Concurrent Markov Decision Processes for Robust Robot Team Learning under Uncertainty

by

Justin Girard

A thesis submitted in conformity with the requirements for the degree of Master of Applied Science

Institute for Aerospace Studies

University of Toronto

© Copyright by Justin Girard 2014
Concurrent Markov Decision Processes for Robust Robot Team Learning under Uncertainty

Justin Girard

Master of Applied Science

Institute for Aerospace Studies

University of Toronto

2014

Abstract

For robots to become a more common fixture in private and public industries, they must exhibit compliant individual and social learning. To achieve social compliance, while maintaining individual performance, robots must represent knowledge accurately in both certain and uncertain environments. Robots also need to quantify effective decision making both when isolated and when teamed with peer robots and humans. Thus, this thesis considers improvements to the Concurrent Individual and Social Learning (CISL) approach [30, 31], and addresses all of the above problems by exploring three subjects: learning problem representation using Markov Decision Processes (MDPs) [17], state uncertainty and state estimation [18], and advice sharing from both robot and human advisors [17, 19].
Acknowledgement

Thanks go to my supervisor Dr. M. Reza Emami, who tirelessly guided me in my research. It would have been impossible to have such an experience as I have had, without both consistent access to his time and his character. To my family and friends I deliver as much heartfelt appreciation as possible, as they have lent me support despite my being vacant both emotionally and physically. Lastly, I thank my grandparents, my mother, and my father, who always taught me to free my imagination, to remain curious, and to enjoy hard work.
# Table of Contents

Abstract ........................................................................................................................................... ii

Acknowledgement ................................................................................................................................... iii

Table of Contents ................................................................................................................................... iv

List of Tables ......................................................................................................................................... vi

List of Figures ....................................................................................................................................... vii

Chapter 1: Introduction ....................................................................................................................... 1

1.1 Motivation ..................................................................................................................................... 1

1.2 Outline ......................................................................................................................................... 2

1.3 Background ................................................................................................................................. 3

1.3.1 Decentralized Representations ............................................................................................. 4

1.3.2 Partial State Representations .............................................................................................. 5

1.3.3 Concurrent MDP Representations ....................................................................................... 6

1.3.4 State Observation Uncertainty ............................................................................................ 7

1.3.5 Human Advice, Crowdsourcing and Learning from Demonstration .................................. 8

Chapter 2: Concurrent Individual and Social Learning Theory ........................................................... 10

2.1 Single Agent Concurrent Markov Decision Process ................................................................. 10

2.2 Decentralized Concurrent Markov Decision Processes ............................................................ 12

2.3 Concurrent Individual and Social Learning ............................................................................. 13

2.3.1 Collective Learning: Individual Performance Policy Regression ........................................ 15

2.3.2 Cooperative Learning: Task Allocation Policy Regression .................................................. 16

Chapter 3: State Observation Uncertainty ......................................................................................... 22

3.1 Individual Performance MDP and Uncertainty ........................................................................ 22

3.2 Measuring State Observation Uncertainty ................................................................................ 24

3.2.1 Expected False Observation: ................................................................................................. 25
List of Tables

Table 1: Reward function for the Individual Performance MDP ................................................. 40
Table 2: Reward function for a Task Allocation MDP .............................................................. 43
Table 3: Paired p-values for Total Effort .................................................................................. 54
Table 4: Select pairwise p-values for the CISL and RCISL uncertainty experiments .......... 63
Table 5: Advice Sharing pairwise p-values ............................................................................. 68
List of Figures

Figure 1: Single-agent CMDP ....................................................................................................... 10
Figure 2: A decentralized CMDP ............................................................................................... 12
Figure 3: A representation of the simulation environment ........................................................... 38
Figure 4: Secret Collect Simulation Game .................................................................................... 47
Figure 5: Qualitative individual performance of a robot ............................................................ 49
Figure 6: Average Individual Reward (RI) for a typical robot .................................................... 50
Figure 7: Mean performance of concurrent and decentralized MDPs ........................................... 51
Figure 8: CMDP vs Single Q-Learning, Cooperative Effort ......................................................... 53
Figure 9: One standard deviation unit for a) Simulation Time and b) Total Effort ....................... 54
Figure 10: Uncertainty: Robot foraging performance under uncertainty ..................................... 57
Figure 11: Uncertainty: Average Individual Reward under uncertainty ..................................... 59
Figure 12: Uncertainty: Expected False Observation and Expected False Reward .................... 60
Figure 13: Uncertainty: Mean performance of CISL and RCISL ................................................. 61
Figure 14: Uncertainty: Robot’s physical Cooperative Effort ...................................................... 62
Figure 15: Advice Sharing: Average Individual Reward RI ....................................................... 64
Figure 16: Advice Sharing: Advice Sharing experiment metrics ............................................... 65
Figure 17: Advice Sharing standard deviation .......................................................................... 68
Chapter 1

Introduction

1.1 Motivation

The focus of robust robot learning, in both popular culture and the research community, is frequently on the single agent. This bias correlates with the difficulty presented by team robotics. If noiseless multi-agent problems are formulated as single Multi-agent Markov Decision Processes (MMDP) [9], it is immediately evident that an intractable expansion of state space is encountered. Further, the consideration of uncertainty makes such Markov Decision Processes (MDPs) even less tractable, as the multi-agent problems become Partially Observable Markov Decision Processes (POMDPs), whose theoretical solutions require the maintenance of many simultaneous state beliefs. Given the difficulty, it is understandable that robust multi-agent regression models exhibiting theoretical rigor are less frequent in both popular culture and technical literature. In addition, due to the difficulty inherent in autonomous team robotics, there is a demand for robots to collaborate both with human advisors, and with peer robots that exhibit varying levels of experience and capability. Indeed, in the future, robots will likely be tasked with learning alone, in teams, and/or from human advisors, both in certain and uncertain scenarios. Unfortunately, literature on frameworks that simultaneously address theoretically optimal multi-agent learning, taking into account state measurement uncertainty, and access to human advisors, is lacking.

Thus, there is opportunity for a framework that can solve large multi-agent problems theoretically, empirically, under uncertainty and with feedback from both peer and human advisors. Previously, the Concurrent Individual and Social Learning (CISL) mechanism has been shown to addresses the need for simultaneous individual and team behavioural improvements [30], but has not has demonstrated theoretically optimal behaviour, robustness to state uncertainty, or compliancy in response to human advice. This thesis shows that the CISL can be expanded in each above direction.
1.2 Outline

This work covers three categories. First, a theoretical model coined Concurrent Markov Decision Processes (CMDPs) is introduced, and used to analyze multiple multi-agent regression approaches. Second, the effect of state uncertainty on one CISL approach, generally called “the CISL mechanism”, is considered both theoretically and within a foraging case study. Lastly, two advice sharing mechanisms are integrated into the CISL mechanism, to enable both agent-agent and human-agent advice sharing.

Chapter 1 outlines this thesis, and explores the background of MDP research, highlighting several topics: decentralized MDPs, degenerate or partial state representations for MDPs, Concurrent MDP models, state uncertainty and robustifying techniques, and methods to leverage advice both from other agents and human advisors.

Chapter 2 introduces three CISL implementations that solve for three corresponding CMDPs. These approaches are contrasted with a control approach, a decentralized policy improvement algorithm that solves for a DEC-MDP [5]. Chapter 2 analyzes the theoretical basis of the CISL approaches, and the properties of the three corresponding CMDP models.

Chapter 3 discusses the effect of state uncertainty on the CISL mechanism, by exploring both the theoretical properties of MDPs and the performance of the CISL mechanism within a noisy environment. The consideration of state uncertainty leads to characterizing metrics, which express the robustness of a regression approach. The Robust Concurrent Individual and Social Learning (RCISL) approach is developed, which requires the use of a state estimator. The state estimator utilizes the expected value of a particle filter and is comprehensively derived. The empirical performance of the CISL and RCISL mechanisms are examined both by investigating the characterizing metrics and by investigating the observed behaviour of the multi-agent team.

In Chapter 4 methods to augment multi-agent reinforcement learning with suggested actions from both human and robot advisors are discussed. The suggested actions, called advice, are retrieved from both the Human Advice Layer (HAL), and the Advice Exchange (AE) mechanism. The advice retrieved from the AE mechanism is characterized by Q-learning utility values. The advice retrieved from the HAL is characterized by a Gaussian Mixture Model (GMM) trained with data obtained from human advisors.
Chapter 5 studies the performance of the aforementioned multi-agent policy regression approaches within a heterogeneous foraging case study. To complete the foraging scenario, items must be returned to a target zone by robots. To return any item, each robot must effectively learn from the simulation environment, including how to avoid obstacles and collisions, how to physically cooperate with other robots, how to accept advice from agents or humans, and how to initiate or acquiesce from item assignments. Afterward, the results are studied by considering theoretical metrics, such as the quantity of reward obtained from an MDP, and empirical metrics, such as the total agent effort required when completing the foraging objective.

In Chapter 6 several discussions and conclusive results are made. In summary, the CISL approach and the introduced CMDP model are shown to be efficacious for the enhancement of heterogeneous team foraging performance, and theoretically succinct in terms of representing complex and interdependent multi-agent learning behaviours. The RCISL mechanism is shown to maintain efficient performance despite all tested levels of state uncertainty, and all advice sharing mechanisms conferred a benefit to the robot team’s learning performance.

1.3 Background

In this section, previous research on learning process representation using Multi-agent Markov Decision Processes (MMDPs) is discussed. First, decentralized approaches are discussed, which are methods that decompose an MMDP into smaller identical sub problems (Section 1.3.1). Afterward, partial state representations are discussed, which note the ability of MMDPs to undergo state space pruning or compression (Section 1.3.2). Lastly for theory, literature behind CMDP models is discussed (Section 1.3.3). The subject of state uncertainty is discussed next (Section 1.3.4), followed by human advice, crowdsourcing and learning from demonstration (Section 1.3.5).

In this research Fully Observable MDP’s (FOMDPs) are generally referenced, with $< S, A, T, R >$ tuples where $S$ denotes a finite set of discrete states, $s$; $A$ denotes a set of finite actions, $a$; $T(s, a, s')$ represents a true transitional probability between states, and $R(s, a, s')$ denotes a reward function. When solving for an optimal policy, an infinite time horizon and a discounted reward setting is assumed. In some cases other MDP definitions will be noted.
specifically. Attention will be restricted to *multi-agent planning* problems where all agents collectively seek a uniform outcome [9].

In a direct sense, the simplest solution to such a planning problem is to include all relevant state information into a FOMDP, treating it as a Multi-Agent Markov Decision Process (MMDP); each agent can be given complete knowledge of the entire system as well as all other agents’ utility functions [9]. Locally optimal behavioural *policies* can be found using any reinforcement learning method. The MMDP modelling approach is attractive because issues related to competition, timing, and scheduling are not present. There is really one ideal utility function across all agents, and all agents regress to it together. These problems are P-complete [28] or PSPACE-complete [35], and can be approached in polynomial time.

Two major issues exist with this the aforementioned centralized approach. First, the state space size for real-world problems can be prohibitively large; hundreds of dimensions can easily exist, leading to problems of sparsity, interdependence, and computation. Secondly, a single robot is unlikely to know the positions, utility functions, and future decisions of an entire robot team with much certainty. To deal with these major multi-agent issues a few approaches have been taken:

### 1.3.1 Decentralized Representations

A paper in 2002 showed that game theoretic worst-case estimates can be applied to multi-agent teams, in a partially or completely observable sense by considering systems as *decentralized* [5]. Decentralized MDP modelling approaches formalize the multi-agent model by adding a set of agents, actions per agent and observations per agent to the MDP definition. In a finite-horizon sense, worst case computation to solve such problems has been readily shown to be super-exponential in computation time and NEXP-Hard [5].

Some attempts have been made to solve these intractable problems in an approximate sense. If it assumed that all the agents are completely independent, such that they do not affect each other or each other’s observations, then a factored representation of the DEC-MDP can be solved [4]. The *anytime* implementation of this model, an algorithm that is interruptible, is intractable for large problems, even if typical simulations land it typically within 90% of the value earned by an optimal policy. Recent approaches, while encouraging in the amount of
agents simulated, still only approach simple games, and lack the complexities of realistic simulations [34].

It is clear that the DEC-MDP model is a complete and accurate model of a multi-agent system. It is also one of the hardest models for this problem in terms of complexity, as it is NEXP-Hard. It lacks strong real-time algorithms for agents with limited memory and computational resources.

1.3.2 Partial State Representations

To directly convert an MMDP or DEC-MDP problem into a more tractable one, a designer can choose to represent a partial subset of the full state. For example, it may be determined that some state information is irrelevant for each individual agent, or that a certain subset of this state information is unstable or unavailable. In this case it may be desired to prune the state representation to a more reasonable size. Approaches range from learned pruning, such as Principal Component Analysis (PCA) [6, 43, 42] or clustering [22], to any arbitrary information excluded by the designer of a state space. This reduces the potential state space required for the agent to explore and therefore lessens the computational burden across all agents. It is rather direct to find research that is focused on state representation as an aid to machine learning, to speed convergence to an optimal policy. In the case of learned pruning, such as PCA or clustering, a reduction of state space may also result in noise reduction and performance improvement.

Another approach is to compress the state representation using various methods without loss of information. For example, a learned factor representation [9] can lead to less memory usage, even if theoretical worst-case performance is equal to the full state space; the agents learn which state variables are dependent through experience. Conversely to discovering and modelling many dependencies between states, many state variables can be assumed to be independent. This leads to exponentially less memory usage. In both of these schemes no information loss takes place in terms of raw state data, but the predictive power of the utility functions can decrease due to misrepresentation of variable independence.

Lastly, it may be desired to strictly limit the MDP to some local scope that is directly observable by an agent, or smaller, using an ad-hoc method. In these cases a designer can take an
MDP of extremely high dimension and convert it to an MDP of lower dimension; an intractable problem can be formulated as a weaker, inaccurate, and tractable model. A typical state vector in this case may include communications from other agents, sensor readings, and a variety of internal metrics. For robotics, this step is a general requirement if any aspects of realism are to be represented in simulation and experiment.

1.3.3 Concurrent MDP Representations

A last approach, and the primary contribution of this thesis, is a method of converting an intractable multi-agent MDP problem into separate dependent MDP’s. Groups of behaviours can be addressed separately, and the main benefit conferred is reduction in computation since two classes of behaviours are assumed as independent. In fact, the nature of policies derived in a Concurrent MDP setting do not need to be of the same class or rigor, allowing the designer more freedom when compared with previous approaches. This approach, while empirically common, is rarely formalized or analytically explored. I define this approach as the Concurrent MDP (CMDP) approach and analytically describe its application to simulation.

In general, if care is not taken in the design of DEC-MDPs they may exhibit degeneracies when solved by a policy regression technique. In practice, such degenerate DEC-MDPs are solved by both a policy regression approach, such as Q-Learning, and a coordination heuristic. One example is “Q-Learning with Search” where individual agents search for foraging targets, while their exploration regions are chosen by a coordination heuristic called “Search” [20]. Another example is the control of a robot soccer player, where each agent learns many ways to represent the state space in a simulated soccer game, and a coordinating heuristic selects the best action given multiple behavioural policies [1]. The Alliance framework [38] uses a sub mechanism called L-Alliance [39] to divide a foraging problem into an individual learning problem and separate task allocation problem in a completely heuristic fashion. Thus, it is possible to model a degenerate DEC-MDP, and to use a coordination heuristic to compensate for model degeneracy. In the literature these models enjoy empirical success, but generally do not follow through to consider how such coordination heuristics theoretically solve for the complete centralized MDP.
The Concurrent and Individual and Social Learning (CISL) approach addresses multi-agent learning by solving two independent classes of MDP problems concurrently [30, 17]. First, a limited scope Individual Performance MDP is solved. Secondly, a Task Allocation MDP is solved. Both of these MDPs are compact and of low quality relative to a corresponding decentralized MDP. For the CISL approach, the individual MDP is seen as completely independent of other non-cooperating agents—pairwise agent interactions are considered as unmodeled noise. The Task Allocation MDP is developed to characterize the process of assigning tasks to agents, such that the reward obtained by the team of agents is a random process, which reflects team performance toward a team objective.

