Non-Markovian State Aggregation for
Reinforcement Learning

David Johnston

October 30, 2015

Contents
1 Introduction 1
2 Reinforcement Learning 2
  2.1 The Value Function 3
  2.2 Markov Decision Processes 4
3 Feature Reinforcement Learning 5
  3.1 Feature Maps 5
    3.1.1 Notation 6
    3.1.2 State Aggregation and φ-uniformity 6
  3.2 Counterexamples to Open Problem 10 for \( V^* \) Aggregation 8
    3.2.1 Transient counterexample 8
    3.2.2 Second Counterexample 10
    3.2.3 Experimental Performance of Ergodic Counterexample 14
4 Using Q-learning on Aggregated State Representations 15
  4.1 Generating random aggregated MDPs 15
    4.1.1 Single Action 15
    4.1.2 Multiple Actions 16
  4.2 Experimental Parameters 17
    4.2.1 \( Q^* \) aggregation 17
    4.2.2 \( V^* \) aggregation 18
  4.3 Results 18
  4.4 Dependence on \( \epsilon \) 20
  4.5 Estimating the deviation from a Markovian process 20
  4.6 non-Markovianity of Randomly Aggregated MDPs 22
  4.7 State Aggregation with the Mountain Car Problem 26
  4.8 Experimental Setup 26
Reinforcement learning studies agents that operate in environments that provide them with rewards in response to actions that the agents take. The agent’s aim (and the agent designer’s) is that the agent takes actions that lead to the maximum possible accumulated reward. Reinforcement learning is a very general framework for goal directed learning; a reinforcement learner must be able to explore an environment sufficiently to “discover” where to find the maximum reward, and it must be able to learn enough about the dynamics of the environment to devise a policy which delivers the maximum reward in expectation. Supervised learning, in contrast, involves extra knowledge of the environment by way of examples of desirable behaviour. Unsupervised learning may not involve much prior knowledge of the problem domain, but it is not goal-directed.

The generality of the reinforcement learning setting makes it very difficult to design a reinforcement learning agent that learns well in every possible environment. Solomonoff induction \[15\] has been proposed as a universal solution to the problem of learning the dynamics of any computable environment. Solomonoff induction is, however, incomputable and so any real agent must at best be an approximation to it. AIXI \[1\] is a reinforcement learning agent built on Solomonoff induction, and achieves universal optimality in the sense that no agent exists that learns the on-policy dynamics of any environment significantly faster than AIXI. However, it is not clear that AIXI has the right balance of exploration and exploitation\[9\].

On the other hand, in a restricted class of environments known as finite state fully observable Markov Decision Processes (FS-FO-MDPs), relatively simple algorithms such
as Q-learning are known to converge to the optimal policy [20]. A recent result due to Hutter [2] has suggested that through the use of state aggregation, techniques applicable to FS-FO-MDPs may be applicable to general environments. This report examines the use of state aggregation in extending the applicability of Q-learning, and investigates algorithms for learning a state aggregation.

The report consists of three sections. Section 2 introduces key notation and definitions in reinforcement learning and Hutter’s state aggregation results. Section 4 details experiments examining the behaviour of a Q-learning agent on aggregated processes. Finally, section 5 examines algorithms for learning an aggregated representation of the environment, including a brief experimental investigation of McCallum’s U-Tree [6] and an extension of the sample complexity bound of MERL [4].

2 Reinforcement Learning

The agent-environment model is a standard conceptual framework for reinforcement learners and other AI agents [13]. In this model, an agent $\Pi$ interacts with an environment $P$ in discrete cycles indexed by a time step $t \in \mathbb{N}$. At each time step the agent may take an action $a \in \mathcal{A}$, and in response the environment will provide an observation $o \in \mathcal{O}$ and a reward $r \in \mathcal{R} \subseteq \mathbb{R}$ to the agent. The sequence of observations, rewards and actions up to time $t$ forms the history:

$$h_t := o_1 r_1 a_1 \ldots o_{t-1} r_{t-1} a_{t-1} o_t r_t \in \mathcal{H}_t := (\mathcal{O} \times \mathcal{R} \times \mathcal{A})^{t-1} \times \mathcal{O} \times \mathcal{R}$$

At cycle $t + 1$, the agent selects its next action $a_{t+1}$ according to the current history $h_t$. In general, the agent aims to take the action that maximises the expected return, defined as the discounted sum of rewards [18]:

$$G_t = \sum_{i=t}^{\infty} \gamma^{i-t} r_i$$

where $\gamma \in [0, 1]$ is the discount rate. If the problem terminates in finite time, we can add a terminal cycle $T$ after which all rewards are 0.

The agent $\Pi$ can be seen as a (possibly stochastic) function from histories to actions, and the environment $P$ as a (possibly stochastic) function from histories and actions to observations and rewards:

$$P : \mathcal{H} \times \mathcal{A} \rightarrow \mathcal{O} \times \mathcal{R}$$

$$\Pi : \mathcal{H} \rightarrow \mathcal{A}$$
We can express the fact that the agent chooses action $a_t$ given history $h_t$ by $\Pi(h_t) = a_t$. If the agent chooses actions stochastically, the distribution $\Pi(a_t|h_t)$ may be used instead. $P(o_{t+1}|r_{t+1}|h_t, a_t)$ refers to the probability that the environment yields observations $o_{t+1}$ and reward $r_{t+1}$ given history $h_t$ and action $a_t$.

### 2.1 The Value Function

A wide variety of approaches to reinforcement learning are concerned with learning the value function. The value function $V : \mathcal{H} \rightarrow \mathbb{R}$ maps a history $h_t$ to the expected return under the agent’s policy $\Pi$:

$$V^\Pi(h_t) := E^\Pi_P[G_{t+1}|h_t]$$

Related to this is the action-value or q-value function, defined as the expected return of history $h_t$ when taking action $a_t$ and subsequently following policy $\Pi$:

$$Q^\Pi(h_t, a_t) := E^\Pi_P[G_{t+1}|h_t, a_t]$$

The value and q-value functions can be written pseudo-recursively as

$$Q^\Pi(h_t, a_t) = E^\Pi_P[r_{t+1} + \gamma V^\Pi(h_{t+1})|h_t, a_t]$$ (1)

$$V^\Pi(h_t) = Q^\Pi(h_t, \Pi(h_t))$$ (2)

We can define the optimal (q-)value functions as the value of history $h_t$ (and action $a_t$) under the policy which maximises the expected return

$$V^\star(h_t) = \max_\Pi V^\Pi(h_t)$$

$$Q^\star(h_t, a_t) = \max_\Pi Q^\Pi(h_t, a_t)$$

These definitions lead to the pseudo-recursive equations

$$Q^\star(h_t, a_t) = E_P[r_{t+1} + \gamma V^\star(h_{t+1})|h_t, a_t]$$ (3)

$$V^\star(h_t) = \max_{a \in \mathcal{A}} Q^\star(h_t, a_t)$$ (4)

$$\Pi^\star(h_t) \in \text{arg max}_{a \in \mathcal{A}} Q^\star(h_t, a_t)$$ (5)

The last line is membership because in general there might be multiple actions that lead to the same expected return. As noted earlier, the objective of a reinforcement learning
agent is to maximise expected return, and if an agent can learn the optimal q-value function it can do so via equation 5.

These equations have been presented in terms of the histories \( h_t \in \mathcal{H} \), rather than in terms of states. Because histories never repeat, \( h_i = h_j \) if and only if \( i = j \) and these forms do not lead to a closed set of equations. In fully general environments, it is difficult to learn value functions from experience.

### 2.2 Markov Decision Processes

Markov Decision Processes (MDPs) form an important class of environments that permit learning from experience. In an MDP, the environment is at any time step \( t \) in some state \( s_t \in \mathcal{S} \). These states exhibit the Markov property, which informally asserts that the history \( h_t \) affects the probability of transition to the state \( s_{t+1} \) and receiving reward \( r_{t+1} \) only through the current state \( s_t \) and action \( a_t \):

\[
P(s_{t+1} | h_t a_t) = P(s_{t+1} | s_t a_t)
\]

To connect the idea of a state with previous notation, we note that in a fully-observable MDP the agent may directly observe the environment’s state, and so there is a bijective map between observations \( o \) and states \( s \).

\[
P(o_{t+1} | h_t a_t) = P(o_{t+1} | o_t a_t)
\]

If we suppose further that the state (or observation) space is finite, then for sufficiently long action sequences states must repeat. Together this forms the class of finite-state fully-observable MDPs, for which we can write closed analogues of the pseudo-recursive equations 2 to 4 known as the Bellman equations. For the optimal (q-)values, these are

\[
Q^*(o_t, a_t) = E_P[r_{t+1} + \gamma V^*(o_{t+1})]
\]

\[
V^*(o_t) = \max_{a \in \mathcal{A}} Q^*(o_t, a_t)
\]

Reinforcement learning algorithms based on temporal difference learning such as Q-learning and the TD(\( \lambda \)) family are known to converge to the optimal policy for finite state fully observable MDPs [18][10].
3 Feature Reinforcement Learning

3.1 Feature Maps

Many interesting problems are neither fully observable nor finite state. In such environments, naively applying algorithms developed for FS-FO-MDP’s cannot be expected to produce acceptable performance.

One approach to dealing with problems like this is to develop a feature map \( \phi : \mathcal{H} \rightarrow \mathcal{S} \) that takes the history \( h_t \) and maps it to a state \( s_t \) that, ideally, summarises all the relevant information of the history \( h_t \) with respect to developing a policy.

Ideally, the map \( \phi \) will have enough state distinctions that the agent can eventually learn to take the best action given the information it has access to, but not too many more than this, as a proliferation of states will slow learning. Given that simple agents perform well on MDPs, we might guess that reducing an environment to an MDP might be a reasonable goal. Reducing the environment to a FS-FO-MDP will allow an agent to learn an optimal policy, but as we will discuss later this is a stronger condition than necessary.

Formally, we will consider feature maps \( \phi : \mathcal{H} \rightarrow \mathcal{S} \). This induces a reduced process \( P_\phi \) on the environment

\[
P_\phi(s_{t+1}r_{t+1}|h_t a_t) = \sum_{\tilde{a}_{t+1}: \phi(h_t a_t h_{t+1} r_{t+1}) = s_{t+1}} P(\tilde{a}_{t+1} r_{t+1}|h_t a_t)
\]

Such a process is Markov if \( P_\phi \) is the same for all histories mapped to the same state

\[
P_\phi \in \text{MDP} \iff \exists p : P_\phi(s_{t+1}r_{t+1}|\tilde{h}_t a_t) = p(s_{t+1}r_{t+1}|s_t a_t) \quad \forall \tilde{h}_t : \phi(\tilde{h}_t) = s_t \quad (6)
\]

The process \( P_\phi \) may not be Markov. We can, however define a Markov process \( p \) through a stochastic inverse \( B(h|sa) \). Formally, we require of \( B \) only that \( \sum_{h \in \mathcal{S}} B(h|sa) = 1 \). The Markovian process \( p \) is defined as follows

\[
P(h_{t+1}|sa) = \sum_{h \in \mathcal{S}} B(h|sa) P(h_{t+1}|h_t a_t)
\]

\[
p(s'|sa) = \sum_{h \in \mathcal{S}'} P(h_{t+1}|sa)
\]

Note the process \( p \) is related to, but not the same as the possibly non-Markovian process \( P_\phi \). Nonetheless, Hutter has established that under appropriate conditions (defined
later), the value function of the process $p$ will match the value function of the original process $P[2]$.

### 3.1.1 Notation

$\Pi^*$, $V^*$, $Q^*$ refer to the optimal policy, value and q-value functions of the unaggregated process $P$.

$\pi^*$, $v^*$ and $q^*$ refer to the optimal policy, value and q-value functions of the process $p$.

The shorthand $\forall \phi(h) = \phi(\tilde{h})$ may be used to indicate $\forall h, \tilde{h}: \phi(h) = \phi(\tilde{h})$.

### 3.1.2 State Aggregation and $\phi$-uniformity

Given a process $P$ and feature map $\phi$, as defined above, Hutter [2] has shown that such a map that respects the conditions

\begin{align}
|Q^*(\tilde{h}, a)Q^*(\tilde{h}, a)| &\leq \epsilon \text{ for all } \phi(h) = \phi(\tilde{h}) \\
\Pi^*(h) = \Pi^*(\tilde{h}) &\text{ for all } \phi(h) = \phi(\tilde{h})
\end{align}

will have the following properties:

(i) $|Q^*(h, a) - q^*(\pi(h), a)| \leq \frac{\epsilon}{1 - \gamma}$ and $|V^*(h, a) - v^*(\pi(h), a)| \leq \frac{\epsilon}{1 - \gamma}$

(ii) $0 \leq V^*(h) - V^\Pi(h) \leq \frac{2\epsilon}{(1 - \gamma)^2}$ where $V^\Pi(h) = \pi^*(\phi(h))$

(iii) If $\epsilon = 0$ then $\Pi^*(h) = \pi^*(\phi(h))$

If $\epsilon = 0$, we refer to the aggregation as exact. Otherwise, it is called approximate.

