Action Priors for Learning Domain Invariances

Benjamin Rosman, *Member, IEEE,* and Subramanian Ramamoorthy *Member, IEEE*

***Abstract*—An agent tasked with solving a number of different decision making problems in similar environments has an opportunity to learn over a longer timescale than each individual task. Through examining solutions to different tasks, it can uncover behavioural invariances in the domain, by identifying actions to be prioritised in local contexts, invariant to task details. This information has the effect of greatly increasing the speed of solving new problems. We formalise this notion as action priors, defined as distributions over the action space, conditioned on environment state, and show how these can be learnt from a set of value functions. We apply action priors in the setting of reinforcement learning, to bias action selection during exploration. Aggressive use of action priors performs context based pruning of the available actions, thus reducing the complexity of lookahead during search. We additionally define action priors over observation features, rather than states, which provides further flexibility and generalisability, with the additional benefit of enabling feature selection. Action priors are demonstrated in experiments in a simulated factory environment and a large random graph domain, and show significant speed ups in learning new tasks. Furthermore, we argue that this mechanism is cognitively plausible, and is compatible with findings from cognitive psychology.**

***Index Terms*—Search pruning, action selection, action ordering, transfer learning, reinforcement learning**

I. Introduction

Consider some learning agent, such as a robot, operating in a building for a prolonged period of time. This robot is required to perform multiple tasks in this domain, for example couriering different items to different people around the building, monitoring access to certain areas, etc. The difficulty with this setting is that not only are these tasks varied, but their specifications are also not completely known *a priori*, e.g. goal states may differ between tasks.

It is clear that this robot would need to learn to perform a number of different tasks. However, we wish to accelerate this learning via transferring knowledge. When approaching a new task, the robot should be able to reuse knowledge from having experienced other tasks. The challenge here is how to manage this experience, and how to organise it so as not just to store vast numbers of behaviours which are tailor-learnt to specific tasks. Instead, we desire some mechanism for generalisation.

Although the agent may be learning different tasks, the fact that they exist within the same environment means there is some underlying structure common to them all. It is this structure we wish to exploit to facilitate faster learning through forming better abstractions of the domain. Hence, we are interested in building agents capable of learning domain invariances. Invariances in this sense act as a form of common sense knowledge of the domain: these are constraints of preferences which are common to a large class of behaviours. This form of knowledge can provide insights into learning what *not* to do in particular situations, e.g. learning to avoid behaviours in a simulator which may not be realisable in reality [1]. In this way, we are interested in a form of model learning, where the model consists of the commonalities between a set of tasks, rather than the reward structure for any individual task.

Learning new behaviours, e.g. through reinforcement learning, is typically slow in that it necessarily requires extensive exploration of the space of possibilities. This is particularly the case when the specification of the task (given by the exact reward structure) is difficult to obtain, such as in the case of delayed reward reinforcement learning. In addition, learning *safely* is an important consideration for an agent which has to deal with the typical exploration-exploitation trade-off: while exploration must occur to discover optimal behaviour, this cannot be done at the cost of damage to the agent (particularly if it is embodied, such a robot). Our goal is thus to be able to inject weak knowledge into the problem, which is a prior of sensible behaviours in the domain, and can be learnt autonomously from previous tasks. To this end we introduce the notion of *action priors*, being localised distributions over the action sets of a learning agent, which are used to bias learning of new behaviours, and ensure safe exploration.

Learning action priors equates to finding invariances in the domain, across policies. These invariances are the aspects of the domain with which interaction of the agent remains the same, independent of the task. For instance, when one is driving a car the “rules of the road”, techniques for driving the car, and interaction protocols with other vehicles remain unchanged regardless of the destination. Learning these domain invariances is useful in a lifelong sense, as it simplifies the problem of learning to perform a new task, by factoring out those elements which remain unchanged across task specifications. We thus regard this as a form of transfer learning [2], [3] where the agent is required to learn to generalise knowledge gained from solving some tasks, to apply to others.

