots can move in the cardinal directions. The transition from adjacent states has an equal probability of being a type I or type II transition. The start and end location in the grid is randomly generated.

In both the fixed path and general domains, the human success probabilities are randomly generated by sampling from a continuous range of possible values. The algorithms cannot directly observe the human's success probabilities, but they are given a continuous range of possible values. We fix the reward of attempting the transition at -2, the reward of entering the fail state is -4, and the reward of the human helping the robot is -0.75.

6.2.2 Algorithm. **Fixed Path Domain** Dahiya et al. [12] proposes a decentralised method for human help allocation in a multi-robot shared autonomy system, where the robot with the highest need for human assistance is given human help. The robot determines the need for human assistance based on their MDP. The MDP outlines a direct path of eight transitions from the start to the end location and gives the probabilities of success for each transition for the human

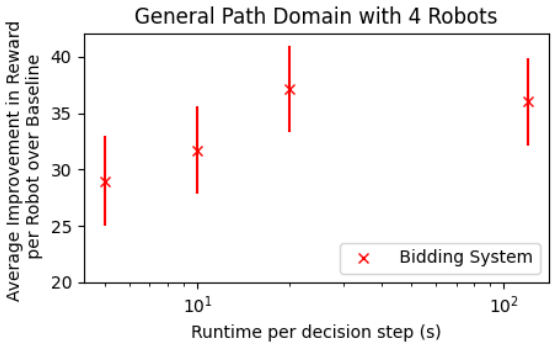


Figure 5: The difference in average reward per agent between our bidding system and the baseline, Dahiya et al. [12].

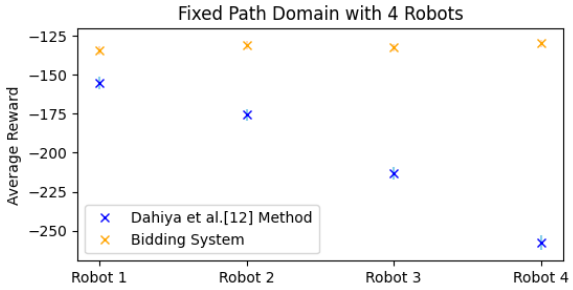


Figure 6: The average reward of the robots in the fixed path domain, using our bidding system and the method proposed by Dahiya et al. [12].

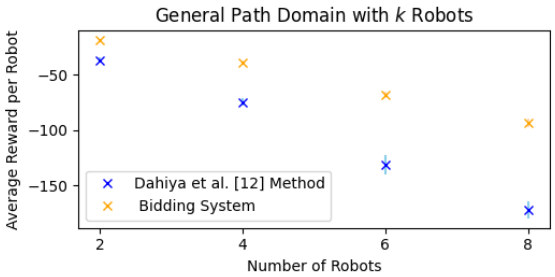


Figure 7: Average reward per robot in the general domain, where the number of robots is varied.

and autonomous operator. This is used to compute the Whitter Index [1] for each state. The index represents the additional human cost required to add into the reward function such that the expected total reward of the robot attempting the task is higher than when the human attempts the task. This involves an implicit assumption that the human will always outperform the robot, and the method assumes that the performance of the human is known. The robot with the highest Whitter index is given human help.

Generalized Domain In Dahiya et al. [12], the MDP is generated by placing seven waypoints between the start and end location, and the path planning between the two points is considered to be a separate problem. In this general domain, the path between the start and end location is found by solving the MDP where we only consider autonomous actions. This path is then used to generate a sequence of transitions from the start to the end location. Unlike the fixed path domain, the number of transitions is not fixed at eight.

Bidding System In both the fixed path and generalized domain, we apply our bidding system with a planning horizon that terminates once the robot reaches the next waypoint, and $\alpha = 0.99$. We set α high to balance the relative magnitudes of the first and second term in Equation 7. Every robot was given two minutes to compute their bid. As the algorithm was implemented in python, these computation times could be substantially improved. The two minutes bid computation time was fixed to ensure convergence of the policy. As shown in Fig. 5, the bidding system outperforms the baseline even at low computation times. In comparison, Dahiya et al. [12] took 15.2 ± 4.8 seconds to compute their static policy.

6.2.3 Results. **Fixed Path Domain** In the fixed path domain, under Dahiya et al. [12] the robots had an average reward of -200.44 ± 2.27, while our bidding system had an average reward of -132.07 ± 1.29. In a deadlock situation, where multiple robots submit the same bid, the robot with the lowest index is given human help, and this can be seen in Figure 6.