If a process $P$, under a map $\phi$, respects equations 7 and 8 then we will say it’s q-value function is $\phi$-uniform.

Similarly, for a map which respects

\begin{align}
|V^*(h, a) - V^*(\tilde{h}, a)| &\leq \epsilon \text{ for all } \phi(h) = \phi(\tilde{h}) \\
\Pi(h) = \Pi(\tilde{h}) &\text{ for all } \phi(h) = \phi(\tilde{h})
\end{align}

we have the following properties:

(i) $|V^*(h) - v^*(\phi(h))| \leq 3\epsilon(1 - \gamma)^2$ and $\langle q^*(\phi(h), a) - \langle Q^*(h, a) \rangle_B \rangle \leq \frac{3\epsilon \gamma}{P}(1 - \gamma)^2$

(ii) If $\epsilon = 0$ then $\Pi^*(h) = \pi^*(\phi(h))$
Here $\langle \rangle_B$ represents the expectation under the dispersion probability $B$.

Note that in the approximate case for a $\phi$-uniform value function, there is no bound on the difference between $V^*(h)$ and $V^H(h)$. Open problem 10 of [2] asks if it is possible to establish a similar bound as for $Q^*$ aggregation in this case, and it will be demonstrated that this is not the case.

Given a process $P$ that has a $\phi$-uniform q-value function under a map $\phi$, the above results establish that learning the q-value function of the related process $p$ will be sufficient to (approximately) learn the q-value function of $P$. The question then is, can a reinforcement learning agent like Q-learning in such an environment, using an aggregation $\phi$, learn the q-value function of $p$? The actual process experienced by such an agent will be $P_\phi$ rather than $p$, which is not necessarily Markovian, so this is beyond the class of problems for which Q-learning is known to work. Nonetheless, Q-learning has been observed to converge on non-Markovian problems, and a proof that it does so under the conditions described is in development.
3.2 Counterexamples to Open Problem 10 for $V^*$ Aggregation

Here two counterexamples are presented to the following proposition:

Supposing that $\Pi^*(h) = \Pi^*(\tilde{h})$ and $|V^*(h) - V^*(\tilde{h})| \leq \epsilon$ for all $h$ such that $\phi(h) = \phi(\tilde{h})$, can we establish a bound on $V^{\tilde{\Pi}}(h)$ of the form:

$$V^*(h) - V^{\tilde{\Pi}}(h) = O\left(\frac{\epsilon}{(1 - \gamma)^2}\right)$$

Before presenting the problems, we will note that we can establish some related bounds. Given stochastic inverse $B$, from [2] we have

$$\langle V^{\Pi}(h) \rangle_B \geq V^*(h) - \frac{3\epsilon}{(1 - \gamma)^2}$$

For this bound on the expectation to hold, we must have $V^{\Pi}(h)$ not much smaller than $V^*(h)$ with high probability. Thus both of the counterexamples construct an MDP which, under aggregation, there is a particular history $h_l$ that is encountered with very low probability for which $V^*(h) - V^{\Pi}(h)$ can be very large.

Because the counterexamples are constructed from MDPs which are more naturally described in terms of states than histories, we will adopt the terminology of “raw state” to refer to any state of the original MDP, symbolised by capital letters $A, B, C, ...$, and “aggregated state” to refer to any state in the codomain of $\phi$, symbolised by $\phi(\cdot)$. A raw state $A$ can be understood to be any history that ends in the state $A$, so we do not lose generality by switching to this terminology.

3.2.1 Transient counterexample

This example is due to Jan Leike. The actions, transition probabilities and rewards for the raw MDP are given in Fig. [1].

For this problem, $\Pi^*(C) = \Pi^*(D) = \beta$. Under this policy:
Figure 1: The first counterexample. All transition probabilities are 1. $R$ is understood to be a large positive number.

\[
V^*(E) = \frac{1}{1 - \gamma} + \epsilon \\
V^*(G) = \frac{1}{1 - \gamma} \\
V^*(F) = \frac{1}{1 - \gamma} \\
V^*(H) = \frac{1}{1 - \gamma} + \epsilon \\
V^*(C) = V^*(D) = \frac{\gamma}{1 - \gamma} + \gamma \epsilon
\]

We will define the stochastic inverses $B(F|\phi(F)) = 1$, and $B(H|\phi(H)) = 1/2$ and $B(C|\phi(C)) = 0$
Returning to the problem description, the value functions of $\phi(H)$ and $\phi(F)$ will be:

$$v^*(\phi(F)) = V^*(F) = \frac{1}{1 - \gamma}$$

$$v^*(\phi(H)) = \frac{1}{2} (1 + \epsilon + 1 - \gamma \epsilon) + \gamma v^*(\phi(H))$$

$$= \frac{1}{1 - \gamma} + \frac{\epsilon}{2}$$

$$= v^*(\phi(F)) + \frac{\epsilon}{2}$$

Given this, we can calculate q-value functions of $\phi(C)$:

$$q^*(\phi(C), \beta) = \gamma v^*(\phi(F))$$

$$q^*(\phi(C), \alpha) = \gamma v^*(\phi(H))$$

Clearly, $\pi^*(\phi(C)) = \alpha$. If $\pi^*(\phi(C)) = \bar{\Pi}(C) = a$, then $V^\Pi(C) = \frac{-R}{1 - \gamma}$, which can be made arbitrarily large in magnitude for any $\gamma$, and so we can find an arbitrarily large difference in $V^* - V^\Pi$.

### 3.2.2 Second Counterexample

The above counterexample relies on the stochastic inverse of the raw state $C$ being 0. This second counterexample establishes that 9 can be violated even when the stochastic inverse of the raw state (or history) in question is finite under the optimal policy. The MDP graph is given in Fig [2]. There are two actions available in raw states $A$ and $B$, and one in the rest; the transition probabilities for these actions are given in the graph, but to keep the diagram clear other transitions have been omitted.

The transition probabilities for the raw states $C, D, E$ and $F$ is equal, such that it is given by:

$$T_A = T_C = T_E = \frac{1 - \delta}{3}$$

$$T_B = \delta$$

Where $\delta$ can be substituted for any of $C, D, E, F$, and $\delta \ll 1$. 11
Figure 2: The second counterexample. Note that only transitions from nodes $A$ and $B$ are affected by choosing action $\alpha$ or $\beta$. The rewards for the various transitions are given in the text.

The problem has a high degree of symmetry, so the values of nodes $C$, $D$, $E$ and $F$ are all given by a similar expression:

$$V^\Pi(\cdot)_{(CDEF)} = \Sigma_{a \in \Pi, s'} R_{s'} + \gamma(T_A V^\Pi(A) + T_B V^\Pi(B) + T_C V^\Pi(C) + T_E V^\Pi(E))$$

Where, $\cdot$ is substituted for the state of interest, and $\Sigma_{s'} R_{s'} = R_A + R_B + R_C + R_D$ is the sum of the rewards for the transitions from that node under action $a$. Because they have common transition probabilities, the value for each state depends only on the immediate reward given, regardless of $\gamma$. 

12
The rewards for each transition are as follows:

\[
\begin{align*}
R_{A_{a\cdot s'}} &= 0 \quad \text{all } a, s' \\
R_{BD} &= 0 \\
R_{BF} &= -r \\
R_{C_{a\cdot s'}} &= -\frac{\epsilon}{6} \quad \text{all } a, s' \\
R_{D_{a\cdot s'}} &= \frac{\epsilon}{12} \quad \text{all } a, s' \\
R_{E_{a\cdot s'}} &= \frac{\epsilon}{6} \quad \text{all } a, s' \\
R_{F_{a\cdot s'}} &= -\frac{\epsilon}{12} \quad \text{all } a, s'
\end{align*}
\]

The action superscript has been omitted for each state that only has one action, and \(a \in \alpha, \beta\).

The values of raw states \(A\) and \(B\) are given by:

\[
\begin{align*}
V^{\Pi}(A) &= \gamma(\Pi(\alpha|A)V^{\Pi}(D) + \Pi(\beta|A)V^{\Pi}(F)) \\
V^{\Pi}(B) &= -\Pi(\beta|B)r + \gamma(\Pi(\alpha|B)V^{\Pi}(D) + \pi(\beta|B)V^{\Pi}(F))
\end{align*}
\]

It can be verified that, for any policy \(\Pi\),

\[
\begin{align*}
V^{\Pi}(D) - V^{\Pi}(C) &= \epsilon \\
V^{\Pi}(D) - V^{\Pi}(F) &= \frac{2\epsilon}{3} \\
V^{\Pi}(F) - V^{\Pi}(E) &= -\epsilon
\end{align*}
\]

From this, we can see that \(\Pi^*(A) = \Pi^*(B) = \alpha\), so we also have

\[
\begin{align*}
V^*(A) &= V^*(B) = V^*(D) \\
V^*(B) - V^*(\beta(B)) &= r
\end{align*}
\]

Together, this establishes \(|V^*(h) - V^*(\tilde{h})| \leq \epsilon\) for any \(h, \tilde{h} \in \phi(h)\), and that the value of raw state \(B\) differs by \(r\) between actions \(\alpha\) and \(\beta\).

We will now show that we can find a value \(\delta\) such that for any \(\epsilon\) and \(\gamma\), \(r\) can be arbitrarily large and the optimal policy of the aggregated problem will be \(\pi(\phi(A)) = \beta\).

To find the values of the aggregated states, we will use the stationary distributions \(\rho^{\alpha}\) and \(\rho^{\beta}\) of the MDP for the stochastic inverses \(B(h|sa)\) under the policies \(\Pi(\cdot) = \alpha\) and
\( \Pi() = \beta. \)

\[ \rho^\alpha = \frac{1}{N} [1, \frac{3\delta}{1 - \delta}, 1, \frac{1 + 2\delta}{1 - \delta}, 1, 0] \]

\[ \rho^\beta = \frac{1}{N} [1, \frac{3\delta}{1 - \delta}, 1, 0, 1, \frac{1 + 2\delta}{1 - \delta}] \]

Where \( N \) is a normalising factor. From this, we can see that

\[
\begin{array}{c|c|c}
\text{action} = \alpha & \text{action} = \beta \\
\hline
B(A|\phi(A)) & \frac{1 - \delta}{1 + 2\delta} & \frac{1 - \delta}{1 + 2\delta} \\
B(B|\phi(A)) & \frac{1 + \delta}{1 - \delta} & \frac{1 + \delta}{1 - \delta} \\
B(C|\phi(C)) & \frac{1 + \delta}{1 + \delta} & 1 \\
B(D|\phi(C)) & \frac{1 + 2\delta}{1 + \delta} & 0 \\
B(E|\phi(E)) & 1 & \frac{1 - \delta}{1 + 2\delta} \\
B(F|\phi(F)) & 0 & \frac{1 - \delta}{1 + 2\delta} \\
\end{array}
\]

Table 1: Stochastic inverses for counterexample 2.

With these inverses, we find that

\[
v^\beta(\phi(E)) - v^\alpha(\phi(C)) = \frac{2\epsilon}{3(2 + \delta)}
\]

And therefore

\[
v^\beta(\phi(A)) - v^\alpha(\phi(A)) = B(B|\phi(A))(-r) + \gamma (v^\beta(\phi(E)) - v^\alpha(\phi(C)))
\]

\[
v^\beta(\phi(A)) - v^\alpha(\phi(A)) = -\frac{3\delta}{1 + 2\delta} r + \gamma \frac{2\epsilon}{3(2 + \delta)}
\]

To satisfy the assumptions of a counterexample, we want to find values of \( \epsilon, \delta \) and \( r \) so that \( \pi(\phi(A)) = \beta \) is the optimal policy of the aggregated MDP, which will happen if \( v^\beta(\phi(A)) - v^\alpha(\phi(A)) > 0 \). This gives us the following condition:

\[
\frac{3\delta r}{1 + 2\delta} < \frac{2\epsilon \gamma}{3(2 + \delta)}
\]

This will be satisfied when

\[
\delta r < \frac{2}{27} \epsilon \gamma
\]
### Table 2: Q-values of the raw and aggregated problem after 1e5 steps of Q-learning.