The key assumption leveraged in this work is that there is certain structure in a domain in which an agent is required to perform multiple tasks over a long period of time. This is in the form of local contexts in which, regardless of the task, certain actions may be commonly selected, while others should always be avoided as they are either detrimental, or at best do not contribute towards completing any task. This induces local sparsity in the action selection process. By learning this structure, and posed with a new task, an agent can focus exploratory behaviour away from actions which are seldom useful in the current situation, and so boost performance in expectation. Extracting this structure thus provides weak constraints similar to manifold learning [4].

B. Rosman is with Mobile Intelligent Autonomous Systems, Council for Scientific and Industrial Research (CSIR), South Africa.

S. Ramamoorthy is in the School of Informatics, University of Edinburgh, UK.

Action priors thus allow a decision making agent to bias exploratory behaviour towards actions that have been useful in the past in similar situations, but different tasks. This formalism also provides a platform which can be used to inject external information into the agent, in the form of teaching.

We pose this framework in a reinforcement learning context, but note it is applicable to other decision making paradigms. Indeed, we argue in Section V that this resembles techniques which humans are believed to invoke in order to facilitate decision making under large sets of options [5]. Our approach is generally applicable to domains with two properties:

- The domain must support multiple tasks, with related structure (described further in Section II).
- The action set of the agent must be large, for prioritising actions to provide benefit (see experiments in Section IV).

A. *Contributions*

The problem addressed in this paper is that an agent learning to perform a range of tasks in the same domain is essentially relearning everything about the domain for every new task, and thus learning is slow. We seek a middle ground between model-based and model-free learning, where a model of the regularities in behaviours within a domain can be acquired.

Our approach to tackling this problem is to extract the invariances in the domain from the set of tasks that the agent has already solved. This is related to the notion of extracting *affordances* from the domain, discussed further in Section VI. These invariances are termed action priors, and provide the agent with a form of prior knowledge which can be injected into the learning process for new tasks.

Additionally, we show

- an alternative version of the action priors which is suitable for changing domains and partial observability,
- a method for selecting domain features so as to maximise the effect of these priors,
- that action priors are based on mechanisms which are cognitively plausible in humans.

B. *Paper Structure*

This paper is structured as follows. We introduce our core innovation, action priors, in Section II. We then discuss how action priors can be used in scenarios where the structure of the domain changes, and use this reformulation to perform feature selection in Section III. We demonstrate our methods in experiments in Section IV, and discuss the relation of our approach to psychological findings in human behaviour in Section V. Finally, we present related work in Section VI.

II. Action Priors

A. *Preliminaries*

In keeping with the standard formalism of reinforcement learning, let an environment be specified by a Markov Decision Process (MDP). An MDP is defined as a tuple (S, A, T, R, γ), where S is a finite set of states, A is a finite set of actions which can be taken by the agent, $T : S \times A \times S \to [0, 1]$ is the state transition function where $T(s, a, s')$ gives the probability of transitioning from state s to state s' after taking action a, $R : S \times A \to \mathbb{R}$ is the reward function, where $R(s, a)$ is the reward received by the agent when transitioning from state s with action a, and $\gamma \in [0, 1]$ is a discount factor.

A Markovian policy $\pi : S \times A \to [0, 1]$ for an MDP is a distribution over state-action space. The return, generated from an episode of running the policy π is the accumulated discounted reward $\bar{R}^\pi = \sum_k \gamma^k r_k$, for r_k being the reward received at step k. The goal of a reinforcement learning agent is to learn an optimal policy $\pi^* = \arg\max_\pi \bar{R}^\pi$ which maximises the total expected return of the MDP, where typically T and R are unknown.

