

https://dspace.org/

Learning who to trust: policy learning in single-stage decision problems with unreliable expert advice

Love, Tamlin

2022
https://hdl.handle.net/10539/33893

Downloaded from a DSpace repository. https://dspace.org/

Learning Who to Trust: Policy Learning in Single-Stage Decision Problems with Unreliable Expert Advice

Tamlin Love
1438243

February, 2022

Supervised by Dr. Ritesh Ajoodha and Prof. Benjamin Rosman



Submitted to the School of Computer Science and Applied Mathematics, Faculty of Science, University of the Witwatersrand, Johannesburg, South Africa

Abstract

Work in the field of Assisted Reinforcement Learning (ARL) has shown that the incorporation of external information in problem solving can greatly increase the rate at which learners can converge to an optimal policy and aid in scaling algorithms to larger, more complex problems. However, these approaches rely on a single, reliable source of information; the problem of learning with information from multiple and/or unreliable sources of information is still an open question in ARL. We present CLUE (Cautiously Learning with Unreliable Experts), a framework for learning single-stage decision problems with policy advice from multiple, potentially unreliable experts. We compare CLUE against an unassisted agent and an agent that naïvely follows advice, and our results show that CLUE exhibits faster convergence than an unassisted agent when advised by reliable experts, but is nevertheless robust against incorrect advice from unreliable experts.

Portions of this work have appeared in the Workshop on Human-aligned Reinforcement Learning for Autonomous Agents and Robots as *Should I Trust You? Incorporating Unreliable Expert Advice in Human-Agent Interaction* [Love *et al.* 2021].

An implementation of the CLUE framework presented in this work is available at `https://anonymous.4open.science/r/CLUE_SSDP-4425`.

Declaration

University of the Witwatersrand, Johannesburg
School of Computer Science and Applied Mathematics
Senate Plagiarism Policy

I, Tamlin Love, (Student number: 1438243) am a student registered for MSc Dissertation (COMS8003A) in the years 2020 and 2021.

I hereby declare the following:

- I am aware that plagiarism (the use of someone else's work without their permission and/or without acknowledging the original source) is wrong.
- I confirm that ALL the work submitted for assessment for the above course is my own unaided work except where I have explicitly indicated otherwise.
- I have followed the required conventions in referencing the thoughts and ideas of others.
- I understand that the University of the Witwatersrand may take disciplinary action against me if there is a belief that this in not my own unaided work or that I have failed to acknowledge the source of the ideas or words in my writing.

Signature: 

Date: 17/02/2022

Contents

1 Introduction **6**
- 1.1 Introduction . . . 6
- 1.2 Contributions . . . 7
- 1.3 Structure . . . 8

2 Background and Related Work **9**
- 2.1 Reinforcement Learning and Single-Stage Decision Problems . . . 9
  - 2.1.1 Representing Single-Stage Decision Problems . . . 9
  - 2.1.2 Solving Single-Stage Decision Problems . . . 12
- 2.2 Assisted Reinforcement Learning . . . 13
- 2.3 Related Work . . . 15

3 Methodology **16**
- 3.1 Introduction . . . 16
  - 3.1.1 Assumptions . . . 17
- 3.2 Cautiously Learning with Unreliable Experts . . . 17
  - 3.2.1 Modelling Reliability . . . 18
  - 3.2.2 Making Decisions . . . 18
  - 3.2.3 Updating Reliability Estimates . . . 20
- 3.3 CLUE Framework . . . 23
- 3.4 Theoretical Analysis . . . 24

4 Experiments **28**
- 4.1 Experiment Set-Up . . . 29
  - 4.1.1 Environment . . . 29
  - 4.1.2 Agents . . . 29
  - 4.1.3 Experts . . . 30
- 4.2 Panel Compositions . . . 30
  - 4.2.1 Adversarial Advice . . . 32
  - 4.2.2 Comparison with Probabilistic Policy Reuse . . . 32
  - 4.2.3 Alternate Approach to Expert Simulation . . . 33
- 4.3 Reliability Estimates . . . 34
  - 4.3.1 Reliability Estimates Over Time . . . 34
  - 4.3.2 Prior Parameters . . . 36
- 4.4 Expert Parameters . . . 38
- 4.5 Hardware and Software Specifications . . . 39

5 Conclusion **40**
- 5.1 Future Work . . . 40

References **42**

Chapter 1

Introduction

1.1 Introduction

Single-Stage Decision Problems (SSDPs) are a type of Reinforcement Learning (RL) problem with a wide range of useful applications, including recommendation systems [Li *et al.* 2010], investment portfolio selection [Huo and Fu 2017] and clinical trials [Varatharajah *et al.* 2018]. For example, consider the problem of a doctor who can observe a patient's symptoms and medical history and must prescribe the right set of treatments to improve the patient's condition and avoid harmful side effects. A simple version of such a problem is represented in Figure 1.1. These types of problems have attracted research looking to augment the doctor with a software agent, with the long-term goal of making such diagnoses more comprehensive and widely available [Lauritzen and Spiegelhalter 1988; Heckerman and Nathwani 1992; Kao *et al.* 2018].

As another example, consider the scenario of a robot frail-care assistant, tasked with monitoring its patient and assisting in daily tasks. Suppose this robot has already learned how to optimally perform each individual task (e.g. mobility assistance, calling emergency services, dispensing medicine, etc.), but has yet to learn which tasks to perform in which situations, based on the observations it can make through its sensors (e.g. video footage, audio signal, time of day, etc.). In such a scenario, it is crucial for the robot to learn which tasks to perform for given observations, as there is a great deal of risk involved should the robot perform the wrong task. For example, if the patient has slipped and fallen, the correct response might be to call for help. If the robot does not perform these tasks, serious harm could come to the patient.

In both examples, it is important for the autonomous agent to learn the problem with as few cycles of observation and decision-making as possible, for a number of reasons. Primarily, these types of problems may be very complex, with a large space of possible observations and decisions. For example, the medical diagnosis problem may consist of hundreds of potential symptoms and treatments. Additionally, data acquisition may be difficult, either because the agent is acting in the real-world, thus potentially damaging itself and its surroundings, or because of ethical and safety issues, especially when dealing with human patients.

One approach to tackling this complexity and the need for sample efficiency is to incorporate external information in the learning process [Bignold *et al.* 2020]. For example, an autonomous medical diagnosis system could be advised by a doctor who instructs the agent to prescribe certain treatments in response to certain combinations of symptoms and medical history. Given the potential complexity, it may not always be feasible to elicit all of this information before learning starts. Instead, the human advisor can advise the agent as it learns, in response to its performance. Indeed, previous work has shown

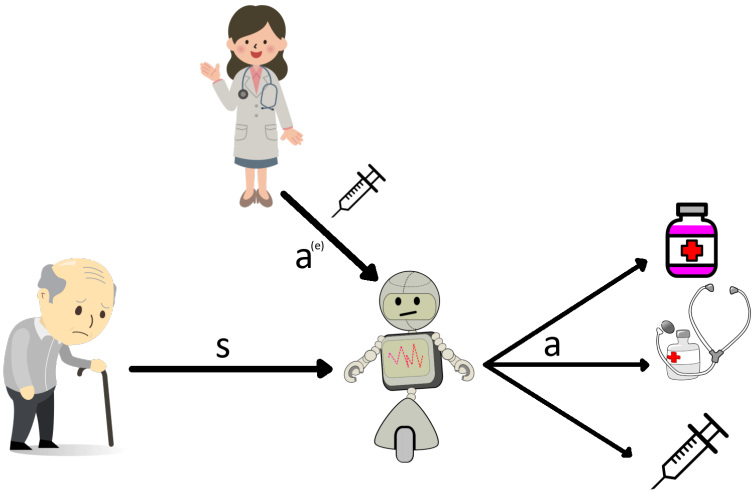


Figure 1.1: A simple example of the medical diagnosis problem, where an agent observes a patient and their symptoms and prescribes some set of treatments. Here, the agent is advised to perform a certain action by a human doctor.

that the interactive incorporation of expert advice can improve the rate at which an RL agent converges to a given performance threshold, provided that said advice is correct [Torrey and Taylor 2013].

It may be desirable to incorporate advice from multiple experts, either because a single expert does not have enough expertise to cover the full breadth of the problem, or simply because being able to incorporate more advice results in better sample efficiency [Shelton 2000]. For example, an agent learning the medical diagnosis problem could be advised by a whole panel of doctors, consisting of cardiologists, oncologists, etc. Incorporating advice from multiple experts introduces its own problems, however, when multiple experts offer conflicting advice for the same situation. Here the agent must decide which advice to follow and which to ignore. In general, expert advisors, especially humans, can give incorrect advice, either in error or through active malice [Efthymiadis *et al.* 2013]. Overcoming these problems has been identified as an open problem in the field of Assisted RL [Bignold *et al.* 2020].

In order to address these issues, we present CLUE, a framework for learning SSDPs with policy advice from multiple, potentially unreliable experts. We demonstrate that CLUE, when advised by reliable experts, converges faster than an equivalent agent that does not incorporate advice, but is robust to advice given by experts that may be unreliable to some degree.

1.2 Contributions

The contributions of this work are as follows.

- The CLUE framework for learning SSDPs with the policy advice of multiple, potentially unreliable experts, which can be integrated into the SSDP learning process over any existing learning algorithm (see Section 2.1.2). This consists of the following contributed components.
  - A model of the reliability of an expert advisor (Section 3.2.1).
  - A decision-making rule that uses the reliability model of each expert in a panel of experts to determine which advice, if any, to follow during exploration (Section 3.2.2). This decision-making rule is able to incorporate advice from multiple, potentially unreliable experts and

can handle contradictions in the advice given.

  - A Bayesian update rule that incorporates the agent's experience of the environment in updating the reliability model of each expert (Section 3.2.3).

- Theoretical analysis which shows the conditions under which a CLUE agent has a higher probability of acting optimally when exploring compared to some default exploration strategy in a simple environment (Section 3.4).
- Empirical results which demonstrate that CLUE:
  - can benefit from advice given by consistently reliable experts (Section 4.2).
  - is robust to advice given by consistently unreliable experts (Section 4.2).
  - can correctly rank multiple experts by their reliabilities in order to potentially benefit from advice given by multiple experts (Section 4.3).
  - can exploit information revealed by consensus and contradictions among multiple experts (Section 4.2), as well as consistently suboptimal advice (adversarial advice, Section 4.2.1), to improve performance.
  - is robust against choices of initial estimates of reliability (Section 4.3.2) and varying degrees of expert-agent interaction (Section 4.4).

1.3 Structure

The structure of this work is as follows. In Chapter 2, we discuss the concepts upon which this research is built, including Reinforcement Learning and Single-Stage Decision Problems in Section 2.1 and Assisted Reinforcement Learning in Section 2.2, as well as other approaches to solving the problems of multiple experts and unreliable advice in Section 2.3. In Chapter 3, we discuss the methodology of this work, including key definitions and assumptions in Section 3.1, the contributions of this work in Section 3.2, the CLUE framework and how it fits into the assisted reinforcement learning paradigm in Section 3.3, and theoretical analysis of the algorithm in a simple environment in Section 3.4.

In Chapter 4, we present a number of experiments that test the performance and functionality of CLUE, including a discussion of experimental set-up in Section 4.1, a comparison of different panels of experts in Section 4.2, an investigation into how CLUE estimates the reliability of experts in Section 4.3, and an investigation into the effects of varying degrees of interaction between the agent and experts in Section 4.4. A final summary of this work and directions for future research are provided in Chapter 5.

Chapter 2

Background and Related Work

2.1 Reinforcement Learning and Single-Stage Decision Problems

Reinforcement Learning (RL) is a field of machine learning in which decision-making entities, known as *agents*, learn how to interact with an environment in order to maximise some cumulative reward signal [Sutton and Barto 2018]. Of the many types of RL problems, this research concerns itself with *single-stage decision problems* (SSDPs), also known as *contextual bandits* [Langford and Zhang 2007], with discrete states and actions. In this setting, the agent observes some *state* $s \in S$ (Algorithm 1, line 3), selects some *action* $a \in A$ (Algorithm 1, line 4), and receives some *reward* or *utility* $r(s, a) \in \mathbb{R}$ from the environment (Algorithm 1, line 5). Each round of observation, action-selection and environment feedback is referred to as a *trial*, and each trial is independent from previous trials.

The medical diagnosis example from the previous chapter can be posed as an SSDP, with the set of observable symptoms and medical history forming the state space, the set of available treatments forming the action space, and the reward signal being a function of the patient's overall health, whether or not they have experienced negative side-effects, etc. The frail-care assistance robot example (also from the previous chapter) can also be posed as an SSDP, with each state being composed of the observations made by the robot. Unlike the medical diagnosis example, the action space here is made up of high-level strategies, rather than low level actions such as joint angles and motor velocities. In this example, the reward could be related to the well-being of the patient.

A *policy* $\pi : S \to A$ is a function that maps each state to an action, and the goal of an agent within an SSDP is to learn the *optimal policy*

$$\pi^* = \underset{\pi}{\operatorname{argmax}}\ EU(\pi(s)|s) \quad \forall s \in S,$$

where $EU(a|s)$ denotes the *expected utility* (i.e. *expected reward*) of choosing an action a in state s. The expected utility function is typically not given, and must be learned by the agent through its interactions with the environment (see Section 2.1.2).

The SSDP framework is provided in Algorithm 1.

2.1.1 Representing Single-Stage Decision Problems

Before we can address the problem of learning an optimal policy for an SSDP, it is important to discuss how SSDPs can be represented by software systems. Recall that an SSDP trial involves observing a state $s \in S$, selecting an action $a \in A$ and receiving a reward $r(s, a) \in \mathbb{R}$. Assuming that S and A are

Algorithm 1 Single Stage Decision Problem

procedure STANDARD_SSDP($environment, N$) ▷ N = number of trials
  for $t \in [0, ..., N-1]$ **do**
    $s_t \leftarrow$**sample_state**($environment$) ▷ environment picks state
    $a_t \leftarrow$**act**(s_t) ▷ agent picks action (e.g. Algorithm 2)
    $r_t \leftarrow$**execute_action**($a_t, environment$) ▷ environment returns reward
    learn(s_t, a_t, r_t) ▷ agent learns (e.g. Algorithm 2)

discrete, a simple approach to representing an SSDP is to create a table of size $|S| \times |A|$ which maps each state-action pair $\langle s, a \rangle$ to an expected utility $EU(a|s)$.

Such an approach is particularly useful for an agent attempting to solve an SSDP. The agent can maintain an estimate $Q(s, a) \approx EU(a|s)$, and with each trial t, can update the value for $Q(s_t, a_t)$ using the reward r_t [Sutton and Barto 2018]. Such an approach is less useful if one is simulating an SSDP environment, as this requires knowing the distribution of states and the tabular approach makes the assumption that all states are equally likely to occur.