<table>
<thead>
<tr>
<th>State</th>
<th>$Q^\alpha$</th>
<th>$Q^\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>1.45·10^{-2}</td>
<td>-1.0·10^{-2}</td>
</tr>
<tr>
<td>$B$</td>
<td>1.45·10^{-2}</td>
<td>-1.01</td>
</tr>
<tr>
<td>$C$</td>
<td>-4.62·10^{-2}</td>
<td>-4.62·10^{-2}</td>
</tr>
<tr>
<td>$D$</td>
<td>2.79·10^{-2}</td>
<td>2.89·10^{-2}</td>
</tr>
<tr>
<td>$E$</td>
<td>5.4·10^{-2}</td>
<td>5.39·10^{-2}</td>
</tr>
<tr>
<td>$F$</td>
<td>-2.12·10^{-2}</td>
<td>-2.05·10^{-2}</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>State</th>
<th>$Q^\alpha$</th>
<th>$Q^\beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi(A)$</td>
<td>-2.72·10^{-2}</td>
<td>-3.96·10^{-4}</td>
</tr>
<tr>
<td>$\phi(C)$</td>
<td>-5.47·10^{-2}</td>
<td>-5.47·10^{-2}</td>
</tr>
<tr>
<td>$\phi(E)$</td>
<td>5.06·10^{-3}</td>
<td>5.08·10^{-3}</td>
</tr>
</tbody>
</table>

### 3.2.3 Experimental Performance of Ergodic Counterexample

The second counterexample above was implemented and a Q-learning agent was used to learn the value functions of the raw and aggregated problems. The parameters chosen were $\varepsilon = 0.3$, $\delta = 10^{-3}$, $\gamma = 0.5$ and $r = 1$. The Q-learning agent learned the q-value function with a random action selection and a step size $\alpha = \frac{1}{n}$ at step $n$.

The agent learned a q-value function which corresponded to the important features of the example above: in the raw case, $V^*(A) = V^\alpha(A) > V^\beta(A)$ and $V^*(B) = V^\alpha(B) > V^\beta(B)$, while in the aggregated case $V^\beta(\phi(A)) > V^\alpha(\phi(A))$. In addition, $V^*(h) - V^*(\tilde{h}) < 10^{-1}$ for all $h, \tilde{h} \in \phi(h)$ while $V^\alpha(\phi(A)) - V^\beta(B) \approx 1$ (see Table 1). This difference could in principle be made much larger, but it would require a proportionally small $\delta$, which could make it take a long time to learn.
4 Using Q-learning on Aggregated State Representations

Recall that an environment $P$ that has a $\phi$-uniform q-value for some map $\phi$ will preserve the value function of the original problem. As discussed previously, the aggregation $\phi$ can induce non-Markovian behaviour in the induced process $P_{\phi}$ (Eq. [6] is violated), which places the task of learning the value function of a general $Q^*$ aggregated process beyond the realm in which Q-learning is known to work. Nonetheless, the relationship between the value function of the Markovian process $p$ and the original process $P$ suggest it is worth investigating the applicability of Q-learning to this task. The experimental investigation conducted here has found promising results. Two domains were chosen to investigate the performance of Q-learning on aggregated problems:

First, random Markov Decision Processes were generated that admitted $Q^*$ and $V^*$ aggregation. The rate of convergence of Q-learning on the raw MDP was compared to the rate of convergence of Q-learning operating on the (generally non-Markovian) aggregated problem. This was repeated across many problems with varying parameters to investigate any dependencies that could be found.

Secondly, for a more organic test of value aggregation, an aggregation of the mountain car task described by Singh [14] was developed from an estimate of the task’s value function.

4.1 Generating random aggregated MDPs

To test the performance of a Q-learning agent on aggregated state representations, a scheme was developed to generate random MDPs with properties that allowed their states to be aggregated under $Q^*$ or $V^*$ aggregation.

4.1.1 Single Action

The goal is to generate a transition matrix $T_{ss'}$ and reward matrix $R_{ss'}$ for an MDP which produces groups of states with the same value that can be aggregated. A single action scheme is developed first for simplicity. We begin by specifying an arbitrary value vector $v$ that satisfies the desired equality properties for $Q^*$ or $V^*$ aggregation. A matrix $T$ is generated with entries selected from Uniform(0,1). This matrix is verified to be aperiodic and irreducible (and regenerated if it is not), and each row is normalised so that it sums to 1. Given $T$ and $v$, we require a reward matrix $R$ to satisfy the Bellman
equation:

\[ v(s) = \sum_{s'} T_{ss'}(R_{ss'} + \gamma v(s')) \]  

The problem at this point is underspecified, and extra degrees of freedom are dealt with by specifying that the reward for any transition depends only on the final state: \( R_{xs'} = R_{ys'} \) for all \( x, y, s' \in S \). We can then write a reward vector \( r \) where \( r_s = R_s \), and rewrite the Bellman equation:

\[ Tr = (I - \gamma T)v \]  

Solving equation 20 for \( r \) yields a complete MDP. Eq. 20 also appears in Ng and Russell’s linear programming algorithm for inverse reinforcement learning [8].

4.1.2 Multiple Actions

Under multiple actions, the Bellman equation is no longer linear, so it is non-trivial to extend the above approach to generate MDPs with multiple actions available.

Instead, we begin by generating a problem as above with a single action \( a_1 \). We are then free to add actions to the problem without changing the optimal value as long as we ensure that for all actions \( a_i \neq 1 \) and \( s \in S \),

\[ q^*(s, a_2) = \sum_{s'} T_{ss'}(r_{ss}^{a_2} + \gamma v^*(s')) < v^*(s) \]  

We are free to choose values of \( q^*(s, a_i \neq 1) < v^*(s) \) and a transition matrix \( T \) as inputs to our generation scheme. We must find a set of rewards \( R^a \) that satisfies 21. We will apply the same assumption regarding rewards as in the single action case, allowing us to write a reward vector \( r^a \). The value of the vector can then be found from the equation:

\[ r^{a_i \neq 1} = T^{-1}(q(\cdot, a_i \neq 1) - \gamma T v^*) \]

This method yields only problems for which the optimal policy consists of taking the same action \( a_1 \) in each state. However, given that the labelling of actions in any state is arbitrary, we can apply an arbitrary state dependent permutation of the action labels so this is not a limitation.
4.2 Experimental Parameters

In all cases, the value function is learned using the standard Q-learning algorithm. The agent begins in state $s$, takes action $a$ and observes the next state $s'$ and the reward $r_{ss'}^a$. It then updates its estimate of the value of the state-action pair $(s, a)$ as

$$q_{i+1}(s, a) = q_i(s, a) + \frac{1}{n_{s,a}}(r_{ss'}^a + \gamma \max_{a'}(q_i(s', a')) - q_i(s, a))$$

Here, $n_{s,a}$ is the number of times the agent has observed the state action pair $(s, a)$. The value estimate was initialised to $q_0(\cdot, \cdot) = 0$ for all states and actions.

For each generated problem, the Q-learning algorithm was run on the raw MDP and the aggregated MDP for $1e6$ steps.

Both $Q^*$ and $V^*$ aggregation were investigated. While counterexamples exist for $V^*$ aggregation, it is still of interest whether these are common limitations.

64 problems were generated which represented all possible combinations of the following parameters when they were allowed to vary in powers of 2 between their minima and maxima (note that not every available numerical combination yields a possible MDP):

| $|S|$ | $|\phi(S)|$ | $b$ | $|A|$ |
|-----|-------------|-----|-----|
| min | 4           | 2   | 2   | 1   |
| max | 64          | 32  | 32  | 8   |

Where $b$ is the “branching factor” of the transition matrix - the number of transitions out of each state for each particular action. It was required that $|S_\phi| \leq |S|/2$ and $b < |S|$.

4.2.1 $Q^*$ aggregation

For exact $Q^*$ aggregation, a target value vector $\mathbf{v}$ was formed by sampling $|S_\phi|$ real numbers $v_1^*, ..., v_{|S_\phi|}^*$ from the uniform distribution on $(0, 10|S_\phi|)$ and setting the first $|S_\phi|/|S|$ entries of $\mathbf{v}$ to $v_1^*$, the next to $v_2^*$ and so on. For each non-optimal action $a \neq a_1$ added to the problem, a further $|S_\phi|$ real numbers were sampled such that $v_j^* \sim U(v_j^* - 40, v_j^* - 1)$. The values $q^*(h, a)$ were initialised so that the Q-value of each state in the same group was identical.

Approximate $Q^*$ aggregation proceeded in almost the same manner, except a noise $\epsilon \sim (0, \epsilon_{max})$ was independently sampled for each state and added to the value function.
4.2.2 \( V^* \) aggregation

The target value vector for \( V^* \) aggregation was initialised in the same manner as the value vector for \( Q^* \) aggregation. Instead of sampling \( |S^\phi| \) numbers for suboptimal actions, however, \(|S|\) numbers were randomly sampled, one for each raw state, subject to the restriction that they were less than the optimal value for that state. Thus the Q-value functions were not \( \phi \)-uniform for problems generated for \( V^* \) aggregation.

4.3 Results

Part of the reason for this investigation was the question of whether or not Q-learning converges to the correct value function on MDPs if it is presented with an aggregated but non-Markovian representation of the state. It is, of course, impossible to conclusively demonstrate that Q-learning does or doesn’t converge precisely to the correct value function in the infinite limit by experiment. In all cases investigated, the value function qualitatively looked like it was converging, in some cases faster than others, and in most cases it the value function converged to the true value in fewer cycles under the aggregated representation than under the raw representation. The raw representation should always converge as the raw representation does yield an MDP. Thus, to study the performance of the aggregated representation, the relative error of the raw and aggregated value functions was taken. The relative error is defined as \( R = \frac{\Delta'}{\Delta^a} \), where \( \Delta'_n = \frac{1}{|S|} \sum_{a,s} |\hat{q}'_n(a,h) - q^*(a,h)| \) and \( \hat{q}'_n(a,s) \) is the q-value estimate of the raw agent at timestep \( n \). \( \Delta^a_n \) is similarly defined for the aggregated agent, except it is instead normalised by \( \frac{1}{|S^\phi|} \). In many cases the quantity log(\( R \)) is used, as this symmetrises relative differences in \( \Delta'_n \) and \( \Delta^a_n \) and led to a more stable regression fit.

Running Q-learning on the random MDPs produced yielded a dataset recording \( R \) and the problem parameters \(|S|\), \(|S^\phi|\), \( b \) and \(|A|\).

To explore the importance of different problem parameters, an ordinary least squares linear regression was run on the data with log(\( R \)) as the target value and explanatory variables

- \( n \) - the number of steps q-learning has taken
- \(|S|\) - the number of states in the MDP
- \( \frac{|S|}{|S^\phi|} \) - the number of states divided by the number of states after aggregation
- \(|A|\) - the number of actions

19
| Parameter | n | |S| | b | |A| | |S|/|S₀| |R²|
|---|---|---|---|---|---|---|---|---|---|---|
| V*—aggregation | 0.03 | -0.14 | 0.29 | 0.32 | 3.00 | 0.12 |
| Q*—aggregation | 0.06 | -0.24 | 0.81 | 0.17 | 1.93 | 0.26 |

Table 4: Ordinary Least Squares regression on the improvement in performance afforded

![Figure 3: log(R) against |S|/|S₀| for V* and Q* aggregation. log(R) > 0 implies the aggregated agent outperformed the raw agent.](image)

- **b** - the number of possible transitions for every (state, action) pair (identical for all problems studied)

All parameters were normalised to take values between 0 and 1 before the regression was performed, and a few extreme outliers were filtered out.

The data was randomly partitioned into a training and a test set. The fit was performed on the training set and the variance score $R^2$ calculated on the test set.