Many approaches to learning an optimal policy involve learning the value function $Q^\pi(s, a)$, giving the expected return from selecting action a in state s and thereafter following the policy π [6]. Q is typically learnt by iteratively updating values using the rewards obtained by simulating trajectories through state space, e.g. Q-learning [7]. A greedy policy can be obtained from a value function defined over $S \times A$, by selecting the action with the highest value for any given state.

B. *State Based Action Priors*

We define a domain by the tuple $D = (S, A, T, \gamma)$, and a task as the MDP $\tau = (D, R)$. In this way we factorise the environment such that the state set, action set and transition functions are fixed for the whole domain, and each task varies only in the reward function. Given an arbitrary set of tasks $\mathcal{T} = \{\tau\}$ and their corresponding optimal policies $\Pi = \{\pi^*_\tau\}$, we wish to learn for each state $s \in S$ its action priors $\theta_s(A)$: a distribution over the action set, representing the probability of each action in A being used in an optimal policy in the state s, aggregated over tasks. This is then, in subsequent tasks, used to prioritise the actions in each state.

The idea is that by considering a set of policies, each selecting actions in a state s according to different distributions, one can gain insights into properties of s. These insights are in terms of the "usefulness" of different actions in that state[1]. For example, if one action is favoured by all policies when in s, then that action could be considered as very useful in s, and should be favoured during exploration in subsequent tasks. Conversely, if an action is not selected by any policy in s, then that action is likely to have negative consequences, and should be avoided. This notion thus informs weak constraints on policies at s, in that an action not selected by previous policies should not be prioritised in solving future tasks.

To provide further intuition, we say that an action a_1 is preferable to an action a_2 in a state s, if a_1 is used in s by a larger number of optimal policies than a_2. The setting in which we study the phenomenon of accelerated learning in a lifelong sense is that the tasks seen so far are sampled from a distribution of all possible tasks, and are representative of that task space. By examining the optimal policies that arise from multiple tasks in the same domain, we aim to learn about the structure of the underlying domain, in terms of identifying local behaviours which are invariant across tasks.

[1] This "usefulness" is defined in terms of the utility of that action over a bank of different tasks. This is formalised in Equation (3).

1) Combining Policies to Learn Priors: Consider the setting in which the agent has prolonged experience in the domain D. This means the agent has solved a set of tasks in D, and we use the resulting set of optimal policies to extract the action priors as a form of structural information about the domain.

For each state $s \in S$, we model the action priors $\theta_s(A)$ as a distribution over the action set A, describing the usefulness of each action in optimally solving the tasks in $\mathcal{T}$ using the policies Π. To do so, we first define the utility of an action a in a state s under a policy π as a boolean variable describing whether or not a was optimal in s using π. Formally,

$$U_s^{\pi}(a) = \delta(\pi(s,a), \max_{a' \in A} \pi(s,a')), \tag{1}$$

where $\delta(\cdot,\cdot)$ is the Kronecker delta function: $\delta(a,b) = 1$ if $a = b$, and $\delta(a,b) = 0$ otherwise. As a result $U_s^{\pi}(a) = 1$ if and only if a is the best (or tied best) action in s under π. This formulation of a utility function is used rather than $Q(s,a)$, as the values stored in $Q(s,a)$ relate to the rewards allocated for the task that π solves. These values are thus relative, unlike those of U_s^{π} which provide a point-estimate of the value of an action in a state, and as such are comparable across policies for very different tasks a reward scales.

Now consider the utility $U_s^{\Pi}(a)$ of an action a in a state s under a *policy library* Π. This value is the weighted sum,

$$U_s^{\Pi}(a) = \sum_{\pi \in \Pi} w(\pi) U_s^{\pi}(a), \tag{2}$$

where $w(\pi) \in \mathbb{R}$ is a weight for the policy $\pi \in \Pi$. This weight factor allows us to include suboptimal policies in the formulation, by giving them lower weights.