In this research three CISL approaches is focused on, which are examples of possible CMDP regression mechanisms. The benefit of CMDP models is their solutions may be analytically sound, computationally tractable, and broadly applied in experimental scenarios with off-the-shelf technologies—a step toward further unification of practical decision theoretic approaches and experimental robotics.

1.3.4 State Observation Uncertainty

In general, most approaches to handling state observation uncertainty can fall into two categories. One approach to handling uncertainty is to directly model a system as a Partially Observable Markov Decision Process (POMDP) [23]. Although computationally costly, it is possible with the processing available within a single laptop to run a single well designed agent [40]. Albeit powerful, POMDP models are typically sequestered to games with limited state space and uncertainty, such as card games and small discrete space games. There is some recent work on more tractable solutions to the POMDP computational burden [34], but it still remains a challenging problem to apply such accurate models to realistic scenarios for multi-agent learning. Partially observable models require the maintenance of many, potentially infinite, separate beliefs about the current state and their likelihood as time progresses.

Another approach to compensate for partial observability is to continue with a fully observable assumption for the MDP, and to use a state estimator, also called a state filter. To illustrate this approach, a true state \( s \), a state observation \( o \) and a belief state \( b \) are defined, \((s, o, b \in S_t)\). Since a state observation is not fully observable \((o \neq s)\), this approach involves
regressing to a belief state using a state estimator ($b = S_e(o)$), with the underlying assumption (hope) that any belief state is the same as (close to) a true state ($b \approx s$). [27]. A popular approach is to use a Bayes filter, for example a particle filter [21], to naively infer the system state from several observations and prior evidence. Although computationally efficient, this approach has some theoretical difficulties. One common issue occurs when the beliefs from a state estimator exhibit high variance; if the state estimator does not capture or utilize a measure of uncertainty when regressing to a belief state, the agent cannot use this uncertainty in its learning. In practice, particle filtering has been demonstrated to work well and is applied to a wide variety of problems including simultaneous localization and mapping (SLAM) problems [41, 26].

A fully observable model with a naïve ground truth, while computationally efficient, is inaccurate and can develop divergent policies, whereas partially observable models encourage a computational bottleneck, and are more cumbersome during simulation and experimentation. In this thesis the addition of a particle filter to the CISL approach is discussed, allowing for realistic simulation of a moderately large team of robots.

1.3.5 Human Advice, Crowdsourcing and Learning from Demonstration

For the purpose of this work, crowdsourcing will be defined as the process of acquiring data from a crowd of web-based human contributors. This research is concerned with a subset of crowdsourcing research applied to machine learning and robotics, which is further concerned with acquiring labelled or unlabeled training data that can be generalized into agent behaviour examples, which can be dubbed human advice.

The training data can be acquired through any combination of simulation and remote operation. In terms of simulation, a video game, called Mars Escape!, has been designed to gather human social interaction examples [13]. In terms of remote operation, data has been collected though teleoperation of a maze navigating robot [16], and piloting of a helicopter [15]. In terms of parametric distributions, Gaussian Mixture Models (GMMs) have been widely used to represent prior understanding of decision-making state spaces [14]. Specifically, it is possible to capture complex dynamical control signals online using Gaussian Mixture Models [11, 12].
Few works in the literature address the augmentation of reinforcement learning systems with human advice, which is of significant interest for this thesis. One approach is to enhance the reward function using a parametric distribution, called Inverse Reinforcement learning [32]. A second approach is to consider a parametric distribution as a model for each agent’s state space model, such that the human advice would be how to perceive the environment [25]. Another method, which does not involve redefining any part of a Markov Decision Process (MDP), is to use the GMM as an ideal advisor in an advice-sharing mechanism; such a mechanism may operate similarly to another approach called Advice Exchange [33, 30, 17].

Since in this paper I am only interested in the possibility of augmentation, and not a redefinition of the problem model, a GMM is chosen as a method to obtain human advice. Behaviour examples are included using a method similar to Advice Exchange.
Chapter 2
Concurrent Individual and Social Learning Theory

This chapter begins by explaining the Concurrent Markov Decision Process problem model in Section 2.1, which serves as the basis for this research. The model is then broadened into a multi-agent context in Section 2.2. In Section 2.3 some Concurrent Individual and Social Learning regression approaches are presented, which solve a DEC-CMDP problem model, along with a Single Q-Learning approach, which solves a DEC-MDP problem model.

2.1 Single Agent Concurrent Markov Decision Process

A single agent’s learning MDP can be factored into an Individual Performance MDP and a Task Allocation MDP, which run concurrently as a Concurrent Markov Decision Process (CMDP). Figure 1 illustrates the proposed concurrent learning model. A Concurrent MDP is a set of two or more MDPs that have at least one dependent MDP relationship, meaning that one MDP somehow depends on the properties of another separate MDP. A relationship of this type is more general than a hierarchical relationship [3], and a hierarchical MDP can be considered as a special type of Concurrent MDP.

![Figure 1: Single-agent CMDP](image)

The CMDP problem model assumes the individual performance and Task Allocation MDPs run in parallel as a CMDP set with two specific dependencies. A task selection dependency models the reliance of an individual MDP on the Task Allocation MDP; for every state \( s_i \in S_i \) a subset \( e_{IT} \in s_i \) is assumed to exist whose value is determined solely by the Task Allocation MDP. Conversely, the team performance dependency defines the reward...
function \( R_T(s_T, a_T, s'_T) \) and transition function \( T_T(s_T, a_T, s'_T) \) for the Task Allocation MDP based on the performance of individual agents. Other than the two dependencies, these MDPs are independent, and can be treated as separate processes for the purposes of finding an optimal policy.

**Individual Performance MDP:**

Individual agent progress toward a sub-task can be defined as a \( < S_I, A_I, T_I, R_I > \) tuple,

- \( S_I \) denotes a discrete set of states, whose intrinsic value is partially specified by the task allocation process through a set of evidence \( E_{IT} \).
- \( A_I \) denotes a discrete set of actions.
- \( T_I(s_I, a_I, s'_I) \) denotes a stable transition model, i.e., the probability of executing action \( a_I \) starting from state \( s_I \) and ending up in state \( s'_I \).
- \( R_I(s_I, a_I, s'_I) \) denotes a positive real number as a reward received for transitioning from state \( s_I \) into state \( s'_I \) using action \( a_I \).

The evidence value \( e_{IT} \in E_{IT} \) is defined by the Task Allocation MDP, such that maximal visitation of all states at time infinity is assured. Such a single-agent MDP can be seen as a straightforward process for the reinforcement learning.

**Task Allocation MDP:**

Team progress toward a goal can be defined as a \( < S_T, A_T, T_T, R_T > \) tuple,

- \( S_T \) denotes a discrete set of states, which capture the individual performance characteristics between all agents and all tasks.
- \( A_T \) denotes a discrete set of actions, i.e., a function that assigns all available tasks to available agents.
- \( T_T(s_T, a_T, s'_T) \) denotes a stable transition model, i.e., the probability of executing action \( a_T \) starting from state \( s_T \) and ending up in state \( s'_T \). The transition function is completely specified by the individual performance process through a set of evidence \( E_{TI} \).
- \( R_T(s_T, a_T, s'_T) \) denotes a positive real number as a reward that is received after taking a certain action \( a_T \) within state \( s_T \) when arriving in state \( s'_T \). The reward function is completely specified by the individual performance process through a set of evidence \( E_{TI} \).

The value of the reward \( R_T \) and transition functions \( T_T \) are assumed to be unpredictable up to some known discrete iteration \( t_s \). After \( t_s \) the task allocation process’s reward and transition
function are assumed to have a constant expectation value, which may be affected by some unknown amount of zero mean noise.

It is worth emphasizing that within the Concurrent MDP model the two individual performance and task allocation processes are explored concurrently. This concurrent learning approach distinguishes the model from a single decentralized MDP that may consist of a set of states $S_U = S_I \times S_T$, a set of actions $A_U = A_I \cup A_T$, a reward function $R_U$, and a transitional model $T_U$. Indeed, the decentralized MDP is used in this thesis as a control to contrast with other learning models.

In all proposed MDPs it is assumed that the time horizon is infinite, the reward is discounted, and the time setting is discrete.

## 2.2 Decentralized Concurrent Markov Decision Processes

Further, the set of CMDPs is considered as a *transition independent Decentralized CMDP (DEC-CMDP)*, which is an extension of a DEC-MDP [4]. These models characterize a multi-agent problem. In this thesis, the singular CMDP for each agent is solely discussed, as DEC-CMDP models applying to multi-agent problems can be expressed as a collection of single-agent CMDPs under some strong assumptions, explored below.

![Figure 2: A DEC-CMDP](image-url)
The above collection of independent CMDPs (Figure 2) characterizes a corresponding DEC-CMDP without degeneracy under some assumptions: Each dependent MDP is either completely independent of other agents, as in the case of the Individual Performance MDP, or each dependent MDP characterizes a problem for the whole team, as does the Task Allocation MDP.

For the independence of the Individual Performance MDP, it is asserted that if each agent can understand and solve its local Individual Performance MDP, then the individual performance components of the DEC-CMDP will be solved. As formulated, the reward function for the individual partition of the DEC-CMDP is the sum of the Individual Performance MDP reward functions for each agent in the process. Additionally each agent is assumed to be independent of all other agents in terms of state transitions and reward. Each agent is assumed to act and observe at the same instants. Under these conditions each agent can maximize reward in an independent context, and neglect to model multiple agents within one Individual Performance MDP.

For the centralization of the Task Allocation MDP, it is asserted that one instance of the Task Allocation MDP completely characterizes the task allocation partition of the DEC-CMDP. Each agent may maintain an identical instance of the Task Allocation MDP, for the definition of its CMDP. The strong assumption in is that the agents can communicate perfectly, and regress to a joint action behavioural policy for the Task Allocation MDP in an identical manner.

Thus, the DEC-CMDP mentioned above can be fully characterized as an independent collection of CMDPs. It is noted that each CMDP when solved, solves the larger DEC-CMDP if the above assumptions are satisfied. If any of the above assumptions cannot be applied to a certain learning problem, then more complicated instances of the DEC-CMDP approach or DEC-POMDP approaches can be considered, along with the resulting implications for regression approaches.

2.3 Concurrent Individual and Social Learning

For each agent, the CISL mechanism solves for two independent Markov Decision Processes; an Individual Performance MDP models agent behaviour toward an assigned task, while Task Allocation MDP models agent task selection. The team learning CMDP model can be
solved by a three layered approach, where each layer represents three paradigms of multi-agent learning [38]: Collective, Cooperative, and Collaborative.

Collective learning in a multi-agent environment denotes a set of independent learning mechanisms developing their own independent behaviour policies. The collective layer learning problem is indeed the individual performance partition of the CMDP, which can be solved by a reinforcement learning algorithm, such as Q-Learning [44].

Cooperative behaviours are described as behaviours that solve for the task allocation partition of the CMDP, i.e., which agent should work toward which task. To formulate this layer three algorithms are contrasted in this thesis. First, Q-Learning is applied to the task allocation concurrent to the collective layer. Secondly, a heuristic approach is used, called L-ALLIANCE [38], and lastly, a second version of the L-Alliance algorithm is applied, which is a combination of stochastic gradient descent with the L-Alliance algorithm.

Collaborative behaviours enable agents to directly influence each other’s policies. Both a heuristic mechanism called Advice Exchange [33], and a mechanism called the Human Advice Layer are used. These systems allow an agent to accept utility values of actions, as advice, from other agents or from human advisors. The agents will then possibly benefit from the enhanced behaviours for a certain number of iterations, and will include these lessons in their own policy regression.

This section discusses how the collective and cooperative mechanisms regress to a policy for their corresponding Markov Decision Process. A number of agents $n$ are assumed in the multi-agent team, where each agent $i$ derives its own policy, $\pi^i$, and utility function, $Q^i$, for each MDP. Each agent has two policies and two utility functions: $\pi_i^i$, $\pi_T^i$, $Q_i^i$, $Q_T^i$, where the subscript identifies individual or task allocation policy or utility function. In the absence of any subscript $I$ or $T$ in the notation, $I$ is assumed; many sections focus only on the Individual Performance MDP.
2.3.1 Collective Learning: Individual Performance Policy Regression

The individual policy $\pi^i(t)$ regression is performed by a Q-Learning algorithm [44]. The algorithm is a form of policy improvement in Decision Theory, which takes incremental steps to improve an agent’s behavioural policy [8]. After random initialization of all the utility values, $Q_{i,0}(s_t, a_t) \in S_t \times A_t$, each agent will execute reinforcement learning using a basic Q-Learning update rule:

$$Q_{i,(t+1)}(s_t, a_t) = Q_{i,t}(s_t, a_t) + \alpha_{i,t} \left( R_{i,t}(s_t, a_t, s_{i,(t+1)}) + \gamma_{i,t} Q^*_{i,(t+1)}(s_t, a_t) - Q_{i,t}(s_t, a_t) \right)$$

(1)

where $Q^*_{i,(t+1)}(s_t, a_t)$ is the greatest positive utility value that can be expected in state $s_{i,(t+1)}$ at time $(t + 1)$ if action $a$ is selected at time $t$. The expected utility values are discounted by a constant decay factor $\gamma \in [0,1]$ to account for uncertainty in the expectation. The parameter $\alpha \in (0,1]$ is the learning rate, which approaches zero as $t$ approaches $\infty$ satisfying two convergence criteria, $\sum_{t=0}^{\infty} \alpha_{i,t}(s_t, a_t, s_{i,(t+1)}) = \infty$ and $\sum_{t=0}^{\infty} \left( \alpha_{i,t}(s_t, a_t, s_{i,(t+1)}) \right)^2 \in \mathbb{R}$.

The function $R_{i,t}(s_t, a_t, s_{i,(t+1)})$ is a real positive reward function. Given these definitions it is expected that Q-Learning will settle on a locally optimal policy for an infinite time-horizon problem [44].

To determine a suitable action $a_{i,t}$ in state $s_{i,t}$, a probabilistic distribution is sampled, which is based on an agent’s known utility values for a state $s_{i,t}$:

$$\pi^i_{i,t}(a_{i,t}|s_{i,t}) = \frac{e^{Q^i_{i,t}(s_{i,t}, a_{i,t})/\vartheta}}{\sum_{a_{i,t}} e^{Q^i_{i,t}(s_{i,t}, a_{i,t})/\vartheta}}$$

(2)

where the action probability distribution $\pi^i_{i,t}(a_{i,t}|s_{i,t})$ is the probability that agent $i$ will select action $a_{i,t} \in A_{i,t}$ in the given state $s_{i,t}$ at time $t$. The temperature parameter $\vartheta$ modifies the smoothness of the action probability distribution through scaling of the utility values. The
optimal policy for maximization of agent reward is can be defined as
\[ \pi_{i,t}^*(s_{i,t}) = \arg\max_{a_{i,t}} \pi_{i,t}^i(a_{i,t}|s_{i,t}). \]

2.3.2 Cooperative Learning: Task Allocation Policy Regression

This section begins by outlining three methods of finding and executing an optimal task allocation behavioural policy, namely Concurrent Q-Learning, L-Alliance, and RL-Alliance. Afterward, this section covers a tractable modification to both L-Alliance and RL-Alliance to accommodate physical cooperation, enabling two agents to cooperate toward a single task.