A more complex approach to representing SSDPs involves *Bayesian networks* (BN), a type of probabilistic graphical model representing a set of variables $\mathcal{Z}$, consisting of a structure $\mathcal{G}$ and parameters Θ [Pearl 1988]. The structure $\mathcal{G}$ is a directed acyclic graph (DAG) whose vertices represent variables and whose directed edges represent conditional dependencies between these variables. Thus if the edge $(Z_i, Z_j) \in \mathcal{G}$, then Z_j is conditionally dependent on Z_i. More precisely, if $Pa^{\mathcal{G}}_{Z_i}$ denotes the parents of Z_i in $\mathcal{G}$ and $ND^{\mathcal{G}}_{Z_i}$ denotes the non-descendants of Z_i in $\mathcal{G}$, then Z_i is conditionally independent of $ND^{\mathcal{G}}_{Z_i}$ given $Pa^{\mathcal{G}}_{Z_i}$ [Koller and Friedman 2009]. The parameters $\theta_{Z_i} \in \Theta$ define the conditional probability distributions (CPDs) $\theta_{Z_i} = P(Z_i|Pa^{\mathcal{G}}_{Z_i})$.

A BN allows for a compact representation of the joint probability distribution over $\mathcal{Z}$ and for an easy visualisation of the conditional dependencies present between variables in $\mathcal{Z}$. An example BN, complete with graphical structure and CPDs (given as tables), is provided in Figure 2.1. In this example, the variables upon which each variable is conditionally dependent are explicitly clear (e.g. $P(WetGrass)$ can be determined given only the values of $Sprinkler$ and $Rain$).

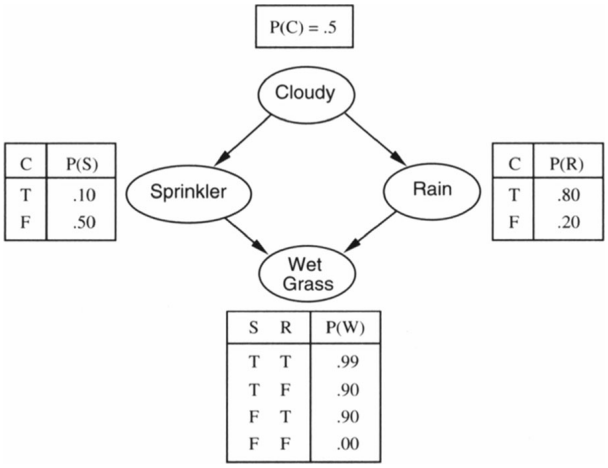


Figure 2.1: A famous example of a Bayesian network, illustrating the compact representation afforded by exploiting the conditional dependencies between variables [Norvig and Russell 1994].

An *influence diagram* (ID), alternately known as a *decision network*, is a type of Bayesian network

whose set of variables Z can be partitioned into three disjoint sets $\mathcal{Z} = \mathcal{X} \cup \mathcal{D} \cup \mathcal{U}$, where $\mathcal{X}$ denotes the set of *state* or *chance variables*, $\mathcal{D}$ denotes the set of *action* or *decision variables*, an assignment of which functions as an action, and $\mathcal{U}$ denotes the set of *reward* or *utility variables*, the sum of whose values adds up to reward returned by the environment [Howard and Matheson 2005; Koller and Friedman 2009].

For each action variable $D \in \mathcal{D}$, the parent set $Pa_D^{\mathcal{G}}$ denotes the observations made before deciding on a value for D. A common convention is to use edges between action variables to denote the order in which these variables are assigned values. For example, the edges $(D1, D2)$ and $(D2, D3)$ imply the ordering $D1 \rightarrow D2 \rightarrow D3$. For an ID to be used to represent an SSDP, it is required that only a single round of observation and decision-making occurs, and thus there cannot exist a state variable $X \in \mathcal{X}$ such that X has action variables as both descendants and ancestors. The set $\mathcal{B} \subseteq \mathcal{X}$ denotes the set of state variables observed before decision-making, an assignment of which comprises the state, whereas the set $\mathcal{O} \subseteq \mathcal{X}$ (which does not share any variables with $\mathcal{B}$), denotes the set of state variables observed after decision-making, which can serve to add noise to the final utility [Innes and Lascarides 2019].

In an ID, every state variable $X \in \mathcal{X}$ is associated with a CPD $\theta_X = P(X|Pa_X^{\mathcal{G}})$ and every utility variable $U \in \mathcal{U}$ is associated with a function $U(Pa_U^{\mathcal{G}})$ which maps to a real number [Koller and Friedman 2009]. The sum of each $U(Pa_U^{\mathcal{G}})$ is the total utility, which the agent seeks to maximise.

For example, a version of the medical diagnosis example, represented as an ID structure, is given in Figure 2.2. In this example, the state is composed of three variables in $\mathcal{B}$ (*RareCondition*, *Symptom1* and *Symptom2*), the action is composed of two variables in $\mathcal{D}$ (*Treatment1* and *Treatment2*), and the reward is composed of a single utility function in $\mathcal{U}$ (Reward), with a degree of stochasticity provided by the two variables in $\mathcal{O}$ (*SideEffect* and *PatientHealth*). The patient's health is dependent on the choice of treatments and the symptoms exhibited by the patient, thus *Symptom1*, *Symptom2*, *Treatment1* and *Treatment2* are all parents of *PatientHealth*. Similarly, the presence of a harmful side-effect is dependent on whether or not a particular treatment (represented by *Treatment1*) has been administered and whether or not the patient has a rare condition, and thus *RareCondition* and *Treatment1* are parents of *SideEffect*. The edge (*Treatment1*,*Treatment2*) indicates that *Treatment1* is assigned a value by the agent before *Treatment2*.

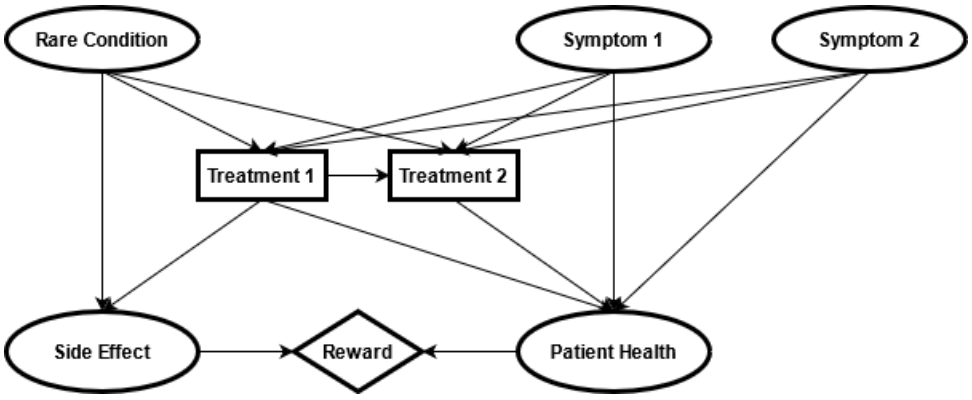


Figure 2.2: The structure of a simple version of the medical diagnosis example as an influence diagram. As per convention, oval nodes denote state variables, rectangular nodes represent action variables, and diamond-shaped nodes represent utility variables.

When used to simulate an environment, an ID can sample a new state by assigning values to each variable $B \in \mathcal{B}$ using the *forward sampling* algorithm, which operates by assigning values to each variable in $\mathcal{B}$ in succession, starting with root nodes (variables with no parents) and only assigning a

value to a variable if all of its parents have been assigned values [Koller and Friedman 2009]. Once the agent has assigned values to each $D \in \mathcal{D}$, forward sampling can be used again to assign values to the remaining variables in $\mathcal{O}$. The value of each utility variable in $\mathcal{U}$ can easily be determined using its corresponding utility function and the total utility can then be calculated as the sum of each value.

Given a $\mathcal{G}$, $\Theta_X \quad \forall X \in \mathcal{X}$ and $U(Pa_U^{\mathcal{G}}) \quad \forall U \in \mathcal{U}$, the *variable elimination* algorithm can be used to calculate the optimal policy, and thus the optimal assignments for each $D \in \mathcal{D}$, for each $s \in S$ [Zhang and Poole 1994]. The algorithm operates by working backwards in an ordering of action variables, and for each $D \in \mathcal{D}$, it finds the value that maximises the total utility for each possible assignment of $Pa_D^{\mathcal{G}}$. Using this new maximum factor, the algorithm can be recursively called until each action variable has an optimal value for each parent assignment. These factors can be used to construct an optimal decision function for each action variable, all of which comprise the optimal policy [Poole and Mackworth 2010].

In principle, an agent can maintain their own ID to model an environment, updating estimates for the structure and parameters of the ID as it interacts with the environment. However, learning a structure and associated CPDs is not a trivial task, and lies outside the scope of this research. Thus, in this work, IDs are used to simulate environments and provide "ground truth" models to simulated experts and true policy agents (see Sections 4.1.1, 4.1.2 and 4.1.3), while other agents use a tabular estimate $Q(s, a)$ to represent the environment (see Sections 2.1.2 and 4.1.2).

2.1.2 Solving Single-Stage Decision Problems

Having discussed how an agent might represent an SSDP, we now turn our attention to the problem of learning an optimal policy in order to maximise the reward obtained from interacting with an environment. As discussed in the previous section, an agent can represent an SSDP environment using a function $Q(s, a) \approx EU(a|s)$, which, in the context of learning to solve a single-stage decision function, is referred to as an *action-value function* [Sutton and Barto 2018].

Suppose in trial t the agent observes state s_t, selects action a_t and receives reward r_t. At the end of this trial, the agent can update the action-value function using the following update rule (see Algorithm 2)

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t - Q(s_t, a_t)), \tag{2.1}$$

where $\alpha \in (0, 1]$ is the *step-size parameter* and controls the rate at which the agent learns [Sutton and Barto 2018]. For stationary problems, α is typically assigned the value of $\frac{1}{k(s,a)}$, where $k(s, a)$ is the number of times the state-action pair (s, a) has been encountered. As this is always calculated at the end of a trial, $k(s, a) \geq 1$.

A common strategy to ensure that an agent's knowledge of the environment is sufficient to learn the optimal policy is *ϵ-greedy exploration*, in which the agent maintains a parameter $\epsilon \in [0, 1]$ and, after observing state s_t, with probability ϵ the agent selects an action from A with uniform random probability (known as "exploration", see Algorithm 2, line 4) [Sutton and Barto 2018]. Otherwise, with probability $1 - \epsilon$, the agent selects the action that maximises $Q(s_t, a)$ (known as "exploitation", see Algorithm 2, line 6). The value of ϵ can be fixed or can decay over the course of learning.

The action-value ϵ-greedy algorithm for SSDPs is outlined in Algorithm 2.

Other popular algorithms for solving SSDPs include LinUCB [Li *et al.* 2010], NeuralBandit [Allesiardo *et al.* 2014], and Contextual Thompson Sampling [Agrawal and Goyal 2013].

Algorithm 2 Baseline Approach for Solving SSDPs

1: **procedure** ACT(s_t,ϵ)
2: $p \leftarrow$**random**() ▷ random value in $[0, 1]$
3: **if** $p < \epsilon$ **then**
4: **return** random $a \in A$ ▷ agent "explores"
5: **else**
6: **return** $\underset{a}{\operatorname{argmax}}\, Q(s_t, a)$ ▷ agent "exploits"
7: **procedure** LEARN(s_t,a_t,r_t)
8: $Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(r_t - Q(s_t, a_t))$

2.2 Assisted Reinforcement Learning

As stated in Chapter 1, it is desirable for an SSDP algorithm to converge to the optimal policy π^* in as few trials as possible. This improved sample efficiency is especially desirable when problems are complex or when gathering data is difficult. One approach to improving convergence in this regard is to incorporate external information in the learning process. In this section, we discuss methods by which this incorporation of external information can be achieved.

The *assisted reinforcement learning* (ARL) framework encompasses a wide range of RL methods that incorporate information external to the environment in the learning process [Bignold *et al.* 2020]. Some examples of ARL approaches include *heuristic reinforcement learning* [Bianchi *et al.* 2004], *reinforcement learning from demonstration* [Taylor and Chernova 2010] and *transfer learning in reinforcement learning* [Taylor and Stone 2009]. Of particular relevance to this research is *interactive reinforcement learning* (IRL), in which an expert (either human of software-based) provides information to the agent during the learning process, usually as a response to the behaviour of the agent [Thomaz *et al.* 2005].

It is important to note that the ARL framework encompasses the full, episodic RL problem, whereas this work is only concerned with SSDPs. However, as SSDPs form a subset of RL problems, it is nevertheless useful to consider approaches to incorporating external information in the general RL case as these approaches can transfer to the specific case of SSDPs.

Bignold *et al.* [2020] identify several components across which different ARL methods can vary, as depicted in Figure 2.3. The first of these is the *information source*. In IRL approaches, this can be a human expert or a software expert that draws advice from either a "ground truth" model of the environment or a learned policy.

The second component is *temporality*, which refers to when and how often information is transferred from the information source to the agent. IRL methods are characterised by *interactive assistance*, in which information can be transferred multiple times throughout the learning process. When exactly an expert offers advice is dependent on the expert itself. In order to simulate the cost of communication, an expert may choose to budget advice by limiting the number of times information can be transferred [Torrey and Taylor 2013]. When budgeting advice, different strategies may be employed to maximise its usefulness. Such strategies include giving advice as early as possible, saving advice for more important states, or offering advice in response to suboptimal behaviour on the part of the agent [Torrey and Taylor 2013]. Advice for a given state can be given before the agent acts in that state or afterwards, as a corrective measure. Although both approaches are common, we focus on the latter in this work (see Algorithm 3, line 6), as this approach allows an expert to assess the performance of an agent when deciding whether or not to offer advice.

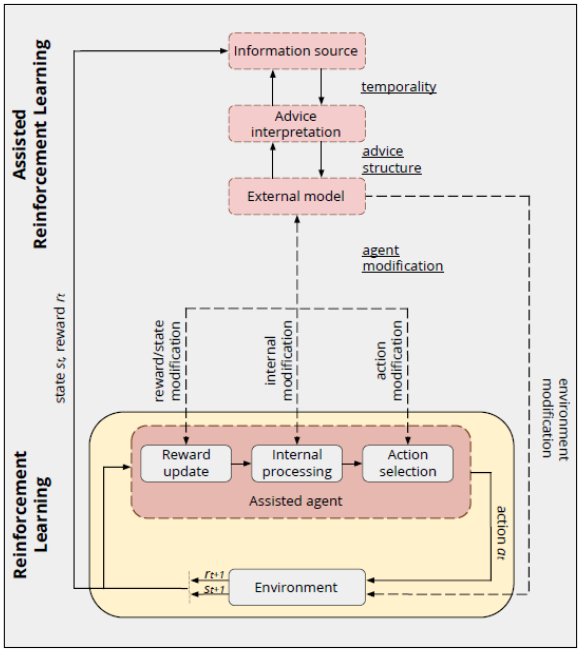


Figure 2.3: The components of the ARL framework [Bignold *et al.* 2020].