The fact that $\theta_s(A)$ is constructed from the policies solving a set of tasks $\mathcal{T}$ raises the possibility that $\mathcal{T}$ is not actually representative of the complete set of possible tasks in D. We counteract this by forming an augmented policy set Π^+, defined as $\Pi^+ = \Pi \cup \pi_0$, where π_0 is the uniform policy: $\pi_0(s,a) = \frac{1}{\|A\|}$, $\forall s \in S, a \in A$. The utility of this policy is then $U_s^{\pi_0}(a) = 1$, $\forall s \in S, a \in A$. In this case, the weight $w(\pi_0)$ is representative of the likelihood of encountering a new task which has not been previously solved.

In the limit of increasing task variance, the probability of an action under the action priors tends to the true probability of being optimal in *any* task. This encapsulates the structure of the domain in terms of the probability of reaching an arbitrary goal from the current state. Then, if the agent is acting with no extra knowledge of the task, this distribution represents the best course of action which should be taken in any state.

Given a state s, for each action a the utility of that action $U_s^{\Pi^+}(a)$ provides an estimate of the value of the state-action pair (s,a) in D under the augmented policy set. We thus choose actions according to these values. To select an action, we sample from a probability distribution $\theta_s(A) = f(U_s^{\Pi^+}(A))$, such that $a \sim \theta_s(A)$. There are many ways in which f may be defined, e.g.,

1) as a proportion:
$f(U_s^{\Pi^+}(A)) = f_p = \frac{U_s^{\Pi^+}(a)}{\sum_{a' \in A} U_s^{\Pi^+}(a')}$,
2) as a Boltzmann softmax distribution:
$f(U_s^{\Pi^+}(A)) = f_s = \frac{\exp\{U_s^{\Pi^+}(a)\}}{\sum_{a' \in A} \exp\{U_s^{\Pi^+}(a')\}}$,
3) as a draw from a Dirichlet distribution:
$f(U_s^{\Pi^+}(A)) = f_d \sim Dir(U_s^{\Pi^+}(a))$.

Throughout the remainder of this paper, we choose to model f as a Dirichlet distribution, although these all remain valid choices. We base this choice on the fact that as the conjugate prior of the multinomial distribution over the action set, the Dirichlet distribution can be interpreted as the belief in the probabilities of rival actions, given some prior observations of their occurrences. Furthermore, the mean of the Dirichlet distribution is its normalised parameters, which is exactly the proportion f_p, although the use of hyperpriors controls against overfitting. We also prefer both f_p and f_d to the softmax f_s as we hypothesise that in situations where a large number of prior policies have been encountered, small discrepancies in action counts could explode under the exponential term, and so some viable options would be explored considerably less than the prior advocates. Additionally, any 'max' function is unlikely to provide the same exploration convergence guarantees[2].

We note that these particular choices of distributions are intended for discrete action sets. If the action space is continuous, for instance if an action is parametrised by a continuous parameter, then other distributions should be used[3]. Alternatively, the continuous space could be discretised into intervals of a desired granularity, and the method described in the rest of the paper then applies.

2) Action Priors as a Dirichlet Distribution: For each state s, draw the action priors $\theta_s(A)$ from a Dirichlet distribution conditioned on s. The Dirichlet distribution is parametrised by concentration parameters $(\alpha(a_1), \alpha(a_2), \ldots, \alpha(a_{\|A\|}))^T$ and so for each state s, we maintain a count $\alpha_s(a)$ for each action $a \in A$. The initial values of $\alpha_s(a) = \alpha_s^0(a)$ are known as the pseudocounts, and can be initialised to any value by the system designer to reflect prior knowledge. If these counts are the same for each action in a state, i.e. $\alpha_s(a) = k$, $\forall a \in A$ this returns a uniform prior, which results in each action being equally favourable in the absence of further information.