2.3.2.1 Q-Learning for Task Allocation

A direct approach to the task allocation problem can be the use of a separate Q-Learning mechanism concurrently with the individual performance Q-Learning mechanism (equation 1):

\[ Q_{t+1}(s_{t+1}, a_{t+1}) = Q_t(s_t, a_t) + \alpha_t \left( R_t(s_t, a_t, s_{t+1}) + \gamma_t Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t) \right) \]

To operate this mechanism no online communication between agents is intrinsically required.

2.3.2.2 L-Alliance

To assign all the available tasks to the agents, a task selection mechanism called L-Alliance [38] can be used. The L-Alliance mechanism regresses to a task allocation policy \( \pi_{i,t}^l \) that chooses a task selection action \( a_{i,t} \in A_T \) for every agent \( i \). At each iteration, every task \( j \) is assigned to the available agent \( i \) that has the highest motivation \( m_t(i,j) \), as described by each agent’s state \( (s_{i,t} \in S_T) \). This motivation is initialized as zero and can be updated in every iteration \( t \) of the Individual Performance MDP:

\[ m_t(i,j) = (m_{t-1}(i,j) + p_t(i,j))g_t(i,j) \]

The motivation \( m_t(i,j) \) is derived using a pervious motivation value \( m_{t-1}(i,j) \), a gating multiplier \( g_t(i,j) \) and an impatience factor \( p_t(i,j) \). The binary multiplier \( g_t(i,j) \) ensures that certain conditions are all satisfied at time \( t \) to keep the robot motivated; otherwise \( m_t(i,j) \) is reset to zero. These conditions include a) whether task \( j \) is still incomplete, b) no other task \( j \) is assigned to agent \( i \), c) no more than two agents are assigned to the task \( j \) at time \( t \), and d) no other agents are present at time \( t \) that can accomplish task \( j \) better than robot \( i \). If any of these
conditions is not satisfied the multiplier $g_t(i,j)$ is set to zero; otherwise it is set to one. To check these conditions robots need to communicate online.

The impatience factor $p_t(i,j)$ is a real-positive function that is broken into three formulations depending on the average trial time $\tau_t(i,j)$, which denotes the moving average of the trial time of agent $i$ when assigned to task $j$:

$$
\begin{align*}
&\frac{\theta}{\tau_t(i,j)} : \text{Initial } \tau_t(i,j), \\
&\frac{\theta}{\tau_{\text{max}} - \left(\tau_t(i,j) - \frac{\min_{k,l} \tau_t(k,l)}{\tau_t(i,j)}\right) \times K_{ij}} : \text{(Superior) lowest } \tau_t(i,j) \forall i \\
&\frac{\theta}{\tau_{\text{min}} + \left(\tau_t(i,j) - \frac{\min_{k,l} \tau_t(k,l)}{\tau_t(i,j)}\right) \times K_{ij}} : \text{(Mediocre) not lowest } \tau_t(i,j) \forall i
\end{align*}
$$

The parameter $\theta$ denotes a positive non-zero constant, and the scaling factor $K_{ij}$ is used to reduce the idle time of the agents, and is defined as:

$$
K_{ij} = \frac{\tau_{\text{max}} - \tau_{\text{min}}}{\max_{k,l} \tau_t(k,l) - \min_{k,l} \tau_t(k,l)}
$$

The parameters $\tau_{\text{min}}$ and $\tau_{\text{max}}$ are defined as the minimum and maximum permitted delay. Further, $\max_{k,l} \tau_t(k,l)$ and $\min_{k,l} \tau_t(k,l)$ are also defined, respectively, as the maximum and minimum trial time over all robots $k$ and tasks $l$. The smaller the average trial time, $\tau_t(i,j)$, the larger agent $i$’s impatience rate becomes in the future. The Superior impatience rate is used when agent $i$’s average trial time $\tau_t(i,j)$ is lower than other agents in the team; otherwise a Mediocre impatience rate is used.

Thus, the L-Alliance mechanism is a greedy algorithm that assigns the most motivated agent $i$ to a corresponding task $j$. The L-Alliance policy $\pi_T^*$ uses an agent’s latent transitional model, $T_T$, to maximize for the reward function, $R_T$. The maximization of reward can be characterized by minimizing the agents’ average trial values:
where $s_{a,t}(i,j)$ represents the assignment status of agent $i$ to task $j$, and is valued as either one (assigned) or zero (unassigned).

It may be desired to enable the L-Alliance mechanism to assign two agents to one task to enable physical cooperation, by considering every coalition average trial time, $\tau_t(i, i', j)$, where $i$ and $i'$ represent two agents cooperating toward a task $j$ [37]. The motivation equation and impatience can then still be used in the case of physical cooperation, but resulting in coalition motivation $m(i, i', j)$ and coalition impatience $p_t(i, i', j)$.

It may be intractable to consider every task coalition pair. To enable tractable physical cooperation, it is assumed that a simplified agent effort metric can be utilized in place of all coalition-task trial averages $\tau_t(i, i', j)$:

$$\bar{\tau}_t(i, i', j) = \begin{cases} 
\tau_t(i, j) & \sum_{k \neq i} s_{a,k}(k, j) = 0 \\
\tau_t(i, j) + \tau_t(i', j) & \text{otherwise} \end{cases}$$

Additionally, to discourage unneeded cooperation, a limiting condition is added to the gating condition $g_t(i, j)$: e) The gating function $g_t(i, j)$ is set to zero if another robot $i'$ is currently engaged with task $j$, and the agent $i'$ is expected to succeed. Lastly, when a task is completed by two cooperating robots, no $\tau_t(i, j)$ or $\tau_t(i', j)$ values are updated, as trial times denote individual effort. Thus, physical cooperation in task allocation is attempted in a tractable manner by executing a minimization of agent trial effort toward tasks.

2.3.3.3 Pareto-optimality for L-Alliance (RL-Alliance)

The L-Alliance mechanism does not always perform optimal task selection. Two examples are task starvation and mediocre assignment. Task starvation occurs if an agent $i$ achieves an anomalously high average trial time $\tau_t(i, j)$ toward task $j$; the agent $i$ may become prevented from the allocation to task $j$ in the future, despite having a historically low average trial time; task starvation can prevent a potentially superior agent $i$ from gaining assignment to
task \( j \). Secondly, a mediocre assignment can occur if the L-Alliance mechanism preferentially assigns mediocre agents over superior agents. The mediocre assignment event can be expressed analytically by formulating when a mediocre impatience rate will be greater than a superior impatience rate, given a superior agent \( s \), a mediocre agent \( i \), and a task \( j \):

\[
\frac{\theta}{\tau_{\text{max}} - \left( \tau_t(s,j) - \min_{k,l} \tau_t(k,l) \right) \times K_{sj}} < \frac{\theta}{\tau_{\text{min}} + \left( \tau_t(l,j) - \min_{k,l} \tau_t(k,l) \right) \times K_{ij}}
\]

(9)

Given equation 9 the mediocre assignment will occur when equation 10 is satisfied:

\[
\tau_t(s,j) + \tau_t(i,j) - 2 \min_{k,l} \tau_t(k,l) < \tau_{\text{max}} - \tau_{\text{min}}.
\]

(10)

The L-Alliance will perform mediocre assignment when \( \tau_{\text{max}} \) is set too high, \( \tau_{\text{min}} \) is set too low or the minimum average trial time across the team \( \min_{k,l} \tau_t(k,l) \) is too large.

A pareto-optimal performance can be achieved by the L-Alliance mechanism if, first, the slow impatience rate \( p_t^{\text{slow}}(i,j) = \theta_1/\tau_t(i,j) \) is used throughout task allocation policy regression, and secondly, a recursive stochastic update formulation is used for the average trial time instead of moving average, as follows:

\[
\tau_t(i,j) = \beta(i,j) \left( \tau_{t-1}(i,j) + \frac{\theta_2}{f_{ij}} (I_{ij} - \tau_{t-1}(i,j)) \right)
\]

(11)

The frequency \( f_{ij} \) denotes the number of attempts agent \( i \) has had toward task \( j \). The parameter \( \theta_2 \) denotes a constant scale factor, and can be called the learning rate. The variable \( I_{ij} \) denotes the time taken during the latest attempt by agent \( i \) toward task \( j \). The function \( \beta(i,j) = e^{(f_{ij}/\theta_4)}/(\theta_3 + e^{(f_{ij}/\theta_4)}) \) defines a softmax expression, which allows each agent to have many attempts and adequate training on all tasks. The parameters \( \theta_3 \) and \( \theta_4 \) control the slope and location of the softmax distribution. Thus, \( \tau_t(i,j) \) would represent the expected trial time required by agent \( i \) when assigned to task \( j \), assuming that the attempt time \( I_{ij} \) is stationary by a certain time \( t_s \).
Three factors indicate that the changed L-Alliance mechanism, called RL-Alliance, finds a Pareto-optimal task allocation and an optimal solution for maximizing of the reward, for an initial allocation of set of tasks to a set of agents:

**Proposition 2.3.1, Task Trial Times will Converge Accurately.**

The parameter $\tau_t(i,j)$ will accurately converge to the expected average trial time of an agent $i$ toward task $j$ [10], due to the usage of stochastic gradient descent, if the trial times are stably stochastic.

**Proposition 2.3.2, RL-Alliance performs Pareto-optimal resource allocation**

By using a gradient descent update for parameter $\tau_t(i,j)$ and a motivation update that only utilizes the slow impatience rate in equation 5, the RL-Alliance mechanism selects tasks in a pareto-optimal manner, because an agent is assigned to the task it is most motivated toward if no other robot is more motivated:

$$m_t(i,j) = \arg\max_{(i',j')} m_t(i',j') \Leftrightarrow a_{T,t}(i,j)$$  \hspace{1cm} (12)

After all items are selected, a pareto-optimal allocation is found, characterized by the inability to reassign tasks without performing a sub-optimal task selection for at least one other agent [24].

**Proposition 2.3.3, RL-Alliance Maximizes Reward**

The RL-Alliance mechanism is a greedy optimization method, which results in maximizing the reward function $R_T$, since the maximization of the motivation towards a task implies a coincident maximization of the reward function:

$$R_T(s_{T,(t-1)}, a_{T,(t-1)}, s_{T,t}) \propto \sum_{i,j} s_{a,t}(i,j) \propto \sum_{i,j} m_t(i,j)s_{a,t}(i,j)$$  \hspace{1cm} (13)

Hence, the RL-Alliance mechanism finds an optimal solution for the reward function, and also finds a pareto-optimal task allocation across agents, for at least the initial assignment of agents to tasks.

The L-Alliance mechanism without alteration invalidates Proposition 2.3.1., through usage of a moving average, Proposition 2.3.2., through the occurrence of mediocre assignments,
and Proposition 2.3.3., through the occurrence of mediocre assignments. Thus, RL-Alliance is a theoretical improvement, and simplification of the L-Alliance mechanism.
In this chapter the subject of state uncertainty is discussed. First, the Individual Performance MDP section is extended to show the effects of noise in section 3.1; this section explains the expected transition function and an expected reward function, called the expected functions, to characterize the theoretical effects on state measurement uncertainty. Section 3.2 derives two theoretical metrics, called expected false observation and expected false reward, to characterize the accuracy of the expected functions, the expected transition function and an expected reward function respectively. Lastly, in section 3.3 a theoretical derivation of a particle filter is considered, which can be used as a basis for a state estimator.

3.1 Individual Performance MDP and Uncertainty

The following arguments show why state observation uncertainty can interfere with a reinforcement learning agent. First, an expected reward function and an expected transition function are defined, called the expected functions, \( \left( \overline{R}_t(s_t, a_t), \overline{T}_t(s_t, a_t, s_{(t+1)}), \right) \). It will be shown that the utility function \( Q_t \) is characterized by the expected functions as time approaches infinity (Proposition 3.1.1). Secondly, it will be demonstrated that the expected functions can characterize their corresponding true functions \( \left( E[R_t(s_t, a_t)], T_t(s_t, a_t, s_{(t+1)}) \right) \), in a noiseless environment at time infinity (Proposition 3.1.2). Thirdly it is shown that the utility function is characterized by the true functions in a noiseless environment (Proposition 3.1.3). Lastly, uncertain state observations can be shown to interfere with the expected functions, possibly invalidating the utility function (Proposition 3.1.4).

**Proposition 3.1.1 Utility is Characterized by the Expected Functions**

The expected transition function can be found by considering visitation frequencies, where a visitation frequency \( f(s_t, a_t, s_{(t+1)}) \) denotes the amount of times the execution of action \( a_t \) in state \( s_t \) has led to arrival in a state \( s_{(t+1)} \).
\[ \hat{T}_t(s_t, a_t, s_{(t+1)}) = \frac{f(s_t, a_t, s_{(t+1)})}{\sum_{s'_{(t+1)} \in \mathcal{S}_t} f(s_t, a_t, s'_{(t+1)})} \] (14)

The expected reward function can then be defined:

\[ \hat{R}_{(t+1)}(s_t, a_t) = \sum_{s_{(t+1)} \in \mathcal{S}_t} \hat{T}_t(s_t, a_t, s_{(t+1)}) \left( R(s_t, a_t, s_{(t+1)}) + \gamma \max_{a \in \mathcal{A}_t} \hat{R}_t(s_t, a) \right) \] (15)

The Q-Learning algorithm (equation 1) approximates this expected future reward function (equation 15) [44], and if the limit of these equations 1 and 15 is taken as they approach infinity, the Q-Learning algorithm can find an exact value for the expected reward function, \((\lim_{t \to \infty} Q_t(s_t, a_t) = \lim_{t \to \infty} \hat{R}_{(t+1)}(s_t, a_t)):

\[ \lim_{t \to \infty} \hat{R}_{(t+1)}(s_t, a_t) = \sum_{s_{(t+1)} \in \mathcal{S}_t} \lim_{t \to \infty} \left( \hat{T}_t(s_t, a_t, s_{(t+1)}) \right) \left( R(s_t, a_t, s_{(t+1)}) + \lim_{t \to \infty} \left( \gamma \max_{a \in \mathcal{A}_t} \hat{R}_t(s_t, a) \right) \right) \] (16)

The utility function found by the Q-Learning algorithm, at time infinity, characterizes the expected future discount reward. The utility values \(Q_t\) and behavioural policy \(\pi_t\) are therefore both characterized by the expected functions at time infinity.