The *agent modification* component refers to the method by which the information source modifies the behaviour of the agent [Bignold *et al.* 2020]. IRL methods can be classified based on the type of advice the expert gives. In *reward-shaping* approaches [Knox and Stone 2009; Gimelfarb *et al.* 2018], the expert modifies the reward signal provided to the agent (e.g. by providing positive or negative feedback when the agent selects certain actions). In *policy-shaping* approaches [Fernández and Veloso 2006; Griffith *et al.* 2013], the expert modifies the agent's policy, typically by advising an action for a given state and having this action override the agent's policy whenever that state is encountered. Both approaches are preferred for different situations and domains. For this research, we focus on policy-shaping, as state-action advice can be more easily elicited from human experts in our domains of interest (particularly the medical diagnosis and frail-care assistance domains), requires minimal similarity between the agent and expert [Torrey and Taylor 2013], and is more robust to infrequent and inconsistent feedback [Griffith *et al.* 2013].

The *advice interpretation* and *advice structure* components refer to the structure of the advice and how it is understood by the agent. A common approach in policy-shaping IRL is to directly advise which actions are optimal for a given state [Griffith *et al.* 2013]. By advising the state-action pair $\langle s, a \rangle$, the expert is asserting that

$$EU(a|s) \geq EU(a'|s) \quad \forall a' \in A, \tag{2.2}$$

or, in other words, that a is the optimal action in state s.

Finally, the *external model* component may store information passed to the agent and model the information source [Bignold *et al.* 2020]. These models can be *retained*, meaning information is stored for later use and can be referred to by the agent, or *immediate*, meaning the information is used by the agent and then discarded.

The framework for learning SSDPs with interactive state-action advice (with advice being given after an agent acts) is provided in Algorithm 3, and can be compared to the standard SSDP loop presented in Algorithm 1. How the agent makes decisions and how it learns given the advice it receives varies

greatly between approaches.

Algorithm 3 Single Stage Decision Problem with Interactive Advice

procedure INTERACTIVE_SSDP($environment, N, e$) ▷ N = number of trials, e = expert
  for $t \in [0, ..., N-1]$ **do**
    $s_t \leftarrow$**sample_state**($environment$) ▷ environment picks state
    $a_t \leftarrow$**act**(s_t) ▷ agent picks action, potentially utilising earlier advice
    $r_t \leftarrow$**execute_action**($a_t, environment$) ▷ environment returns reward
    $a_t^{(e)} \leftarrow$**advise**(e, s_t, a_t, r_t) ▷ expert offers advice (or offers no advice)
    learn($s_t, a_t, r_t, a_t^{(e)}$) ▷ agent learns, potentially utilising advice

2.3 Related Work

Most (but not all) approaches in ARL assume the advice to be coming from a single, infallible expert. However, this assumption does not always hold, especially when the expert is human [Efthymiadis *et al.* 2013]. Suboptimal advice could be the result of communication error, erroneous domain knowledge or a malicious expert. Furthermore, incorporating advice from multiple experts introduces the possibility of two or more experts offering contradicting advice, requiring the agent to choose which advice is more likely to be correct [Shelton 2000]. The problems of incorporating advice from unreliable experts and incorporating advice from multiple experts are considered open questions in ARL [Bignold *et al.* 2020].

Several approaches deal with these problems in different ways. Gimelfarb *et al.* [2018] combine reward-shaping advice from multiple experts as a weighted sum of potential functions, where the weights are updated as the agent learns. Griffith *et al.* [2013] account for incorrect advice by modelling the probability of an expert giving correct advice (here in the form of a label of "right" or "wrong" rather than explicitly telling the agent what to do) with a single, static parameter $C \in [0, 1]$, where $C = 0$ corresponds to always giving suboptimal advice and $C = 1$ corresponds to always giving optimal advice. Both approaches are incompatible with the state-action advice we consider in this work. Nevertheless, these approaches contain elements which are applicable to the state-action advice we consider. The combination of advice weighted by the reliability of each expert forms the basis of the decision-making process outlined in Section 3.2.2, and the model of reliability as the probability of advising optimally is discussed in Section 3.2.1.

Fernández and Veloso [2006] account for potential unreliability in a transferred policy by initially relying on the policy with some probability ψ, and subsequently decaying ψ over time. Although the original approach considers advice given at the beginning of the learning process rather than during the process, this approach can be adapted to the setting this work considers. A comparison of performance with a variety of experts can be found in Section 4.2.2.

Other approaches that consider unreliable information (albeit with different temporalities and types of advice) include the Normalised Actor-Critic algorithm, an RL from demonstration approach which refines an initial policy obtained from potentially imperfect demonstrations [Gao *et al.* 2018], and the joint learning framework of Keswani *et al.* [2021], a classification algorithm, in which a classifier is learnt together with a deferrer which learns when to defer to one or more experts, which may have incorrect domain knowledge or biases.

Chapter 3

Methodology

In the previous chapter we discussed a number of key concepts relating to this work, as well as related work in the field of ARL. In this chapter, we present the contributions of this work and outline how the CLUE framework functions. In Section 3.1, we introduce the problem area and present important definitions and assumptions. In Section 3.2, we provide a high-level outline of CLUE and discuss each contribution in detail. In Section 3.3, we discuss how CLUE fits into the ARL framework described in Section 2.2. Finally, in Section 3.4, we present Theorem 1, which shows the conditions under which a CLUE agent will have increased probability of exploring optimally in a simple environment.

3.1 Introduction

As stated in the previous chapter, it is desirable for an SSDP algorithm to converge to the optimal policy with as few trials as possible, as this can aid in solving complex tasks, in situations where data acquisition is difficult or where safety and ethical concerns arise. In Section 2.2, we discussed Assisted Reinforcement Learning (ARL) - a family of approaches to tackling this problem which incorporate external information (such as advice given by an expert advisor) in the learning process. It has been shown that the rate at which agents learn can be improved when learning with the assistance of a reliable expert [Torrey and Taylor 2013]. Indeed, most ARL approaches assume this advice to be coming from a single, infallible expert. However, this assumption may not hold for all scenarios.

The aim of this research, therefore, is to devise an algorithm that can solve an SSDP with the advice from multiple, potentially unreliable experts. Such an algorithm should benefit from the advice of reliable experts, but be robust against incorrect advice.

In order to accomplish this, we introduce some key definitions. Recall from Equation 2.2 that the state-action advice $\langle s, a \rangle$ asserts that

$$EU(a|s) \geq EU(a'|s) \quad \forall a' \in A, \tag{3.1}$$

where $EU(a|s)$ denotes the *expected utility* (i.e. *expected reward*) from performing action a in state s, A denotes the action-space, and s is some state in S, where S denotes the state-space.

Definition 1. *The advice $\langle s, a \rangle$ is **correct** if it satisfies the inequality in Equation 3.1. Otherwise, it is **incorrect**.*

From this definition, we can define what it means for an expert to be reliable.

Definition 2. *An expert is **reliable** if it offers correct advice for all $s \in S$. Otherwise, the expert is **unreliable**.*

3.1.1 Assumptions

In order to tackle the problem of policy-learning in SSDPs with multiple, potentially unreliable experts, we make two key assumptions.

Assumption 1. *An expert is uniformly reliable across S.*

In other words, for any two states $s_1 \in S$ and $s_2 \in S$, the probability of an expert offering correct advice for either state is equal. In general, this assumption does not hold for all domains. For example, a doctor may specialise in one branch of medicine and they may be more reliable for states within their specialisation than for those outside of that region. However, relaxing this assumption requires dividing the state space into domains of expertise, which could range in size from a single state to the entire state space and could potentially overlap with each other. This division is non-trivial, and lies outside the scope of this research. Thus the problems considered in this research are assumed to be small enough in scope (but not necessarily in size) that a single expert could have expertise over the entire state space.

Assumption 2. *An expert is uniformly reliable across all trials.*

In other words, the probability that an expert gives correct advice does not change from trial to trial. This may not hold for all situations. For example, a human expert may get tired and start making more mistakes in later trials. A malicious expert may attempt to sabotage an agent's performance by giving it correct advice when such advice is less useful, and giving incorrect advice where stakes are high. However, in most cases where the expert is consistent and helpful, we expect the assumption to hold.

3.2 Cautiously Learning with Unreliable Experts

Having discussed important definitions and assumptions, we now present **CLUE** (**C**autiously **L**earning with **U**nreliable **E**xperts), an algorithm for learning SSDPs with the policy advice of multiple, potentially unreliable experts. CLUE involves three actors: an environment, an agent and a panel E of one or more experts. The high-level component view of CLUE is provided in Figure 3.1.

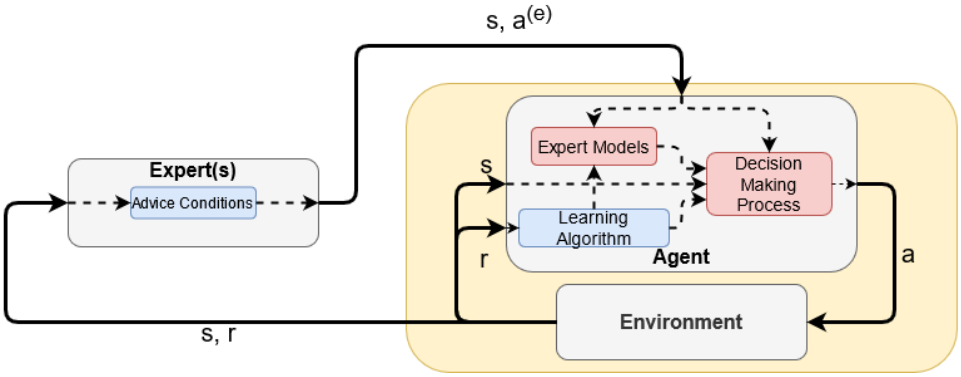


Figure 3.1: A high-level overview of CLUE, showing the interactions between the environment, the agent and expert(s). Components depicted in red represent contributions made by this research.

The environment is identical to any SSDP environment, as discussed in Section 2.1. Namely, for trial t, it samples state s_t, accepts action a_t from the agent and returns reward r_t. At the end of the trial, each expert e in panel E receives $\langle s_t, a_t, r_t \rangle$ and may offer their own advice, $\langle s_t, a^{(e)} \rangle$ on what action the agent should have taken this trial (Algorithm 6, line 6). How exactly an expert decides whether or not to offer advice and which advice to give differs between experts; our approach is outlined in Section 4.1.3 (Algorithm 7).

The agent is composed of three components which facilitate decision making and policy learning. The first of these components is a learning algorithm, which uses the information $\langle s_t, a_t, r_t \rangle$ to learn a policy. The CLUE framework allows for any SSDP learning algorithm to be used. In this work, we use an action-value method as discussed in Section 2.1.2 (Algorithm 2).

The second component, and one of the contributions of this research, is a model of the reliability of each expert (see Section 3.2.1). This model is necessary for learning which pieces of advice are to be followed and which are to be ignored. When an expert offers advice at the end of a trial, the agent uses its own information about the environment (such as a Q function) to evaluate the advice and update the model (see Section 3.2.3, Algorithm 5).

The third component, and another contribution of this research, is a decision making process which uses the information learned by the learning algorithm and the models of each expert to select an action for a state when exploring, given any advice it has previously received for that state (see Section 3.2.2, Algorithm 4).

Having provided a high-level description of CLUE, we now discuss each of the aforementioned contributions in greater detail.

3.2.1 Modelling Reliability

The first contribution we address is how an agent working within the CLUE framework models the reliability of each expert. Intuitively, we can think of an expert as being unreliable to some degree. For example, an expert that offers correct advice in 95% of trials, while still unreliable according to Definition 2, is more reliable than an expert that is always wrong.

Following Griffith *et al.* [2013], we model an expert's reliability, $\rho \in [0, 1]$, as the probability of the expert giving correct advice, where $\rho = 0$ corresponds to an expert whose advice is always wrong and $\rho = 1$ corresponds to a reliable expert. Rather than maintaining a static value for each expert, we can model a probability distribution of the value of ρ using a Beta distribution

$$Beta_\rho[\alpha, \beta] = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} \rho^{\alpha-1}(1-\rho)^{\beta-1}, \tag{3.2}$$

whose shape is determined by the parameters $\alpha, \beta > 0$ [Owen 2008]. These parameters can be thought of as counts, with α and β recording the number of times correct or incorrect advice was given respectively.

The best estimate of the reliability of an expert is therefore the expected value

$$\mathbb{E}[\rho] = \frac{\alpha}{\alpha + \beta}. \tag{3.3}$$

Thus, for each expert $e \in E$, the agent maintains a distribution $Beta_{\rho^{(e)}}[\alpha^{(e)}, \beta^{(e)}]$ from which the reliability estimate $\mathbb{E}[\rho^{(e)}]$ is calculated.

3.2.2 Making Decisions

We now turn our attention to the problem of how $\mathbb{E}[\rho]$ can inform the decision-making process. Suppose that, at the start of trial t, the agent observes state s_t and recalls any advice that some subset $E_t \subseteq E$ of experts offered for state s_t in trials $[0, ..., t-1]$. The agent decides whether to "exploit" - selecting an action according to its policy to maximise reward - or "explore" - selecting some other action to improve estimates of expected reward. If exploring, the agent must choose between randomly selecting an action

or following advice, in which case it must choose which advice to follow. By incorporating advice into exploration, but not exploitation, the agent can still benefit from the advice it receives while not baking this advice into its learned policy, thus allowing it to surpass the performance of an unreliable expert.

We note that there are three cases conditioned on $|E_t|$.

Case 1: $E_t = \varnothing$. In this case, no advice has been offered for s_t, such as will happen when $t = 0$, and thus the agent must act without any advice. The decision-making strategy employed in this research is ϵ-greedy exploration, as discussed in Section 2.1.2.

Case 2: $|E_t| = 1$. In this case, a single expert e has offered advice for s_t. As $\mathbb{E}[\rho^{(e)}]$ is the best estimate of the reliability of the expert, we employ it as a parameter in a similar vein to ϵ in ϵ-greedy methods [Sutton and Barto 2018] or ψ in the probabilistic policy reuse algorithm [Fernández and Veloso 2006], so that, with probability $\mathbb{E}[\rho^{(e)}]$ the agent follows the advice offered by expert e, and with probability $1 - \mathbb{E}[\rho^{(e)}]$ the agent acts as in Case 1. This allows for a trade-off between following advice and exploring as normal, where the former is more likely if the agent's estimate of the expert's reliability is higher.

Case 3: $|E_t| > 1$. In this case, multiple experts have offered (potentially conflicting) advice for s_t. A simple approach might be to pick the expert with the highest value of $\mathbb{E}[\rho^{(e)}]$ and ignore all others, thus reducing this case to Case 2. However, this approach eliminates the information that could be provided by other, less reliable experts, such as information revealed by consensus among experts (the "wisdom of the crowd" [Yi *et al.* 2012]) or the information provided by adversarial experts (experts who are almost always wrong, thus informing the agent which actions not to take).