The pseudocounts $\alpha_s^0(a)$ are a hyperprior which model prior knowledge of the tasks being performed by the agent. If the variance in the tasks is expected to be small, or alternatively a large number of training tasks are provided, then this hyperprior is set to a smaller value. However, if there is great diversity in the tasks, and the agent will not be expected to sample them thoroughly, then a larger hyperprior will prevent the action priors from over-generalising from too little data[4].

We wish these counts to describe the number of times an action a was considered optimal in a state s, across a set of

[2] Empirical validation of this choice has been omitted for space reasons [8].

[3] As an example, one could use a Normal-gamma distribution (as a conjugate prior to a Gaussian distribution) in the same way we use a Dirichlet distribution: by specifying a hyperprior, and then using the optimal parameter values from the training tasks to update the model parameters. Online, draw a (μ, σ) from this distribution, and use that Gaussian to draw an action parameter value. Depending on the range of these parameter values, they may need to be transformed with a logit function.

[4] It can be shown that the convergence guarantees of Q-learning can be maintained by using the action priors with a non-zero hyperprior [8].

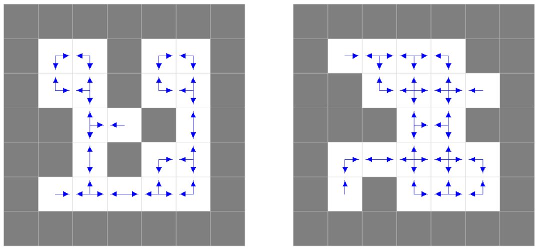


Fig. 1: Example action priors learned for 5×5 maze worlds, from 50 random optimal Q-functions. Indicated directions in each cell have a non-zero probability mass, but in every cell the agent can execute any of four directional movements. Grey cells are obstacles, and white cells are free space.

policies Π. We thus set

$$\begin{aligned} \alpha_s(a) &= U_s^{\Pi^+}(a) = \sum_{\pi \in \Pi^+} w(\pi) U_s^{\pi}(a) && (3) \\ &= \sum_{\pi \in \Pi} w(\pi) U_s^{\pi}(a) + \alpha_s^0(a). && (4) \end{aligned}$$

This provides a natural intuition for the counts as the weighted utility of Π, and the hyperprior is then $\alpha_s^0(a) = w(\pi_0)$.

Typically, one would not want to maintain a full library of policies. The α counts can alternatively be learnt by the agent in an online manner as it learns the solutions to new tasks. In this way, when the agent solves some task τ^{t+1}, the counts for each state-action pair can be updated by the values in π^{t+1}. This provides an online update rule for Equation (3) as

$$\alpha_s^{t+1}(a) \longleftarrow \begin{cases} \alpha_s^t(a) + w(\pi^{t+1}) & \text{if } \pi^{t+1}(s,a) = \\ & \max_{a' \in A} \pi^{t+1}(s,a') \\ \alpha_s^t(a) & \text{otherwise.} \end{cases} \quad (5)$$

To obtain the action priors $\theta_s(A)$, sample from the Dirichlet distribution: $\theta_s(A) \sim Dir(\alpha_s)$. Note that $\theta_s(A)$ is a probability distribution over A, and so $\sum_a \theta_s(a) = 1$, $\forall s \in S$.

Note that in Equation (5) we increment the α counts by $w(\pi^{t+1})$, rather than the probability $\pi^{t+1}(s,a)$. This enumerates the actions used by the different policies, rather than simply averaging the effects of the actions taken by the individual policies, which could result in the agent being drawn towards a local optimum in state space, by one policy dominating others. Instead we weight each action by the number of independent tasks which require the selection of that particular action, which is then used as a prior probability of that action choice in that state over all tasks in the domain.

Figure 1 demonstrates the result of using our method on 5×5 maze worlds, extracted from policies which were the optimal solutions to 50 random navigation tasks. An arrow in a cell is drawn in a direction only if any mass was allocated to that direction by any policy. Note that this results in the "useful" actions of the domain, being the actions that do not cause collisions with obstacles. Action priors effectively reduce the set of actions from four in each cell to the subset which were used in the training tasks (55.26% and 63.16% of the full action sets respectively in the examples shown in Figure 1).