From Table 4 it is clear that the most significant effect of the difference in learning rate between the raw and aggregated problems is the “degree of aggregation” (ratio of $|S|$ to $|S₀|$), with a significantly better fit for $Q^*$ aggregation. In fact, a model with $|S|$/|S₀| as the sole explanatory variable minimised the Bayesian information criterion for both aggregation types. For $Q^*$ aggregation, the simple model achieved a BIC of 6 while the best two-variable model scored 12.5, while analogous models for $V^*$ aggregation scored 0.5 higher in each case.
log(R) is plotted against $|S|/|S_0|$ in Fig. 3. It is apparent from this plot that while the vast majority of the time the aggregated agent learned the value function faster, there were some tests for which the aggregated agent performed worse than the raw agent, particularly for low values of $|S|/|S_0|$. This could be due either to trial-to-trial variation, or because some problems were found for which no aggregation consistently performed better. To test this, ten problems were selected from those that contributed points below the line, and ten repeated trials were run with each. The mean of all trials was $\langle R \rangle = 2.5$ with standard deviation $\sigma = 3.0$. Only a single problem had a mean $R$ below 1, with $\langle R \rangle = 0.82$ and individually $\sigma = 0.74$. There is a probability of approximately 0.85 of getting at least one result this low if the true means were all above 1. This evidence suggests that the instances in which the raw agent performed better were due to random variation.

4.4 Dependence on $\epsilon$

Problems were separately generated to investigate the change in convergence when noise sampled independently for each state from Uniform(0, $\epsilon_{\text{max}}$) was added to the value function. For the problems described above, dependence was usually not visible even for relatively large values of $\epsilon_{\text{max}}$, which may have been because the absolute convergence of the randomly generated problems was often poor. To get a clearer picture of the dependence on $\epsilon_{\text{max}}$, MDPs with $|S| = 4$, $|S_0| = 2$, $b = 4$ and $a = 1$ were generated. After $1e4$ iterations, these had typically converged to an average deviation from the true value function of $\Delta \approx 0.01 - 1$, where the state values themselves were between 0 and 40.

Under these conditions, some dependence on $\epsilon_{\text{max}}$ is visible, as can be seen in Fig. 4. The lack of dependence for problems that exhibited slower convergence suggests that noise chiefly affects the ultimate precision achieved and not the rate of convergence.

4.5 Estimating the deviation from a Markovian process

The aggregated problems generated here are not necessarily Markovian, but it is interesting to investigate how much they deviate from Markov processes in practice. To estimate this quantity, the degree to which the process conditioned on the two most recent states differed from the process conditioned on the most recent state was used. Define a process $P$ with $N$ raw states (“histories”) and an aggregation $\phi$ with $n$ aggregated states $s_1, ... s_n \in S_\phi$. $S_t$ indicates the state of the process at time $t$. The induced process $P_\phi$ yields $n$ probability distributions $P_\phi(S_t = s|S_{t-1} = s')$, and $n^2$ distributions $P_\phi(S_t = s|S_{t-1} = s', S_{t-2} = s'')$ where $s, s', s'' \in S_\phi$. If the process were Markovian, then $P_\phi(S_t|S_{t-1} = s', S_{t-2} = s'') = P_\phi(S_t|S_{t-1} = s')$ for all $s, s', s'' \in S_\phi$. 
Define $D_{ss'}$ as the Kullback-Liebler divergence between the process conditioned on one and two timesteps:

$$D_{ss'} = D_{KL}(P_\phi(S_t|S_{t-1} = s, S_{t-2} = s') \parallel P_\phi(S_t|S_{t-1} = s))$$

$$= \sum_{s'' \in S_\phi} P_\phi(s''|s, s') \log \frac{P_\phi(s''|s, s')}{P_\phi(s''|s)}$$

Where we use the shorthand $P_\phi(s|s') := P_\phi(S_t = s|S_{t-1} = s')$.

To calculate the KL divergence, we need to calculate transition probabilities for the aggregated problem. To do this, we will exploit the fact that all aggregated problems we are studying have a stationary distribution $\rho(h)$ under the optimal policy in which every (raw) state has a nonzero occupation. For an agent following the optimal policy, then, each aggregated state has a well defined stochastic inverse $B(h|\phi(h)) = \frac{\rho(h)}{\sum_{h' \in \phi(s)} \rho(h')}$ which is nonzero for all $s \in \phi(s)$. Define the conditional distribution

$$\rho(s_t|\phi(s_{t-1})) := \sum_{s' \in \phi(s_{t-1})} P(s_t|s')B(s'|\phi(s_{t-1}))$$

We can then find the transition probabilities of the aggregated process by summing over
\[ s \in \phi(s_t): \]
\[ P_\phi(\phi(s_t)|\phi(s_{t-1})) = \sum_{s' \in \phi(s_t)} \rho(s'|\phi(s_{t-1})) \]

Defining the second-order stochastic inverse \( B(s_t|\phi(s_t), \phi(s_{t-1})) = \frac{\rho(s_t|\phi(s_{t-1}))}{\sum_{s' \in \phi(s_t)} \rho(s'|\phi(s_{t-1}))} \), we can follow similar steps to calculate the second-order transition probabilities of the aggregated process:
\[ P_\phi(\phi(s_t)|\phi(s_{t-1}), \phi(s_{t-2})) = \sum_{s' \in \phi(s_t)} \sum_{s'' \in \phi(s_{t-1})} P(s''|s') B(s'|\phi(s_{t-1}), \phi(s_{t-1})) \]

If we were analysing problems in which we might find \( \rho(s) = 0 \) for all \( s \) in some aggregated state \( \phi(s) \), we would need an alternative treatment for aggregated states with 0 probability. However, this is not necessary here as \( \rho(s) > 0 \) for all \( s \in S \) for the problems we will analyse.

The KL divergence is only defined if the first argument is 0 wherever the second argument is 0. We note above that if \( P_\phi(\phi(s_t)|\phi(s_{t-1})) = 0 \), then \( \rho(s|\phi(s_{t-1})) = 0 \) for all \( s \in \phi(s) \) and so \( P_\phi(\phi(s_t)|\phi(s_{t-1}), \phi(s_{t-2})) = 0 \) also. This is not necessarily true in the other direction.

Using distributions \( P_\phi \) calculated from \( P \) and \( \phi \) as described above, we calculate the \( n^2 \) divergences \( D_{ij} \). To estimate the “non-Markovianity” of the process, we then average these \( \hat{D} := \frac{1}{n^2} \sum i = 0^n \sum_j=0^n D_{ij} \).

We will finally note that \( \hat{D} > 0 \) implies that an aggregated problem is non-Markovian, but the converse is not true. We are checking the difference between the first and second order transitions as a matter of convenience, but a lack of distinguishability based on second order transitions doesn’t imply that this is true of higher order transitions.

### 4.6 non-Markovianity of Randomly Aggregated MDPs

To begin with, I investigated the behaviour of randomly aggregated MDPs with respect to the measure \( \hat{D} \). A number of MDPs were generated with varying parameters \( N := |S|, n_{agg} := S_\phi \) and \( b \) the “branching factor” which measures the average number of transitions from each state. The results are plotted in Fig. [5]. There is clearly a strong and highly predictable dependence of \( \hat{D} \) on all three parameters.

Two general trends which can be seen in the results are that \( \hat{D} \) seems to be maximal for \( n_{agg} \approx N/2 \) and for low \( b \). As expected, \( \hat{D} = 0 \) where \( N = n_{agg} \), because the aggregated problem is exactly the same as the raw problem in that case.
Figure 5: Variation of “non-Markovianity” \( \tilde{D} \) with parameters \( N, n_{agg} \) and \( b \). Points have been coloured by value of \( \tilde{D} \) in the last plot.
An informal reason for why $\tilde{D}$ might peak at $n_{\text{agg}} \approx N/2$ comes to mind: Knowing $\phi(s_{t-2})$ narrows down the number of possible raw states to $|\phi(s_{t-2})|$. Obviously, this is usually more informative if $|\phi(s_{t-2})|$ is small than if it is large. This does not necessarily imply that knowing $\phi(s_0)\ldots\phi(s_{t-2})$ is more informative where the average $|\phi(s)|$ is small, though; it could be the case that denser aggregations tend to benefit from longer histories.

The fact that $\tilde{D}$ increases with decreasing $b$ has a ready explanation. Informally again, a small $b$ means that $P(s_t|s_{t-1})$ is zero for many choices of $s_t$. Given that the remaining probabilities are randomly sampled, this suggests that $P(s_t|s_{t-1})$ will usually be more sharply peaked where $b$ is small; that is, it will have lower entropy. Holding a fixed $B(s|\phi(s))$, we would then expect a small $b$ to usually give a $\rho(s_t|\phi(s_{t-1}))$ with lower entropy. In other words, a small $b$ will usually mean that knowing $\phi(s_{t-1})$ will give us more information about $s_t$ than a similar problem with a large $b$.

The arguments presented are informal, as the questions addressed are not central to the analysis in this report. We are concerned with the performance of reinforcement learning agents on aggregated problems, and the degree to which these problems are Markovian does not have a strong theoretical bearing on this question.

The key question is whether $\tilde{D}$ has any predictive power regarding the performance of a Q-learning agent on an aggregated problem. To evaluate this, 70 random aggregated problems were generated with $N = 32$, $n_{\text{agg}} = 16$ and $b$ that varied from 4 to 32 in steps of 4. The choice of fixing $N$ and $n_{\text{agg}}$ was made because we know that $R$ depends strongly on $N/n_{\text{agg}}$, and we have theoretical reasons unrelated to $\tilde{D}$ to believe it should. $R$ may depend weakly on $b$, but fixing $b$ also leads to a very restricted range of $\tilde{D}$, so it was allowed to vary.

The results of the experiment are plotted in Fig. 6. There does not appear to be any relationship between $R$ and $\tilde{D}$ in the data. This test has many caveats and certainly does not exclude the possibility that certain types of non-Markovian behaviour pose a problem to a Q-learning agent. However, it does suggests that if a value function can be represented by a $Q^\dagger$-aggregation, whether or not the resulting problem is approximately Markovian does not appear to affect the performance of a Q-learning agent.

Several caveats of using $\tilde{D}$ have already been mentioned. A final significant caveat is the definition of Markovian used. Here, the Markovian property of the process is investigated in terms of the history of aggregated states, while in practice the Markovian property of interest is in terms of the history of raw observations. Thus perhaps the more relevant measure for the aggregated Markovian processes investigated here is how much better an outcome (rewards and observations) can be predicted if the agent is given access to the raw state instead of the aggregated state. This should not be too difficult to implement, but due to time constraints it hasn’t been done here.
Figure 6: $\log(R)$ against $\tilde{D}$ for a random selection of problems.
4.7 State Aggregation with the Mountain Car Problem

The MDPs generated specifically for the task of exploring value aggregation may not be very similar to any “real” problems in reinforcement learning. For a more natural test of state aggregation, aggregation was applied to the mountain car problem described by Sutton and Singh [14].

4.8 Experimental Setup

The mountain car problem, illustrated in Fig. 7, places a stationary car at $x \approx -0.4$ between two hills, and its goal is to climb the right-hand hill to reach the goal state at $x = 0.5$. The car may either accelerate forwards, accelerate backwards or coast, but the car’s engine is weaker than the force from gravity on the steepest slopes of each hill, so it must first travel backwards to build the momentum to escape. For each step that the car has not reached the exit state, it receives a reward of $-1$, and for the goal state it receives a reward of 0. Further details of the problem are described at [16].

To create aggregated states, a close approximation to the true value function of the task was required. The state space is continuous, so some sort of function approximation was required. A tile coding as described in [18][17] with 16 tilings was used as the approximator. For each episode the car started randomly positioned in the interval $(-1.2,0.5)$ with random velocity in the interval $(-0.07,0.07)$. A Q-learning agent reached the minimum average episode length after approximately 5000 episodes, and the agent was executed for a further 25 000 to learn a high fidelity representation of the value function. The value function learned can be seen in Fig. 8.
Figure 8: Left: Value function learned by the tile-coded Q-learning agent. Right: Visualisation of 16 aggregated classes. The classes are delineated by contour lines, and some classes have multiple separated regions. The classes visibly trace the value function learned by the tile-coded agent.

This function was then used to generate an aggregation $\phi(s)$. First, the state space was discretised into $100 \times 100$ equally sized position and velocity states, and a value function derived from the tile coding estimate was mapped to these cells.

The q-values of each cell form points in $\mathbb{R}^3$. To produce an aggregation, K-means clustering was applied to the cells on the basis of their q-values with a custom metric designed for the purpose. The metric $\rho_q : \mathbb{R}^3 \times \mathbb{R}^3 \rightarrow \mathbb{R} \cup \{\infty\}$ was defined as follows for two states $c_1, c_2 \in \mathbb{R}^3$:

$$
\rho_q(c_1, c_2) = \begin{cases} 
\infty & \pi^*(c_1) \neq \pi^*(c_2) \\
\max_a |q(c_1, a) - q(c_2, a)| & \text{otherwise}
\end{cases}
$$

A visualisation of the classes derived using $k = 16$ classes is presented in Fig. 8.