To take advantage of the information provided by all experts, we instead employ a Bayesian approach to calculate the probability of each action being optimal given the available advice, inspired by similar approaches in crowd-sourced data labelling [Burke and Klein 2020] and potential-based reward shaping [Gimelfarb *et al.* 2018]. Let a^* denote the optimal action for state s_t, and $v_t^{(e)}$ denote the advice utterance $\langle s_t, a_t^{(e)} \rangle$ given by expert e for s_t, with V_t denoting the set $\{v_t^{(e)} | e \in E_t\}$. Thus, our aim is to calculate $P(a = a^* | V_t)$ for each $a \in A$. By Bayes' rule,

$$P(a = a^* | V_t) = \frac{P(V_t | a = a^*) P(a = a^*)}{\sum_{k=0}^{|A|} P(V_t | a_k = a^*) P(a_k = a^*)}. \tag{3.4}$$

We now assume that each expert gives advice independently of every other expert, i.e. $P(v_t^{(e_i)} \cap v_t^{(e_j)} | a = a^*) = P(v_t^{(e_i)} | a = a^*) P(v_t^{(e_j)} | a = a^*)$ for any arbitrary pair of experts e_i and e_j. Under this assumption, Equation 3.4 can be expressed as

$$P(a = a^* | V_t) = \frac{\prod_{e \in E_t} P(v_t^{(e)} | a = a^*) P(a = a^*)}{\sum_{k=0}^{|A|} \prod_{e \in E_t} P(v_t^{(e)} | a_k = a^*) P(a_k = a^*)},$$

which, under the assumption that the prior probability $P(a = a^*)$ is equal for all $a \in A$ (a reasonable assumption to make in the absence of a more informative prior), reduces to

$$P(a = a^* | V_t) = \frac{\prod_{e \in E_t} P(v_t^{(e)} | a = a^*)}{\sum_{k=0}^{|A|} \prod_{e \in E_t} P(v_t^{(e)} | a_k = a^*)}. \tag{3.5}$$

All that remains in order to calculate $P(a = a^* | V_t)$ is to determine the value of $P(v_t^{(e)} | a = a^*)$. As discussed in Section 3.2.1, $\mathbb{E}[\rho^{(e)}]$ is the probability that expert e offers correct advice for a given state.

Thus, under the assumption that if an expert does not advise a correct action, they select a suboptimal action in $A \setminus \{a^*\}$ with uniform probability [Masegosa and Moral 2013],

$$P(v_t^{(e)}|a = a^*) = \begin{cases} \mathbb{E}[\rho^{(e)}] & v_t^{(e)} \text{ advises } a \\ \frac{1-\mathbb{E}[\rho^{(e)}]}{|A|-1} & v_t^{(e)} \text{ does not advise } a \end{cases} \tag{3.6}$$

Substituting Equation 3.6 into Equation 3.5, we can calculate the probability of each action $a \in A$ being optimal, and set $a_{best} = \underset{a}{\operatorname{argmax}}\, P(a = a^*|V_t)$. As in Case 2, we use $P(a_{best} = a^*|V_t)$ as a parameter, selecting action a_{best} with probability $P(a_{best} = a^*|V_t)$ and acting as in Case 1 otherwise. Indeed, following this procedure for $|E_t| = 1$ results in an identical process to that outlined for Case 2, and thus we need only consider cases 1 and 3.

Of course, the above formulation assumes that $\mathbb{E}[\rho^{(e)}]$ accurately models the reliability of expert e, which may not always be the case, as discussed in Section 3.2.3. In particular, the over-estimation of the reliability of particularly unreliable experts may result in the over-selection of suboptimal actions. Erring on the side of caution, we introduce a threshold parameter $T \in [0, 1]$, such that if $P(a_{best} = a^*|V_t) < T$, the agent acts without advice. Thus the agent only follows advice if it is sufficiently confident that it is correct.

The decision-making process outlined above is presented as pseudocode in Algorithm 4. This can be compared and contrasted with the unassisted decision-making process provided in Algorithm 2.

3.2.3 Updating Reliability Estimates

Finally we discuss how each expert's reliability estimates are updated as they advise the agent and as the agent interacts with the environment. After selecting some action a_t, the agent receives reward r_t and some subset of experts offer their advice for state s_t. The learning algorithm then uses $\langle s_t, a_t, r_t \rangle$ to update its policy. The agent must now update $P(\rho^{(e)})$ for each expert (if any) that offered its advice for state s_t. Suppose expert e advises action $a^{(e)}$. Using the agent's own learned information (e.g. a Q function), it can estimate $EU(a|s_t)\ \forall a \in A$ to determine if $a^{(e)}$ is the optimal action for s_t. Early in training the agent's own understanding of the environment is limited, and so these evaluations will be poor. As the agent learns however, the accuracy of these evaluations will improve. Poor estimates may also be the result of violating the assumptions listed in Section 3.1.1.

Across t trials, with the advice of expert e having been evaluated $n^{(e)}$ times, let $x^{(e)}$ denote the number of optimal evaluations. Thus $n^{(e)} - x^{(e)}$ denotes the number of suboptimal evaluations. For ease of readability we omit the superscript denoting expert e. In order to update the reliability estimate, we wish to set the beta distribution $Beta_\rho[\alpha, \beta]$ to be equal to $P(\rho|x)$, which by Bayes' rule equals the following:

$$Beta_\rho[\alpha, \beta] = P(\rho|x) = \frac{P(x|\rho)P(\rho)}{\int_0^1 P(x|\rho)P(\rho)d\rho}. \tag{3.7}$$

As x and $n - x$ represent counts of correct and incorrect evaluations respectively, a natural choice is to model the likelihood $P(x|\rho)$ as a binomial distribution [Etz 2018]

$$P(x|\rho) = B_x[n, \rho] = \binom{n}{x}\rho^x(1-\rho)^{n-x}. \tag{3.8}$$

As we wish to model the posterior $P(\rho|x)$ as a beta distribution, we can model the prior $P(\rho)$ as a beta distribution

Algorithm 4 Acting with Advice from a Panel of Potentially Unreliable Experts

procedure ACT_WITH_ADVICE(s_t,t,T,ϵ,$\mathbb{E}[\rho^{(e)}]$)
  $p \leftarrow$**random**() ▷ random value in $[0, 1]$
  if $p < \epsilon$ **then** ▷ agent "explores"
    $E_t \leftarrow \{e | e$ advised for s_t in $\tau \in [0, ..., t-1]\}$
    if $|E_t| = \varnothing$ **then**
      return random $a \in A$
    else
      for $a \in A$ **do**
        $L_a \leftarrow 0$
        for $e \in E_t$ **do**
          if expert e advised (s_t, a) **then**
            $L_a \leftarrow L_a \times \mathbb{E}[\rho^{(e)}]$
          else
            $L_a \leftarrow L_a \times \frac{1-\mathbb{E}[\rho^{(e)}]}{|A|-1}$
      for $a \in A$ **do**
        $P(a = a_*) \leftarrow \frac{L_a}{\sum_{i=0}^{|A|} L_{a_i}}$
      $a_{best} \leftarrow \arg\max_a P(a = a_*)$
      if $P(a_{best} = a_*) < T$ **then**
        return random $a \in A$ ▷ act randomly
      else
        $q \leftarrow$**random**()
        if $q < P(a_{best} = a_*)$ **then**
          return a_{best} ▷ select the best action given received advice
        else
          return random $a \in A$ ▷ act randomly
  else ▷ agent "exploits"
    return $\arg\max_a Q(s_t, a)$

$$P(\rho) = Beta_\rho[\alpha_0, \beta_0] = \frac{\Gamma(\alpha_0 + \beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)} \rho^{\alpha_0 - 1}(1-\rho)^{\beta_0 - 1}, \tag{3.9}$$

which is conjugate to the binomial likelihood $P(x|\rho)$ [Etz 2018]. The prior parameters α_0 and β_0 can be thought of as prior counts of x and $n - x$ respectively. Therefore, α_0 may be thought of as the number of times (prior to the start of training) that the expert's advice was evaluated to be optimal, and β_0 the number of times the expert's advice was evaluated to be suboptimal.

Substituting Equations 3.8 and 3.9 into Equation 3.7, we arrive at

$$\begin{aligned} P(\rho|x) &= \frac{B_x[n,\rho] Beta_\rho[\alpha_0, \beta_0]}{\int_0^1 B_x[n,\rho] Beta_\rho[\alpha_0, \beta_0] d\rho} \\ &= \frac{\binom{n}{x}\rho^x(1-\rho)^{n-x}\frac{\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)}\rho^{\alpha_0-1}(1-\rho)^{\beta_0-1}}{\int_0^1 \binom{n}{x}\rho^x(1-\rho)^{n-x}\frac{\Gamma(\alpha_0+\beta_0)}{\Gamma(\alpha_0)\Gamma(\beta_0)}\rho^{\alpha_0-1}(1-\rho)^{\beta_0-1} d\rho} \\ &= \frac{\rho^{x+\alpha_0-1}(1-\rho)^{n-x+\beta_0-1}}{\int_0^1 \rho^{x+\alpha_0-1}(1-\rho)^{n-x+\beta_0-1} d\rho}. \end{aligned} \tag{3.10}$$

Now, because a beta distribution is a valid probability distribution,

$$\begin{aligned} 1 &= \int_0^1 Beta_\rho[\alpha, \beta] d\rho \\ &= \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} \rho^{\alpha-1}(1-\rho)^{\beta-1} d\rho, \end{aligned}$$

and thus

$$\int_0^1 \rho^{\alpha-1}(1-\rho)^{\beta-1} d\rho = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)}. \tag{3.11}$$

Substituting Equation 3.11 into Equation 3.10,

$$\begin{aligned} P(\rho|x) &= \frac{\rho^{x+\alpha_0-1}(1-\rho)^{n-x+\beta_0-1}}{\frac{\Gamma(x+\alpha_0)\Gamma(n-x+\beta_0)}{\Gamma(x+\alpha_0+n-x+\beta_0)}} \\ &= \frac{\Gamma(n+\alpha_0+\beta_0)}{\Gamma(x+\alpha_0)\Gamma(n-x+\beta_0)} \rho^{x+\alpha_0-1}(1-\rho)^{n-x+\beta_0-1} \\ &= Beta_\rho[x+\alpha_0, n-x+\beta_0] \end{aligned} \tag{3.12}$$

Substituting the parameters of Equation 3.12 into Equation 3.3, we can calculate the expected value

$$\mathbb{E}[\rho] = \frac{x+\alpha_0}{n+\alpha_0+\beta_0}.$$

Thus, as the agent encounters states for which expert e has given advice, it need only update $n^{(e)}$ and $x^{(e)}$ and recompute $\mathbb{E}[\rho^{(e)}]$, which can be used in future decision-making.

The update process provided above is presented as pseudocode in Algorithm 5.

Algorithm 5 Updating Unreliability Estimates

procedure UPDATE_ESTIMATES(s_t,a_t,r_t,V_t)
  $E'_t \leftarrow \{e | e$ advised for s_t in $\tau \in [0, ..., t]\}$
  for $e \in E$ **do**
    if e advised some action in trial t **then**
      store_advice(e,s_t,$v_t^{(e)}$)
    if $e \in E'_t$ **then**
      $best_reward \leftarrow \max_a EU(s_t, a)$
      $advice_reward \leftarrow EU(s_t, v_t^{(e)})$
      if $advice_reward \geq best_reward$ **then**
        $\alpha_{t+1}^{(e)} \leftarrow \alpha_t^{(e)} + 1$
        $\beta_{t+1}^{(e)} \leftarrow \beta_t^{(e)}$
      else
        $\alpha_{t+1}^{(e)} \leftarrow \alpha_t^{(e)}$
        $\beta_{t+1}^{(e)} \leftarrow \beta_t^{(e)} + 1$
      $\mathbb{E}[\rho_{t+1}^{(e)}] \leftarrow \frac{\alpha_{t+1}^{(e)}}{\alpha_{t+1}^{(e)} + \beta_{t+1}^{(e)}}$

3.3 CLUE Framework

Having discussed each of the core components of the CLUE framework in the previous sections, we now present the CLUE framework in Algorithm 6, which can be viewed as an extension of the interactive reinforcement learning (IRL) SSDP algorithm (Algorithm 3) that allows for multiple, potentially unreliable experts through the Bayesian methods of decision-making (Algorithm 4) and reliability estimation (Algorithm 5).

Algorithm 6 Cautiously Learning with Unreliable Experts

1: **procedure** CLUE($environment, E, N$) ▷ E = panel of experts, N = number of trials
2: **for** $t \in [0, ..., N-1]$ **do**
3: $s_t \leftarrow$**sample_state**($environment$) ▷ environment picks state
4: $a_t \leftarrow$**select_action**(s_t) ▷ Algorithm 4
5: $r_t \leftarrow$**execute_action**($a_t, environment$) ▷ environment returns reward
6: $advice_t \leftarrow$**panel_advise**(E, s_t, a_t, r_t) ▷ e.g. Algorithm 7
7: **learn**(s_t, a_t, r_t) ▷ e.g. Algorithm 2
8: **update_estimates**($s_t, a_t, r_t, advice_t$) ▷ Algorithm 5

Although the CLUE framework as presented in this chapter is only equipped for SSDPs and not full, episodic RL problems, it can nevertheless be analysed through the lens provided by the ARL framework of Bignold *et al.* [2020] as outlined in Section 2.2, and can be compared to other IRL methods. Such an analysis may prove useful in future work extending the CLUE framework to solve full RL problems (see Section 5.1).

In contrast to most other IRL approaches, the information source in the CLUE framework can consist of multiple experts, each of which is not assumed to be reliable (although they are assumed to be consistent, as discussed in Section 3.1.1). Due to the policy-shaping, state-action advice adopted in this work, the experts advising the agent need only share a common action-space, and can vary in how they represent the environment. Both human and software-based experts can be employed in the CLUE framework.

The temporality of the CLUE framework is typical of IRL methods, as advice is given interactively by the experts during learning [Thomaz *et al.* 2005]. As stated in Section 2.2, in this work we adopt the approach of the experts advising the agent after it acts, so that the experts can assess the performance of the agent before deciding whether or not to offer advice. In general however, experts can also offer advice before the agent acts without requiring any significant modifications to the CLUE framework.

The advice interpretation and structure components of the ARL framework are also typical of policy-shaping IRL approaches [Bignold *et al.* 2020]. Namely, each expert offers state-action advice in the form described by Equation 3.1. The advice is given in a form readable by the agent (e.g. the index of an action in A, an assignment of action variables in an influence diagram, etc.).

As with many other IRL approaches, a CLUE agent maintains a retained model of the information source; storing all advice it receives for later reference. However, where it differs from other approaches is that the agent maintains a separate model for each expert. These models not only store all advice offered by each expert, but also maintain an estimate of the experts' reliabilities, as described in Sections 3.2.1 and 3.2.3.

The type of agent modification exhibited by the CLUE framework is policy-shaping, which alters the agent's behaviour rather than the reward signal it receives or directly modifying internal components. A CLUE agent alters its behaviour when exploring by calculating the probability of each action being

optimal given the advice it has received and its estimates of experts' reliabilities, and, with some degree of randomness, selects the action with the highest probability, as described in Section 3.2.2.