Generalized Path Domain We varied the number of robots in the domain from 2 to 8 and compared the average reward per robot for our bidding system and the method proposed by Dahiya et al. [12]. As shown in Figure 7, our bidding system outperforms the method proposed by Dahiya et al. [12] in all cases. The average reward per robot decreased as the number of robots increased for both approaches. This is because the human is being shared among more robots, and thus the human is less likely to be able to help an robot.

7 CONCLUSION

We have presented a efficient approach to allocating human assistance in a multi-robot shared autonomy system when there is uncertainty over the human operator's performance. Our online bidding system explicitly considered reducing the variance over the latent parameters governing the human operator. We demonstrated the computational infeasiblity of using a joint model for shared autonomy systems with large number of robots, and compared our approach to one where the uncertainty over the human's performance was not considered. In future work, we will consider systems with multiple human operators, and where the human operator's performance can change during execution.

ACKNOWLEDGMENTS

This work was supported by the Defence Science and Technology Laboratory, the EPSRC Programme Grant 'From Sensing to Collaboration' (EP/V000748/1), the Clarendon Fund at the University of Oxford, and a gift from Amazon Web Services. This document is an overview of UK MOD's Defence Science and Technology Laboratory (DSTL) sponsored research and is released for informational purposes only. The contents of this document should not be interpreted as representing the views of the UK MOD, nor should it be assumed that they reflect any current or future UK MOD policy.

REFERENCES

[1] Nima Akbarzadeh and Aditya Mahajan. 2022. Conditions for indexability of restless bandits and an O(K^3) algorithm to compute Whittle index. *Cambridge University Press* 54, Advances in Applied Probability (2022), 1164–1192. https://doi.org/10.1017/apr.2021.61 arXiv:2008.06111 [cs, eess, math].

[2] Sterling J Anderson, Steven C. Peters, Karl D. Iagnemma, and Tom E. Pilutti. 2009. A unified approach to semi-autonomous control of passenger vehicles in hazard avoidance scenarios. In *2009 IEEE International Conference on Systems, Man and Cybernetics*. IEEE International Conference on Systems, Man and Cybernetics, 2032–2037. https://doi.org/10.1109/ICSMC.2009.5346330 ISSN: 1062-922X.

[3] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas, and Shlomo Zilberstein. 2020. Learning to Optimize Autonomy in Competence-Aware Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS '20)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 123–131.

[4] Raphen Becker, Victor Lesser, and Shlomo Zilberstein. 2005. *Analyzing Myopic Approaches for Multi-Agent Communication*. Vol. 2005. IEEE/WIC/ACM International Conference on Intelligent Agent Technology. https://doi.org/10.1109/IAT.2005.44 Journal Abbreviation: Proceedings - 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT'05 Pages: 557 Publication Title: Proceedings - 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, IAT'05.

[5] Craig Boutilier. 1996. Planning, Learning and Coordination in Multiagent Decision Processes. In *TARK '96: Proceedings of the 6th conference on Theoretical aspects of rationality and knowledge*. Morgan Kaufmann Publishers Inc., The Netherlands, 195–210. https://dl.acm.org/doi/10.5555/1029693.1029710

[6] Craig Boutilier and Tyler Lu. 2016. Budget allocation using weakly coupled, constrained Markov decision processes. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence (UAI'16)*. AUAI Press, Arlington, Virginia, USA, 52–61.

[7] Yifan Cai, Abhinav Dahiya, Nils Wilde, and Stephen L. Smith. 2022. Scheduling Operator Assistance for Shared Autonomy in Multi-Robot Teams. In *2022 IEEE 61st Conference on Decision and Control (CDC)*. IEEE, Cancun, Mexico, 3997–4003. https://doi.org/10.1109/CDC51059.2022.9993014 ISSN: 2576-2370.

[8] Jack-Antoine Charles, Caroline P. C. Chanel, Corentin Chauffaut, Pascal Chauvin, and Nicolas Drougard. 2018. Human-Agent Interaction Model Learning based on Crowdsourcing. In *Proceedings of the 6th International Conference on Human-Agent Interaction (HAI '18)*. Association for Computing Machinery, New York, NY, USA, 20–28. https://doi.org/10.1145/3284432.3284471

[9] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. 2022. Shared Autonomy Systems with Stochastic Operator Models. In *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, Vienna, Austria, 4614–4620. https://doi.org/10.24963/ijcai.2022/640

[10] Clarissa Costen, Marc Rigter, Bruno Lacerda, and Nick Hawes. 2023. Planning with Hidden Parameter Polynomial MDPs. *Proceedings of the AAAI Conference on Artificial Intelligence* 37, 10 (June 2023), 11963–11971. https://doi.org/10.1609/aaai.v37i10.26411 Number: 10.