**Proposition 3.1.2 Expected Functions are Characterized by True Functions**

In a noiseless environment, the expected functions can characterize their true counterparts. First, the expected transition function \(\hat{T}_t\) at time infinity can characterize the true transitional probability \(T\):

\[ T(s_t, a_t, s_{(t+1)}) = \lim_{t \to \infty} \hat{T}_t(s_t, a_t, s_{(t+1)}) = \lim_{t \to \infty} \frac{f(s_t, a_t, s_{(t+1)})}{\sum_{s'_{(t+1)} \in \mathcal{S}_t} f(s_t, a_t, s'_{(t+1)})} \] (17)

Second, through substitution of equation 17 into equation 15, a common definition for reward can be obtained:

\[ E[R_t(s_t, a_t)] = \sum_{s_{(t+1)} \in \mathcal{S}_t} T_t(s_t, a_t, s_{(t+1)}) \left( R(s_t, a_t, s_{(t+1)}) + \gamma \max_{a \in \mathcal{A}_t} E \left[ R_{(t+1), a} \right] \right) \] (18)
Proposition 3.1.3 Utility is Characterized by the True Functions

It follows from Proposition 3.1.1 and Proposition 3.1.2 that in a noiseless environment, the Q-Learning algorithm’s utility values characterize the true functions:

$$\lim_{t \to \infty} Q_t(s_t, a_t) = \lim_{t \to \infty} \mathcal{R}_t(s_t, a_t) \iff \lim_{t \to \infty} \mathcal{R}_t(s_t, a_t) = E[R_t(s_t, a_t)]$$

(19)

Proposition 3.1.4 Uncertainty in State Observation invalidates the Utility

The utility function $Q_t$ may be inaccurate if the measured expected functions are inaccurate ($\lim_{t \to \infty} \mathcal{R}_t(s_t, a_t) \neq E[R_t(s_t, a_t)]$), which invalidates Proposition 3.1.3. Specifically, uncertainty in state observation can affect utility value regression by affecting an agent’s perception of its environment’s visitation frequencies, $f(s_t, a_t, s_{t+1}) \neq f(o_t, a_t, o_{t+1})$, invalidating equation 6 in proposition 3.1.2. The estimated functions in this circumstance may not converge on the true functions, and the utility function will still converge on the (now divergent) estimated functions (Proposition 3.1.1).

Two questions can be asked. How can the utility function be quantified? And, is there a way to compensate for observation uncertainty? In Section 3.2 issue of characterizing state observation uncertainty; metrics called expected false observation and expected false reward are used to characterize the accuracy of the expected functions, (the expected transition function and expected reward function). In Section 3.3 the use of a state estimator based on a Bayesian particle filter is discussed, to compensate for the encountered state uncertainty.

3.2 Measuring State Observation Uncertainty

The effects of state observation uncertainty can be characterized by two metrics, labelled as expected false observation and expected false reward. The expected false observation metric quantifies the inaccuracy of an agent’s expected transition function, through measuring the chance of an agent’s incorrectly identifying its own state. The expected false reward metric quantifies the inaccuracy of the regression approach to computing the agent’s expected reward function, through measuring how much an agent’s perceived reward differs from the true reward. To define each metric, a situation where zero-mean Gaussian noise is added over all state
readings $s_t \in S_t$ is considered. A general probability density function $\eta(o_t|s_t)$ is used over a true state $s_t$, where a sample from the random observation population is defined as an observation $o_t$.

### 3.2.1 Expected False Observation:

A *false observation probability* ($p(e_t|s_t)$) can be defined to characterize the ability of an agent to both understand and execute a policy within an uncertain environment. Given a known state $s_t$ and a random observation $o_t$, a probability can be derived by defining a distance function $d_\delta$, based on the Kronecker delta function $\delta$, which reflects the occurrence of an incorrect observation given a true state:

$$d_\delta(o_t,s_t) = 1 - \delta(o_t,s_t)$$  \hspace{1cm} (20)

Then, the probability of an incorrect observation given a state value can be defined as:

$$p(e_t|s_t) = \sum_{o_t \in S_t} \eta(o_t|s_t)d_\delta(o_t,s_t)\Delta o_t$$  \hspace{1cm} (21)

The value $\Delta o_t = 1$ is assumed. The expected value of the false observation probability, called the *expected false observation*, can be derived using the Euclidian distance between the observation and the true state, $d_E(o_t,s_t)$:

$$D(s_t) = \sum_{o_t \in S_t} d_E(o_t,s_t)\eta(o_t|s_t)\Delta o_t$$  \hspace{1cm} (22)

Both the false observation probability and its expected value may be sufficient to characterise the ability of the Q-Learning algorithm to act under state observation uncertainty.

### 3.2.2 Expected False Reward:

The uncertainty in state observation may increase the probability that a non-optimal policy will be learned. This is due to the fact that in many MDPs the expected reward $R_t(s_t,a_t)$ relies on the transitional model. To define the expected false reward metric, the *observation probability* $p(o_{(t+1)}, o_t, s_{(t+1)}|s_t, a_t)$ can be formulated; the probability an agent observes itself in state $o_{(t+1)}$ while it is really in state $s_{(t+1)}$, after having acted on observation $o_t$, (given a ground truth $s_t, a_t$). To formulate this probability, the chain rule can be readily applied:
\( p(o_{(t+1)}, o_t, s_{(t+1)}|s_t, a_t) = p(o_{(t+1)}|o_t, s_{(t+1)}, s_t, a_t)p(o_t|s_{(t+1)}, s_t, a_t)p(s_{(t+1)}|s_t, a_t) \) (23)

Assuming independence of some above conditions, and knowing \( \Delta o_t = 1 \), I can express the observation probability using known distributions:

\[ p(o_{(t+1)}, o_t, s_{(t+1)}|s_t, a_t) = \eta(o_{(t+1)}|s_{(t+1)})\eta(o_t|s_t)T(s_t, a_t, s_{(t+1)}) \] (24)

Afterwards the expected perceived reward \( \bar{R}_t \) and the expected false reward \( \Delta \bar{R}_t(s_t, a_t) \) can be formulated, given any specific state action pair:

\( \bar{R}_t(s_t, a_t, s_{(t+1)}) = \sum_{o_{(t+1)} \in S_t} \sum_{o_t \in S_t} p(o_{(t+1)}, o_t, s_{(t+1)}|s_t, a_t) R(o_t, a_t, o_{(t+1)}) \) (25)

\( \Delta \bar{R}_t(s_t, a_t) = \sum_{s_{(t+1)} \in S_t} \left( \bar{R}_t(s_t, a_t, s_{(t+1)}) - T(s_t, a_t, s_{(t+1)}) R(s_t, a_t, s_{(t+1)}) \right) \)

After simplification and substitution, a final formulation is obtained for the value of the expected false reward \( \Delta \bar{R}_t(s_t, a_t) \):

\( \Delta \bar{R}_t(s_t, a_t) = \sum_{s_{(t+1)} \in S_t} T(s_t, a_t, s_{(t+1)}) \left( \sum_{o_{(t+1)} \in S_t} \eta(o_{(t+1)}|s_{(t+1)}) \sum_{a_t \in S_t} \eta(o_t|s_t) R(o_t, a_t, o_{(t+1)}) \right) - R(s_t, a_t, s_{(t+1)}) \) (26)

Thus, the expected false observation and expected false reward metrics can represent the extent of uncertainty in state observation. These metrics can also be used to investigate the efficacy of uncertainty reduction approaches that are designed for Fully Observable Markov Decision Processes.

### 3.3 Robust Concurrent Individual and Social Learning

One approach to maintaining individual policy performance despite the occurrence of uncertain observations is to utilize a belief value using a state estimator, which can be denoted by \( S_e(*) \). The state estimator relies on a collection of Bayesian particle filters. Thus, in this section a single particle filter will be derived first as an extension of the works on particle filters [2, 41], and then the derived particle filter is applied to a multivariate state.
The particle filter model estimates the probability, \( p(x_k | u_{1:k}, z_{1:k}) \), of observing a real-valued variable \( x_k \) at step \( k \). The model will be formulated using the same distribution at a previous step \((k - 1)\), a series of signals \((u_{1:k})\), and a series of observations \((z_{1:k})\). To begin, I consider the sensor reading \( z_k = h_k(x_k, n_k) \) representing an observation of \( \{x_k \in \mathbb{R}, k \in \mathbb{N}\} \) subject to noise \( \{n_k \in \mathbb{R}, k \in \mathbb{N}\} \). Next, a random population, \( \{x^i_{1:k}, w^i_k\}^{N_k}_{i=1} \), can be defined which characterizes \( p(x_k | z_{1:k}) \) with increasing accuracy as \( N_k \) approaches infinity. The probability of a value \( x_k \) can be readily approximated given a collection of observations by calculating a weighted sum over the population:

\[
p(x_k | z_{1:k}) \approx \sum_{i=1}^{N_k} w^i_k \delta(x_k - x^i_k) \tag{27}
\]

where \( \delta(\cdot) \) signifies an instance of a Dirac delta function. The weight value \( w^i_k \) at each step \( k \) is assumed to be proportional to the current observation probability, \( p(z_k | x^i_k) \), multiplied by \( w^i_{k-1} \):

\[
w^i_k \propto n_o w^i_{k-1} p(z_k | x^i_k) \tag{28}
\]

where \( n_o \) defines a normalization factor ensuring \( \sum_{i=1}^{N_k} w^i_k = 1 \). Given equation 28, the probability of \( x^i_k \) can be expressed using \( p(z_k | x^i_k) \):

\[
p(x^i_k | z_{1:k}) \approx n_o p(z_k | x^i_k) p(x^i_k | z_{1:k-1}) \tag{29}
\]

Using the Chapman–Kolmogorov equation, it is possible to consider the distribution in terms of the observation at the previous step \((k - 1)\):

\[
p(x^i_k | z_{1:k}) \approx n_o p(z_k | x^i_k) \int p(x^i_k | x'_{k-1}) p(x'_{k-1} | z_{1:k-1}) dx'_{k-1} \tag{30}
\]

Next, I represented \( p(x_k | u_{1:k}, z_{1:k}) \) by including the control signal series \( u_{1:k} \):

\[
p(x^i_k | u_{1:k}, z_{1:k}) \approx n_o p(z_k | x^i_k) \int p(x^i_k | x'_{k-1}, u_k) p(x'_{k-1} | u_{1:k-1}, z_{1:k-1}) dx'_{k-1} \tag{31}
\]

Thus, the classic particle filter is formulated in equation 31. The formulation \( p(z_k | x^i_k) \) represents the latest observation, and the formulation \( p(x^i_k | x'_{k-1}, u_k) \) represents the movement model. The integration in equation 31 can be readily estimated through the following summation:
Lastly, the probability \( p(x_k^i | u_{1:k}, z_{1:k}) \) can be approximated using linear algebra, beginning with matrix definitions, where \( Z_k, U_k, W_{k-1}, \) and \( W_k \) are matrix representations of their respective probabilities in equations 33-38:

\[
\begin{align*}
W_{k-1} &= \begin{bmatrix}
    p(x_{k-1}^1 | u_{1:k-1}, z_{1:k-1}) \\
    \vdots \\
    p(x_{k-1}^{N_k} | u_{1:k-1}, z_{1:k-1})
\end{bmatrix} \\
U_k &= \begin{bmatrix}
    p(x_k^1 | x_{k-1}^1, u_k) & \cdots & p(x_k^{N_k} | x_{k-1}^{N_k}, u_k) \\
    \vdots & \ddots & \vdots \\
    p(x_k^{N_k} | x_{k-1}^1, u_k) & \cdots & p(x_k^{N_k} | x_{k-1}^{N_k}, u_k)
\end{bmatrix} \\
U_k W_{k-1} &= \begin{bmatrix}
    p(x_k^1 | x_{k-1}^1, u_k) & \cdots & p(x_k^{N_k} | x_{k-1}^{N_k}, u_k) \\
    \vdots & \ddots & \vdots \\
    p(x_k^{N_k} | x_{k-1}^1, u_k) & \cdots & p(x_k^{N_k} | x_{k-1}^{N_k}, u_k)
\end{bmatrix} \\
U_k W_{k-1} &= \begin{bmatrix}
    p(x_k^1 | u_{1:k}, z_{1:k-1}) \\
    \vdots \\
    p(x_k^{N_k} | u_{1:k}, z_{1:k-1})
\end{bmatrix} \\
Z_k^T I &= \begin{bmatrix}
    p(z_k | x_k^1) & 0 & 0 \\
    0 & \ddots & 0 \\
    0 & 0 & p(z_k | x_k^{N_k})
\end{bmatrix} \\
W_k &= Z_k^T I U_k W_{k-1} = \begin{bmatrix}
    p(x_k^1 | u_{1:k}, z_{1:k}) \\
    \vdots \\
    p(x_k^{N_k} | u_{1:k}, z_{1:k})
\end{bmatrix}
\end{align*}
\]

Thus, the final filter can be expressed succinctly:

\[
p(x_k | u_{1:k}, z_{1:k}) \approx n_o \delta(x_k C - X_k)^T Z_k^T I U_k W_{k-1}
\]
where \( Z_k, U_k \) and \( W_k \) are characterized by \( p(z_k|x_k^i), p(x_k^i|x_{k-1}^i, u_k) \) and \( p(x_k^i|u_{1:k}, z_{1:k}) \), respectively. The matrix \( C \) represents a column matrix of ones, while the function \( \delta \) represents a value based on the inverse of the Euclidean distance bounded by \([0,1]\), approximating a Dirac delta function \( \delta \). The matrix \( I \) denotes the identity matrix and \( X_k \) describes a column matrix containing the current value of the \( x_k \) series. Given a current matrix \( W_k \), an estimation can also be defined as:

\[
p(x_k | u_{1:k}, z_{1:k}) \approx n_0 \delta(x_k C - X_k)^T W_k
\]  

Therefore, the state estimator can be defined as:

\[
\arg\max_{x_k} p(x_k | u_{1:k}, z_{1:k}) \approx n_0 X_k^T Z_k^T I U_k W_{k-1} = n_0 X_k^T W_k
\]  

Using the defined theoretical basis, a general particle filtering algorithm can be described.

First, the population \( \{ x_k^i, w_k^i \}_{i=1}^{N_s} \) is initialized by choosing an initial probability \( W_k \):

---

**Step 1: Create Single Dimension Particle Filter**

Function \([X_1, W_1] = \text{SIS}_\text{Define} [z_1, u_1] : \)

Draw \([X_1, W_1]\) from \(q(x_1 | u_1, z_1)\), where \(q\) represents a hypothesized initial density function.

Normalize \(W_1\)

End

---

Afterward, the particle filter is updated for the next step:

**Step 2: Update Single Dimension Particle Filter**

Function \([X_k, W_k] = \text{SIS}_\text{Update} [X_{k-1}, W_{k-1}, z_k, u_k] : \)

Create \(X_{k-1}^{\text{expanded}}\) by sampling \([p(x_k^i | x_{k-1}^i, u_k), u_k, W_{k-1}, X_{k-1}]\)
Define $[U_k]$ using $[u_k, X_{k-1}^{\text{expanded}}, p(x_k^i|x_{k-1}^j, u_k)]$

Define $[Z_k]$ using $[z_k, X_{k-1}^{\text{expanded}}]$ 

Define $W_k := \eta Z_k^T I U_k W_{k-1}$

Normalize $W_k$

End

At desired times, particles can be resampled:

**Step 3: Resample Single Dimension Particle Filter**

Function $[X_k^{\text{expanded}}, W_k^{\text{expanded}}] = \text{SIS}_{\text{Resample}} [X_k, W_k]$:

Draw $[X_k^{\text{expanded}}, W_k^{\text{expanded}}]$ from $p(x_k|u_{1:k}, z_{1:k})$, characterized by $[X_k, W_k]$

Normalize $W_k^{\text{expanded}}$

End

Lastly, the state estimator can be queried at any time steps 1 and 2 are not executing:

**Step 4: Estimate from Single Dimension Particle Filter**

Function $[x_k] = \text{SIS}_{\text{Estimate}} [X_k, W_k]$:

Calculate $x_k = X_k^T W_k$

End

Step 2 can be performed in a single update subroutine. There are various methods to derive the number of samples $N_s$, to resample the population of particles and model prior probabilities.