$V^*$ aggregation was also trialled. For $V^*$ aggregation, a slightly different metric $\rho_v$ was used:

$$
\rho_v(c_1, c_2) = \begin{cases} 
\infty & \pi^*(c_1) \neq \pi^*(c_2) \\
|c(c_1) - c(c_2)| & \text{otherwise}
\end{cases}
$$

4.9 Results

The performance of the aggregated agent with state representation $\phi_a(s)$ was compared to a discretised agent using a square grid state representation $\phi_d(s)$ with the same total
number of states. The relative performance is plotted in Fig. 10 and some key numbers are summarised in the table 9.

<table>
<thead>
<tr>
<th>$\phi_a(S)$</th>
<th>$\phi_d(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>episode length, $c = 32$</td>
<td>555</td>
</tr>
<tr>
<td>episode length, $c = 1024$</td>
<td>159</td>
</tr>
<tr>
<td>episode length, $c = 32768$</td>
<td>200</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>21</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi_a(S)$</th>
<th>$\phi_d(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>episode length, $c = 32$</td>
<td>473</td>
</tr>
<tr>
<td>episode length, $c = 1024$</td>
<td>186</td>
</tr>
<tr>
<td>episode length, $c = 32768$</td>
<td>144</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>10</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi_a(S)$</th>
<th>$\phi_d(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>episode length, $c = 32$</td>
<td>848</td>
</tr>
<tr>
<td>episode length, $c = 1024$</td>
<td>250</td>
</tr>
<tr>
<td>episode length, $c = 32768$</td>
<td>128</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\phi_a(S)$</th>
<th>$\phi_d(S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>episode length, $c = 32$</td>
<td>1465</td>
</tr>
<tr>
<td>episode length, $c = 1024$</td>
<td>420</td>
</tr>
<tr>
<td>episode length, $c = 32768$</td>
<td>124</td>
</tr>
<tr>
<td>$\epsilon$</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 9: Average episode lengths for a $Q^*$ aggregated agent on the mountain car task. $|\phi_a(S)|$ refers to the number of states of the aggregated agent, $|\phi_d(S)|$ refers to the number of states of the discretised agent and $c$ is the number of completed episodes. The original tile coded agent had an average episode length of 121 once the value function converged.

The performance of $Q^*$ and $V^*$ aggregation was nearly identical given the same number of target states. Both were able to approximate the value function significantly better than a discretised agent with the same number of states, and in many cases better than said agent with significantly more states. The coarser aggregations $|\phi_a(S)| = 16$ and $|\phi_d(S)| = 64$ also exhibited somewhat faster initial convergence than the finer aggregations, though they were ultimately unable to approximate the value function as precisely.
Figure 10: Comparison of gridded and aggregated state representations.
4.10 Agreement with theoretical limit

We examine here whether the results found are in agreement with the theoretical limit derived by Hutter in [2]. This limit tells us that, given an upper bound $\epsilon \geq |q^*(s, a) - q^*(s', a)|$ for all $s \in \phi(S)$, we should expect

$$|V^*(s, a) - v^*(\pi(s), a)| \leq \frac{\epsilon}{1 - \gamma}$$  \hspace{1cm} (22)

This limit was derived in the context of rewards being between 0 and 1, while the mountain car task has rewards of $-1$ and 0. A task yielding the same optimal policy can be defined in which the agent receives reward 0 for all transitions except the final one, for which it receives reward 1. Thus the value of a state with $d$ steps until the terminal state is reached under the optimal policy is $\gamma^d$. We require $0 < \gamma < 1$ here, but $\gamma < 1$ is necessary to make Eq. useful in any case.

Helpfully, the values of states in the original mountain car task are approximately the negative of the number of steps remaining in the episode under the optimal policy. If we denote by $q$ the q-value function of the original task and $q_\gamma$ the q-value function on the modified task, letting $d = \max(q^*(s, a), q^*(s, a))$ we can show

$$|q_\gamma^*(s, a) - q_\gamma^*(s', a)| = |\gamma q^*(s, a) - \gamma q^*(s', a)| \leq \gamma^d - \gamma^{d+\epsilon} \leq (1 - \gamma^\epsilon)$$  \hspace{1cm} (23)

With the third line following from $d \geq 1$ for all $s \in \phi(S)$.

Combining Eq. 4.10 with 23, we get

$$\gamma^V - \gamma^v \leq \frac{1 - \gamma^\epsilon}{1 - \gamma}$$

Given $\epsilon > 1$ (which is true of all cases studied here), the right-hand side is greater than 1, while the left hand side must be less than 1. Thus this limit is quite straightforwardly satisfied in the case at hand. This seems to be partly a product of the fact that $\epsilon$ was quite large for any aggregation with a reasonable number of states. This suggests that the bound given by equation 4.10 may be quite loose for some problems, and a fairly relaxed aggregation may perform quite well.
5 Learning the Aggregation

The work so far discusses the performance of agents on aggregated problems with aggregations derived from a precise knowledge of the fundamental problem’s true value function. This is, of course, an unrealistic requirement for any useful agent. If the designers already know the value function, they would have little need for an agent to learn it. It is important to ask how $Q^*$-aggregation can be used to improve the performance of agents when we do not know the value function we wish them to learn. There are two main avenues through which aggregation could help an agent to learn a policy: first, an aggregation reduces the number of states in a problem, and so the same amount of exploration will provide more samples per state (this is especially significant if we are comparing an aggregation to a problem working with the entire history, as each history can have at most one sample). This much has been demonstrated by the experiments conducted so far, but to take advantage of this we require agents that can learn an aggregation. Secondly, if an agent performs some sort of search through a space of possible environment representations, if we require that the search space contains a $Q^*$-aggregation of the true environment rather than an exact representation, we may be able to define a smaller space to be searched.

5.1 Existing work

Extensive work exists on agents which learn an adaptive representation of a state space. Munos and Moore [7] have examined approaches to refining a discretisation of a continuous state space. Their approach represents the state space in a structure known as a \textit{kd-trie} that divides the state space into hierarchically organised rectangles and defines a value at the corner of each “leaf” rectangle. The value of a particular point is linearly interpolated from the corners that surround it. The state space can be refined by splitting these rectangles, and the authors test a number of local heuristics for deciding when to do this:

- The average difference in value between the corners of a rectangle
- The variance in the value of the corners of a rectangle
- Policy disagreements between the corners of a rectangle

These local heuristics were found to offer reasonable performance for a 2D car on a hill task similar but not identical to the mountain car task discussed here, but these heuristics struggled in higher dimensions. They also proposed two non-local criteria involving measures they termed \textit{influence}, that measures the degree to which one state influences
other states in the problem, and variance which is the expected squared deviation of actual returns from the value function of that state. These global measures were found to perform better on higher dimensional tasks. Reynolds [12] employs a method for refining continuous state space discretisations similar to the policy disagreement heuristic trialled by Munos and Moore.

McCallum [6] has developed the U-Tree algorithm for discrete state spaces, which maintains a tree representation of the state space and uses a statistical test on the transitions observed each time a state has been visited to determine if dividing the state could better represent the value function. Uther and Veloso [19] extend U-Tree to continuous state spaces. A similar approach is taken by Pyeatt and Howe [11], who instead of recording the full history of actions and observations, record the history of value updates for each leaf node to decide when to split states.

5.2 U-Tree

McCallum’s U-Tree algorithm represents a model of its environment in a tree, in which the leaves represent the states and each internal node is associated with a distinction. An internal node’s distinction maps a partial history \( h_k = o_1a_1r_2...a_{k-1}r_ko_k \) to exactly one of that node’s children on the basis of a particular component of observation \( o_{k-d} \) or action \( a_{k-d-1} \) where the time index \( d \) is a parameter of the distinction. Thus every partial history of the agent up to \( h_t \) is associated with one of the agent’s leaf nodes.

At regular intervals, the agent computes candidate distinctions for each of its leaf nodes. That node is expanded into a number of “fringe nodes”, and partial histories are associated with the fringe nodes of the candidate distinction. The q-values of each history are calculated by \( q_p(h_k) = r_{k+1} + \gamma v(\phi(h_{k+1})) \). If a Kolmogorov-Smirnoff test indicates that the values of the histories of different fringe nodes shows sufficient evidence of being drawn from different distributions, the candidate distinction is accepted and the fringe nodes are converted to leaf nodes.

This heuristic for splitting states attempts to distinguish states that have different q-values, and this aligns it in principle with \( Q^* \)-aggregation. In practice the algorithm cannot detect every possible q-value distinction. Distinctions are drawn on the basis of an observation or action and a time index - for example, a distinction might be drawn on the basis of a component of the \( o_{t-3} \) observation for a given partial history to time \( t \). If there is a relevant distinction to be drawn on the basis of two observation components from, say \( o_{t-3} \) and \( o_{t-2} \), then U-Tree can learn the distinction if a relevant distinction can be drawn of \( o_{t-3} \) or \( o_{t-2} \) individually. It is possible to modify U-Tree to expand two layers of fringe nodes, which would permit it to learn all distinctions that require two observation components, but increasing the fringe depth comes with an exponential cost in computation time and in the number of samples needed to make meaningful tests.
Nonetheless, U-Tree is an algorithm that learns value distinctions and, as far as possible, learns only value distinctions, which is what is wanted for a $Q^*$-aggregation.

An implementation of U-Tree was tested with the cases investigated so far - randomly generated “aggregatable” MDPs, and the mountain car task. For the randomly generated problems, the agent was presented with the index of the raw state, the aggregated state, or both. For the mountain car task, the agent was presented with the car’s position and velocity, and it was able to make state divisions by splitting the current state’s position and velocity ranges in half.

U-Tree cannot make distinctions coarser than the perceptions given to it, so if it is given the perceptions of the raw states of an MDP it cannot learn an aggregation of that MDP. In practice, when given raw MDP states as observations, the agent typically learned to use only the most recent observation to draw distinctions, which is the minimal number of distinctions it was able to learn. Similarly, when given aggregated states as observations, the agent typically added distinctions based only on the most recent observation though it was somewhat less likely to make a distinction at all.

When presented with the raw and aggregated observations combined, the agent never opted to distinguish states based only on the aggregated states. Either the agent would distinguish states based on the raw observation, or it would distinguish states based on aggregated observations and raw observations (see Fig. 11). This latter feature may be because U-Tree adds distinctions when it detects samples drawn from different

Figure 11: Visualisation of the state representation learned by U-Tree in one run of a problem with 4 raw states, 2 aggregated states and 1 action. The observation vector presented to U-Tree was $(\phi(s), s)$. U-Tree first distinguished on the aggregated observation, and then further distinguished on the raw observation, even though the value function was quite uniform throughout.
Figure 12: The number of states in the representation learned by U-Tree. The combined representation was in principle able to settle on an 8-state representation, but it never did.

Ideally, we would want an agent that learns that the $Q^*$-aggregated representation is sufficient, but the straightforward implementation of U-Tree did not do this. In all cases studied U-Tree made the most helpful distinctions first.

The implementation of U-Tree tested was not able to find a useful state representation for the mountain car task. The culprit appeared to be the Kolmogorov-Smirnov test. The Kolmogorov-Smirnov test gives a likelihood that two sets of samples are drawn from different distributions by looking at the maximum deviation of the cumulative distributions of the samples. If both sample sets contain a large number of identical
samples and just a few that differ greatly, the Kolmogorov-Smirnov test will not assign a high likelihood of different distributions for each. In the random exploration phase of the mountain car task, the agent will typically take around 20000 steps of exploration with the same reward of $-1$ before a single step terminates the episode with a reward of 0, so this presents a extremely adversarial case for the Kolmogorov-Smirnov test.

Three issues were found with the U-Tree algorithm:

1. The minimal representation the agent can learn is limited by the perceptions given to the agent
2. A node is split if the values of its partial histories appear to be drawn from different distributions, but if these distributions have the same mean then that should be sufficient for $Q^*$-aggregation
3. The Kolmogorov-Smirnov does not assign any weight to how far samples deviate between sample sets as long as it is only a few samples that do deviate

Points 2 and 3 suggest that it is worth exploring different splitting criteria. One criterion which appears to address both concerns is used by Uther and Veloso [19], in which the sum-squared error of the partial histories’ $q$-values and the state’s $q$-value is considered.

If a problem is structured so that there is some a priori reason to believe some states may be aggregated together, it is possible to add a preprocessing step to U-Tree and add preprocessed features as extra dimensions to the percept vector. For appropriate problems, this may allow a U-Tree agent to learn coarser distinctions than its raw percept sequence permits. If U-Tree also retains access to raw percepts, it may be possible to learn a mixed state representation, drawing distinctions on preprocessed features where appropriate and on raw percepts where necessary. More investigation is required to determine the viability of this approach, however.