[11] Murat Cubuktepe, Nils Jansen, Mohammed Alshiekh, and Ufuk Topcu. 2021. Synthesis of Provably Correct Autonomy Protocols for Shared Control. *IEEE Trans. Automat. Control* 66, 7 (July 2021), 3251–3258. https://doi.org/10.1109/TAC.2020.3018029 Conference Name: IEEE Transactions on Automatic Control.

[12] Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan, and Stephen L. Smith. 2022. Scalable Operator Allocation for Multi-Robot Assistance: A Restless Bandit Approach. *IEEE Transactions on Control of Network Systems* 9 (Sept. 2022), 1397–1408. https://doi.org/10.1109/TCNS.2022.3153872 arXiv:2111.06437 [cs, eess].

[13] Frits de Nijs, Erwin Walraven, Mathijs de Weerdt, and Matthijs Spaan. 2017. Bounding the Probability of Resource Constraint Violations in Multi-Agent MDPs. *Proceedings of the AAAI Conference on Artificial Intelligence* 31, 1 (Feb. 2017). https://doi.org/10.1609/aaai.v31i1.11037 Section: Main Track: Planning and Scheduling.

[14] Francesco Duchetto, Ayse Kucukyilmaz, Luca Iocchi, and Marc Hanheide. 2018. Do Not Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation From Human Demonstrations. *IEEE Robotics and Automation Letters* PP (July 2018), 1–1. https://doi.org/10.1109/lra.2018.2861080

[15] Lu Feng, Clemens Wiltsche, Laura Humphrey, and Ufuk Topcu. 2016. Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems. *IEEE Transactions on Automation Science and Engineering* 13, 2 (April 2016), 450–462. https://doi.org/10.1109/TASE.2016.2530623 Conference Name: IEEE Transactions on Automation Science and Engineering.

[16] Anna Gautier, Marc Rigter, Bruno Lacerda, Nick Hawes, and Michael Wooldridge. 2023. Risk-Constrained Planning for Multi-Agent Systems with Shared Resources. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS '23)*. International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 113–121.

[17] Arthur Guez, Nicolas Heess, David Silver, and Peter Dayan. 2014. Bayes-Adaptive Simulation-based Search with Value Function Approximation. In *Advances in Neural Information Processing Systems*, Vol. 27. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2014/hash/839ab46820b524afda05122893c2fe8e-Abstract.html

[18] Kazi Mahmud Hasan, Abdullah-Al-Nahid, and Khondker Jahid Reza. 2014. Path planning algorithm development for autonomous vacuum cleaner robots. In *2014 International Conference on Informatics, Electronics Vision (ICIEV)*. 1–6. https://doi.org/10.1109/ICIEV.2014.6850799

[19] Emilie M. D. Jean-Baptiste, Pia Rotshtein, and Martin Russell. 2015. POMDP Based Action Planning and Human Error Detection. In *Artificial Intelligence Applications and Innovations (IFIP Advances in Information and Communication Technology)*, Richard Chbeir, Yannis Manolopoulos, Ilias Maglogiannis, and Reda Alhajj (Eds.). Springer International Publishing, Cham, 250–265. https://doi.org/10.1007/978-3-319-23868-5_18

[20] Sebastian Junges, Nils Jansen, Joost-Pieter Katoen, Ufuk Topcu, Ruohan Zhang, and Mary Hayhoe. 2018. Model Checking for Safe Navigation Among Humans. In *Quantitative Evaluation of Systems (Lecture Notes in Computer Science)*, Annabelle McIver and Andras Horvath (Eds.). Springer International Publishing, Cham, 207–222. https://doi.org/10.1007/978-3-319-99154-2_13

[21] C. Longjard, Prayoth Kumsawat, Kitti Attakitmongkol, and Arthit Srikaew. 2007. Automatic lane detection and navigation using pattern matching mode. In *Proceedings of the 7th WSEAS International Conference on Signal, Speech and Image Processing (SSIP'07)*. World Scientific and Engineering Academy and Society (WSEAS), Stevens Point, Wisconsin, USA, 44–49.

[22] Amal Nanavati, Christoforos Mavrogiannis, Kevin Weatherwax, Leila Takayama, Maya Cakmak, and Siddhartha Srinivasa. 2021. Modeling Human Helpfulness with Individual and Contextual Factors for Robot Planning. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation. https://doi.org/10.15607/RSS.2021.XVII.016

[23] Frits de Nijs, Matthijs Spaan, and Mathijs de Weerdt. 2018. Preallocation and Planning Under Stochastic Resource Constraints. *Proceedings of the AAAI Conference on Artificial Intelligence* 32, 1 (April 2018). https://doi.org/10.1609/aaai.v32i1.11592 Number: 1.

[24] Martin Puterman. 1994. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Ltd.