The Robust Concurrent Individual and Social Learning (RCISL) mechanism filters the sensor noise and other uncertainty for the concurrent individual/team learning.
Chapter 4
Collaboration through Advice Sharing

Advice Sharing, between agents, can be seen as a mechanism that enables agents or humans to cooperate with each other without redefinition of multi-agent reward functions. Specifically, the sharing of both agent-agent and human-agent advice is considered. In Section 4.1, the integration of the Advice Exchange mechanism is considered, which allows agents to accept and give advice to peers in a multi-agent team. In Section 4.2, the integration of human advice, using a Human Advice Layer is discussed. In both sections, it is demonstrated that the local optimality of the MDP solutions as found by the CISL approaches are not compromised though the acceptance of advice.

4.1 Collaboration: Advice Exchange

In order to allow agents to share their experience an Advice Exchange mechanism [33] is used, which enables collaboration through the exchange of advised actions between agents. An agent $i$ uses actions from a superior agent $k$’s policy $\pi_{i,t}^k$, when agent $k$ is acquiring it’s reward $R_t$ in greater quantity on average than agent $i$. the Advice Exchange mechanism is implemented by replacing the individual policy $\pi_{i,t}^i$ of and agent $i$ with an advised policy, $\tilde{\pi}_{i,t}^i$, for the exploration of the Individual Performance MDP:

$$\tilde{\pi}_{i,t}^i = \begin{cases} \pi_{i,t}^k, & \text{IF } k = \arg\max_z (\bar{q}_h^k) \text{ AND } \bar{q}_{ih}^i < \bar{q}_h^k - d|\bar{q}_h^k| \\ \pi_{i,t}^i, & \text{AND } \bar{q}_h^i < \bar{q}_h^k \text{ AND } \sum_{a_i} Q_{i,t}^i(s_{i,t}, a_i) < \rho \sum_{a_i} Q_{i,t}^k(s_{i,t}, a_i) \end{cases}$$

(42)

Equation 42 states that three advice conditions must be met for agent $i$ to accept advice from advisor agent $k$. These conditions are formulated using three quality parameters. These quality parameters are based on the reward $q$ received in a given iteration from the Individual Performance MDP, and they include the average quality over a single epoch ($\bar{q}$), the current average quality ($\hat{q}$), and the best average quality ($\bar{q}$). The subscript $h$ represents the current epoch, a set of consequent iterations, the parameter $d$ represents a self-confidence parameter ranging from 0 to 1, and the potential advisor set $Z^i$ contains the indexes of all suitable advisors.
for agent $i$. The parameter rho $\rho$ represents the significance of advisor experience. The average quality parameter for agent $i$ can be calculated as a simple moving average at the end of an epoch with a total of $M$ iterations:

$$\bar{q}_{th}^i = \frac{\sum_{m=1}^{M} q_{mh}^i}{M} \tag{43}$$

The current average quality of an epoch $h$ is defined as:

$$\bar{q}_h^i = (1 - \eta)\bar{q}_{M(h-1)}^i + \eta\bar{q}_{(h-1)}^i \tag{44}$$

where $0 \leq \eta \leq 1$ is a decay parameter. Lastly, the best average quality for epoch $h$ is defined as:

$$\bar{q}_h^i = \max(\bar{q}_{M(h-1)}^i, \rho\bar{q}_{(h-1)}^i) \tag{45}$$

where $0 \leq \rho \leq 1$ is a weighting factor. The initial value of all the three qualities can be chosen as zero.

**Optimality for Advice Exchange**

The Advice Exchange mechanism does not prevent the Q-Learning algorithm from finding a locally optimum utility function $Q_{I,t}^i$, because the mechanism still guarantees infinite visitations to all reachable states $(s_I \in S_I)$ at time infinity, since given any state-action pair $(a_I, s_I)$ the action selection probability $\bar{\pi}_I^i(a_I|s_I)$ is a real positive number. Further, no parameters given to the Q-Learning formulation change when implementing the Advice Exchange mechanism, as the reward function $(R_t)$ and transitional model $(T_t)$ remain identical. Thus, it is expected that the advised policy, $\bar{\pi}_I^i$, will maintain discovery of a locally optimum utility function $Q_{I,t}^i$ for the maximization of individual reward $R_I$.

The effect of the Advice Exchange mechanism on task selection can be examined through the following deconstruction of the Team Performance dependency, (which is described in Figure 1). The dependency is entirely characterized by the environment and individual behaviour policies $\{\pi_1^n\}$. More specifically, the reward function $R_T$ is calculated using the expected average trial times $\tau_t(i,j)$ and item assignment status $s_{a,t}(i,j)$, as observed through monitoring all individual behaviour policies. Thus, for the Task Allocation MDP to be solvable by the RL-Alliance mechanism, the transitional model and reward function must both be stationary processes after a certain iteration, $t_s$. This stationary requirement is satisfied when
using Q-Learning regression and softmax action selection alone, as the set of individual policies \{\pi_1^t, ..., \pi_n^t\} converge to stationary probabilities at time infinity. The Advice Exchange mechanism may invalidate this assumption, because there is always a possibility that at least one of the advised policies \{\bar{\pi}_1^t, ..., \bar{\pi}_n^t\} is not a stationary process; hence, the actual average trial time may \bar{\tau}_t(i,j) vary due to the acceptance of advice:

\[
\bar{\tau}_t(i,j) = \begin{cases} 
\tau_t(k,j) & \text{IF } \bar{\pi}_1^t(i,j) = \pi_{i,t}^k \\
\tau_t(i,j) & \text{otherwise}
\end{cases}
\]

The RL-Alliance mechanism, after the trial at time \(t\), will update the average trial time \(\tau_{(t+1)}(i,j)\) based on the advised behaviour policy \(\bar{\pi}_1^t\). In effect, the RL-Alliance mechanism will regress to a value for the agent’s expected advised policy \(\bar{\tau}_t(i,j)\), which is not necessarily a stationary process. Despite this effect, the Advice Exchange mechanism can support pareto-optimal task allocation under two circumstances: a) all potential advisor policies converge to identical policies \(\{\pi_1^t = \cdots = \pi_n^t\}\) by a certain time \(t_s\); or b) the Advice Exchange mechanism will cease at the time \(t_s\). Therefore, one direct method to ensure that the Advice Exchange mechanism will not interfere with pareto-optimal task selection is to vary the self-confidence parameter \(d\) such that the Advice Exchange mechanism will cease advice taking before a certain time \(t_s\). A softmax expression such as the following can provide the desirable behaviour:

\[
d_t = e^{(Kt/t_s)}/(1 + e^{(Kt/t_s)})
\]

The parameter \(K\) adjusts how the effect of Advice Exchange mechanism is decreased in time, such that:

\[
K = ln(\theta_s) - ln(100 - \theta_s)
\]

where \(\theta_s\) is the inactivity percentage of the Advice Exchange mechanism at time \(t_s\). Thus, the Advice Exchange mechanism allows sharing of behaviours between agents while still allowing the L-Alliance mechanism to converge toward a pareto-optimal solution for the Task Allocation MDP.

4.2 Collaboration: Human Advice Layer

In this section the acquisition of human advice from a trained Gaussian Mixture Model (GMM) is discussed. First, the structure of the GMM, also called the Human Advice Function \(H\),
and its integration with the CISL approach is discussed. Next, the general Expectation
Maximization method used to train the Human Advice Function $H$ is explained. The formulation
for describing a multivariate Gaussian $g$ as a function of an agent’s state $s \in S_I$ follows:

$$g_k(s) = \frac{1}{(2\pi)^{d/2}|C_k|^{1/2}} e^{-\frac{1}{2}(s-\mu_k)^T C_k(s-\mu_k)}$$  \hspace{1cm} (49)$$

where $s \in \mathbb{R}^d$, and $\mu_k \in \mathbb{R}^d$ and $C_k \in \mathbb{R}^{d \times d}$ are mean and covariance of the Gaussian
distribution, respectively. I also denote the prior probability of observing a Gaussian with index
$k$ as $p_{k,a}$, where $a$ denotes an observed (human) action $a \in A_I$. The Human advice function $H$
is constructed by using a mixture of Gaussian probability density functions $g$. A prior probability
$p_{k,a}$ for each distribution is computed \textit{a priori}.

$$H(s,a) = \sum_{k \in K_a} p_{k,a} g_k(s)$$  \hspace{1cm} (50)$$

Each Gaussian distribution within the Human Advice Layer is referenced by an index $k \in$
$\{1, ..., N\}$, which is within a set of distribution indexes for an action $a$, $K_a \subset \{1, ..., N\}$. Additionally, the union of all distribution index sets retrieves all distribution indexes,
$(\bigcup_{a \in A_I} K_a) = \{1, ..., N\}$, and the intersection of any two distribution index sets is the empty set:
$\forall a, b \in A_I: (K_a \cap K_b) = \{\}$. The total distribution count $N$ and inclusion of distribution indexes
$k$ into each distribution index set $K_a$ is chosen by the designer.

To accept human advice given the function $H$, an agent’s action selection policy $\pi(a | s)$ is
enhanced to deliver human advice, $\bar{\pi}_I(a | s)$:

$$\bar{\pi}_I(a | s) = \begin{cases} \arg \max_{a \in A_I} H(s,a); & \max_{a \in A_I} H(s,a) > \beta \text{ AND } f_h(s) < \rho \\
\pi_I(a | s); & \text{otherwise} \end{cases}$$  \hspace{1cm} (51)$$

where a minimum human confidence threshold $\beta \in [0,1]$ and a maximum advice count $\rho$ limit
the amount of advice given to an agent. The advice count function $f_h(s)$ returns the number of
times the current agent has taken advice when in state $s$. Thus, it is straightforward to accept
advice from a human advisor, given appropriate basis functions, $p_{k,a}, \mu_k, C_k, k \in \{1, ..., N\}$. In
This paper unnormalized density functions are used to maintain the relative densities between the GMM distributions, $H(s, a), a \in A_t$. Another approach would be to normalize all GMM models $H$ over the set of all actions $a \in A_t$, by considering the relative set sizes for each action $S_H(a)$. Thus, the action with the highest likelihood of human selection is preferentially chosen when an agent lacks enough experience in a state $s$.

To train the Human Advice Functions, the expectation maximization method can be used. The expectation maximization algorithm is executed in two steps, the E-step and the M-step. In the E-step a prior expectation $q_{k,t}$ is obtained from the GMM distributions of the current step $t$:

$$q_{k,t}(s, a) = \frac{p_{(k,a),t}g_{k,t}(s)}{\sum_{j \in K_t} p_{(j,a),t}g_{j,t}(s)} \quad (52)$$

where an agent’s state density function $q_{k,t}(s, a)$ quantifies the likelihood that one state $s$ belongs to a Gaussian distribution with index $k$ at step $t$. The human training data set $S_H(a)$ contains all recorded states $s$ for which a human action $a$ has been observed in the Secret Collect game. In the set $S_H(a)$ it is possible to have duplicated state entries $s$.

In the M-step, the values of the prior probability, mean, and covariance matrix are updated recursively. The diagonal values $\sigma_k^2$ of covariance matrix $C_k$ are defined, assuming mutual independence between all dimensions $d$:

$$\mu_{k,(t+1)} = \frac{\sum_{s \in S_H(a)} q_{k,t}(s, a) \cdot s}{\sum_{s \in S_H(a)} q_{k,t}(s, a)}$$

$$\sigma_{k,(t+1)}^2 = \frac{\sum_{s \in S_H(a)} q_{k,t}(s, a) \left\| s^2 - \mu_{k,t}^2 \right\|}{\sum_{s \in S_H(a)} q_{k,t}(s, a)} \quad (53)$$

$$p_{(k,a),(t+1)} = \frac{\sum_{s \in S_H(a)} q_{k,t}(s, a)}{\sum_{j \in K_t} \sum_{s \in S_H(a)} q_{j,t}(s, a)}$$

The E-Step and M-Step can be repeated iteratively to monotonically maximize the likelihood that the GMM models explain the human advice data. Multiple GMM models are
trained, each pertaining to a corresponding human training data set \( S_H(a) \). Each change in both the human training data \( S_H(a) \) and the advice function \( H(s,a) \) will not affect prior advice functions \( (H(s,b), b \neq a) \).

4.2.1 Optimality for the Human Advice Layer

The HAL operates by changing an agent’s policy \( \pi_{l,t} \) to a policy capable of accepting human advice \( \bar{\pi}_{l,t} \). After a certain experience threshold, advice will cease being given by the HAL, \( (f_h(s_t) \geq \rho \rightarrow \bar{\pi}_{l,t} = \pi_{l,t}) \).

If the experience threshold \( \rho \) can be exceeded \( (f_h(s_t) \geq \rho) \) for all states \( s \in S_l \) before time \( t_s \), then each agent will behave according to its own stably stochastic policy \( \pi_l \). In this case the RL-Alliance mechanism and Q-Learning mechanism can both converge on accurate policies, due to the satisfaction of the both the Task Selection and Team Performance dependencies.

Otherwise, satisfaction of the Task Selection and Team Performance dependency, will occur at a later time than \( t_s \), after all agents cease accepting human advice \( (f_h(s_t) \geq \rho) \).

Thus, the Human Advice Layer does not prevent the discovery of locally optimal or pareto-optimal policies for the multi-agent CMDP. Attention should be given to controlling the experience threshold \( \rho \), to speed the desired convergence behaviour.
Chapter 5
Simulation and Results

In this section both the design of and results from the heterogeneous team foraging case study are discussed. The case study design covers three areas: the implementation of the four multi-agent regression models, the addition of a state estimator into a CISL mechanism, and the implementation of two advice sharing mechanisms. The results section follows and explores the team performance conferred by the three aforementioned case study areas.

5.1 Heterogeneous Team Foraging

The objective of the case study is to move 8 items to a target zone, which has a diameter of 2m. The 2 dimensional simulation world is sized as 10m by 10m. Each item is classified as heavy or light. There are 8 heterogeneous robots that are classified in equal number within four categories: strong-slow, strong-fast, weak-slow, and weak-fast. Heavy items can be manipulated by two robots together, or one strong robot alone. The team must cooperate to forage all items into the target zone within 15,000 iterations, or the simulation is terminated. Each item is considered as a circle with diameter 0.50m. Robots have a diameter of 0.25m. There are four

Figure 3: A representation of the simulation environment
circular obstacles in the area, each with a diameter of 1m. For every foraging trial, called a run, the initial positions of all simulation objects are randomized.

In the following sections, three MDP models are presented for the aforementioned foraging scenario. Initially, a Concurrent MDP will be formulated for the Q-Learning and L-Alliance approach. Afterward, a second Concurrent MDP will be formulated using separate Q-Learning algorithms for individual performance and task allocation. Lastly, a decentralized MDP will be used as the control, solved by single Q-Learning algorithm for both individual performance and Task Allocation MDPs.

5.1.1 Concurrent MDP: Q-Learning and L-Alliance

In this section the implementation of Concurrent MDP problem model is described for the foraging scenario. Initially, the Individual Performance MDP is explained, and afterward three definitions of the Task Allocation MDP are shown.

**Individual Performance MDP:**

The individual MDP used in this case study consists of a \( <S_I,A_I,T_I,R_I> \) tuple defined as follows:

**State** \( s_I \in S_I \): is defined:

\[
 s_I = \{r_x, r_y, r_\theta, o_x, o_y, l_x, l_y, l_p, g_x, g_y, b_x, b_y\} 
\]  

(54)

The state \( s_I \) includes information about the robot \( r \), nearest obstacle \( o \), assigned item \( l \), target zone \( g \), and world border \( b \). The subscripts \( x \) and \( y \) represent the x-y coordinates of the objects, and \( r_\theta \) represents the robot’s orientation from the x-axis of the world frame. The variable \( l_p \) represents the type of the assigned item as 1 for light and 2 for heavy.