### 5.3 Exploration Algorithms

Exploration algorithms aim to learn the optimal policy for an environment while exploring in a manner that is in some sense optimal. Measures of optimality often include regret, the amount of reward the agent missed in comparison to an agent that acts optimally from the start, or sample complexity which is the number of timesteps for which the agent’s value function is more than $\epsilon$ from the optimal value function. Exploration algorithms often operate by maintaining a class $\mathcal{M}$ of plausible environments and gradually reducing this class as more information is learned. $Q^*$-aggregation may benefit exploration algorithms by weakening assumptions they must make of models in $\mathcal{M}$, such as that the true environment $\mu \in \mathcal{M}$ or that models in $\mathcal{M}$ must be MDPs.
The UCRL(γ) algorithm [3] is an exploration algorithm for finite state discounted MDPs with a known state count |S| based on a modification of the UCRL2 algorithm. The BLB algorithm [5] works with a set Φ of state representations, at least one of which is a Markov representation of the true environment φμ. BLB uses the UCRL2 algorithm to learn the transition structure of the models φ ∈ Φ, and optimistically explores this set of models, eliminating models that predict returns too different from those actually obtained by “exploiting” that model. MERL [4] is an algorithm for general reinforcement learning with discounted rewards that learns an approximately optimal value function given a finite set of models M that contain the true environment μ.

This work extends the MERL algorithm to the case where an exact Q∗-aggregation φμ ∈ M, but the true environment μ is not known to be in M.

5.4 The MERL Algorithm

Given a finite set of models M that contains the true environment model μ with |M| = N, a precision ε and a confidence δ a sample complexity bound for MERL has been derived by [4]. With probability 1 − δ, the number of timesteps MERL runs for until it’s policy π is ε-optimal is less than

\[ \hat{O}\left( \frac{N}{\varepsilon^2(1 - \gamma)^3} \log^2 \frac{N}{\delta \varepsilon (1 - \gamma)} \right) \]

We are interested here in whether a similar bound holds if we relax the assumption that the true environment μ ∈ M, requiring only an exact aggregation of the true environment φ(μ) ∈ M. I show that if φ(μ) yields a Markov decision process, then MERL yields the same sample complexity bound. On the other hand, in the general case of exact value aggregation a slightly modified version of MERL named MERL-A yields the a bound with differences of lower order logarithmic factors.

The full proof for the sample complexity bound is given in [4]. Here I will focus on elements of the algorithm and proof that must be altered under these different assumptions, including parts of the original as they are needed to explain the argument.

5.4.1 Notation

Sample complexity: A policy π is ε-optimal in environment μ for history h if \( V^*_\mu(h) - V^*_\pi(h) < \varepsilon \). The sample complexity of a policy π is the lowest Λ such that with high
probability, $\pi$ is $\epsilon$-optimal for all but $\Lambda$ timesteps. Define $L_{\mu,\pi}^\epsilon : \mathcal{H}^\infty \to \mathbb{N} \cup \{\infty\}$ to be the number of times $\pi$ is not $\epsilon$-optimal:

$$L_{\mu,\pi}^\epsilon(h) = \sum_{t=1}^{\infty} [V_\mu^*(h_t) - V_\mu^\pi(h_t) > \epsilon]$$

The sample complexity of $\pi$ is $\Lambda$ with respect to accuracy $\epsilon$ and confidence $\delta$ if $P(L_{\mu,\pi}^\epsilon(h) > \Lambda) < \delta$ for $\forall \mu \in \mathcal{M}$.

**D-step return:** the expected d-step return is defined as

$$V_\mu^\pi(h_t; d) = \mathbb{E}[\sum_{k=t}^{t+d} \gamma^{k-t} r_k | h_t]$$

The value function is the expected return in the infinite limit: $V_\mu^\pi(h_t) = \lim_{d \to \infty} V_\mu^\pi(h_t; d)$.

In addition, define the empirical d-step return $R_t := \sum_{j=t}^{t+d} \gamma^{j-t} r_j$.

**Aggregation:** given an aggregation $\phi : \mathcal{H} \to \mathcal{S}$, we define the induced process

$$P_\phi(s_{t+1}|h_t) := \sum_{\tilde{h}_{t+1}: \phi(\tilde{h}_{t+1}) = s_{t+1}} P(\tilde{h}_{t+1}|h_t)$$

An aggregated environment representation $\phi$ is a Markov aggregation iff the process $P_\phi$ is Markov with respect to states $s \in \mathcal{S}$.

$$P_\phi(s_{t+1}r_{t+1}|\tilde{h}_t a_t) = P(s_{t+1}r_{t+1}|s_t a_t) \quad \forall \phi(\tilde{h}_t) = s_t$$

**5.4.2 Algorithm**

At each time step $t$, MERL computes the policy $\pi$ and the environments $\nu, \sigma \in \mathcal{M}$ maximising

$$\Delta := V_\pi^\nu(h_t; d) - V_\pi^\sigma(h_t; d)$$

If $\Delta > \frac{\epsilon}{4}$, then MERL enters an exploration phase, following policy $\pi$ for $d$ time steps. Otherwise, MERL will follow the optimal policy with respect to the first model in $\mathcal{M}$ for one time step.

An exploration phase is a $\kappa$-exploration phase if $\Delta \in [\epsilon 2^{\kappa-2}, \epsilon 2^{\kappa-1})$, where

$$\kappa \in \mathcal{K} = \{0, 1, 2, ..., \log_2 \left( \frac{1}{\epsilon (1 - \gamma)} \right) + 2\}$$
For each environment $\nu$ and $\kappa$, MERL keeps a count $E(\nu, \kappa)$ of the number of $\kappa$-exploration phases associated with that environment. With each phase, it notes the empirical return $R_t$ received during that phase and records the values of $X(\nu, \kappa) = (1 - \gamma)(V^\pi_\nu(h_t; d) - R_t)$ and $X(\nu, \kappa) = (1 - \gamma)(R_t - V^\pi_\nu(h_t; d))$, where $h_t$ is the history at the beginning of the phase.

MERL will eliminate environment $\nu$ if the cumulative sum of $X(\nu, \kappa)$ is sufficiently large, but it only tests this when $E(\nu, \kappa)$ has increased sufficiently since the last test. In particular, it tests the sum of $X(\nu, \kappa)$ whenever $E(\nu, \kappa) = \lceil \alpha j \rceil$ for some $j \in \mathbb{N}$ and $\alpha := \frac{4\sqrt{N}}{4\sqrt{N} - 1} \in (1, 2)$. If the sum of $X(\nu, \kappa)$s is greater than $\lambda(\alpha_j) := \sqrt{2\alpha_j \log \frac{\alpha_j}{\delta_1}}$, then environment $\nu$ is eliminated. Here, $\delta_1 := \frac{\delta}{32|K|N^{3/2}}$.

For later analysis of the algorithm, we define a failure phase as a period of $d$ timesteps beginning at time $t$ if

- $t$ is not part of an earlier exploration phase or failure phase
- $V^*_\mu(h_t) - V^\pi_\mu(h_t) > \epsilon$

An exploration or failure phase is proper if $\mu \in \mathcal{M}$ at time $t$ for MERL. For MERL-A a phase is proper if $\phi(\mu) \in \mathcal{M}$ at time $t$.

A pseudocode description of MERL is given in [4]. Here I’ll detail the closely related MERL-A, which differs only in the parameter $d$ and the criterion for eliminating models.

### 5.4.3 D-step returns for Markov aggregation

If the aggregated problem is Markovian, then the same sample complexity bound holds as in [4].

**Lemma 1:** The d-step return of any state $s$ of an exact Markov aggregation $\phi_\mu$ is identical to the d-step return of every history $\tilde{h}_t$ for which $\phi_\mu(\tilde{h}_t) = s$.

**Proof:** The value of state $s_t$ and history $h_t$ can be written in terms of its d-step return:

$$
V^\pi_\phi(\phi_\mu(h_t)) = V^\pi_\phi(\phi_\mu(h_t; d)) + \gamma^d \sum_{h_{t+d} \in \Phi_\mu} P(h_{t+d}|h_t, \pi(\phi_\mu(h_t))) V^\pi_\phi(h_{t+d})
$$

$$
V^\pi_\phi(h_t) = V^\pi_\phi(h_t; d) + \gamma^d \sum_{h_{t+d}} P(h_{t+d}|h_t, \pi(h_t)) V^\pi_\phi(h_{t+d})
$$

39
\begin{align*}
 \text{Inputs: } & \epsilon, \delta \text{ and } M = \{\nu_1, \nu_2, ..., \nu_N\} \\
 & t = 1 \text{ and } h \text{ is the empty history} \\
 & d := \frac{1}{1 - \gamma} \log \left( \frac{\epsilon(1 - \gamma)}{24 \sqrt{N}} \right), \delta_1 = \frac{\delta}{32 |K| N^{\alpha / 2}} \\
 & \alpha := \frac{4\sqrt{N}}{4\sqrt{N - 1}}, \alpha_j := \lceil \alpha^j \rceil \\
 & E(\nu, \kappa) := 0 \text{ for all } \nu \in M, \kappa \in N \\
 \text{loop} & \quad \text{repeat} \\
 & \quad \quad \Pi := \{\pi^*_\nu : \nu \in M\} \\
 & \quad \quad \nu, \nu_0, \pi := \arg \max_{\nu, \nu_0 \in M, \pi \in \Pi} (V^\pi_\nu(h; d) - V^\pi_{\nu_0}(h; d)) \\
 & \quad \quad \text{if } \Delta := V^\pi_\nu(h; d) - V^\pi_{\nu_0} > \epsilon / 4 \text{ then} \\
 & \quad \quad \quad \tilde{h} = h \text{ and } R = 0 \\
 & \quad \quad \quad \text{for } j = 0 \rightarrow d \text{ do} \\
 & \quad \quad \quad \quad R = R + \gamma^j r_t(h) \\
 & \quad \quad \quad \quad \text{ACT}(\pi) \\
 & \quad \quad \quad \end{align*}

\begin{align*}
 & \quad \text{end} \\
 & \quad \kappa := \min \{\kappa \in N : \Delta > \epsilon 2^{\kappa - 2}\} \\
 & \quad E(\nu, \kappa) = E(\nu, \kappa) + 1 \text{ and } E(\nu, \kappa) = E(\nu, \kappa) + 1 \\
 & \quad X(\nu, \kappa) E(\nu, \kappa) = (1 - \gamma)(V^\pi_{\nu_0}(h; d) - R) \\
 & \quad X(\nu, \kappa) E(\nu, \kappa) = (1 - \gamma)(V^\pi_{\nu_0}(h; d) - R) \\
 & \quad \text{else} \\
 & \quad \quad i := \min \{i : \nu_i \in M\} \text{ and ACT}(\pi^*_\nu_i) \\
 & \quad \text{end} \\
 & \quad \text{end} \\
 & \quad \exists \nu \in M, \kappa, j \in N \text{ such that } E(\nu, \kappa) = \alpha_j \text{ and} \\
 & \quad \sum_{i=1}^{E(\nu, \kappa)} X(\nu, \kappa)_i \geq \sqrt{2E(\nu, \kappa) \log \frac{E(\nu, \kappa)}{\delta_1} + \frac{E(\nu, \kappa)\epsilon}{2\sqrt{N}}} \\
 & \quad \text{; } M = M\backslash\{\nu\} \\
 & \text{end} \\
 & \text{Fn ACT}(\pi): \quad \text{Take action } a_t = \pi(h) \text{ and receive reward and observation } r_t, o_t \text{ from environment} \\
 & \quad t \leftarrow t + 1 \text{ and } h \leftarrow ha_t o_t r_t \\
 & \text{end}
\end{align*}

\textbf{Algorithm 1:} MERL-A algorithm
It has been shown by [2] that the (infinite horizon) value function of a state $s_t$ $v_{\phi\mu}(s_t)$ agrees with the values of all histories mapped to that state

$$v_{\phi\mu}(s_t) = V_{\phi}(h_t) \quad \forall \phi(h_t) = s_t$$

Combining this with the definition of $P_{\phi}$, we can rewrite (25):

$$V_{\mu}(h_t) = V_{\mu}(h_t; d) + \gamma^d \sum_{s_{t+d}} P(s_{t+d}|h_t\pi(h_t))V_{\mu}(s_{t+d})$$

$$= V_{\phi\mu}(s_t)$$

We now note that the Markov property gives us $P(s_{t+d}|h_t\phi(h_t)) = P(s_{t+d}|s_t\phi(s_t))$. Thus the sums in (24) and (25) are equal, and so the remaining terms must be equal:

$$v_{\phi\mu}(s_t; d) = V_{\mu}(h_t; d)$$

\[\square\]

### 5.4.4 Sample complexity bound for exact Markov aggregation

The assumption that the true environment $\mu \in M$ only enters the proof of a sample complexity bound via it’s value function and d-step return. Both of these quantities are preserved identically in a Markov aggregation, so the same proof holds.