[25] Marc Rigter, Bruno Lacerda, and Nick Hawes. 2020. A Framework for Learning From Demonstration With Minimal Human Effort. *IEEE Robotics and Automation Letters* 5, 2 (April 2020), 2023–2030. https://doi.org/10.1109/LRA.2020.2970619

[26] Ariel Rosenfeld, Noa Agmon, Oleg Maksimov, and Sarit Kraus. 2017. Intelligent agent supporting human–multi-robot team collaboration. *Artificial Intelligence* 252 (Nov. 2017), 211–231. https://doi.org/10.1016/j.artint.2017.08.005

[27] Stephanie Rosenthal and Manuela Veloso. 2012. Mobile robot planning to seek help with spatially-situated tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence (AAAI'12)*. AAAI Press, Toronto, Ontario, Canada, 2067–2073.

[28] Gokul Swamy, Siddharth Reddy, Sergey Levine, and Anca D. Dragan. 2020. Scaled Autonomy: Enabling Human Operators to Control Robot Fleets. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 5942–5948. https://doi.org/10.1109/ICRA40945.2020.9196792 ISSN: 2577-087X.

[29] Erwin Walraven and Matthijs T. J. Spaan. 2018. Column Generation Algorithms for Constrained POMDPs. *Journal of Artificial Intelligence Research* 62 (July 2018), 489–533. https://doi.org/10.1613/jair.1.11216

[30] Kyle Hollins Wray, Luis Pineda, and Shlomo Zilberstein. 2016. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI'16)*. AAAI Press, New York, New York, USA, 517–523.

[31] Bo Wu, Bin Hu, and Hai Lin. 2017. Toward efficient manufacturing systems: A trust based human robot collaboration. In *2017 American Control Conference (ACC)*. 1536–1541. https://doi.org/10.23919/ACC.2017.7963171 ISSN: 2378-5861.

[32] Jianhui Wu and Edmund H. Durfee. 2010. Resource-Driven Mission-Phasing Techniques for Constrained Agents in Stochastic Environments. *Journal of Artificial Intelligence Research* 38 (July 2010), 415–473. https://doi.org/10.1613/jair.3004 arXiv:1401.3845 [cs].

[33] kirk A. Yost and Alan R. Washburn. 2000. *The LP/POMDP Marriage: Optimization with Imperfect Information*. Technical Report. NAVAL POSTGRADUATE SCHOOL MONTEREY CA DEPT OF OPERATIONS RESEARCH. https://apps.dtic.mil/sti/citations/ADA487441 Section: Technical Reports.

7.1 Limitations

The multi-agent system presented in this paper has several limitations that result in sub-optimal behaviours. The first limitation is that the robots' planning horizon is one task. This approach comes from the possibility that the robot may receive new observations of the human's behaviour from the other robots, which can subsequently alter the Q-values of the state-action pairs. As a result, planning for long horizons is impractical as the robot cannot anticipate all possible observations that might be made by other agents. This limitation can lead to the robot acting greedily, as the robot does not explicitly consider how it's actions may affect future states.

Another significant limitation of this work is the necessity of tuning the hyper-parameter α in the reward function. As shown in Figure 3 in the paper, the system's performance is highly sensitive to the value of α. Therefore, the user must tune this hyper-parameter prior to execution, otherwise the system may exhibit sub-optimal choices. We found that the algorithm acts closest to optimal when α is set so that the relative magnitudes of the rewards from the environment and the reward from the variance are similar in magnitude. This can be used as a heuristic for selecting an appropriate value for α.

The final limitation of this framework came from the use of the HPP-MDP model to represent the human operator. As highlighted in Chapter 5, the HPP-MDP requires the latent parameters to be static during execution. This can be an unrealistic assumption for human operators, where their behaviours may alter due to hidden factors, such as fatigue. However, the HPP-MDP formulation allows us to maintain a closed-form distribution over the human's internal state, and thus efficiently allocate human help to the best robot for any given scenario.

Chapter 8

Conclusion

In this thesis, we addressed the challenges of modelling human behaviour in sequential decision-making problems for shared autonomy system. We focused on modelling three features of human behaviour: the uncertainty of human behaviour, the continuous nature of the space of possible behaviour, and the variation in human performance over the course of a mission. We explored a range of MDP frameworks to capture these features, and their performance was compared against frameworks presented in the literature.

We began by exploring how assumptions over the evolution of human behaviour impacts the performance of the shared autonomy system. In Chapter 4, we compared the framework proposed by Feng et al. (2016), which assumes full observability of the human's internal state, with our proposed MOMDP framework, which explicitly considers the hidden nature of the human's internal state. Our experiments found that the MOMDP framework outperformed the baseline under conditions where the stochastic transition dynamics of the human's internal state are known. This improvement came from the MOMDP's ability to track their true internal state based on the observations, while the baseline assumed the knowledge of the current internal state, independent on the observations they were making. However, even when our MOMDP framework was implemented with incorrect model assumptions, it still outperformed the baseline that used correct

model assumptions. This outcome highlights the significance of explicitly modelling the hidden nature of the human's internal state in shared autonomy systems.