3.4 Theoretical Analysis

In this section, we show the conditions for which CLUE, when exploring, will have a higher probability of selecting the optimal action for a given state than some default unassisted exploration strategy, when acting in an environment where $|A| = 2$ and being advised by a single expert that operates under the following assumption.

Assumption 3. *Assume an expert has a true reliability $\rho_{true} \in [0, 1]$ such that, when giving advice, it advises the optimal action with probability ρ_{true} and otherwise advises some suboptimal action with probability $\frac{1-\rho_{true}}{|A|-1}$.*

To do this, we define a function $W(a)$ which represents the probability of selecting a given action when exploring. For example, if selecting actions with uniform random probability, $W(a) = \frac{1}{|A|}\ \forall a \in A$. We show the values of $\mathbb{E}[\rho]$ (the agent's estimate of the reliability, which may or may not be accurate) for which the probability of selecting the optimal action a^* is greater than or equal to $W(a^*)$, given W and ρ_{true}.

To aid in the proof of this theorem (Theorem 1), we first prove Lemma 1, which shows the conditions for which a CLUE agent is guaranteed to identify a given action as optimal.

Lemma 1. *Suppose an environment with $|A| = 2$ and a panel consisting of a single expert. Let $\mathbb{E}[\rho]$ denote the agent's estimate of the reliability of the expert. For any given state the expert has advised for, the optimal action a^* will be identified as such by the agent if one of the following holds*

- *The expert advised a^* and $\mathbb{E}[\rho] > \frac{1}{2}$*
- *The expert did not advise a^* and $\mathbb{E}[\rho] < \frac{1}{2}$*

If $\mathbb{E}[\rho] = \frac{1}{2}$, the agent is equally as likely as not to identify a^* as the optimal action.

Proof. From Equation 3.5, we have that

$$P(a_j = a^*|V_t) = \frac{\prod_{e \in E_t} P(v_t^{(e)}|a_j = a^*)}{\sum_{k=0}^{|A|} \prod_{e \in E_t} P(v_t^{(e)}|a_k = a^*)}, \tag{3.13}$$

which, given that $|E_t| = 1$ and $|A| = 2$, reduces to

$$P(a_j = a^*|V_t) = \frac{P(v_t^{(e)}|a_j = a^*)}{P(v_t^{(e)}|a_0 = a^*) + P(v_t^{(e)}|a_1 = a^*)}. \tag{3.14}$$

Without loss of generality, let a_0 denote the optimal action for s_t. Substituting in Equation 3.6, Equation 3.14 is equal to

$$\begin{aligned} P(a_j = a^*|V_t) &= \frac{P(v_t^{(e)}|a_j = a^*)}{\mathbb{E}[\rho] + 1 - \mathbb{E}[\rho]} \\ &= P(v_t^{(e)}|a_j = a^*), \end{aligned} \tag{3.15}$$

which is equal to $\mathbb{E}[\rho]$ if the expert advised a_j and $1 - \mathbb{E}[\rho]$ otherwise. Let a_{best} denote the action that maximises $P(a_j = a^*|V_t)$.

We consider 2 cases.

Case 1: The expert has advised a_0. Thus,

$$P(a_0 = a^*|V_t) = \mathbb{E}[\rho]$$
$$P(a_1 = a^*|V_t) = 1 - \mathbb{E}[\rho].$$

$P(a_0 = a^*|V_t) > P(a_1 = a^*|V_t)$ is therefore only true when $\mathbb{E}[\rho] > \frac{1}{2}$, and thus the agent will identify a_0 as the optimal action if $\mathbb{E}[\rho] > \frac{1}{2}$. If $\mathbb{E}[\rho] = \frac{1}{2}$, the agent will do so with probability $\frac{1}{2}$.

Case 2: The expert has advised a_1. Thus,

$$P(a_0 = a^*|V_t) = 1 - \mathbb{E}[\rho]$$
$$P(a_1 = a^*|V_t) = \mathbb{E}[\rho].$$

$P(a_0 = a^*|V_t) > P(a_1 = a^*|V_t)$ is therefore only true when $\mathbb{E}[\rho] < \frac{1}{2}$, and thus the agent will identify a_0 as the optimal action if $\mathbb{E}[\rho] < \frac{1}{2}$. If $\mathbb{E}[\rho] = \frac{1}{2}$, the agent will do so with probability $\frac{1}{2}$. □

Theorem 1. *Suppose an environment with $|A| = 2$ and a panel consisting of a single expert. Let $W(a)$ denote the probability of selecting action a when exploring unassisted. Then the probability of a CLUE agent selecting the optimal action a^* when exploring is greater than or equal to $W(a^*)$ if one of the following holds*

- $\mathbb{E}[\rho] = \frac{1}{2}$ *and* $W(a^*) \leq \frac{1}{2}$
- $\mathbb{E}[\rho] < \frac{1}{2}$ *and* $W(a^*) \leq 1 - \rho_{true}$
- $\mathbb{E}[\rho] > \frac{1}{2}$ *and* $W(a^*) \leq \rho_{true}$

Proof. Let $P(a)$ denote the probability of selecting action a. Let $a^{(e)}$ denote the action advised by the expert. From the decision-making process described in Section 3.2.2, we have that

$$P(a^*) = \mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*) \qquad \text{if } a_{best} = a^* \tag{3.16}$$
$$P(a^*) = (1 - \mathbb{E}[\rho])W(a^*) \qquad \text{if } a_{best} \neq a^* \tag{3.17}$$

We consider three cases.

Case 1: Let $\mathbb{E}[\rho] = \frac{1}{2}$. From Equations 3.16 and 3.17, we have that

$$P(a^*) = \frac{1}{2} + \frac{1}{2}W(a^*) \qquad \text{if } a_{best} = a^*$$
$$P(a^*) = \frac{1}{2}W(a^*) \qquad \text{if } a_{best} \neq a^*$$

From Lemma 1, $P(a_{best} = a^*) = \frac{1}{2} = P(a_{best} \neq a^*)$ and thus

$$\begin{aligned} P(a^*) &= \frac{1}{2}(\frac{1}{2} + \frac{1}{2}W(a^*)) + \frac{1}{2}(\frac{1}{2}W(a^*)) \\ &= \frac{1}{4} + \frac{1}{2}W(a^*), \end{aligned}$$

which is greater than or equal to $W(a^*)$ if and only if $W(a^*) \leq \frac{1}{2}$.

Case 2: Let $\mathbb{E}[\rho] < \frac{1}{2}$. From Equations 3.16 and 3.17, and from Lemma 1, we have that

$$P(a^*) = (1 - \mathbb{E}[\rho])W(a^*) \qquad a^{(e)} = a^*$$
$$P(a^*) = \mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*) \qquad a^{(e)} \neq a^*$$

From Assumption 3, $P(a^{(e)} = a^*) = \rho_{true}$ and $P(a^{(e)} \neq a^*) = 1 - \rho_{true}$. Therefore,

$$\begin{aligned} P(a^*) &= P(a^{(e)} = a^*)(1 - \mathbb{E}[\rho])W(a^*) + \\ &\quad P(a^{(e)} \neq a^*)(\mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*)) \\ &= \rho_{true}(1 - \mathbb{E}[\rho])W(a^*) + \\ &\quad (1 - \rho_{true})(\mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*)) \\ &= \mathbb{E}[\rho](1 - \rho_{true}) + W(a^*)(1 - \mathbb{E}[\rho]), \end{aligned}$$

which is greater than or equal to $W(a^*)$ if and only if $\mathbb{E}[\rho](1 - W(a^*) - \rho_{true}) \geq 0$. As $\mathbb{E}[\rho] \geq 0$, it is sufficient to prove that $1 - W(a^*) - \rho_{true} \geq 0$.

$$\begin{aligned} 1 - W(a^*) - \rho_{true} &\geq 0 \\ -W(a^*) &\geq \rho_{true} - 1 \\ W(a^*) &\leq 1 - \rho_{true} \end{aligned}$$

Thus $P(a^*) \geq W(a^*)$ if and only if $W(a^*) \leq 1 - \rho_{true}$.

Case 3: Let $\mathbb{E}[\rho] > \frac{1}{2}$. From Equations 3.16 and 3.17, and from Lemma 1, we have that

$$\begin{aligned} P(a^*) &= \mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*) && a^{(e)} = a^* \\ P(a^*) &= (1 - \mathbb{E}[\rho])W(a^*) && a^{(e)} \neq a^* \end{aligned}$$

From Assumption 3, $P(a^{(e)} = a^*) = \rho_{true}$ and $P(a^{(e)} \neq a^*) = 1 - \rho_{true}$. Therefore,

$$\begin{aligned} P(a^*) &= P(a^{(e)} = a^*)(\mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*)) + \\ &\quad P(a^{(e)} \neq a^*)(1 - \mathbb{E}[\rho])W(a^*) \\ &= \rho_{true}(\mathbb{E}[\rho] + (1 - \mathbb{E}[\rho])W(a^*)) + (1 - \rho_{true})(1 - \mathbb{E}[\rho]), \end{aligned}$$

which is greater than or equal to $W(a^*)$ if and only if $\mathbb{E}[\rho](\rho_{true} - W(a^*)) \geq 0$. As $\mathbb{E}[\rho] \geq 0$, $P(a^*) \geq W(a^*)$ if and only if $W(a^*) \leq \rho_{true}$.

□

Theorem 1 shows that whether or not the probability of selecting the optimal action using CLUE is higher than the probability of selecting the optimal action using some default exploration strategy W is dependant on the true reliability ρ_{true}, the agent's estimate of the reliability $\mathbb{E}[\rho]$ and the exploration strategy W itself.

For a uniform random exploration strategy ($W(a) = \frac{1}{|A|}\ \forall a \in A$), the implication of Theorem 1 is that there will be improved performance provided that the estimate $\mathbb{E}[\rho]$ is on the same side of $\frac{1}{2}$ as the true reliability ρ_{true}. For another exploration strategy, the improvement may increase or decrease depending on the probability of selecting a^* under that strategy.

Ultimately, Theorem 1 demonstrates that CLUE is a more effective exploration strategy provided that $\mathbb{E}[\rho]$ is sufficiently close to ρ_{true} (see Section 4.3 for empirical results for the convergence of $\mathbb{E}[\rho]$), given Assumption 3 and $|A| = 2$.

In this chapter we discussed the problem of incorporating advice from multiple, potentially unreliable experts in the SSDP learning process, introduced the CLUE framework as a means of addressing this problem, including a decision-making process for exploration and a means of modelling and updating reliability estimates of experts, discussed how CLUE fits within the broader ARL framework of

Bignold *et al.* [2020], and showed the theoretical conditions for CLUE having a higher probability of acting optimally when exploring, given a particular model of expert and a simple action space.

All in all, the CLUE framework represents a novel approach to addressing two open questions in ARL - the incorporation of multiple experts and the possibility of those experts being unreliable - within the sub-problem of single-stage decision problems. The goal of this framework is to have an agent that **1)** can benefit from advice from reliable experts, **2)** is robust to advice from unreliable experts and **3)** can exploit information revealed by consensus and contradictions among multiple experts to better determine the optimality of actions.

In the next chapter, we present a number of experiments to demonstrate that the CLUE framework presented in this chapter has satisfied these goals and to investigate the effects of various parameters on the functionality of the algorithm.

Chapter 4

Experiments

In the previous chapter we presented the CLUE framework for learning single-stage decision problems (SSDPs) with the advice of multiple, potentially unreliable experts. In particular, we presented a decision-making strategy that employs Bayes' rule to pool all available advice to calculate the action with the highest probability of being optimal for a given state, and a Bayesian method of updating the reliability estimate of each expert.

In this chapter, we present a number of experiments to demonstrate that the CLUE framework satisfies the following properties:

1. When advised by at least one reliable (or nearly reliable) expert, a CLUE agent converges to a given performance threshold faster than an unassisted agent with an identical learning algorithm, thereby benefiting from correct advice.

2. When advised by unreliable experts that are likely to offer incorrect advice, a CLUE agent asymptotically converges to the same threshold of performance achieved by an equivalent unassisted agent, thereby being robust against incorrect advice.

3. When advised by multiple experts, a CLUE agent correctly ranks each expert by the degree to which each is reliable, thereby identifying which experts are more likely to offer correct advice.

Additionally, these experiments aim to show the effect of each hyperparameter of the CLUE framework and to demonstrate that these hyperparameters need not be finely tuned for the above properties to hold.

In Section 4.1, we detail how the environments with which the agents interact are generated (Section 4.1.1), the logic and parameters of each agent are tested (Section 4.1.2), and how experts are simulated (Section 4.1.3). In Section 4.2 we compare the performance of each agent with different compositions of expert panels, in order to determine how CLUE handles experts of different reliabilities. Within this section, we also demonstrate how CLUE can benefit from adversarial advice (Section 4.2.1), provide a comparison with the π-reuse and PRQ algorithms of Fernández and Veloso [2006] (Section 4.2.2), and show that CLUE performs well when working with an alternate simulation of experts that behaves differently from those presented in Section 4.1.3 (Section 4.2.3). In Section 4.3, we examine how the reliability estimates change over time when interacting with different experts (Section 4.3.1) and how the initial beta parameters affect the performance of the CLUE agent (Section 4.3.2). Finally, in Section 4.4, we examine the effects of varying degrees of agent-expert interaction on the performance of CLUE, and in Section 4.5, we detail the hardware and software used in these experiments.

4.1 Experiment Set-Up

4.1.1 Environment

To simulate an SSDP environment of the form discussed in Section 2.1, we use an influence diagram (ID) to represent the state- and action-spaces, utility function and the conditional probability distributions governing the distribution of states, as discussed in Section 2.1.1. To show that the performance of CLUE generalises, we aggregate results across multiple, randomly generated ID environments.

Each state variable $X \in \mathcal{X}$ and each action variable $D \in \mathcal{D}$ have a binary domain $\{0, 1\}$, so that $|S| = 2^{|\mathcal{X}|}$ and $|A| = 2^{|\mathcal{D}|}$. In order to ensure that each ID represents a well-formed SSDP, we restrict the graph structure to a directed acyclic graph and ensure that all state nodes are parents of action nodes (so that the problem is fully observable), all action nodes are parents of the reward node (so that all actions have some effect on the reward), no action nodes are descendants of another action node (so that only a single round of decision-making occurs), and that no state nodes are children of an action node (as states are observed before decision-making). Rewards are scaled between -1 and 1, so that results across environments are easily comparable. Some randomly generated ID environments ($|\mathcal{X}| = 3$, $|\mathcal{D}| = 2$) are presented in Figure 4.1.