3) Using the Action Priors: An action prior provides the agent with knowledge about which actions are sensible in situations in which the agent has several choices to explore. As such, they are useful for seeding search in a policy learning process. Although this applies to any algorithm which takes exploration steps as is typical in reinforcement learning, we demonstrate this modified exploration process with an adaptation of traditional Q-learning [6], called ϵ-greedy Q-learning with State-based Action Priors (ϵ-QSAP) [9], shown in Algorithm 1. Note, in this algorithm, $\alpha^Q \in [0,1]$ denotes the learning rate, and should not be confused with the Dirichlet distribution counts $\alpha_s(a)$. The parameter $\epsilon \in [0,1]$ controls the trade-off between exploration and exploitation. Both α^Q and ϵ are typically annealed after each episode.

Algorithm 1 ϵ-greedy Q-learning with State-based Action Priors (ϵ-QSAP)

```
Require: action prior $\theta_s(a)$
 1: Initialise $Q(s,a)$ arbitrarily
 2: for every episode $k = 1 \dots K$ do
 3:     Choose initial state $s$
 4:     repeat
 5:         $a \longleftarrow \begin{cases} \arg\max_a Q(s,a), & \text{w.p. } 1-\epsilon \\ a \in \mathcal{A}, & \text{w.p. } \epsilon\theta_s(a) \end{cases}$
 6:         Take action $a$, observe $r$, $s'$
 7:         $Q(s,a) \longleftarrow Q(s,a) + \alpha^Q[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
 8:         $s \longleftarrow s'$
 9:     until $s$ is terminal
10: end for
11: return $Q(s,a)$
```

The difference between this and standard ϵ-greedy Q-learning can be seen on line 5. This is the action selection step, consisting of two cases. The first case deals with exploiting the current policy stored in $Q(s,a)$ with probability $1-\epsilon$, and the second case with exploring other actions $a \in A$ with probability ϵ. The exploration case is typically handled by choosing the action uniformly from A, but instead we choose with probability based on the prior $\theta_s(A)$ to shape the action selection based on what were sensible choices in the past.

The effect is that the agent exploits the current estimate of the optimal policy with high probability, but also explores, and does so with each action proportional to the number of times that action was favoured in previous tasks. This highlights the assumption that there is inherent structure in the domain which can be identified across multiple tasks.

We emphasise here that although this paper utilises action priors as an exploration policy within the context of Q-learning, this idea applies to a broader range of algorithms. In the general case, action priors may instead be used to provide an initial seed policy for a learning algorithm, or instead as a filter for selecting or ranking actions during learning.

III. Priors from Observations

In Section II-B, action priors were defined as distributions over actions, conditioned on the current state. In this section we extend these definitions such that the action priors are instead conditioned on observations. There are several reasons for this representation change. The first is that the transition function T or even the state space S may not be task independent, and may instead differ between task instances. This may be the case, for example, when an agent is exploring buildings, and the layout of each differs. It is not sensible to condition action priors on states, if the connectivity of those states changes between task instances. Instead, the agent should condition action priors on observable features of the states – features which persist across tasks, even if state identities do not. This representation change therefore allows the action priors to generalise further, to tasks in related environments.

Alternatively, one may not always have full observability of the state, and so different states cannot be uniquely distinguished. This is the case in partially observable reinforcement learning problems [10] which typically require the solution of partially observable Markov decision processes (POMDPs). Full state information is not always accessible. Similarly, there may be states which have not been explored during training time, and so no action prior would be available for these states, which may be required at test time. In both these scenarios, it is again sensible to instead base the action priors on the observable features of the state. The observation based action priors thus transfer to unseen state and action combinations.