In Chapter 5 and Chapter 7, we explored how to represent the continuous nature of the space of possible human behaviour in shared autonomy systems. Typically, a BAMDP would be used to represent a continuous and hidden parameter in a MDP, and the belief would be maintained using a particle filter. The HPP-MDP presented in Chapter 5 was able to represent the continuous and hidden nature of a human's internal state, while also maintaining a closed-form distribution over the latent parameter space. This would prove to be advantageous in the multi-agent setting presented in Chapter 7, where the robot must evaluate the benefits of learning about the human to help other agents. The closed-form distribution allowed the robot to efficiently compute the variance in belief over the latent parameters, and use this variance as an indicator of how much the robot should prioritise learning. However, the HPP-MDP framework's assumption of a human's static internal state may not be practical in the real world, when considering features such as fatigue.

To represent the dynamic, continuous and hidden nature of human behaviour, we proposed a cont-MOMDP framework in Chapter 6. This framework allowed us to represent the richest model describing human behaviour, but we were unable to maintain a closed-form distribution over the latent parameter space. As shown in Chapter 5, closed-form distributions result in faster belief updates than particle-based belief representations, and thus more efficient computation of policies. This represents a clear trade-off between the complexity of the model and the efficiency of the computation.

Ultimately, the most effective and suitable framework presented in this thesis depends heavily on the specific context of the shared autonomy system, and what our objective is. For example, a simple MOMDP framework is ideal for assessing whether autonomy outperforms human performance in tasks like computer games. In contrast, the HPP-MDP framework is best suited for multi-robot shared autonomy system (such as in a

warehouse), where the robot must share information about the human helper's skills in various tasks to optimize task allocations. Finally, in scenarios where a shared autonomy system is monitoring a human driver, ready to take over control if the human shows signs of fatigue, the cont-MOMDP is best suited. It enables the agent to reason over the evolution of the driver's fatigue if they continue to drive, and decide whether to alter the route to take control to minimise risks. Shared autonomy systems become increasingly integrated into our daily lives, choosing the appropriate frameworks for each scenario is crucial to ensuring optimal performance of the system.

8.1 Future Work

Application on Real-World Domains

Chapter 5, while we conducted experiments on a real computer game, these experiments used pre-programmed software to play the game, and represent the human. The next step to further our experimental contributions would be to implementing our framework onto a shared autonomy system with actual human participants. We have begun developing an experimental domain using the CARLA driving simulator to collect data from human drivers. In this set up, the mission involves navigating a car from A to B, and the agent must decide the driver (human or autonomous) and the route the car takes (there exists multiple routes from the same start and finish). The goal is to demonstrate a shared autonomy system in which the agent evaluates the capabilities of the human driver, and adapts the route or operator of the car to minimise the travel time, while minimising the risk of accidents.

Complex Human Models

In Chapter 6 and Chapter 7, we modelled human performance as a linear combination of MDPs.

This is based on the assumption that if a "normal" human has a success probability

of a task p_{norm}, and a "tired" human has a success probability of p_{tired}, a human who is half-tired would have a success probability of $(p_{norm} + p_{tired})/2$. However, this model may not hold true for certain tasks; for example, a human driving down a straight road will not exhibit signs of fatigue until they are very close to a fully"tired" state. A future research topic is developing a human model where different tasks have unique rates of fatigue, and these rates are learnt during execution. This approach will enable the agent to learn which tasks the human can reliably complete, even after many tasks.

Blended Autonomy

This thesis primarily focused on modelling the human in a shared autonomy system, with a switching system to transfer control from one operator to another. Blended autonomy, where the input from multiple operators are combined to control a robot, is an interesting extension to our shared autonomy system. This approach is often used to maintain safety of the robot, in scenarios where the human operator may be unaware of environmental risks. The models developed in this thesis allow the agent to evaluate the confidence they have over the human's capabilities. We can incorporate blended autonomy into our shared autonomy system, where the agent adjusts the weighting of each operator's input based on it's belief over the human's capabilities. As the agent gains more information about the human, the human's input can be more heavily weighed in tasks with high confidence, while the autonomous controller can be given more control in tasks where the human is known to be less successful.