**Action** \( a_I \in A_I \): is move_forward, move_backward, turn_right, turn_left, or interact. Each robot moves at either 0.2m (slow robot) or 0.4m (fast robot) per move action. All turns are fixed at \( \pi/4 \) radians per turn. Interact attaches the robot to an assigned item when the robot is currently not gripping another item.
**Transition** $T_t(s_t, a_t, s'_t)$: represents the probability of transitioning from one state $s_t$ into another state $s'_t$, given an action $a_t$.

**Reward** $R_t(s_t, a_t, s'_t)$: administers the reward (penalty) points based on whether the robot moves closer (farther) to the assigned item or moves the item closer (farther) to the target zone or delivers the item to the target zone. Additional reward is administered if the robot moves further from the target zone while not assigned to an item (Table 1). The parameter $\Delta \omega$ represents a threshold, which is currently set to 17cm.

Table 1: Reward function for the Individual Performance MDP.

<table>
<thead>
<tr>
<th>Event</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>Item is moved at least $\Delta \omega$ cm closer to the target zone</td>
<td>0.5 reward</td>
</tr>
<tr>
<td>Item is moved at least $\Delta \omega$ cm away from the target zone</td>
<td>-0.3 reward</td>
</tr>
<tr>
<td>Robot moved at least $\Delta \omega$ cm closer to the assigned item</td>
<td>0.5 reward</td>
</tr>
<tr>
<td>Robot moved at least $\Delta \omega$ cm away from the assigned item</td>
<td>-0.3 reward</td>
</tr>
<tr>
<td>The robots assigned item is returned</td>
<td>10 reward</td>
</tr>
</tbody>
</table>

**Task Allocation MDP**

The task allocation learning problem is to determine the optimal assignment of items to the set of agents, where a task selection action at each iteration $t$ is considered optimal when the robots minimize their average item return effort, $\overline{\tau}_t(i, i', j)$. First, the single-item-single-robot assignment will be discussed using the average trial duration, $\tau_t(i, j)$ and afterward, the robots’ pairwise physical cooperation will be discussed. The Task Allocation MDP is modeled as a $< S_T, A_T, T_T, R_T >$ tuple:

**State** $s_T \in S_T$: contains three pieces of information: each robot’s average trial time toward each task, each items current delivery status and, each robot’s assignment status toward each task. The state consists of three sub-states, $\{s_{T,1}^1, s_{T,2}^2, s_{T,3}^3\}$.

1) The robot performance sub-state is described by

$$s_{T,1}^1 = \begin{bmatrix} \tau_t(1,1) & \cdots & \tau_t(1,k) \\ \vdots & \ddots & \vdots \\ \tau_t(n,1) & \cdots & \tau_t(n,k) \end{bmatrix},$$

(55)
where \( n \) and \( k \) denote the number of robots and items, respectively. For the end of this section an inversed robot performance matrix is defined:

\[
\mathbf{s}^{-1}_{r,t} = \begin{bmatrix}
\frac{1}{\tau_t(1,1)} & \cdots & \frac{1}{\tau_t(1,k)} \\
\vdots & \ddots & \vdots \\
\frac{1}{\tau_t(n,1)} & \cdots & \frac{1}{\tau_t(n,k)}
\end{bmatrix}
\]

(56)

2) The task status sub-state is described:

\[
\mathbf{s}^2_{r,t} = \begin{bmatrix} c_{1,t} \\ \vdots \\ c_{k,t} \end{bmatrix}
\]

(57)

The variables \( c_{j,t} \in \{0,1\} \) represent the delivery status of an item, such that 0 indicates a returned (delivered) item, and 1 denotes an undelivered item.

3) The current assignment of robots to items follows:

\[
\mathbf{s}^3_{r,t} = \begin{bmatrix}
s_{a,t}(1,1) & \cdots & s_{a,t}(1,k) \\
\vdots & \ddots & \vdots \\
s_{a,t}(n,1) & \cdots & s_{a,t}(n,k)
\end{bmatrix}
\]

(58)

The variables \( s_{a,t}(i,j) \in \{0,1\} \) represent the assignment status of a robot \( i \) to item \( j \), such that 0 denotes unassigned and 1 indicates assigned. Note that robots cannot be assigned to items that are returned.

**Action** \( \mathbf{a}_t \in \mathbf{A}_t \): is the assignment of robot \( i \) to item \( j \).

\[
a_{t,t}(i,j): \begin{cases} 
    s_{a,(t+1)}(i,j) = 1 \\
    s_{a,(t+1)}(i, k \neq j) = 0
\end{cases}
\]

(59)

**Transition** \( \mathbf{T}_t(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t) \): represents the probability of transitioning from one state \( \mathbf{s}_t \) into another state \( \mathbf{s}'_t \), given an action \( \mathbf{a}_t \).

**Reward** \( \mathbf{R}_t(\mathbf{s}_t, \mathbf{a}_t, \mathbf{s}'_t) \): administers the reward for the allocation of items to the best preforming robots in the team, and it is characterized by the minimization of the average trial duration, \( \tau_t \), at state \( \mathbf{s}_{r,t} \).

\[
\mathbf{R}_{r,t}(\mathbf{s}_{r,(t-1)}, \mathbf{a}_{r,(t-1)}, \mathbf{s}_{r,t}) = \sum_{i=1}^{n} \sum_{j=1}^{k} \frac{s_{a,t}(i,j)}{\tau_t(i,j)} = \text{trace} \left( \mathbf{s}^{-1}_{r,t} \mathbf{s}^3_{r,t} \right)
\]

(60)
Physical cooperation

To support physical cooperation, the coalition average trial time $\tau_t(i,i',j)$ can be considered, as discussed in 3.2.2. A first approach can modify the Task Allocation MDP’s state to characterize all a pairwise robot coalition average trial times. The expansion of the sub state $s_{T,t}$ can turn into:

$$s_{T,t}^1 = \left[ \begin{array}{cccc} \tau_t(1,1,1) & \ldots & \tau_t(1,1,k) \\ \vdots & \ddots & \vdots \\ \tau_t(n,1,1) & \ldots & \tau_t(n,1,k) \end{array} \right] = \left[ \begin{array}{cccc} \tau_t(1,n,1) & \ldots & \tau_t(1,n,k) \\ \vdots & \ddots & \vdots \\ \tau_t(n,n,1) & \ldots & \tau_t(n,n,k) \end{array} \right]$$

(61)

As mentioned in previously, such an increase in state space by a factor of $n$ may be intractable due to computational limitations. A second less direct approach is used to minimize the size of the state space matrix; each robot’s effort metric $\bar{\tau}_t(i,i',j)$ is stored to characterize the total team effort.

5.1.2 Concurrent MDP: Individual and Task Allocation

Q-Learning algorithms

As a contrast to the full information MDP, as designed for the L-Alliance mechanism presented in section 5.1.1, a Task Allocation MDP suitable for Q-Learning is considered. The model for the Individual Performance MDP is kept identical with the approach discussed in section 5.1.1. The Task Allocation MDP for the Q-Learning mechanism can be modeled as a $< S_T, A_T, T_T, R_T >$ tuple.

State $s_T \in S_T$: represents the set of the three nearest available items to robot $i$,

$$s_{T,t} = \{s_{k,t}(i,1), s_{a,t}(i,1), s_{k,t}(i,2), s_{a,t}(i,2), s_{k,t}(i,3), s_{a,t}(i,3)\},$$

(62)

where value $s_{k,t}(i,j)$ is the robot $i$’s perception of the presence and type of the $j^{th}$ nearest item, and is valued $\{0: \text{none}, 1: \text{light}, 2: \text{heavy}\}$. Similarly, the value $s_{a,t}(i,j)$ is the assignment status of the robot $i$ with item $j$, $\{0: \text{unassigned}, 1: \text{another_robot_only}, 2: \text{current_robot_only}, 3: \text{cooperating}\}$. Unavailable or fully utilized items are excluded from the state entirely. If an item is currently assigned, it will be represented in the first state position ($j = 1$), regardless of the simulation geometry. Lastly, empty slots are filled with none or unassigned values. Thus, this state model does not reflect other robots’ performance values, drawing a contrast with the L-
Alliance approach. The RL-Alliance approach requires the monitoring of all robots’ average trial time.

**Action** $a_T \in A_T$: is the assignment or release of robot $i$ to/from an item $j$. The action $assign_j$ ($j \in \{1,2,3\}$) assigns the robot to the $j^{th}$ item; and $drop$ releases any gripped item. If only a single item is available in the state $s_{T,t}$ any action $assign_j$ will assign the robot to the single available item.

**Transition** $T_T(s_T, a_T, s'_T)$: represents the probability of transitioning from one state $s_T$ into another state $s'_T$, given an action $a_T$.

**Reward** $R_T(s_T, a_T, s'_T)$: for the Task Allocation MDP is defined as the average reward acquired by the robot $i$ within a duration $\theta_d$ of the Individual Performance MDP. In addition to the average reward acquired by the individual process, which is assumed to be zero-mean by a time $t_s$ (Table 2).

<table>
<thead>
<tr>
<th>Event</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>The robot currently assigned to an otherwise unoccupied item.</td>
<td>5 reward</td>
</tr>
<tr>
<td>The robot is gripping an item, and the item has moved less than</td>
<td>-10 reward</td>
</tr>
<tr>
<td>17cm since the last task allocation iteration.</td>
<td></td>
</tr>
<tr>
<td>The robots assigned item is returned.</td>
<td>10 reward</td>
</tr>
</tbody>
</table>

**5.1.3 Decentralized MDP: Single Q-Learning Algorithm**

The final team learning model, which is provided as a control, is denoted as a decentralized MDP, which degenerately characterizes the Individual Performance MDP (section 5.1.1) and the Task Allocation MDP (section 5.1.2) using a *transition independent decentralized* model [4, 29, 36]. The model is expressed as a $<S_U,A_T,T_U,R_U>$ tuple:

**State** $s_U \in S_U$: is a combination of the Individual Performance MDP state set (section 5.1.1) and the Task Allocation MDP state set (section 5.2.2, two nearest items), $S_U \subset S_T \times S_I$. A
*drop_available* state value $d_a$ is added, which represents the availability (one) or non-availability (zero) of drop action:

$$s_U = \{r_x, r_y, r_\theta, o_x, o_y, l_x, l_y, l_p, g_x, g_y, b_x, b_y, s_{k,t}(i, 1), s_{a,t}(i, 1), s_{k,t}(i, 2), s_{a,t}(i, 2), d_a\}$$ (63)

**Action** $a_U \in A_U$: is an action in the union of the Individual Performance MDP action set and the Task Allocation MDP action set, $(A_U \subset A_I \cup A_f)$. To limit the *drop* action, one constraint is added to the model: When a robot transitions from not gripping to gripping an item, the *drop* action is unavailable for a duration $\theta_d$ of decentralized MDP iterations. The duration parameter $\theta_d$ allows for a similar implementation of the trial time for the L-Alliance mechanism, robots are given at least $\theta_d$ iterations with each item attempt.

**Transition Probabilities** $T_U(s_U, a_U, s'_U)$: is the probability of transitioning from one state $s_U$ into another state $s'_U$ given an action $a_U$.

**Reward** $R_U(s_U, a_U, s'_U)$: is a direct sum of the Individual Performance MDP reward and Task Allocation MDP reward as indicated in sections 5.1.1 and 5.1.2.

$$R_U(s_U, a_U, s'_U) = R_I(s_U, a_U, s'_U) + R_T(s_U, a_U, s'_U)$$ (64)

### 5.1.4 Particle Filter Integration for Uncertainty

The state estimator for heterogeneous team foraging can be constructed using a collection of particle filters. The system observation is extended into two dimensions $\{x_{x,k}, x_{y,k}\}$. The observation $x$ can represent any of the state variables $o$ for the nearest obstacle, $l$ for the nearest item, $g$ for the Target Zone, and $b$ for the nearest world border. The subscript $x$ and $y$ denote for Euclidian $x$ and $y$ positions, respectively. Given the particle filter formulation, the state estimator $s_e(u_k, z_k)$ can be expressed as:
Where $u_k$ represents an action defined by the Individual Performance MDP and $z_k = \{o_{x,k}, o_{y,k}, l_{x,k}, l_{y,k}, g_{x,k}, g_{y,k}, b_{x,k}, b_{y,k}\}$ is a collection of sensor readings which characterize the system state observations with noise. To derive the state estimator the control probability, observation probability and resampling method are defined:

**Control Probability:**

The control probability $p\left(\{x_{x,k}, x_{y,k}\}|u_k, \{x_{x,k-1}, x_{y,k-1}\}\right)$ is a motion model, which describes the likelihood that a point $x_k$ is accurate given a robot was at a point $x_{k-1}$ during the execution of action $u_k$. A value for discrepancy $d$ is found as the Euclidean distance between $x_k$ and the expected value $\bar{x}_k$.

$$d = \sqrt{(x_{x,k} - \bar{x}_{x,k})^2 + (x_{y,k} - \bar{x}_{y,k})^2} \quad (66)$$

The probability is calculated using a zero-mean Gaussian density function with a standard deviation of $\sigma$. To estimate $\{\bar{x}_{x,k}, \bar{x}_{y,k}\}$ the action $u_k$ is applied to a given point $\{x_{x,k-1}, x_{y,k-1}\}$.

The movement actions for the particle filter of the foraging scenario are 'move forward,' 'rotate right' and 'rotate left.' The action $u_k = 'move forward'$ adds a value on the $x$ axis, such that $\bar{x}_{x,k} = x_{x,k-1} - 20cm$ or $\bar{x}_{x,k} = x_{x,k-1} - 40cm$, depending on the speed of the robot. The action $u_k = \{'rotate right', 'rotate left'\}$ alters the state variables as perceived by the robot, such that:

$$\begin{bmatrix} \bar{x}_{x,k} \\ \bar{x}_{y,k} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & -\cos\theta \end{bmatrix} \begin{bmatrix} x_{x,k-1} \\ x_{y,k-1} \end{bmatrix}, \quad (67)$$

where $\theta$ equals $\pi/4$ or $-\pi/4$. The movement model assumes that objects cannot move through each other and that the objects cannot move outside the 10m by 10m area.
Observation Probability:

To define the observation probability \( p(z_k \mid x_k) \) for the state variables, the Euclidean distance is used once again.

Resampling Method:

Resampling is performed in stages. Having \( N_s \) samples, the particle filter formulation is first used to draw \( N_s \) new particles, raising the total particle count to \( 2N_s \). Then, \( N_d \) new particles are added using the control probability raising the total particle count to \( 2N_s + N_d \). Lastly, the particles are pruned down to the size of \( N_s \) by discarding the least likely values according to the particle filter formulation. Some ‘random’ particles are also added around each expected value during resampling, to help prevent the population of particles from becoming degenerate.

5.1.5 Collaboration: Advice Exchange

To enable advice sharing through Advice Exchange, a set of potential advisors \( Z^i \) is required for every robot \( i \). Thus, the set of advisors is defined to be any agent of the same strength, i.e. if robot \( i \) is weak-fast, then the potential advisors are all the weak-slow and weak-fast robots. For all CISL approaches, the quality \( q_t^i \) will be defined as the reward obtained as a result of the current state \( s_{l,t} \) and action \( a_{l,t} \) at iteration \( t \), \( q_t^i = R_l(s_{l,t}, a_{l,t}, s_{l,(t+1)}) \).