### 5.4.5 D-step returns for exact q-value aggregation

When we drop the assumption that the aggregated problem $\phi(\mu)$ forms an MDP, we can no longer claim that the d-step returns of the aggregated problem are the same as those of the true environment. Fig. 13 shows an example of an aggregated problem where the expected d-step returns for states $A$ and $B$ differ by $\frac{2^d}{1-\gamma}$ despite them having the same value. The d-step return for $\phi(A)$ isn’t uniquely defined, but it must be somewhere between $V_{\mu}(A; d)$ and $V_{\mu}(B; d)$, and must disagree with at least one of them. Fig. 13 does give us an upper limit on how much the d-step returns may disagree; given some $\gamma$ and $d$, the returns may differ by no more than $\frac{2^d}{1-\gamma}$.

A sample complexity bound still holds in this situation, but it requires significant modification of the algorithm.
5.4.6 A sample complexity bound for exact q-value aggregation

Define $E_{\text{max}} := \frac{2^{18}N}{\epsilon^2(1-\gamma)^2} \log^2 \frac{2^{15}N}{\epsilon^2(1-\gamma)^2\delta_1}$ and $G_{\text{max}} := \frac{2^{19}N|K|}{15\epsilon^2(1-\gamma)^2} \log^2 \frac{2^{15}N}{\epsilon^2(1-\gamma)^2\delta_1}$. $E_{\text{max}}$ and $G_{\text{max}}$ represent high probability upper bounds on the number of exploration phases and the number of failure phases respectively.

The proof for a sample complexity bound for MERL-A rests on three lemmas:

**lemma 2**: The true environment aggregation $\phi(\mu)$ remains in $M_t$ for all $t$ with probability $1 - \delta/4$.

**lemma 3**: The number of proper failure phases is bounded by $G_{\text{max}}$ with probability $1 - \delta/2$.

**lemma 4**: The number of proper exploration phases is bounded by $E_{\text{max}}$ with probability $1 - \delta/4$.

**Theorem 1** Let $\mu$ be the true environment and $\phi(\mu) \in M$ be an exact q-aggregation of $\mu$. Let $\pi$ be the policy of Algorithm 1. Then

$$P\{L_{\mu,\pi}^\epsilon(h) \geq d \cdot (G_{\text{max}} + E_{\text{max}})\} \leq \delta$$

If lower order logarithmic factors are dropped, this is the same as the sample complexity bound for MERL-A: $O\left(\frac{N}{\epsilon^2(1-\gamma)^2} \log^2 \frac{N}{\delta\epsilon(1-\gamma)}\right)$

**Proof of Theorem 1**: By lemma 2, all exploration and failure phases are proper with probability $1 - \delta/4$. If the bounds of lemma 2, 3 and 4 hold, then the number of steps
for which MERL-A is more than $\epsilon$ from the correct value function is $d(E_{\text{max}} + G_{\text{max}})$.

By the union bound, the bounds for lemma 2, 3 and 4 will all hold together with probability $1 - \delta$. \qed

**Lemma 2** I will begin by giving a modified proof of lemma 2. Here, I will show that with a modification to the algorithm, the aggregated environment $\phi(\mu)$ remains in $\mathcal{M}_t$ for all $t$ with probability $1 - \frac{\delta}{4}$.

Recall that MERL will eliminate an environment from its set of possible environments if

$$\sum_{i=1}^{\alpha_j} X(\nu, \kappa)i \geq \sqrt{2\alpha_j \log \frac{\alpha_j}{\delta_1}}$$

In the original proof, if the true environment $\mu \in \mathcal{M}$ then defining $B_{\alpha_j}^\mu = \sum_{i=1}^{\alpha_j} X(\mu, \kappa)i$ it can be shown that $P(B_{\alpha_j}^\mu \geq \sqrt{2\alpha_j \log \frac{\delta_1}{\alpha_j}}) \leq \frac{\delta_1}{\alpha_j}$, which is ultimately sufficient to show that $\mu$ will remain in $\mathcal{M}_t$ with probability $1 - \frac{\delta}{4}$ [4]. This hinges on the fact that $B_{\alpha_j}^\mu$ is a Martingale with expectation 0, which follows from the fact that $E[\mu][X(\mu, \kappa)] = 0$. This is not necessarily true if $\phi(\mu) \in \mathcal{M}$ but $\mu \notin \mathcal{M}$.

Defining $Y_{\alpha_j} = \sum_{i=1}^{\alpha_j} (X(\phi(\mu), \kappa)i - E[X(\phi(\mu), \kappa)i])$, it is easy to see that $Y_{\alpha_j}$ is a Martingale with expectation 0. Using an argument identical to [4], can then conclude that $P(Y_{\alpha_j} \geq \sqrt{2\alpha_j \log \frac{\delta_1}{\alpha_j}}) \leq \frac{\delta_1}{\alpha_j}$. Furthermore, using the upper bound $E[\mu][X(\phi(\mu), \kappa)] \leq \gamma d$, we can show that $Y_{\alpha_j} \leq B_{\alpha_j}^{\phi(\mu)} - \alpha_j \gamma d$ and so $P(B_{\alpha_j}^{\phi(\mu)} \leq \sqrt{2\alpha_j \log \frac{\delta_1}{\alpha_j}} + \alpha_j \gamma d) \leq \frac{\delta_1}{\alpha_j}$, which again will imply that $\phi(\mu)$ will remain in $\mathcal{M}_t$ with probability $1 - \frac{\delta}{4}$.

Thus, by altering the elimination criterion for MERL to

$$\sum_{i=1}^{\alpha_j} X(\nu, \kappa)i \geq \sqrt{2\alpha_j \log \frac{\alpha_j}{\delta_1}} + \gamma d \alpha_j$$

We can have high confidence that our target environment will not be eliminated. For a sample complexity bound to hold, however, we need to also show that environments will be eliminated quickly enough given our relaxed condition for elimination.

**Lemma 4** The proof here follows the first part of the argument in [4] closely. Recall that $E_l(\nu, \kappa)$ counts the number of exploration phases involving environment $\nu$ of length $\kappa$ at time $t$. $E_{\nu, \kappa} = \sum_{\nu} E_l(\nu, \kappa)$ is twice the total number of exploration phases at time
$t$ and $E_{\infty, \kappa} := \lim_t E_{t, \kappa}$ is twice the number of exploration phases in total. $E_{\max, \kappa}$ is a constant to be defined later.

A $(\nu, \kappa)$ exploration phase is $\nu$-effective if
\[
E[X(\nu, \kappa)|\mathcal{F}_t] = (1 - \gamma)(V^\nu_\mu(h_t; d) - V^\nu_\nu(h_t; d)) > (1 - \gamma)\frac{\epsilon_\kappa}{2}
\]
and $\nu$-effective if an analogous condition holds for $\nu$. Supposing $\phi(\mu) \in \mathcal{M}_t$, we know
\[
V^\nu_\mu(h_t; d) + \epsilon/8 \geq V^\nu_\mu(h_t; d) \geq V^\nu_\nu(h_t; d) - \epsilon/8
\]
\[
V^\nu_\mu(h_t; d) - V^\nu_\nu(h_t; d) > \epsilon_\kappa
\]
The first follows from our limit on the variation of $d$-step returns, and the second is by definition in the algorithm. Thus every exploration phase must be $\nu$-effective or $\nu$-effective or both. We will denote by $F_t(\nu, \kappa)$ the number of $\nu$-effective $(\nu, \kappa)$ exploration phases up to time $t$.

We are interested in the chance that $E_{\infty, \kappa} > E_{\max, \kappa}$. From [4], if there is a $t'$ and $\nu$ such that $E_{t', \nu} > E_{\max, \kappa}$ then
\[
\sqrt{\frac{E_{\max, \kappa} E_{t', \nu}}{4N}} \geq \sqrt{\frac{E_{\max, \kappa} E_{t', \nu}}{4N}} \geq \frac{E_{t', \nu}}{\sqrt{4N}} \tag{26}
\]
For a fixed $\nu \in \mathcal{M}$, let $X_1, X_2, ..., X_{E_{\infty, \nu, \kappa}}$ be a sequence with $X_i := X(\nu, \kappa)_i$ and $t_i$ the time step at the start of the $i$th $(\nu, \kappa)$-exploration phase. Define
\[
Y_i := \begin{cases} X_i - E[X_i|\mathcal{F}_{t_i}] & \text{if } i \leq E_{\infty}(\nu, \kappa) \\ 0 & \text{otherwise} \end{cases}
\]
If a $t'$ exists as above, then there is a largest $t \leq t'$ such that $E_t(\nu, t) = \alpha_j$ for some $j \in \mathbb{N}$. Again from [4], we have $t$ satisfying
\[
1. \ E_t(\nu, \kappa) = \alpha_j \text{ for some } j
\]
\[
2. \ E_{\infty}(\nu, \kappa) > E_t(\nu, \kappa)
\]
\[
3. \ F_t(\nu, \kappa) \geq \sqrt{\frac{E_{t, \nu, \kappa} E_{\max, \kappa}}{16N}}
\]
\[
4. \ E_t(\nu, \kappa) \geq \frac{E_{\max, \kappa}}{16N}
\]
Define $\lambda_1(E) = \sqrt{2E \log \frac{E}{\delta_1}}$ and $\lambda_2(E) = \lambda_1(E) + E^{\gamma(1-\gamma)/2N}$. Here, $\lambda_1(\cdot)$ corresponds to the original environment rejection threshold, and $\lambda_2(\cdot)$ corresponds to the updated version for aggregated environments. By assumption $E_\infty(\nu, \kappa) > E_t(\nu, \kappa)$, so at the end of the exploration phase beginning at time step $t$, $\nu$ must remain in $\mathcal{M}$. Thus

$$\lambda_2(E) \geq \sum_{i=1}^{\alpha_j} X_i$$

(27)

\[
\geq \sum_{i=1}^{\alpha_j} Y_i + \frac{\epsilon_\kappa(1-\gamma)F_t(\nu, \kappa)}{2} \\
\geq \sum_{i=1}^{\alpha_j} Y_i + \frac{\epsilon_\kappa(1-\gamma)}{8} \sqrt{\frac{\alpha_j E_{\text{max},\kappa}}{N}}
\]

Where $(a)$ follows from the fact that $E[X_i] \geq 0$ for all $i$ and $E[X_i] \geq \epsilon_\kappa(1-\gamma)/2$ if $X_i$ corresponds to a $\nu$-effective exploration phase. Item (3) above gives us $(b)$.

Define $E_{\text{raw},\kappa} := \int_1^{21N} \frac{2^{9N}}{\epsilon_\kappa(1-\gamma)^2 \delta_1}$. Following [4], we have the following inequality for $E_{\text{raw},\kappa}$:

$$\int_1^{21N} \frac{2^{9N}}{\epsilon_\kappa(1-\gamma)^2 \delta_1} \geq 2\lambda_1(\alpha_j)$$

The main addition the argument of Lattimore et al. follows. Define $E_{\text{max},\kappa} := 4E_{\text{raw},\kappa}$, and we have

$$\frac{\epsilon_\kappa(1-\gamma)}{8} \sqrt{\frac{\alpha_j E_{\text{max},\kappa}}{N}} \geq 2\lambda_1(\alpha_j) + \frac{\epsilon_\kappa(1-\gamma)}{16} \sqrt{\frac{\alpha_j E_{\text{max},\kappa}}{N}}$$

(27a)

\[
\geq 2\lambda_1(\alpha_j) + \frac{\epsilon_\kappa(1-\gamma)}{16\sqrt{N}} \alpha_j \\
\geq 2\lambda_1(\alpha_j) + \frac{\epsilon(1-\gamma)}{2\sqrt{N}} \alpha_j \\
= 2\lambda_2(\alpha_j)
\]

Inequality $(a)$ follows because $E_{\text{max},\kappa} \geq \alpha_j$ and $b$ follows from the fact that $\epsilon_\kappa \geq \frac{\epsilon_\kappa}{2}$. Combining this with (27), if $E_{\infty,\kappa} > E_{\text{max},\kappa}$ then there exists an $\alpha_j$ such that $\sum_{i=1}^{\alpha_j} Y_i \leq -\lambda_2(\alpha_j)$. As with the proof of lemma 2, $B_n = \sum_{i=1}^{n} Y_i$ is a martingale with expectation 0, and so Azuma’s inequality implies

$$P(\sum_{i=1}^{\alpha_j} Y_i \leq -\lambda(\alpha_j)) \leq \frac{\delta_1}{\alpha_j}$$

45
Following [4], this is sufficient to show that the number of proper exploration phases is less than or equal to $E_{\text{max}} \equiv E_{\text{max,0}}$ with probability $1 - \frac{\delta}{4}$.