Long-term missions

The research presented in this thesis focused on one-shot missions, where the hidden parameters or states are reset every mission. However, in real-world applications, many of the shared autonomy systems will often interact the same person repeatedly, such as a semi-autonomous car driver, or a warehouse worker. In such scenarios, it is important to retain certain information across missions, such as the individual's performance in various tasks, or their rate of fatigue. However, other information, such as their current level of

tiredness, may only be relevant for the particular mission and can be discarded upon completion. Future work could explore models that incorporate both hidden states and parameters, allowing the system to retain and adapt relevant information over multiple missions while discarding information unique to the mission. This approach would result in a system that continuously improves and adapts to the human as the human continues to interact with the system.

Bibliography

C. Basich, J. Svegliato, K. H. Wray, S. Witwicki, J. Biswas, and S. Zilberstein. Learning to Optimize Autonomy in Competence-Aware Systems. In *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '20, pages 123–131, Richland, SC, May 2020. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-7518-4.

J. Casper and R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the World Trade Center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, June 2003. ISSN 1941-0492. doi: 10.1109/TSMCB.2003.811794. URL `https://ieeexplore.ieee.org/document/1200160`. Conference Name: IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics).

A. Cass, L. Kaelbling, M. Littman, A. Cassandra, L. Pack, K. Michael, and L. Littman. Acting Optimally in Partially Observable Stochastic Domains. Mar. 1995.

J.-A. Charles, C. P. C. Chanel, C. Chauffaut, P. Chauvin, and N. Drougard. Human-Agent Interaction Model Learning based on Crowdsourcing. In *Proceedings of the 6th International Conference on Human-Agent Interaction*, HAI '18, pages 20–28, New York, NY, USA, Dec. 2018. Association for Computing Machinery. ISBN 978-1-4503-5953-5. doi: 10.1145/3284432.3284471. URL `https://doi.org/10.1145/3284432.3284471`.

J. Crandall and M. Goodrich. Experiments in adjustable autonomy. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, volume 3, pages 1624–1629 vol.3, Oct. 2001. doi: 10.1109/ICSMC.2001.973517. ISSN: 1062-922X.

M. Cubuktepe, N. Jansen, M. Alshiekh, and U. Topcu. Synthesis of Provably Correct Autonomy Protocols for Shared Control. *IEEE Transactions on Automatic Control*, 66(7): 3251–3258, July 2021. ISSN 1558-2523. doi: 10.1109/TAC.2020.3018029. Conference Name: IEEE Transactions on Automatic Control.

A. Dahiya, N. Akbarzadeh, A. Mahajan, and S. L. Smith. Scalable Operator Allocation for Multi-Robot Assistance: A Restless Bandit Approach. *IEEE Transactions on Control of Network Systems*, 9:1397–1408, Sept. 2022. doi: 10.1109/TCNS.2022.3153872. URL `http://arxiv.org/abs/2111.06437`. arXiv:2111.06437 [cs, eess].

D. Dawson and K. Reid. Fatigue, alcohol and performance impairment. *Nature*, 388(6639):235–235, July 1997. ISSN 1476-4687. doi: 10.1038/40775. URL `https://www.nature.com/articles/40775`. Bandiera_abtest: a Cg_type: Nature Research Journals Number: 6639 Primary_atype: Research Publisher: Nature Publishing Group.

F. Duchetto, A. Kucukyilmaz, L. Iocchi, and M. Hanheide. Do Not Make the Same Mistakes Again and Again: Learning Local Recovery Policies for Navigation From Human Demonstrations. *IEEE Robotics and Automation Letters*, PP:1–1, July 2018. doi: 10.1109/lra.2018.2861080.

M. O. Duff. *Optimal Learning: Computational Procedures For Bayes-Adaptive Markov Decision Process*. PhD thesis, University of Massachusetts, Feb. 2002. URL `https://www.gatsby.ucl.ac.uk/ yael/Okinawa/DuffThesis.pdf`.

FANUC. Assembly Robots | Assembly Line Robots | FANUC America, 2024. URL https://www.fanucamerica.com/solutions/applications/assembly-line-robots.

L. Feng, C. Wiltsche, L. Humphrey, and U. Topcu. Synthesis of Human-in-the-Loop Control Protocols for Autonomous Systems. *IEEE Transactions on Automation Science and Engineering*, 13(2):450–462, Apr. 2016. ISSN 1558-3783. doi: 10.1109/TASE.2016.2530623. Conference Name: IEEE Transactions on Automation Science and Engineering.

A. Fern, S. Natarajan, K. Judah, and P. Tadepalli. A Decision-Theoretic Model of Assistance. *Journal of Artificial Intelligence Research*, 50: 71–104, May 2014. ISSN 1076-9757. doi: 10.1613/jair.4213. URL https://www.jair.org/index.php/jair/article/view/10880.