5.1.6 Collaboration: Human Advice Layer and Secret Collect

To implement the Human Advice Layer into the foraging scenario GMMs were required. In this section the process used to both structure the GMM models and collect the training data is discussed. First, the training data had the following structure:

\[
 s_h = \{r_x, r_y, \tau_{\theta}, h_x, h_y, l_x, l_y, l_p, g_x, g_y, b_x, b_y\} \tag{68}
\]

The structure of a state \( s_h \in S_l \) is discussed in the Individual Performance MDP case study description. To gather the training data, a simulation game called Secret Collect was played by five human advisors, or players. In Secret Collect, players attempt to guide a team of robots through a heterogeneous foraging experiment. There is a finite time limit, so the game can pressure the players into making incorrect decisions as they rush to collect items. The game progresses in a sequence of runs, or rounds. In every way, the Secret Collect operating
environment mirrors the foraging simulation environment, with the exception that the number of items and objects is four respectively; human players were found to have difficulty controlling more than four robots, and the state $s_h$ was independent of the number of robots.

![Game Environment](image)

**Figure 4:** The players control robots by tracing a line of action for each robot.

In Figure 4 screenshots of the game environment are shown. The game’s objective, in general, is to draw a line from each robot, to a chosen task, and then toward a goal location. The player is free to choose actions that don’t contribute to the return of a foraging item. To avoid running out of Simulation Time (*joules*) it is prudent for the player to choose efficient actions.
After a player draws an action line, the selected robot follows the path, completing the required actions $a \in A_i$ along the way. After every executed action, each robot records a feature vector $s_h$ into the unsupervised training data set $S_H(a)$. Duplicate actions are allowed to be inserted into the set $S_H(a)$.

Over the course of the data collection period, The Secret Collect game gathered both beneficial and counter-productive feature vectors. The data was assumed to be of effective quality, and used to train the GMMs without any preparation or filtering.

5.2 Results

In this section the performance results of the concurrent MDPs and the decentralized MDP approaches are compared, as applied to the heterogeneous foraging. Following this, the application of state uncertainty to the CISL approach, (Q-Learning with RL-Alliance,) is considered. The Robust Concurrent and Individual Learning approach is also considered. Lastly, Advice sharing to augment the CISL approach, Q-Learning with RL-Alliance, is considered.

In general, each experiment represents the average of 20 simulations, each consisting of 300 runs. Experiments that required less or more than 20 simulations are noted inline. Each run was limited to 15,000 iterations. For each simulation the utility values and robot performance sub-states were set to their default values. In addition, due to the random nature of learning algorithms, a ten-point moving average was used to smooth the plot of each experiment. Finally, to consider the statistical distinctness of each simulation, a welch T-test was used on characteristic results.

To analyze multi-agent performance, the following metrics have been defined:

- **Simulation Time**: The total simulation time required by individual robots to complete the foraging objective, in terms of number of iterations, which characterizes the agility of team.
- **Total Effort**: The total number of actions executed by individual robots to complete the foraging objective, which characterizes the total team effort.
- **Individual Reward**: The average reward $R_i$ received by a single robot assigned to a single item.
- **Cooperative Effort**: The total number of cooperative actions executed per run, when completing the foraging scenario.
• *Expected False Observation*: The average measurement error an agent encounters when measuring its state.

• *Expected False Reward*: The expected amount of false reward received by an agent in an average iteration.

### 5.2.1 Four Regression Approaches

The enhancement of an individual robot’s foraging ability is considered first, while comparisons between the Concurrent approaches and the Single Q-Learning approach will be studied next. Figure 6 illustrates the foraging performance of a typical robot before (a) and after (b) training by the individual Q-Learning algorithm. It is noted that the Q-Learning mechanism and Individual Performance MDP succeed in enabling robots to enhance their individual performance.

![Figure 5: individual performance of an untrained robot (a) versus the same robot when trained (b).](image-url)
Figure 6: Average Individual Reward ($R_t$) for a typical robot assigned to an item.

Figure 6 depicts the variation in the average individual reward received by a typical robot over the training runs. It is noted that under all CMDP approaches, the individual reward obtained by a typical robot is maximized. The higher values of maximum average reward for the L-alliance approaches, compared to that of Concurrent Q-Learning, indicate the efficacy of task allocation approaches in assigning robots to each item, which results in a better overall performance.

The remaining comparison studies are highlighted in the following stages: First, the mean, standard deviation and mutual statistical significance of all learning approaches are considered. Next, the effect of Advice Exchange mechanism on the learning and converged behaviour is studied. In the following discussions, the Single Q-Learning and Concurrent Q-Learning approaches are denoted as the **Q-Learning approaches**, and the L-Alliance and RL-Alliance approaches as the **L-Alliance approaches**.
Figure 7: Mean performance of concurrent and decentralized MDPs: a) Simulation Time, b) Total Effort
Figure 7 shows the team’s Simulation Time and Total Effort for the foraging scenario experiments. It is shown that convergent policies are empirically executed by all learning approaches. The CMDP approaches seem to dominate the Single Q-Learning method with respect to Simulation Time and Total Effort metrics. With regard to the team’s agility, i.e., Simulation Time (Figure 7a), both L-Alliance approaches tend to start from longer completion times but converge at a steeper rate, whereas the Q-Learning approaches, concurrent and single, start from shorter completion times with a more moderate convergence to their final values. With regard to the team’s effort, the Total Effort metric (Figure 7b), both Q-Learning approaches initially require a larger sum of individual effort to complete the foraging task than their L-Alliance counterparts do. Overall, the superior method, in terms of Simulation Time, seems to be the Concurrent Q-Learning approach, whereas the RL-Alliance and L-Alliance approaches seem to be doing better in terms of Total Effort.

A possible explanation for a superior performance of the CISL approaches over the Single Q-Learning approach is that the task allocation actions have a longer time horizon to generate reward, and thus may not have the same solutions on the scale of individual iterations. That is, it may not be possible to represent long-term rewards of the Task Allocation MDP alongside short-term rewards attained by Individual Performance MDP using identical learning parameters ($\alpha, \gamma$) and a single utility function. Thus, a benefit of a CMDP approach is its ability to represent reward on multiple state space and temporal scales.

The Q-Learning approaches require a lesser number of trials than the L-Alliance approaches for the robots to arrive at their stable performance level. However, under the Q-Learning approaches, the total team’s effort is higher than that of L-Alliance approaches. This is because the L-Alliance approaches allow for physical cooperation only when a single robot is projected as incapable of handling a given item alone, whereas there is no mechanism to limit (unnecessary) physical cooperation in the Q-Learning formulations. Thus, a larger number of physical cooperation actions can occur in the Q-Learning approaches, which results in increasing the total team’s effort. The Physical Cooperation metric is shown in Figure 8; for the given scenario where there are just enough robots of each type to handle items of either weight (heavy and light), the L-Alliance mechanisms minimize cooperative effort, which also reduces the Total Effort:
Figure 8: CMDP vs Single Q-Learning, Cooperative Effort

All learning approaches reduce the standard deviation of their performance metrics through the runs, as illustrated in Figure 9, exhibiting their ability to converge toward foraging performance with a stable expected value.
Figure 9: One standard deviation unit for a) Simulation Time and b) Total Effort

Upon inspection of Figure 9 it appears that both the Concurrent Q-Learning and the RL-Alliance mechanisms exhibit the lowest variance during the stable phase, which contains all of the runs after run 200. The Concurrent Q-Learning approach is optimal during much of the transient phase, which contains all of the runs up and including run 200. This conclusion will be further justified by studying the pairwise T-test results which follow.

To verify that the obtained performance metrics for the different approaches are statistically distinguished, pairwise independent Welch’s T-tests were performed for every run, with a population of 20 simulations [45]. Table 3 summarizes the average p-values of the pairwise T-tests for the transient and steady phases when considering Total Effort, as well as the percentage of runs whose p-value is below the alpha-level of 0.05.

Table 3: Paired p-values for four multi-agent regression mechanisms. Marked (*) pairwise comparisons indicate statistical distinguishability.
<table>
<thead>
<tr>
<th>Pairwise Comparisons</th>
<th>Simulation Time</th>
<th>Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transient Phase</td>
<td>Steady Phase</td>
</tr>
<tr>
<td></td>
<td>average p-value</td>
<td>% distinct runs</td>
</tr>
<tr>
<td>(*) Concurrent Q-Learning vs. L-Alliance</td>
<td>0.2641</td>
<td>0.2450</td>
</tr>
<tr>
<td>(*) Concurrent Q-Learning vs. RL-Alliance</td>
<td>0.1659</td>
<td>0.4500</td>
</tr>
<tr>
<td>(*) Concurrent Q-Learning vs. Single Q-Learning</td>
<td>0.0483</td>
<td>0.7400</td>
</tr>
<tr>
<td>L-Alliance vs. RL-Alliance</td>
<td>0.4498</td>
<td>0.0400</td>
</tr>
<tr>
<td>(*) L-Alliance vs. Single Q-Learning</td>
<td>0.2282</td>
<td>0.3200</td>
</tr>
<tr>
<td>(*) RL-Alliance vs. Single Q-Learning</td>
<td>0.2352</td>
<td>0.3500</td>
</tr>
</tbody>
</table>

As shown in Table 3, all pairwise T-tests verify distinct mechanism behaviour, with the exception of the L-Alliance mechanism compared with the RL-Alliance mechanism, and the Concurrent Q-Learning mechanism compared with the L-Alliance mechanism.

For the minimization of Total Effort, the RL-Alliance approach with Advice Exchange seems to be the most efficient choice. Depending on the application objectives, different CMDP approaches may be appropriate. However, one finding of significance is clear: when state space representation is constrained, a CMDP model may significantly outperform a Single Q-Learning method, and tend to settle toward better performance.

### 5.2.2 Results for Uncertainty Simulations

In this section the performance results of the noiseless and uncertain observation CISL approach simulations are compared with the RCISL approach simulations with uncertain state observation. In this section, each experiment included 5 simulations, each consisting of 300 runs,
and the “CISL” or “RCISL” always refer to Q-Learning and RL-Alliance, without and with the addition of a particle filter for state estimation.

The enhancement of an individual robot’s foraging ability is illustrated in Figure 9, by showing the pre- and post-training behaviours of individual robots tasked with returning one or two items. On the left (a, c, e) untrained robots are shown, whereas on the right (b, d, f) trained robots are depicted:
Figure 10: Robot foraging performance: (a) untrained, (b) trained CISL with no noise; (c) untrained, (d) trained CISL with noise; (e) untrained, (f) trained RCISL with noise.
In Figure 10, the top plots (a, b) show typical behaviour of a robot before (a) and after (b) training using the CISL approach. The middle plots (c, d) show the best untrained (c) and trained (d) behaviour of a robot within a scenario with observation uncertainty (std. = 0.4m) using the CISL approach. The bottom plots (e, f) show typical untrained (e) and trained (f) behaviour of a robot in the same uncertain scenario using the RCISL approach.

Figure 11 shows the average reward per action received by the individual robots versus the simulation runs for various noise levels. The state observation uncertainty levels 0m, 0.05m, 0.2m and 0.4m denote the same levels of standard deviation for the applied zero-mean Gaussian noise. Figures 11a and 11b show the results for the CISL and RCISL mechanisms, respectively. The reward function administers rewards relative to how far each robot either moves toward an item or carries an item toward the target zone. The uncertainty encountered during state observation increases the movement distances perceived by robots, and hence increases the reward that they receive for their actions. This fact is clearly shown in the graphs for the CISL mechanism (Figure 11a). However, as noise increases in Figure 11a, the rate of increase in the reward slows down, indicating that the CISL mechanism may not perform efficiently under noisy conditions. When considering the results for the RCISL mechanism (Figure 11b) the magnitude of the reward received per action is less than that of the CISL mechanism. This may be because the state estimator in the RCISL mechanism generates a more realistic estimate of the robots’ movement distances. Additionally, as the learning proceeds in Figure 11b, the reward tends to increase until it settles at an ultimate level, demonstrating an increase in the reward expected of a learning mechanism.
Figure 11: Average Individual Reward $R$ obtained by the (a) CISL mechanism and the (b) RCISL mechanism.

To corroborate with the above-mentioned argument, Figure 12 shows the expected false measurement distance for the CISL (12a) and RCISL (12b) mechanisms, as well as the amount of expected false reward for the CISL (12c) and RCISL (12d) mechanisms.
As expected, the RCISL experiments demonstrate a lower amount of Expected False Observation and Expected False Reward when compared with their CISL counterparts. The Expected False Reward metric indicates that each robot’s utility function is more accurate when the RCISL mechanism is used, as explained in Sections 3.2 and 3.3.2. In addition, the Expected False Observation metric indicates that more accurate robot behaviour is executed when the RCISL mechanism is used, as explored in Sections 3.2 and 3.3.1. Thus, for the objective of regressing to an optimal policy, the two metrics indicate that using the RCISL mechanism results in a superior learning behaviour.

The foraging team performance can be considered through reflection upon the Simulation Time, Total Effort, and Cooperative Effort metrics:
Figure 13 shows the mean values of team’s Simulation Time (13a for CISL, 13b for RCISL) and Total Effort (13c for CISL, 13d for RCISL) for the foraging scenario experiments. The application of the particle filter enhances the learning performance for both transient (up to run 200) and steady (after run 200) phases. In all RCISL experiments, both the Simulation Time required for the team to complete the objective and the Total Effort expended by the robots during simulations were reduced, when compared with the corresponding CISL experiments. Thus, the RCISL mechanism allows for a significant performance improvement when compared with the CISL counterpart given the uncertainty of noisy observations. Interestingly, for the
0.05m noise level the CISL mechanism performed only marginally worse than the noiseless scenario, suggesting that for minimal levels of noise the CISL mechanism may not benefit significantly from a state estimator.

Figure 14 shows the Cooperative Effort between robots that physically cooperate when returning items. It demonstrates that robots tend to assist each other more readily under uncertainty. The RL-Alliance mechanism, used to assign items to the robots, allows for physical cooperation only when a robot is projected as incapable of handling a given item alone. This condition arises more frequently when state observation becomes uncertain. Thus, a reduction in the uncertainty expectation metrics also correlates with a lessening of unnecessary physical cooperation actions. Hence, filtering noise seems to allow the task allocation mechanism to perform more efficiently in terms of physical cooperation.

![Figure 14: Robot’s physical Cooperative Effort for all experiments](image)

Lastly, to justify the statistical relevance of the above-mentioned results, pairwise independent Welch’s T-tests were performed for various experiments [45]. Table 4 lists the average p-values of the pairwise T-tests for the transient and steady phases, as well as the percentage of runs whose p-value is below the alpha-level of 0.05, for some select experiments.

First, the T-tests verify that the CISL performance degrades statistically significantly as the uncertainty in the observation increases beyond a certain level. For a small level of noise (0.05m in this case), not much degradation can be inferred statistically in the CISL performance. Secondly, under uncertainty the RCISL mechanism shows a statistically significant performance
improvement over the CISL mechanism. Thirdly, the application of the RCISL mechanism makes the learning behaviour independent of the noise level, so that the performance metrics become statistically indistinguishable regardless of the noise level for the RCISL mechanism.

Table 4: Paired p-values for the CISL and RCISL uncertainty experiments. Marked (*) pairwise comparisons indicate statistical distinguishability.