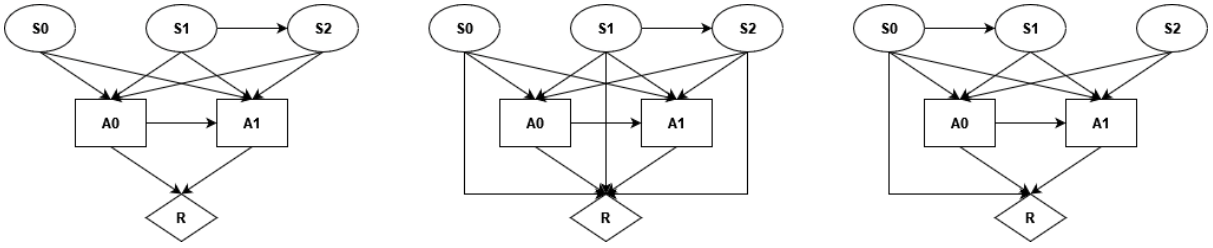


Figure 4.1: Examples of randomly generated ID environments for $|\mathcal{X}| = 3$, $|\mathcal{D}| = 2$.

In implementing these ID environments, we employ a modified version of the ID implementation provided by Poole and Mackworth [2017].

4.1.2 Agents

We compare a number of agents in these experiments. The *True Policy Agent* always selects the optimal action for any state, thus representing the upper-bound performance any agent can achieve. It calculates the optimal policy by applying the variable elimination algorithm (see Section 2.1.1) on the ID representing the environment.

The *Baseline Agent* is an action-value learner with ϵ-greedy exploration, as outlined in Section 2.1.2. The action-value function is initialised to $Q_0(s, a) = 0 \; \forall s \in S, a \in A$, and the learning rate is set to $\alpha = \frac{1}{k(s,a)}$, as described in Section 2.1.2. The value of ϵ decays from 1 to 0 at a constant rate across the first 80% of trials, after which it remains at 0.

As representative of works in ARL that assume a single, infallible expert, we have the *Naïve Advice Follower* (*NAF*), which is identical to the Baseline Agent except that it will always follow any advice it has received for a given state, as it acts under the assumption that each expert is reliable. As these approaches are not designed for use with multiple experts, they are not equipped with methods to handle contradictions between these experts. Thus, in our experiments, if the agent has received multiple pieces of advice for a state, it will randomly select one of those pieces of advice to follow with uniform probability.

Finally, we have a *CLUE* agent, as outlined in Chapter 3, whose learning algorithm and unassisted

decision-making strategy are identical to the Baseline Agent, with initial beta prior values of $\alpha_0, \beta_0 = 1$ for each expert and a decision-making threshold of $T = \frac{2}{|A|}$.

4.1.3 Experts

All experiments are conducted with simulated software-based experts. As discussed in Section 2.2, many assisted RL (ARL) approaches limit the number of interactions between experts and agents, so as to simulate the potential cost of communication, and thus conditions are imposed upon the expert to ensure it gives advice where it is most needed [Torrey and Taylor 2013]. In this research, we adopt the temporality conditions outlined by Innes and Lascarides [2019]. Firstly, we force the expert to wait at least μ trials between advice utterances, thus restricting the total number of interactions. We refer to μ as the *interval parameter*. Secondly, the expert may only offer advice if:

$$\sum_{t' \leq i \leq t} \frac{EU(a_i^*|s_i) - EU(a_i|s_i)}{t - t'} \geq \gamma,$$

where t is the current trial, t' is the last trial for which expert e gave advice, a_i^* is the optimal action for trial i, a_i is the action taken by the agent in trial i, and γ is a *tolerance parameter* that controls how tolerant an expert is of suboptimal performance by the agent. This condition ensures that the expert will only intervene if the agent is under-performing to a significant degree.

In order to simulate reliability, each expert e is controlled by a *true reliability parameter* $\rho_{true}^{(e)}$ [Masegosa and Moral 2013]. When offering advice, the expert will advise the optimal action a^* (obtained from a "ground truth" model of the environment using the variable elimination algorithm) with probability $\rho_{true}^{(e)}$, or else will randomly advise any other action. Thus an expert with $\rho_{true}^{(e)} = 1$ is reliable, while one with $\rho_{true}^{(e)} = 0$ never advises the optimal action.

This process for giving advice is summarised in Algorithm 7.

Algorithm 7 Expert Advice Process

procedure PANEL_ADVISE(s_t, a_t, r_t, E) ▷ E = panel of experts
  $advice \leftarrow [\,]$
  for $e \in E$ **do**
    $a_t^* \leftarrow$**get_optimal_action**(s_t)
    $t' \leftarrow$**last_advice_trial**() ▷ the last trial expert e gave advice
    if $t - t' \geq \mu^{(e)}$ **and** $\sum_{t' < i \leq t} \frac{EU(s_t, a_i^*) - EU(s_t, a_i)}{t - t'} \geq \gamma^{(e)}$ **then**
      $p \leftarrow$**random**()
      if $p < \rho_{true}^{(e)}$ **then**
        $advice[e] \leftarrow a_t^*$ ▷ advise best action with probability $\rho_{true}^{(e)}$
      else
        $advice[e] \leftarrow$random $a \in A \setminus \{a_t^*\}$ ▷ advise any suboptimal action
  return $advice$

4.2 Panel Compositions

In our first set of experiments, we compare the reward obtained in each trial by the agents advised by different panels of experts. The rewards obtained by the agents training over $80,000$ trials across 100 different random environments with 10 state variables ($|S| = 1024$) and 3 action variables ($|A| = 8$) are averaged and plotted against trials. For legibility, the resulting graphs are smoothed using LOWESS

smoothing [Cleveland 1981]. Shaded areas represent one standard deviation above and below the average curve.

We compare the performance of each agent with three panels of experts. The first, a *Single Reliable Expert*, consists of one expert that always gives correct advice ($\rho_{true} = 1$), representing the scenario assumed by most traditional IRL approaches. The second, a *Single Unreliable Expert*, consists of one expert that always gives incorrect advice ($\rho_{true} = 0$), representing a "worst case" scenario for traditional IRL approaches. The third, a *Varied Panel*, consists of seven experts with varying degrees of unreliability ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$), representing a scenario in which correct advice is present, but must be recognised among incorrect advice from unreliable experts. Results for the single expert panels are provided in Figure 4.2, and results for the varied panel are provided in Figure 4.3.

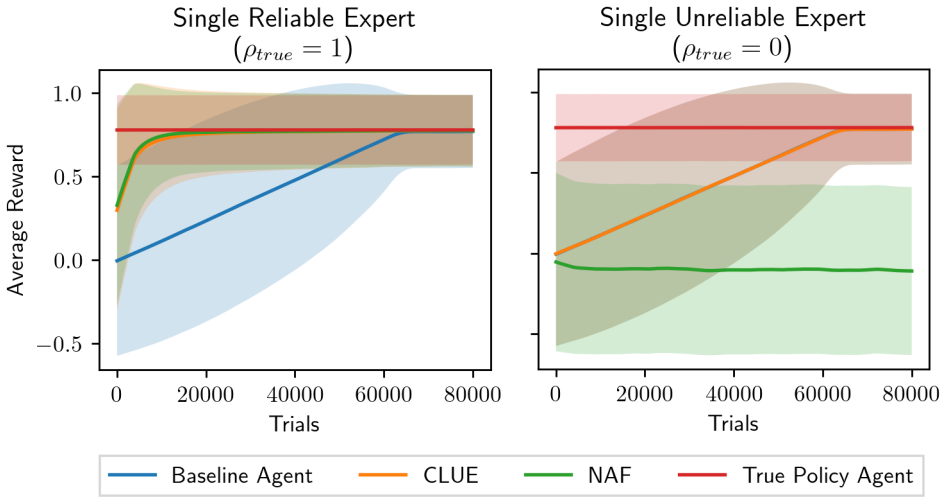


Figure 4.2: A comparison of agent performance, advised by two panels, a *Single Reliable Expert* ($\rho_{true} = 1$) and a *Single Unreliable Expert* ($\rho_{true} = 0$). Note that for the single unreliable expert, the Baseline Agent and CLUE have near-identical performance.

In the first experiment, where $\rho_{true} = 1$, both NAF and CLUE outperform the Baseline Agent, with NAF converging particularly quickly, demonstrating the power of existing ARL methods when the assumption of reliability holds. As CLUE does not assume reliability (with initial $\mathbb{E}[\rho] = 0.5$) and is therefore more cautious, it does not converge as quickly, although it still is able to take advantage of the correct advice to converge faster than the Baseline Agent.

A demonstration of the robustness of CLUE comes in the second experiment, where $\rho_{true} = 0$. In this scenario, NAF exclusively follows sub-optimal advice and therefore performs exceptionally poorly, failing to converge to the optimal policy. CLUE, on the other hand, correctly identifies that the advice is poor and learns not to follow it, and thus has performance almost identical to the Baseline Agent.

In the third experiment, with the varied panel, the performance of NAF lies somewhere between the two single expert cases, as it receives a mix of advice including optimal and suboptimal actions, and cannot discern which advice is advantageous to follow. However, CLUE converges to the optimal policy even faster than it did in the case of a single reliable expert, comparable to the performance of NAF in the same case. This indicates that not only is CLUE learning to assess which experts are worth following and which are not, it is also benefiting from the presence of more advice and the information provided by consensus and disagreement between experts.

Note that in all experiments the standard deviation is quite large relative to the range of possible

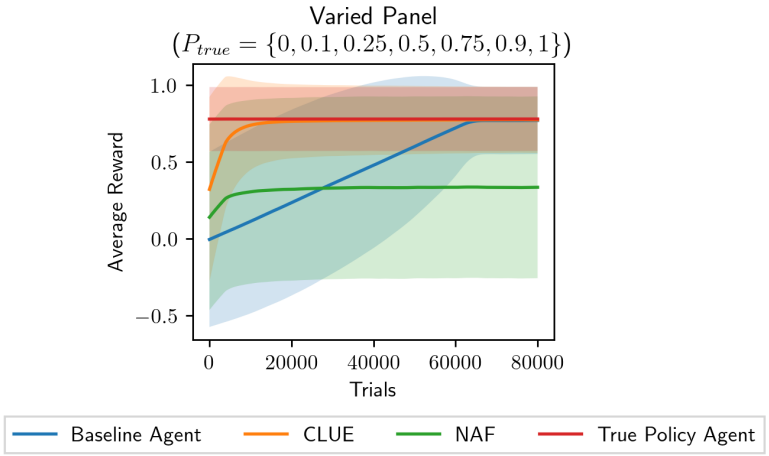


Figure 4.3: A comparison of agent performance, advised by the varied panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$).

rewards. This is due to the fact that these results are aggregated over multiple environments with very different reward functions. It is important to note that, whenever the agent's performance converges to optimal behaviour, the standard deviation in reward matches that of the True Policy Agent.

Thus, the above experiments have demonstrated that, for this class of problems, CLUE is able to benefit from increased sample efficiency with advice from reliable experts while being robust against advice from unreliable experts, and can benefit from advice given by multiple experts even when these experts contradict each other.

4.2.1 Adversarial Advice

To illustrate how CLUE can benefit from adversarial advice (advice from an expert that is consistently wrong), we compare the average reward obtained by the agent advised by a single reliable expert ($\rho_{true} = 1$) and a single unreliable expert ($\rho_{true} = 0$), in an environment where $|\mathcal{X}| = 10$ and $|\mathcal{D}| = 1$, and thus $|A| = 2$, averaged over 100 runs. Results are plotted in Figure 4.4.

For both experts, CLUE shows a similar improvement in performance over the Baseline Agent. This improvement is possible in the case of the single unreliable expert because, having estimated the reliability of the expert to be low, the suboptimal action advised by the expert is deemed to have a low probability of being optimal, thus improving the probability of the other action being optimal.

4.2.2 Comparison with Probabilistic Policy Reuse

In this set of experiments, we compare the average reward per trial obtained by a number of agents in an environment with $|\mathcal{X}| = 7$, $|\mathcal{D}| = 3$, averaged over 100 runs. The panels compared include the three described in Section 4.2, as well as a *Single Random Expert* ($\rho_{true} = 0.5$). In addition to the agents described in Section 4.1.2, we compare adaptations of the π-reuse and PRQ algorithms of Fernández and Veloso [2006], hereafter referred to as the *Decayed Reliance Agent* and *PRQ Agent* respectively.

The Decayed Reliance Agent maintains a parameter $\psi \in [0, 1]$ that decays over the course of learning. With probability ψ, the agent randomly follows advice it has received from the experts, in the same fashion as NAF. Otherwise, the agent acts unassisted, in the same fashion as the Baseline Agent. In these experiments, ψ decays from 1 to 0 across the first 80% of trials.

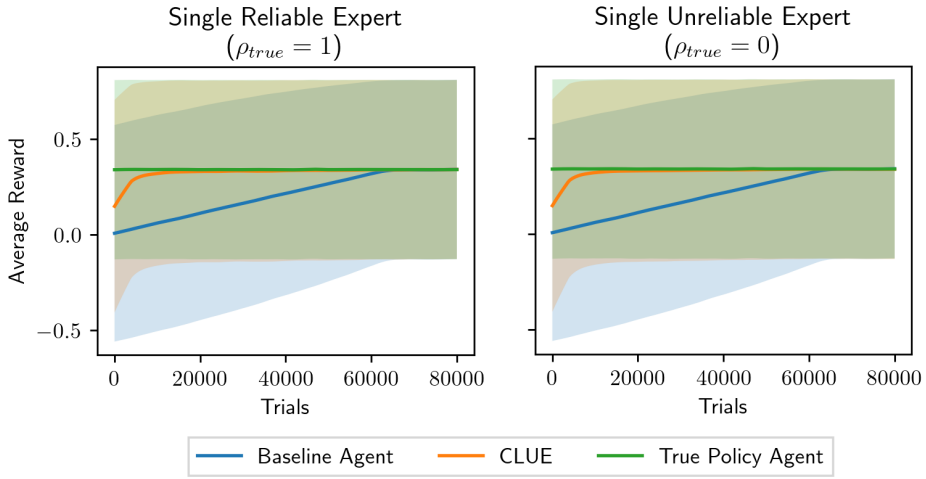


Figure 4.4: A comparison of the performance of CLUE with a single reliable expert ($\rho_{true} = 1$) and a single unreliable expert ($\rho_{true} = 0$) for an environment where $|A| = 2$.

The PRQ Agent maintains a list of policies $L = \{\Pi^{(e)} | \forall e \in E\}$ which retain the advice it has received from each expert, as well as its own learned policy $\Pi^{(0)}$. Every time it selects an action, it follows a policy with probability

$$P(\Pi^{(j)}) = \frac{e^{\tau W^{(j)}}}{\sum_{p=0}^{|E|} e^{\tau W^{(p)}}},$$

where τ is a temperature parameter and $W^{(j)}$ is the average reward obtained by following expert j, with $W^{(0)}$ denoting the average reward obtained by following the agent's policy. Every trial, the temperature parameter is incremented by $\Delta\tau$. In these experiments, $\tau = 0$ and $\Delta\tau = 0.05$.

The Decayed Reliance Agent is outperformed by CLUE in all tested panels, as either its performance is hampered due to following suboptimal advice (as with the single unreliable expert), or it stops relying on useful advice before its policy has converged (as with the other panels).