J. Fu and U. Topcu. Synthesis of Shared Autonomy Policies With Temporal Logic Specifications. *IEEE Transactions on Automation Science and Engineering*, 13 (1):7–17, Jan. 2016. ISSN 1558-3783. doi: 10.1109/TASE.2015.2499164. URL https://ieeexplore.ieee.org/document/7339492/?arnumber=7339492. Conference Name: IEEE Transactions on Automation Science and Engineering.

A. Grau, M. Indri, L. Lo Bello, and T. Sauter. Robots in Industry: The Past, Present, and Future of a Growing Collaboration With Humans. *IEEE Industrial Electronics Magazine*, 15(1):50–61, Mar. 2021. ISSN 1941-0115. doi: 10.1109/MIE.2020.3008136. URL https://ieeexplore.ieee.org/document/9305203. Conference Name: IEEE Industrial Electronics Magazine.

A. Guez, D. Silver, and P. Dayan. Efficient Bayes-Adaptive Reinforcement Learning using Sample-Based Search. In *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc., 2012. URL https://proceedings.neurips.cc/paper/2012/hash/35051070e572e47d2c26c241ab88307f-Abstr

N. Ishikawa and K. Suzuki. Development of a human and robot collaborative system for inspecting patrol of nuclear power plants. In *Proceedings 6th IEEE International Workshop on Robot and Human Communication. RO-MAN'97 SENDAI*, pages 118–123, Sept. 1997. doi: 10.1109/ROMAN.1997.646967.

S. Javdani, S. S. Srinivasa, and J. A. Bagnell. Shared Autonomy via Hindsight Optimization. *arXiv:1503.07619 [cs]*, Apr. 2015. URL `http://arxiv.org/abs/1503.07619`. arXiv: 1503.07619.

E. M. D. Jean-Baptiste, P. Rotshtein, and M. Russell. POMDP Based Action Planning and Human Error Detection. In R. Chbeir, Y. Manolopoulos, I. Maglogiannis, and R. Alhajj, editors, *Artificial Intelligence Applications and Innovations*, IFIP Advances in Information and Communication Technology, pages 250–265, Cham, 2015. Springer International Publishing. ISBN 978-3-319-23868-5. doi: 10.1007/978-3-319-23868-$5_1$8.

S. Junges, N. Jansen, J.-P. Katoen, U. Topcu, R. Zhang, and M. Hayhoe. Model Checking for Safe Navigation Among Humans. In A. McIver and A. Horvath, editors, *Quantitative Evaluation of Systems*, Lecture Notes in Computer Science, pages 207–222, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99154-2. doi: 10.1007/978-3-319-99154-$2_1$3.

L. P. Kaelbling, M. L. Littman, and A. R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101(1):99–134, May 1998. ISSN 0004-3702. doi: 10.1016/S0004-3702(98)00023-X. URL `http://www.sciencedirect.com/science/article/pii/S000437029800023X`.

A.-B. Karami, L. Jeanpierre, and A.-I. Mouaddib. Partially Observable Markov Decision Process for Managing Robot Collaboration with Human. In *2009 21st IEEE International Conference on Tools with Artificial Intelligence*, pages 518–521, Nov. 2009. doi: 10.1109/ICTAI.2009.61. URL

https://ieeexplore.ieee.org/abstract/document/5366844?casa$_{t}oken$ = $rzFUPUzRsxIAAAAA$: $BCwKtaps2TgifCxqup4ipDxMtUGE - 1BHk - ZSDNoE3Q7A -_{H} lL - i9ijf0M74GYX6JG1o_{A}001.ISSN$: 2375 − 0197.

L. Kocsis and C. Szepesvári. Bandit Based Monte-Carlo Planning. In J. Fürnkranz, T. Scheffer, and M. Spiliopoulou, editors, *Machine Learning: ECML 2006*, Lecture Notes in Computer Science, pages 282–293, Berlin, Heidelberg, 2006. Springer. ISBN 978-3-540-46056-5. doi: 10.1007/11871842$_{2}$9.

C.-P. Lam and S. S. Sastry. A POMDP framework for human-in-the-loop system. In *53rd IEEE Conference on Decision and Control*, pages 6031–6036, Dec. 2014. doi: 10.1109/CDC.2014.7040333. URL https://ieeexplore.ieee.org/document/7040333. ISSN: 0191-2216.

S. Mahmud, C. Basich, and S. Zilberstein. Semi-Autonomous Systems with Contextual Competence Awareness. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '23, pages 689–697, Richland, SC, May 2023. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 978-1-4503-9432-1.

Mausam and A. Kolobov. Planning with Markov Decision Processes: An AI Perspective. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–210, June 2012. ISSN 1939-4608, 1939-4616. doi: 10.2200/S00426ED1V01Y201206AIM017. URL http://www.morganclaypool.com/doi/abs/10.2200/S00426ED1V01Y201206AIM017.