Basing action priors on observations rather than states changes the dependence of θ from $s \in S$ to $\phi : S \longrightarrow \mathcal{O}$, where ϕ is the mapping from state space S to the observation space $\mathcal{O}$. The observed features of s are thus described by $\phi(s)$. The state based priors can now be considered as a special case of observation based priors, with $\phi(s) = s$.

Note that we are not solving a partially observable problem, but are instead informing exploration based on some partial information signals. Using observations rather than exact state descriptions allows for more general priors, as the priors are applicable to different states with the same observations. This also enables pooling of the experience collected from different states with similar observations, to learn more accurate action priors. There is, however, a trade-off between this generality, and the usefulness of the priors. This trade-off depends on the observation features, and the amount of action information captured by these features. These, in turn, depend on properties of the task and domain. The more general the observation features, the less informative the action priors will be. On the other hand, the more specific these features are (up to exact state identification), the less portable they are to new states.

Both state and observation based action priors have their uses. For example, maze-like environments stand to benefit from a state based approach, where entire wings of the maze could be pruned as dead-ends, which is not possible based on observations alone. Alternatively, in a rich environment with repeated structure, training policies are less likely to have sufficiently explored the entire space, and so one may pool together priors from different states with the same observations.

A. *Using the Observation Based Priors*

Changing the variables on which the action priors are conditioned from states to observations replaces s with $\phi(s)$ in Algorithm 1. When learning the action priors, Equations (5) and (3) are also still valid, by again replacing s with $\phi(s)$.

Similarly, the α counts are learnt by the agent online from the previous optimal policies, and updated for each $(\phi(s), a)$ pair whenever a new policy π^{t+1} is available:

$$\alpha^{t+1}_{\phi(s)}(a) \longleftarrow \begin{cases} \alpha^{t}_{\phi(s)}(a) + w(\pi^{t+1}) & \text{if } \pi^{t+1}(s,a) = \\ & \max_{a'} \pi^{t+1}(s,a') \\ \alpha^{t}_{\phi(s)}(a) & \text{otherwise.} \end{cases} \tag{6}$$

Thus $\alpha_{\phi(s)}(a)$ is the number of times a was considered a good choice of action in any state s with observations $\phi(s)$ in *any* policy, added to the pseudocounts $\alpha^0_{\phi(s)}(a)$. The corresponding closed form of Equation (6) given a set of policies Π is then:

$$\alpha_{\phi(s)}(a) \quad = \quad \sum_{s \in [s]_\phi} \sum_{\pi \in \Pi} w(\pi) U^{\pi}_{\phi(s)}(a) + \alpha^0_{\phi(s)}(a), \tag{7}$$

where $U^{\pi}_{\phi(s)}(a) = \delta(\pi(\phi(s), a), \max_{a' \in A} \pi(\phi(s), a'))$, and $[s]_\phi = \{s' \in S | \phi(s) = \phi(s')\}$ represents the equivalence class of all states with the same observation features as state s. This additional summation occurs because in the general case, the priors from multiple states will map to the same observation based action priors.

To obtain the action priors $\theta_{\phi(s)}(A)$, again sample from the Dirichlet distribution: $\theta_{\phi(s)}(A) \sim Dir(\alpha_{\phi(s)})$.

B. *Feature Selection*

The choice of the set of observational features is an open question, depending on the capabilities of the agent. Indeed, feature learning in general is an open and difficult question, which has been considered in many contexts, e.g. [11], [12]. The possible features include the state label (as discussed previously), as well as any sensory information the agent may receive from the environment. Furthermore, these features can include aspects of the task description, or recent rewards received from the environment.

As a result, in many domains, there could be a large set of observational features. The size of Φ, the space of possible mappings, is exponential in the number of features. Given a feature space Φ, we are interested in identifying the optimal feature set $\phi^* \in \Phi$, which provides abstraction and dimensionality reduction, with a minimal loss of information in the action prior. Finding such a ϕ^* allows for the decomposition of the domain into a set of *capabilities* [13], being recognisable and repeated observational contexts, with minimal uncertainty in the optimal behavioural responses, over the full set of tasks.