The PRQ agent performs exceptionally well when a reliable expert is present, even outperforming CLUE with the single reliable expert, as it learns to identify and follow the optimal policy. However, its performance is hampered with the single unreliable expert, as it takes longer for it to learn to identify and ignore the suboptimal policy. When the expert's policy is not optimal, but is nevertheless better than a random policy, as with the single random expert, the PRQ agent learns early on that following the expert's advice is better than its initial policy, resulting in an initial boost to performance, but is unable to surpass the performance of the policy. CLUE, on the other hand, is able to benefit from the higher-than-random chance of receiving optimal advice, but is still able to surpass the expert's performance.

These results demonstrate that CLUE is more robust to unreliable experts, while still being able to benefit from the presence of good advice, and that CLUE is able to surpass the performance of the experts advising it.

4.2.3 Alternate Approach to Expert Simulation

In this set of experiments, we consider a different approach to simulating experts, in order to show that the performance of CLUE is not dependent on the specific implementation of software experts detailed in Section 4.1.3. In this approach, each expert only observes a subset of state variables in the ID that

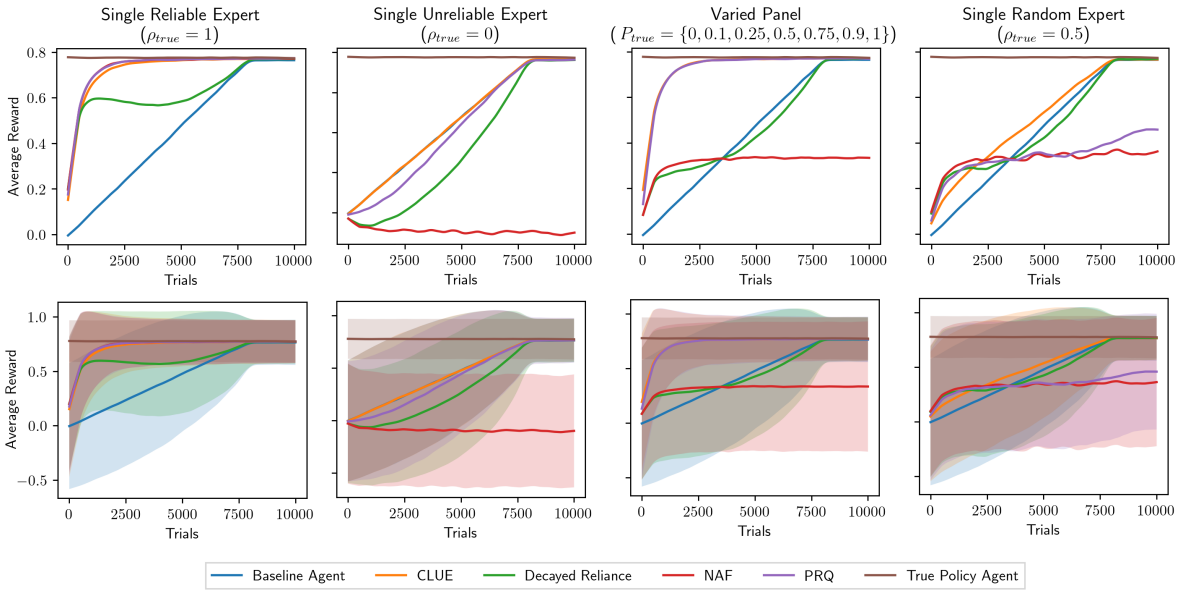


Figure 4.5: Comparisons of CLUE with the Decayed Reliance and PRQ agents, adapted from the π-reuse and PRQ algorithms respectively Fernández and Veloso [2006]. For the sake of clarity, a version without the shaded areas representing one standard deviation is provided. Note that for the varied panel, CLUE and PRQ have near-identical performance, and thus the PRQ curve lies on top of the CLUE curve.

represents the environment, and must advise an action based on this partial observation. Thus an expert with 0 hidden variables is reliable and an expert with $|\mathcal{X}|$ hidden variables always advises the action most likely to be optimal given no observation of the state. A comparison of the performances of CLUE and NAF in an environment where $|\mathcal{X}| = 7$ and $|\mathcal{D}| = 3$, averaged over 10 runs, is given in Figure 4.6. The panels tested include single experts with 0, 1, 3, 5 and 7 hidden variables, as well as a varied panel composed of all of the aforementioned experts.

Here CLUE outperforms the Baseline Agent when the number of hidden state variables is less than $|\mathcal{X}|$, as in these cases the expert is more likely to advise the optimal action than any other action, and performs on par with the Baseline Agent when the amount of information that can be gained from the expert is minimal. CLUE is able to benefit from the varied panel, with performance similar to the case where no state variables are hidden from the expert. NAF, on the other hand, only converges to the optimal policy when the expert is reliable, performing poorly otherwise. These experiments show that CLUE can perform desirably with expert implementations that do not rely on a single "ground truth" reliability parameter as with the implementation presented in Section 4.1.3.

4.3 Reliability Estimates

4.3.1 Reliability Estimates Over Time

In order to further examine the results obtained in Section 4.2, we now compare the value of $\mathbb{E}[\rho]$ for each expert in each panel across the same $80,000$ trials as in the previous experiments. As before, results are averaged over 100 runs in different randomly generated environments and the resulting plots are smoothed using LOWESS smoothing [Cleveland 1981]. Results for the single reliable expert and single unreliable expert are presented in Figure 4.7, and results for the varied panel are presented in Figure 4.8.

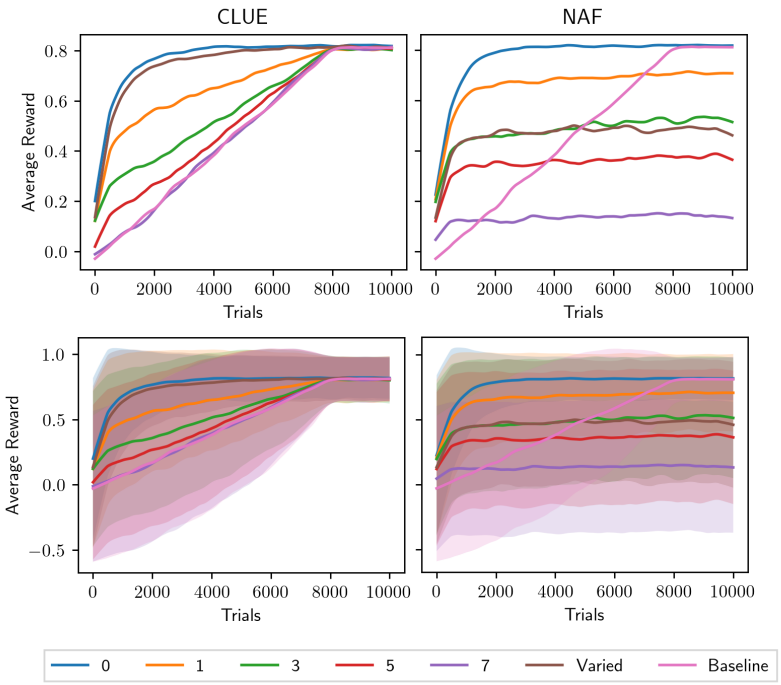


Figure 4.6: A comparison of agent performance advised by a single expert with varying numbers of hidden state variables. The legend denotes the number of hidden state variables, with the varied panel consisting of experts with 0, 1, 3, 5 and 7 variables hidden. The Baseline Agent is also given for comparison. For the sake of clarity, a version without the shaded areas representing one standard deviation is provided.

For the single expert cases, the value of $\mathbb{E}[\rho]$ converges towards the correct value of ρ_{true} (1 and 0 respectively), with the final estimates being $\mathbb{E}[\rho] = 0.996$ for the single reliable expert and $\mathbb{E}[\rho] = 0.005$ for the single unreliable expert. The small degree of error present in these values may be due to poor evaluations of the optimality of the advice received by the agent in early trials, as discussed in Section 3.2.3. Nevertheless, these estimates are close enough (absolute error less than 0.01) to the values of ρ_{true} as to result in the desirable performance presented in Figure 4.2.

For the varied panel, each expert is correctly ranked according to their reliability and the value of $\mathbb{E}[\rho^{(e)}]$ for each expert e correctly converges towards the value of $\rho^{(e)}_{true}$, even faster than the single expert cases. As is to be expected, the variance in the final estimate is larger for experts that randomly choose between suboptimal and optimal advice ($\rho_{true} = 0.5$) than for experts that more consistently offer one or the other. The values to which the estimates converge, as well as the errors in these estimates, are tabulated in Table 4.1.

These experiments show that CLUE is able to estimate ρ_{true} within a small degree of error and is able to correctly rank experts by their reliabilities in the case of the varied panel. As the decision-making rule presented in Section 3.2.2 assumes accurate estimates of reliability, and theoretical analysis in Section 3.4 shows that accurate estimates of reliability lead to a higher probability of acting optimally when exploring (for $|A| = 2$), these results can aid in explaining the desirable performance of CLUE demonstrated in Section 4.2.

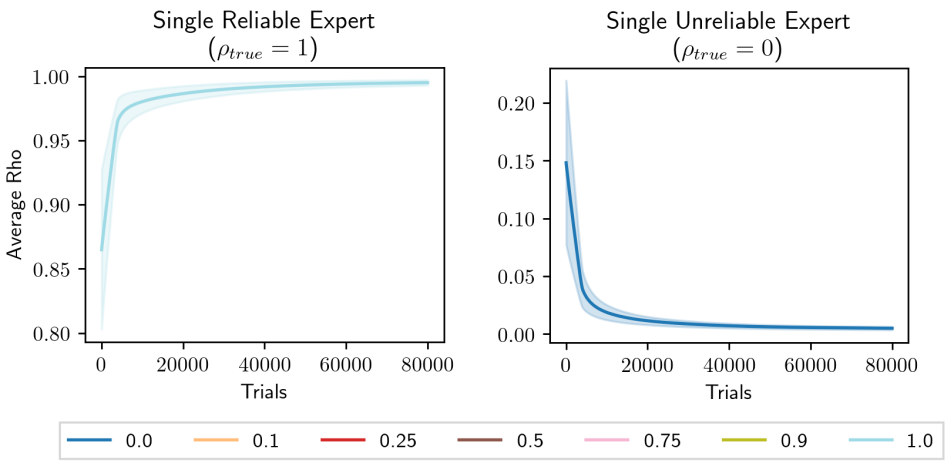


Figure 4.7: The value of $\mathbb{E}[\rho]$ over time as the agent learns, advised by the single reliable expert ($\rho_{true} = 1$) and single unreliable expert ($\rho_{true} = 0$).

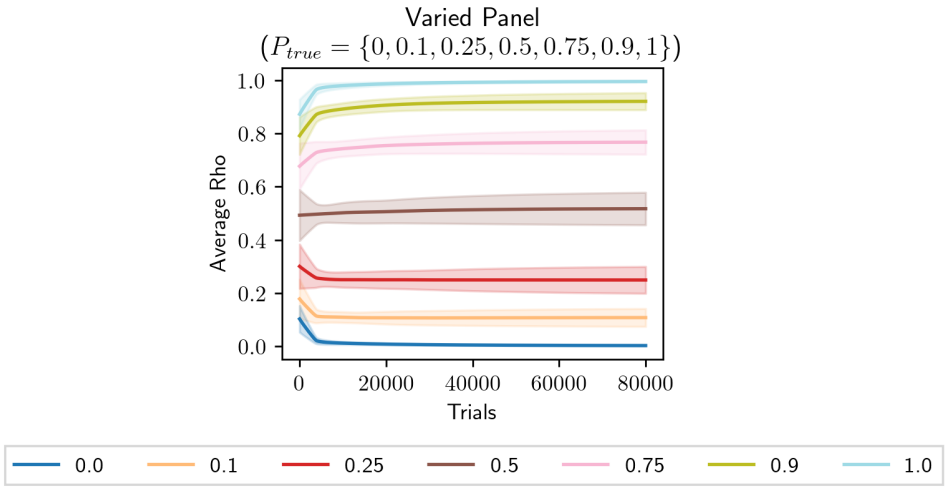


Figure 4.8: The value of $\mathbb{E}[\rho]$ over time as the agent learns, advised by the varied panel ($P_{true} = \{0, 0.1, 0.25, 0.5, 0.75, 0.9, 1\}$). The legend shows the value of $\rho^{(e)}_{true}$ for each expert.

4.3.2 Prior Parameters

In the next set of experiments, we investigate the effect of varying the α_0 and β_0 parameters which determine the prior reliability distribution (see Section 3.2.3). Recall that α_0 and β_0 can be thought of as prior counts of the expert giving incorrect and correct advice respectively. Thus $\alpha_0 > \beta_0$ biases ρ towards 0, and $\alpha_0 < \beta_0$ towards 1, with $\alpha_0 + \beta_0$ determining the strength of that prior against trial data.

To measure their effect, we plot the difference between the average total regret of CLUE and the average total regret of the Baseline Agent,

$$\bar{R}_{CLUE} - \bar{R}_{Baseline}, \tag{4.1}$$

for a number of different α_0 and β_0 values, summed over $10,000$ trials and averaged over 100 runs for a single randomly generated environment with $|S| = 128$ and $|A| = 8$, where the total regret for N trials

is equal to

$$R = \sum_{0 \leq t < N} r(s_t, \pi^*(s_t)) - r(s_t, a_t). \tag{4.2}$$

Therefore, a value of 0 indicates performance equal to the Baseline Agent. Minimising regret is equivalent to maximising reward, and thus negative values indicate better performance than the Baseline Agent and positive values indicate worse performance than the Baseline Agent, with the magnitude indicating the degree to which performance is better or worse. For the sake of simplicity, the same α_0 and β_0 values are used for each expert in a panel, although in practice these values can be initialised differently for each expert.

This process was repeated for each of the panels described in Section 4.2, and plotted in Figure 4.9. For comparison, the average total regret obtained by the Baseline Agent was 3505.0, and the average total regret obtained by NAF was 411.1 for the single reliable expert, 9449.7 for the single unreliable expert, and 4994.5 for the varied panel.