C. L. R. McGhan, A. Nasir, and E. M. Atkins. Human Intent Prediction Using Markov Decision Processes. *Journal of Aerospace Information Systems*, 12 (5):393–397, May 2015. ISSN 2327-3097. doi: 10.2514/1.I010090. URL https://arc.aiaa.org/doi/10.2514/1.I010090.

A. Nanavati, C. Mavrogiannis, K. Weatherwax, L. Takayama, M. Cakmak, and S. Srinivasa. Modeling Human Helpfulness with Individual and Contextual Factors for Robot Planning. In *Robotics: Science and Systems XVII*. Robotics: Science and Systems Foundation, July 2021. ISBN 978-0-9923747-7-8. doi: 10.15607/RSS.2021.XVII.016. URL http://www.roboticsproceedings.org/rss17/p016.pdf.

F. Petric and Z. Kovacic. Design and Validation of MOMDP Models for Child–Robot Interaction Within Tasks of Robot-Assisted ASD Diagnostic Protocol. *International Journal of Social Robotics*, 12(2):371–388, May 2020. ISSN 1875-4805. doi: 10.1007/s12369-019-00577-0. URL https://doi.org/10.1007/s12369-019-00577-0.

J. Pineau, G. Gordon, and S. Thrun. Point-based value iteration: An anytime algorithm for POMDPs. pages 1025–1032, Acapulco, Mexico, Jan. 2003.

M. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Ltd, 1994.

M. Rigter, B. Lacerda, and N. Hawes. A Framework for Learning From Demonstration With Minimal Human Effort. *IEEE Robotics and Automation Letters*, 5(2):2023–2030, Apr. 2020. ISSN 2377-3766, 2377-3774. doi: 10.1109/LRA.2020.2970619. URL https://ieeexplore.ieee.org/document/8976128/.

M. Rigter, B. Lacerda, and N. Hawes. Minimax Regret Optimisation for Robust Planning in Uncertain Markov Decision Processes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13):11930–11938, May 2021. ISSN 2374-3468. URL https://ojs.aaai.org/index.php/AAAI/article/view/17417. Number: 13.

S. Rosenthal and M. Veloso. Mobile robot planning to seek help with spatially-situated tasks. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 2067–2073, Toronto, Ontario, Canada, July 2012. AAAI Press.

P. Schörner, L. Töttel, J. Doll, and J. M. Zöllner. Predictive Trajectory Planning in Situations with Hidden Road Users Using Partially Observable Markov Decision Processes. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2299–2306, June 2019. doi: 10.1109/IVS.2019.8814022. URL https://ieeexplore.ieee.org/abstract/document/8814022?casa$_token$ = $ILbVQmB_{DI}AAAAA$: $EIK0PHZ1BoVYlXBGs7bfC9FjlLCXHMdkt4$ − $Gi6BbCWlHUc4WOHyn4W2 - msjnnqP4UAIVqAow.ISSN$: 2642 − 7214.

D. Silver and J. Veness. Monte-Carlo planning in large POMDPs. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 2*, NIPS'10, pages 2164–2172, Red Hook, NY, USA, Dec. 2010. Curran Associates Inc.

M. Suilen, T. Badings, E. M. Bovy, D. Parker, and N. Jansen. Robust markov decision processes: A place where ai and formal methods meet, 2024. URL https://arxiv.org/abs/2411.11451.

F. van Wyk, A. Khojandi, and N. Masoud. Optimal switching policy between driving entities in semi-autonomous vehicles. *Transportation Research Part C: Emerging Technologies*, 114:517–531, May 2020. ISSN 0968-090X. doi: 10.1016/j.trc.2020.02.011. URL https://www.sciencedirect.com/science/article/pii/S0968090X19300981.

K. H. Wray, L. Pineda, and S. Zilberstein. Hierarchical approach to transfer of control in semi-autonomous systems. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16, pages 517–523, New York, New York, USA, July 2016. AAAI Press. ISBN 978-1-57735-770-4.

K. H. Wray, S. J. Witwicki, and S. Zilberstein. Online Decision-Making for Scalable Autonomous Systems. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence*, pages 4768–4774, Melbourne, Australia, Aug. 2017. International

Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-0-3. doi: 10.24963/ijcai.2017/664. URL https://www.ijcai.org/proceedings/2017/664.

B. Wu, B. Hu, and H. Lin. Toward efficient manufacturing systems: A trust based human robot collaboration. In *2017 American Control Conference (ACC)*, pages 1536–1541, May 2017. doi: 10.23919/ACC.2017.7963171. ISSN: 2378-5861.