<table>
<thead>
<tr>
<th>Pairwise Comparisons</th>
<th>Simulation Time</th>
<th></th>
<th>Total Effort</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transient Phase</td>
<td>Steady Phase</td>
<td>Transient Phase</td>
<td>Steady Phase</td>
</tr>
<tr>
<td></td>
<td>average p-value</td>
<td>% distinct runs</td>
<td>average p-value</td>
<td>% distinct runs</td>
</tr>
<tr>
<td>CISL (N=0.05m)</td>
<td>vs. CISL (N=0.0m)</td>
<td>0.3216</td>
<td>23</td>
<td>0.3477</td>
</tr>
<tr>
<td>(*) CISL (N=0.2m)</td>
<td>vs. CISL (N=0.0m)</td>
<td>0.0909</td>
<td>65</td>
<td>0.0260</td>
</tr>
<tr>
<td>CISL (N=0.4m)</td>
<td>vs. CISL (N=0.2m)</td>
<td>0.1890</td>
<td>39</td>
<td>0.1950</td>
</tr>
<tr>
<td>(*) RCISL (N=0.4m)</td>
<td>vs. CISL (N=0.4m)</td>
<td>0.0209</td>
<td>95</td>
<td>0.0036</td>
</tr>
<tr>
<td>(*) RCISL (N=0.4m)</td>
<td>vs. CISL (N=0.2m)</td>
<td>0.0437</td>
<td>81</td>
<td>0.0260</td>
</tr>
<tr>
<td>RCISL (N=0.4m)</td>
<td>vs. RCISL (N=0.2m)</td>
<td>0.4077</td>
<td>13</td>
<td>0.4388</td>
</tr>
<tr>
<td>RCISL (N=0.4m)</td>
<td>vs. RCISL (N=0.05m)</td>
<td>0.4229</td>
<td>8</td>
<td>0.4070</td>
</tr>
</tbody>
</table>

5.2.3 Results for Advice Sharing

The application of both the Advice Exchange (AE) mechanism and Human Advice Layer (HAL) to the CISL mechanism is considered in this section. The CISL and CISL with AE plots are represented by 50 simulations, whereas the CISL with HAL is represented with 20 simulations, as no additional experiments were required to reach statistical significance.

A first step to evaluate the quality of advice sharing can be the consultation of the Average Reward metric. The agents that utilize the HAL experience a 30.2% increase in the amount of reward during the steady phase, the final 100 runs, while seeming to experience an impaired ability to generate reward within the beginning of the transient phase, before the first 200 runs. Conversely, the AE mechanism moderately increases the reward obtained by CISL mechanism consistently.
Figure 15: Average Individual Reward $R_1$ for a typical robot assigned to an item, with and without usage of advice sharing.

The agents using the HAL, despite initial low level of reward acquisition, learn to maximize the reward received from the Individual Performance MDP more significantly, suggesting more efficient long term performance. The consideration of foraging performance can be reflected upon through study of the Simulation Time and Total Effort metrics, which help characterize team foraging performance and efficiency, respectively:
Figure 16: Advice sharing simulation metrics: a) Simulation Time and b) Total Effort
In Figure 16, the CISTL mechanism with the HAL exhibits the most efficient values for both the Simulation Time and Total Effort metrics, demonstrating an improvement over the CISTL mechanism of approximately 46.7% and 48.9% in the stable phase. As suggested by the Average Reward metric (Figure 15), the usage of the HAL decreased the foraging efficiency for the beginning of the transient phase (Figure 16a), as the advice accepted within the transient phase interfered with each robot’s native action selection policy. Thus, the HAL forced the typical robot to execute more explorative actions than it would normally without access to the HAL advice. However, the team performance encouraged by the HAL surpassed that of the other mechanisms quickly, as a consequence of each robot’s additional experience during the explorative transient phase. By the steady phase, the robots who received suggested actions from the HAL behaved in a preferable manner.

As for the AE mechanism, since less experienced robots could utilize behaviours from higher quality robots, the Simulation Time and Total Effort plots were smoothed and minimized. However, the CISTL with AE mechanism did not reach the same effective level of performance as the CISTL with HAL mechanism did, for at least two reasons. First, the AE mechanism provides the most benefit to an advisee when a robot advisor has a significantly superior utility function, i.e., the AE mechanism confers the most benefit when the utility functions are very heterogeneous in terms of quality. Second, the utility functions are effectively homogenized, as all robots are given a similar amount of trials toward each item during the transient phase; (the RL-Alliance mechanism attempts to learn all pairwise robot-item performance values, a consequence of $\beta(i,j)$ in equation 11). Thus, since no single advisor’s utility function is likely to be acutely superior to any advisee’s utility function, the AE mechanism cannot provide extremely beneficial advice. However, the AE mechanism succeeds in reducing the occurrence of inefficient robot action selection, as the mechanism retrains inefficient utility functions.

Thus, it is noted that the CISTL with AE mechanism performs with more stability than the other mechanisms within much of the transient phase. Figure 17 depicts the standard deviation values. In essence, the CISTL with AE enables any inexperienced or inefficiently trained agents to execute the actions of a superior agent until the inefficient utility function is improved.
Figure 17: Advice sharing standard deviation of metrics a) Simulation Time and b) Total Effort

For both of the advice sharing mechanisms, performance stability is increased in the stable phase when compared with the unenhanced CISL mechanism. As may be expected, application of the HAL increased the standard deviation for the transient phase, as robots attempted some unproductive actions during periods of extensive state space exploration. The CISL mechanism with the HAL demonstrated superior standard deviation for both Simulation Time and Total Effort in the stable phase. The AE mechanism bestowed consistent stability onto the CISL mechanism, reflecting the mechanism’s ability to select only the most productive actions for an advisee from the set of advisors.

Lastly, statistical significance can be considered. Due to the similarity of the performances, a statistically significant difference could not be found between CISL mechanism and the CISL with AE mechanism, even when considering a population of 50 simulations. It is hypothesized that with a larger number of simulations, statistical significance could be observed. A significant difference was found between CISL mechanism and the CISL with HAL mechanism when comparing 20 simulations.

Table 5: Paired p-values. Significant values are underlined. Marked (*) pairwise comparisons indicate statistical distinguishability.

<table>
<thead>
<tr>
<th></th>
<th>Simulation Time</th>
<th></th>
<th>Total Effort</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Transient Phase</td>
<td>Steady Phase</td>
<td>Transient Phase</td>
<td>Steady Phase</td>
</tr>
<tr>
<td></td>
<td>average p-value</td>
<td>% distinct runs</td>
<td>average p-value</td>
<td>% distinct runs</td>
</tr>
<tr>
<td>CISL vs. CISL-AE</td>
<td>0.2783</td>
<td>31</td>
<td>0.2654</td>
<td>31</td>
</tr>
<tr>
<td>(*) CISL vs. CISL-HAL</td>
<td>0.2001</td>
<td>58</td>
<td>0.0476</td>
<td>86</td>
</tr>
<tr>
<td>(*) CISL- AE vs. CISL-HAL</td>
<td>0.2209</td>
<td>54</td>
<td>0.1027</td>
<td>75</td>
</tr>
</tbody>
</table>

Both the HAL and the AE mechanism provided different benefits to the CISL mechanism. The HAL mechanism demanded additional explorative behaviour of agents, which consistently found superior behavioural policies for the multi-agent team. The AE mechanism
increased the efficiency of the learning process, enabling less effective agents to benefit from many superior agents’ utility functions, speeding convergence. In principle, the approaches are not mutually exclusive.
Chapter 6
Discussion and Conclusions

6.1 Discussion: The Behaviour of the Three CISL approaches

In the heterogeneous simulation scenario, the performance of the RL-Alliance mechanism is optimal for minimizing the agent effort, whereas the performance of the Concurrent Q-Learning mechanism is optimal for team performance. An interesting question to ask may be why the Single Q-Learning algorithm performed poorly, as in principle a single Q-Learning method should converge to similar performance metric values as all of the CISL approaches. A possible reason is that the task allocation actions have a much longer time horizon to generate reward, and thus may not have reachable global optimums on the scale of individual iterations; it may not be possible to represent long term rewards of the Task Allocation MDP alongside short-term rewards encoded within the Individual Performance MDP using the an identical alpha and decay parameters ($\alpha, \gamma$) and a single utility $Q$ function. Thus, a benefit of the CMDP modelling approach is the conferred ability to represent reward on multiple state space and temporal scales. Although empirically effective, the underlying CMDP model of the CISL mechanism is not without degeneracy. Thus, interesting further work can involve contrasting many CMDP models in various simulated scenarios, with and without degeneracy. Models that exhibit no degeneracy are typically simpler games, and are less realistic than experimental scenarios.

6.2 Discussion: CMDP Model Benefits

Converting a centralized MDP into a CMDP reduces the state space required to characterize all utility functions. To quantify scale of this space savings, a simple CMDP state reduction ratio can be expressed:

$$\frac{\sum_{t=1}^{k} c_t}{\prod_{t=1}^{k} c_t}$$

(69)
where the count variable $c_i$ represents the characteristic size of the state space for a dependent MDP within a $k$-CMDP, i.e., $c_i = \text{state\_count}(S_i)$. For example, the CISL problem model would be considered a 2-CMDP, and the count of states in states $S_i$ and $S_T$ can be expressed by simply assigning indexes to the state sets, $(S_i = S_1, S_T = S_2, \forall i c_i = \text{state\_count}(S_i))$. Countable state spaces are assumed for this demonstration. A significant reduction in state space can be achieved as the number of dependent MDPs ($k$) increases. To visualize the reduction in space an illustrative example can be considered:

A centralized MDP can be factored into a 3-CMDP, without degeneracy, with state counts, $\{c_1, c_2, c_3\} = \{10^4, 10^4, 10^4\}$. As a consequence of equation 69, this 3-CMDP model can represent all utility functions within $3 \cdot 10^4$ memory locations, $(\sum_{i=1}^{k} c_i)$. The corresponding centralized MDP, without compression or special state space design, requires a trillion state action pairs $(\prod_{i=1}^{k} c_i)$. Techniques that map large intractable state spaces to small tractable spaces can cause information loss, thus, the main benefit conferred by using a CMDP model is to potentially perform this mapping without degeneracy. Thus, the study of equation 69 can encourage the pursuit of reductive representations for multi-agent behaviour.

The CMDP model can be readily generalized to address many types of seemingly dissimilar multi-agent learning problems. Thus, the CISL approach can apply to any MDP that can be deconstructed into an Individual Performance MDP and a Task Allocation MDP.

### 6.3 Discussion: State Uncertainty

A state estimator was developed for a concurrent individual and social robot team learning algorithm, based on particle filters, which improved the team learning behaviour regardless of the noise level. The superior performance of the Robust CISL mechanism was examined and verified through a team foraging scenario exhibiting generous amounts of state uncertainty. It is expected that the RCISL mechanism would be able to handle sensor noise during real-life robot team experimentation. For instance, the localization variance using a Microsoft Kinect, for an absolute position observation $(x, y)$, can likely be found within a low zero-mean standard deviation of $(\sigma_x, \sigma_y) \leq (0.2m, 0.2m)$, and localization using a lidar sensor can perform with minimal standard deviation of $(\sigma_x, \sigma_y) \leq (0.02m, 0.02m)$ [7].
Future research can include hardware experimentation with the RCISL algorithm. In addition, such experimentations can include physical differences in the heterogeneous team, investigating the ability of the RCISL mechanism to account for various degenerate physical platforms. Lastly, and most significantly, considerations for actuation uncertainty and errors can be considered.

6.4 Discussion: Advice Sharing

In this thesis, it has been demonstrated that it is possible for reinforcement learning mechanism to utilize advice from both humans and peer agents. Additionally the advice was shown to support, not add degeneracies to, the quality of the regressed Individual Performance MDP utility functions.

It is noted that the Human Advice Layer, as an advisor, had a marked advantage over the Advice Exchange mechanism; the HAL mechanism has an underlying GMM which is trained on offline human training data, whereas the AE mechanism only has access to the current utility functions within the set of agents. Thus, interesting future work may consider both offline agent utility functions from previous experiments as a basis for the AE mechanism’s advice and online human advice as a basis for the HAL mechanism. These additions would yield two advice sharing mechanisms, which can both utilize online and offline examples from both humans and agents. Future directions might also expand the concept of advice sharing in an attempt exchange the value of MDP elements other than utility functions. For example, human advice can be used to help adjust state space or reward function representations.

Human advice benefited the performance of the reinforcement learning agents enormously, in terms of reducing both the Simulation Time and Total Effort metrics. In addition, the average amount of reward received from the Individual MDP reward function increased significantly. If these results generalize, even partly, to other domains and scenarios, then the practice of using human advice to augment autonomous regression can be seen as an undertreated domain in the literature; using reinforcement learning to derive autonomous agent behaviour is heavily visited, as is learning deep representations for state spaces and reward function spaces. It may be only a matter of time until both methods are heavily integrated.
6.5 Conclusion

It was shown in this thesis that a CMDP model for the representation of multi-agent decision making problems can be more efficient than a corresponding decentralized MDP with high dimensionality. In addition, the CMDP regression approach called the CISL mechanism, was both made robust yielding the RCISL mechanism and enhanced using two advice sharing mechanisms. The heterogeneous foraging case study demonstrated that the CISL mechanism was analytically tractable for large problems, robust to generous levels of state uncertainty, and socially compliant in terms of advice from both superior agents and human advisors.

Several CISL algorithms were developed and subsequently demonstrated as solutions for CMDP models. Through a heterogeneous foraging scenario the performance of these algorithms was demonstrated and contrasted with that of a control approach. The CISL approaches demonstrated superior performance when compared with the Single Q-Learning approach, and the three CISL learning algorithms themselves performed differently with respect to different metrics. For example, the RL-Alliance algorithm seemed to perform better in terms of Total Effort, whereas the Concurrent Q-Learning algorithm showed better results with respect to Simulation Time.

A step towards realism in simulation was considered through the inclusion of state uncertainty. The robustness of the CISL to small amounts of observation uncertainty was demonstrated. The addition of a state estimator to the mechanism yielded the RCISL mechanism, which functions near optimally despite copious amounts of state uncertainty. Thus, the CISL and RCISL mechanisms may be appropriate for usage in an experimental robotics scenario.

Both advice sharing mechanisms, Advice Exchange and the Human Advice Layer, conferred a benefit to the CISL mechanism. Compared with the CISL mechanism, the CISL with AE mechanism demonstrated both a marginal performance improvement and a reduction in standard deviation for all metrics. While the AE mechanism helps agents to homogenize their behaviours, it does not seem to encourage the best agents to learn at an accelerated rate, i.e., the AE mechanism is mainly a behavioural stabilizer. Conversely, advice from the HAL was introduced to dramatic effect; the CISL with HAL mechanism demonstrated enhanced performance in terms of Average Reward, Simulation Time and Total Effort metrics, which are
indicative of the superior utility functions of the agents, team foraging coordination, and reduced team effort, respectively. The HAL also increased the unpredictability of the learning process, leading agents to attempt “human” actions during learning, partially replacing their own nominal action selection policies. Thus, both the AE mechanism and the HAL benefited the CISL mechanism in different ways, calling for future work to both apply all advice sharing methods in unison and to deepen the integration of advice sharing further.

The CISL approaches and their presented modifications consider the importance of concurrency in MDP designs when formulating a policy regression approach. The CMDP model adds analytical rigor to multi-agent problems. In the future, theoretically and empirically solving multi-layer learning problems, (i.e, collective, cooperative, and collaborative,) with multi-layer learning mechanisms may be one direction toward the development of general robots that can learn from each other, within uncertain environments, from both peers and human advisors.
References


[45] B. L. Welch, “The generalization of Student's problem when several different population variances are involved,” *Biometrika*, 1947, Volume 34, Number 1, pp. 28–35.