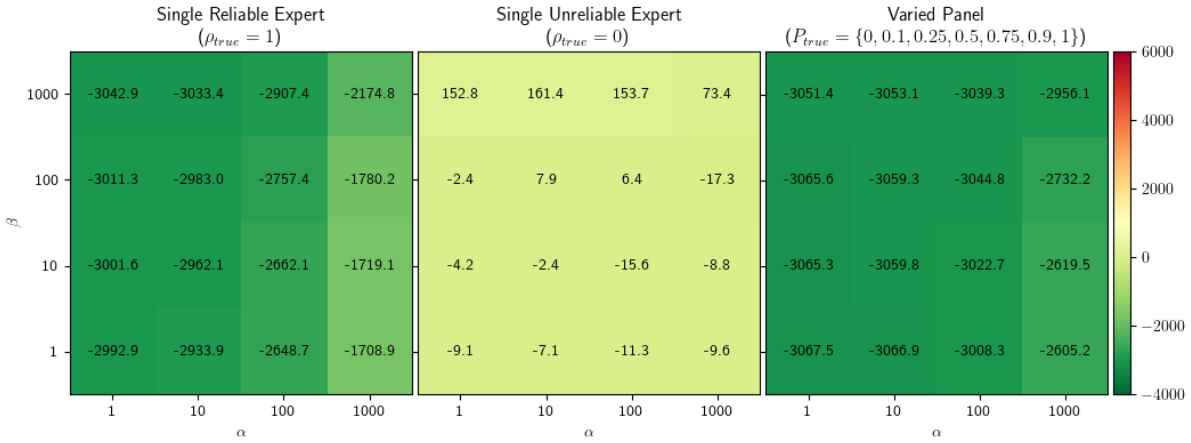


Figure 4.9: The total difference in regret between CLUE and the Baseline Agent for each value of α_0 and β_0 and for each panel. Lower values indicate better performance.

For the single reliable expert, the best performance occurs when the parameters heavily bias the estimate towards 1. However, all tested values result in improved performance over the Baseline Agent, and the gain in performance between $\alpha_0 = 1, \beta_0 = 1$ and $\alpha_0 = 1, \beta_0 = 1000$ is minimal. Results for the single unreliable expert are less varied, with performance close to the Baseline Agent for all tested values. Performance is only degraded when β_0 is large.

The best performance for the varied panel occurs when both parameters are low, as the variety in ρ_{true} means that no single strong prior can bias the reliability distributions correctly for all experts at a

ρ_{true}	**Final estimate**	**Absolute Error**	**Relative Error**
0	0.004	0.004	N/A
0.1	0.098	0.002	0.020
0.25	0.254	0.004	0.016
0.5	0.504	0.004	0.008
0.75	0.759	0.009	0.012
0.9	0.916	0.016	0.018
1	0.994	0.006	0.006

Table 4.1: Comparison of converged $\mathbb{E}[\rho]$ estimates for the varied panel.

time. Across all panels, the performance with a relatively uninformative prior is close to, if not equal to, the best performance, making it a reasonable choice in the absence of strong belief about an expert's reliability. These results also demonstrate that CLUE is robust to the choice of prior, except where $\alpha_0 + \beta_0$ approaches the order of magnitude of the total number of trials.

4.4 Expert Parameters

In the next set of experiments, we investigate the effect of varying the number of interactions between the agent and each expert. Recall from Section 4.1.3 that the number of interactions is determined by the values of μ and γ, with lower values of both resulting in more interactions and higher values of both resulting in fewer interactions. As in Section 4.3.2, we plot the difference in average total regret between the CLUE and the Baseline Agent, and between NAF and the Baseline Agent, for varying values of μ and γ, as depicted in Figure 4.10. The environment and number of trials and runs is identical to Section 4.3.2, and α_0, $\beta_0 = 1$, as in Section 4.2.



Figure 4.10: The total difference in regret between CLUE and the Baseline Agent and NAF and the Baseline Agent for each value of μ and γ and for each panel. Lower values indicate better performance.

With NAF, more advice results in better performance when the advice is always correct, but worse performance when the advice is sometimes incorrect. CLUE on the other hand is robust to the presence of incorrect advice; more correct advice results in better performance, but more incorrect advice has no adverse effect on performance. These results show that increased agent-expert interaction does not lead to worse performance, with performance either improving or not changing significantly depending on the reliability of the experts.

4.5 Hardware and Software Specifications

Computations were performed using High Performance Computing infrastructure provided by the Mathematical Sciences Support unit at the University of the Witwatersrand.

Experiments were run on Ubuntu 18.04 machines with Intel(R) Core(TM) i9-10940X CPU @ 3.30GHz, with 125GiB of RAM.

All code was written and executed in python 3.8.6, with the following libraries:

- *numpy* version 1.19.2
- *matplotlib* version 3.3.2
- *statsmodels* version 0.12.0
- *networkx* version 2.5

Additionally, the implementations of graphical models, influence diagrams, factors and variable elimination are adapted from Poole and Mackworth [2017].

Chapter 5

Conclusion

The aim of this work was to devise an algorithm for learning SSDPs with the assistance of one or more expert advisors, in such a way that an agent can benefit from the advice of reliable experts but can be robust against the advice of unreliable experts. We presented the CLUE algorithm, which operates by explicitly modelling the reliability of each expert and using that model to combine advice in a Bayesian fashion to bias action selection when exploring. Our contributions included the aforementioned decision-making strategy (Section 3.2.2) and a method by which the reliability models can be updated given the agent's own experience of the environment (Section 3.2.3). We also provided theoretical results that show the conditions under which a CLUE agent's exploration is guaranteed to be more likely to act optimally than some default exploration strategy (Section 3.4).

To demonstrate the effectiveness of CLUE, we performed a number of experiments with different simulated environments, types of simulated experts and panel compositions. Our results show that CLUE is able to benefit from advice given by consistently reliable experts, but is also robust against advice given by consistently unreliable experts. When an expert's performance is not optimal, but is better than randomly selecting actions, a CLUE agent is able to benefit from this advice when exploring, but is ultimately able to surpass the performance of the expert advising it. When advised by multiple experts, CLUE is able to exploit the information provided by consensus and contradictions among experts to correctly rank experts by their reliability and improve performance gains. Furthermore, our experiments show that the performance of CLUE is not dependent on a narrow selection of hyperparameters, but is instead robust to a wide choice of initial reliability distributions and varying degrees of agent-expert interaction.

This work may allow for easier integration of external information in the learning process, ultimately contributing towards tackling more complex problems with greater sample efficiency. The explicit modelling of expert reliability allows for a more transparent decision-making process, as it can easily be ascertained why a CLUE agent did or did not follow a given piece of advice. Thus, the integration of the CLUE framework in the learning process does not negatively impact explainability. As with all RL approaches, care must be taken to ensure an ethical process of data gathering from real world interactions (such as in the medical diagnosis example). Additionally, any application of CLUE that uses human experts should ensure that advice is elicited ethically, without exploitation and in a manner that respects participants' privacy.

5.1 Future Work

A natural future extension to CLUE would be to extend the framework to the full, episodic RL problem. Such an extension would need to take into account delayed rewards when evaluating the advice given

by experts. An agent in this setting would also need to decide between following entire policies advised by an expert and following single actions or small sequences of actions.

All learning agents tested in this dissertation use the action-value ϵ-greedy algorithm as described in Section 2.1.2 to learn policies and select actions when unassisted. However, the CLUE algorithm may make use of any learning algorithm and unassisted decision-making strategy. Further research would investigate the performance of a CLUE agent which makes use of other SSDP algorithms (e.g. LinUCB [Li *et al.* 2010], NeuralBandit [Allesiardo *et al.* 2014] or Contextual Thompson Sampling [Agrawal and Goyal 2013]) for policy learning and unassisted action selection.

Other possible future research includes relaxing the assumptions given in Section 3.1, particularly the assumption of uniform reliability across the state space. Relaxing this assumption would almost certainly require a more complex model of the reliability of each expert.

Other related future work may include testing the behaviour of CLUE when advice is given only in certain states. The theoretical analysis performed in Section 3.4 can also be expanded to arbitrary action spaces in order to determine if similar behaviour guarantees are present. This would involve leaving $|A|$ arbitrary when determining $P(a^*)$.

Finally, it remains to be seen how well a CLUE agent would perform in real world environments with the advice of human experts. Further research is needed in these areas to ascertain how well CLUE generalises to these settings.

References

[Agrawal and Goyal 2013] Shipra Agrawal and Navin Goyal. Thompson sampling for contextual bandits with linear payoffs. In *International Conference on Machine Learning*, pages 127–135. PMLR, 2013.

[Allesiardo *et al.* 2014] Robin Allesiardo, Raphaël Féraud, and Djallel Bouneffouf. A neural networks committee for the contextual bandit problem. In *International Conference on Neural Information Processing*, pages 374–381. Springer, 2014.

[Bianchi *et al.* 2004] Reinaldo AC Bianchi, Carlos HC Ribeiro, and Anna HR Costa. Heuristically accelerated Q–learning: a new approach to speed up reinforcement learning. In *Brazilian Symposium on Artificial Intelligence*, pages 245–254. Springer, 2004.

[Bignold *et al.* 2020] Adam Bignold, Francisco Cruz, Matthew E Taylor, Tim Brys, Richard Dazeley, Peter Vamplew, and Cameron Foale. A conceptual framework for externally-influenced agents: An assisted reinforcement learning review. *arXiv preprint arXiv:2007.01544*, 2020.

[Burke and Klein 2020] Pierce Burke and Richard Klein. Confident in the crowd: Bayesian inference to improve data labelling in crowdsourcing. In *2020 International SAUPEC/RobMech/PRASA Conference*, pages 1–6. IEEE, 2020.

[Cleveland 1981] William S Cleveland. LOWESS: A program for smoothing scatterplots by robust locally weighted regression. *American Statistician*, 35(1):54, 1981.

[Efthymiadis *et al.* 2013] Kyriakos Efthymiadis, Sam Devlin, and Daniel Kudenko. Overcoming erroneous domain knowledge in plan-based reward shaping. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1245–1246. Citeseer, 2013.

[Etz 2018] Alexander Etz. Introduction to the concept of likelihood and its applications. *Advances in Methods and Practices in Psychological Science*, 1(1):60–69, 2018.

[Fernández and Veloso 2006] Fernando Fernández and Manuela Veloso. Probabilistic policy reuse in a reinforcement learning agent. In *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 720–727, 2006.

[Gao *et al.* 2018] Yang Gao, Huazhe Xu, Ji Lin, Fisher Yu, Sergey Levine, and Trevor Darrell. Reinforcement learning from imperfect demonstrations. *arXiv preprint arXiv:1802.05313*, 2018.

[Gimelfarb *et al.* 2018] Michael Gimelfarb, Scott Sanner, and Chi-Guhn Lee. Reinforcement learning with multiple experts: A bayesian model combination approach. *Advances in Neural Information Processing Systems*, 31:9528–9538, 2018.

[Griffith *et al.* 2013] Shane Griffith, Kaushik Subramanian, Jonathan Scholz, Charles L Isbell, and Andrea L Thomaz. Policy shaping: Integrating human feedback with reinforcement learning. Georgia Institute of Technology, 2013.

[Heckerman and Nathwani 1992] David E Heckerman and Bharat N Nathwani. An evaluation of the diagnostic accuracy of pathfinder. *Computers and Biomedical Research*, 25(1):56–74, 1992.

[Howard and Matheson 2005] Ronald A Howard and James E Matheson. Influence diagrams. *Decision Analysis*, 2(3):127–143, 2005.

[Huo and Fu 2017] Xiaoguang Huo and Feng Fu. Risk-aware multi-armed bandit problem with application to portfolio selection. *Royal Society open science*, 4(11):171377, 2017.

[Innes and Lascarides 2019] Craig Innes and Alex Lascarides. Learning structured decision problems with unawareness. In *International Conference on Machine Learning*, pages 2941–2950, 2019.

[Kao *et al.* 2018] Hao-Cheng Kao, Kai-Fu Tang, and Edward Y Chang. Context-aware symptom checking for disease diagnosis using hierarchical reinforcement learning. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[Keswani *et al.* 2021] Vijay Keswani, Matthew Lease, and Krishnaram Kenthapadi. Towards unbiased and accurate deferral to multiple experts. *arXiv preprint arXiv:2102.13004*, 2021.

[Knox and Stone 2009] W Bradley Knox and Peter Stone. Interactively shaping agents via human reinforcement: The tamer framework. In *Proceedings of the fifth international conference on Knowledge capture*, pages 9–16, 2009.

[Koller and Friedman 2009] Daphne Koller and Nir Friedman. *Probabilistic Graphical Models - Principles and Techniques*. MIT Press, 2009.

[Langford and Zhang 2007] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 817–824. Citeseer, 2007.

[Lauritzen and Spiegelhalter 1988] Steffen L Lauritzen and David J Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Journal of the Royal Statistical Society: Series B (Methodological)*, 50(2):157–194, 1988.

[Li *et al.* 2010] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th international conference on World wide web*, pages 661–670, 2010.

[Love *et al.* 2021] Tamlin Love, Ritesh Ajoodha, and Benjamin Rosman. Should I trust you? Incorporating unreliable expert advice in human-agent interaction. In *Workshop on Human-aligned Reinforcement Learning for Autonomous Agents and Robots*, 2021.

[Masegosa and Moral 2013] Andrés R Masegosa and Serafín Moral. An interactive approach for bayesian network learning using domain/expert knowledge. *International Journal of Approximate Reasoning*, 54(8):1168–1181, 2013.

[Norvig and Russell 1994] Peter Norvig and Stuart J. Russell. *Artificial Intelligence: A Modern Approach*. Prentice Hall, December 1994.

[Owen 2008] Claire Elayne Bangerter Owen. *Parameter estimation for the beta distribution*. Master's thesis, Brigham Young University-Provo, 2008.

[Pearl 1988] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, 1988.

[Poole and Mackworth 2010] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[Poole and Mackworth 2017] David L Poole and Alan K Mackworth. Python code for artificial intelligence: Foundations of computational agents. *Version 0.7*, 6, 2017.

[Shelton 2000] Christian Shelton. Balancing multiple sources of reward in reinforcement learning. *Advances in Neural Information Processing Systems*, 13:1082–1088, 2000.

[Sutton and Barto 2018] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[Taylor and Chernova 2010] Matthew E Taylor and Sonia Chernova. Integrating human demonstration and reinforcement learning: Initial results in human-agent transfer. In *Proceedings of the Agents Learning Interactively with Human Teachers AAMAS workshop*, page 23. Citeseer, 2010.

[Taylor and Stone 2009] Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(Jul):1633–1685, 2009.

[Thomaz *et al.* 2005] Andrea Lockerd Thomaz, Guy Hoffman, and Cynthia Breazeal. Real-time interactive reinforcement learning for robots. In *AAAI 2005 workshop on human comprehensible machine learning*, 2005.

[Torrey and Taylor 2013] Lisa Torrey and Matthew Taylor. Teaching on a budget: Agents advising agents in reinforcement learning. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1053–1060, 2013.

[Varatharajah *et al.* 2018] Yogatheesan Varatharajah, Brent Berry, Sanmi Koyejo, and Ravishankar Iyer. A contextual-bandit-based approach for informed decision-making in clinical trials. *arXiv preprint arXiv:1809.00258*, 2018.

[Yi *et al.* 2012] Sheng Kung Michael Yi, Mark Steyvers, Michael D Lee, and Matthew J Dry. The wisdom of the crowd in combinatorial problems. *Cognitive science*, 36(3):452–470, 2012.

[Zhang and Poole 1994] Nevin Lianwen Zhang and David Poole. A simple approach to bayesian network computations. In *Proceedings of the biennial conference-Canadian society for computational studies of intelligence*, pages 171–178. Canadian Information Processing Society, 1994.