**Lemma 3**

To prove lemma 3, we will begin with a sub-lemma:

**Lemma 5:** Let $t$ be a time step and $h_t$ the corresponding history. If $\mu \in M_t$ and MERL is exploiting, then $V^*_{\mu}(h_t) - V^*_{\pi_t}(h_t) \leq \frac{5\epsilon}{8} + \frac{\epsilon}{2^{6}\sqrt{N}}$.

**Proof:** MERL is not exploring, so $V^\pi_\nu(h_t; d) < \frac{\epsilon}{8}$ for any $\pi$ and $\nu_1, \nu_2 \in M$.

\[
V^*_{\mu}(h_t) - V^*_{\pi_t}(h_t) \leq V^*_{\mu}(h_t; d) - V^*_{\mu}(h_t; d) + \frac{\epsilon}{8} \\
\leq V^*_{\phi_1(\mu)}(h_t; d) - V^*_{\phi_1(\mu)}(h_t; d) + \frac{\epsilon}{8} + \frac{\epsilon}{2^{6}\sqrt{N}} \\
\leq V^*_{\phi_1(\mu)}(h_t; d) - V^*_{\phi_1(\mu)}(h_t; d) + \frac{5\epsilon}{8} + \frac{\epsilon}{2^{6}\sqrt{N}} \\
\leq \frac{5\epsilon}{8} + \frac{\epsilon}{2^{6}\sqrt{N}} \\
\geq \epsilon(5 \frac{1}{64})
\]

Truncation of the value function gives us (a), (b) is from the d-step return discrepancy. For (c), we use the fact that MERL is not exploring, and (d) follows from the fact that $V^*_{\nu_1}(h_t; d) \leq V^*_{\nu_1}(h_t; d)$. Finally, $N \geq 1$ implies (e).

**Proof of lemma 3:** Let $t$ be the start of a proper failure phase with corresponding history, $h$. Thus $V^*_{\mu}(h) - V^*_{\pi}(h) > \epsilon$. From lemma 5:

\[
V^*_{\mu}(h) - V^*_{\pi}(h) \geq \epsilon(5 \frac{1}{64})
\]

Define $H_{\kappa} \subset H$ as the set of all extensions of $h$ that trigger $\kappa$-exploration phases, and define $H_{\kappa,d} := \{h' : h' \in H_{\kappa} \land \ell(h') \leq t + d\}$, the set of extensions of $h$ that are at most $d$ long and trigger $\kappa$-exploration phases. With these definitions
\[\epsilon \left(3 \cdot \frac{1}{8} \right) \leq V_\mu^{\pi_t}(h) - V_\mu^{\pi}(h)\]

\[
\leq \sum_{\kappa \in K} \sum_{h' \in H_\kappa} P(h'|h) \gamma^{\ell(h') - t}(V_\mu^{\pi_t}(h') - V_\mu^{\pi}(h')) \\
\leq \sum_{\kappa \in K} \sum_{h' \in H_\kappa} P(h'|h) \gamma^{\ell(h') - t}(V_\mu^{\pi_t}(h') - V_\mu^{\pi}(h')) + \frac{\epsilon}{64} \\
\leq \sum_{\kappa \in K} \sum_{h' \in H_\kappa} P(h'|h) \gamma^{\ell(h') - t}(V_\mu^{\pi_t}(h') - V_{\phi(\mu)}^{\pi}(h'; d) - V_\mu^{\pi}(h'; d)) + \frac{\epsilon}{8} \\
\leq \sum_{\kappa \in K} \sum_{h' \in H_\kappa} P(h'|h)4\epsilon_\kappa + \left(\epsilon \cdot \frac{7}{8}\right)
\]

(a) follows from noting that \(\pi = \pi_t\) unless an exploration phase is triggered and (b) from noting that if \(h' \in H_\kappa \setminus H_{\kappa,d}\) then \(\gamma^{\ell(h') - t} \leq (1 - \gamma)^{\ell(h')}\). (c) substitutes \(V_\mu^{\pi_t}(h') \geq V_\mu^{\pi}(h')\) and truncates the value function, while (d) substitutes for \(V_{\phi(\mu)}^{\pi}\) and applies the \(d\)-step mismatch, which allows us to the definition of \(\epsilon_\kappa\) to get (e).

Together, this gives us

\[
\sum_{\kappa \in K} \sum_{h' \in H_\kappa} P(h'|h)2^{\kappa + 3} \geq \frac{15}{8}
\]

Since the maximum of a set is greater than the average, there exists a \(\kappa \in K\) such that \(\sum_{h' \in H_\kappa,d} P(h'|h) \geq \frac{15 - 2^{\kappa - 6}}{|K|}\), which is the probability MERL will trigger a \(\kappa\)-exploration phase in the next \(d\) steps.

Fix a \(\kappa\), and let \(t_1, t_2, \ldots, t_{G_\kappa}\) be the sequence of all timesteps such that \(t_i\) is the start of a failure phase and the probability of triggering a \(\kappa\)-exploration phase is at least \(\frac{15 - 2^{\kappa - 6}}{|K|}\). \(G_\kappa\), then, is the number of failure phases with at least the aforementioned chance of triggering a \(\kappa\)-exploration phase. Let \(Y_i \in \{0, 1\}\) be the event that a \(\kappa\)-exploration phase does occur within \(d\) timesteps of \(t_i\), and let \(\tilde{Y}_i = Y_i\) if \(i \leq G_\kappa\) and 1 otherwise. Let \(E_\kappa\) be the total number of \(\kappa\)-exploration phases and \(G_{\text{max},\kappa}\) a constant to be chosen later.

Suppose \(G_k > G_{\text{max},\kappa}\). Then \(\sum_{i=1}^{G_{\text{max},\kappa}} \tilde{Y}_i = \sum_{i=1}^{G_{\text{max},\kappa}} \tilde{Y}_i\). Since \(Y_i\) counts the number of \(\kappa\)-exploration phases that have occurred, in this case either \(\sum_{i=1}^{G_{\text{max},\kappa}} \tilde{Y}_i \leq E_{\text{max},\kappa}\) or
\[ E_\kappa > E_{\text{max},\kappa} \text{ or both. Thus} \]

\[
P\{G_\kappa > G_{\text{max},\kappa}\} \\
\leq P \left\{ \sum_{i=1}^{G_{\text{max},\kappa}} \bar{Y}_i < E_{\text{max},\kappa} \right\} + P\{E_\kappa > E_{\text{max},\kappa}\} \\
\leq P \left\{ \sum_{i=1}^{G_{\text{max},\kappa}} \bar{Y}_i < E_{\text{max},\kappa} \right\} + \frac{\delta}{4|\mathcal{K}|}
\]

\(G_{\text{max},\kappa}\) must be sufficiently large to bound the first term above by \(\delta/4|\mathcal{K}|\). Recall that we chose \(Y_i\) such that if \(i \leq G_\kappa\), \(E[Y_i | \mathcal{F}_t] = E[\bar{Y}_i | \mathcal{F}_t] = \frac{15 \cdot 2^{\kappa^2-6}}{|\mathcal{K}|}\), and if \(i > G_\kappa\), \(\bar{Y}_i = 1\). Setting

\[
G_{\text{max},\kappa} = \frac{1}{15} \cdot 2^{\kappa^2+7}|\mathcal{K}|E_{\text{max},\kappa}
\]

Guarantees \(E[\sum_{i=1}^{G_{\text{max},\kappa}} \bar{Y}_i] > 2E_{\text{max},\kappa}\). At this point we are again in agreement with [4], which gives an argument by way of Azuma’s inequality and the union bound showing that \(P(G_\kappa > G_{\text{max},\kappa}) < \frac{\delta}{2|\mathcal{K}|}\) and \(P(G > G_{\text{max}}) < \frac{\delta}{2}\). 

48
6 Conclusion

$Q^*$ aggregation raises the possibility that many familiar reinforcement learning algorithms that are known to work on Markov problems may be appropriate under significantly weaker conditions. The weaker conditions may allow coarser state representations for some reinforcement learning problems, which will allow agents to learn values and policies faster. In situations that call for a class of models that contain the true environment, it may also be possible to find smaller classes of models that are known to contain an aggregation of the true environment which will serve equally well.

This work experimentally demonstrates Q-learning converging to the true value function for non-Markovian $Q^*$ aggregated processes. A proof that Q-learning converges in such conditions is still forthcoming, but these results suggest that $Q^*$ aggregation works for at least some classes of process.

Finding an appropriate aggregated state representation remains a significant problem. The U-Tree algorithm for learning the state representation was investigated, but the results are somewhat inconclusive pending a more thorough investigation of state splitting criteria.

The sample complexity bound of MERL-A suggests that some exploration algorithms may be applicable to the case of aggregated state representations with minimal modification, and it would be interesting if UCRL($\gamma$) could be extended in the same manner.

References


INDEPENDENT STUDY CONTRACT

Note: Enrolment is subject to approval by the projects co-ordinator

SECTION A (Students and Supervisors)

UniID: _u4533535________
SURNAME: _Johnston________ FIRST NAMES: _David Orion__________________
PROJECT SUPERVISOR (may be external): __Prof. Marcus Hutter_________________________
COURSE SUPERVISOR (a RSCS academic):         _Prof. Marcus Hutter____________________________
COURSE CODE, TITLE AND UNIT: __COMP6470 Special Topics in Computing________________

SEMESTER ☑ S1 ☑ S2 YEAR: ___2015___________

PROJECT TITLE:
Developing a Feature Reinforcement Learning Agent

LEARNING OBJECTIVES:
• Reasoning mathematically within the framework of reinforcement learning
• Experience implementing and adapting state-of-the-art reinforcement learning algorithms
• Clearly communicating research aims, processes and results

PROJECT DESCRIPTION:
Feature reinforcement learning aims to extend reinforcement learning beyond Markov Decision Processes (MDPs) by learning maps from agent histories to a set of states which can be modelled as an MDP. If a problem can be modelled as an MDP, a number of existing algorithms are known to converge to the correct value function. Recent work [1] has shown that if a map $\phi$ is able to approximately represent the value function of a reinforcement learning problem, then the reduced problem can be modelled as an MDP. This is significantly weaker than the Markov condition.

This project will look at building and analysing the performance of a reinforcement learning agent that learns a map $\phi$ respecting the conditions outlined in [1]. It will investigate adapting existing algorithms such as UCRL($\lambda$) and MERL [2,3] for this purpose, and applying this agent to reinforcement learning problems.


ASSESSMENT (as per course’s project rules web page, with the differences noted below):

Research School of Computer Science
Form updated Jun-
### Assessed project components:

<table>
<thead>
<tr>
<th>Report: name style: <strong>research report</strong>_________ (e.g. research report, software description...)</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>60</td>
<td>23 Oct</td>
<td>Marcus Hutter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Artefact: name kind: <em>software</em>__________________ (e.g. software, user interface, robot...)</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>30</td>
<td>26 Oct</td>
<td>Mayank Dasawani</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Presentation: talk</th>
<th>% of mark</th>
<th>Due date</th>
<th>Evaluated by:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>10</td>
<td>30 Oct</td>
<td>Marcus Hutter</td>
</tr>
</tbody>
</table>

**MEETING DATES (IF KNOWN):**

**STUDENT DECLARATION:** I agree to fulfil the above defined contract:

```
…………………………………………………..  ………………………..
Signature       Date
```

**SECTION B (Supervisor):**

I am willing to supervise and support this project. I have checked the student's academic record and believe this student can complete the project.

```
…………………………………………………..  ………………………..
Signature       Date
```

**REQUIRED DEPARTMENT RESOURCES:**

**SECTION C (Course coordinator approval)**

```
…………………………………………………..  ………………………..
Signature       Date
```

**SECTION D (Projects coordinator approval)**

```
…………………………………………………..  ………………………..
Signature       Date
```