Let a feature f_i be a mapping from a state to a set of values $f_i^1 \cdots f_i^{K_i}$. Let ϕ be a set of these features. We abuse notation slightly and for a particular feature set ϕ_i we enumerate the possible settings of all its constituent features, such that $\phi_i = \phi_i^j$ means that the features in ϕ_i are set to configuration j, where these configurations are uniquely ordered such that $j \in [1, K_{\phi_i}]$, where $K_{\phi_i} = \prod_q K_q$ is the total number of possible feature configurations, q runs over the features of ϕ_i, and K_q is the number of settings for feature q.

Our goal is to find the feature set which can prescribe actions with the most certainty. To this end, define the average entropy of an action a under a particular feature set ϕ as

$$\bar{H}_\phi(a) \quad = \quad \sum_{j=1}^{K_\phi} P(\phi^j) H[P(a|\phi^j)], \tag{8}$$

where $P(a|\phi^j) = \theta_{\phi^j}(a)$ is the value of the action prior for a particular set of feature values, $P(\phi^j)$ is the prior probability of that set of feature values, estimated empirically from data as $\frac{\sum_a \alpha_{\phi^j}(a)}{\sum_i \sum_a \alpha_{\phi^j}(a)}$, and $H[p] = -p \log_2 p$ is the standard entropy. The prior $P(\phi^j)$ weights each component distribution by the probability of that feature combination arising in the data.

By summing the average entropy for each action, we define the entropy of the action set A under a feature set ϕ as

$$H_\phi \quad = \quad \sum_{a \in A} \bar{H}_\phi(a) \tag{9}$$

The optimal feature set is that which minimises the action set entropy. This is thus analogous to the information invariance which is present in the observations of the agent [14]. There is however a caveat with this simple minimisation. The more features are included in the feature set, the sparser the number of examples for each configuration of feature values. We therefore regularise the minimisation, by optimising for smaller feature sets through the application of a penalty based on the number of included features. Finding the optimal feature set ϕ^* is thus posed as solving the optimisation problem

$$\phi^* = \arg\min_{\phi \in \Phi} [H_\phi + c\|\phi\|] \tag{10}$$

where $\|\phi\|$ is the number of features in ϕ, and c is a parameter which controls for the weight of the regularisation term.

The major problem is that the number of possible feature sets is exponential in the number of features, and so we instead focus on an approximation for selecting the best set of features. This is shown in Algorithm 2, which returns the approximate minimal feature mapping $\tilde{\phi}^* \simeq \phi^*$. The key assumption is that each feature f affects the entropy of $\theta_\phi(A)$ independently. For each feature f from the full feature set ϕ_{full}, we marginalise over that feature and compute the entropy of the remaining feature set. Each of these $\|\phi_{full}\|$ individual entropy values is compared to the entropy of the full set $H_{\phi_{full}}$. The greater the increase in entropy resulting from the removal of feature f, the more important f is as a distinguishing feature in the action prior, as the addition of that feature reduces the overall entropy of the action prior distribution. The feature set chosen is then the set of all features which result in an entropy increase greater than some threshold ω.

This independence assumption implies that it is not the case that the importance of some features are conditional on the values of others. This assumption is more likely to hold when the features are already the result of processing, rather than raw sensory input. This could be addressed by considering a feature to be a concatenation of several dependent variables. Alternatively, in the case of dependent features, the optimisation problem in Equation (10) could be solved directly.

In order to use this method for feature selection from a rich data source, such as a vision system, two primary modifications would be required. Firstly, image processing is required of the input data to form hypothesis features. This could be done by identifying common structure in the data (see, e.g. [13]). Secondly, this feature selection procedure could be used during online operation of the agent, which allows for continued re-evaluation of feature importance [15].

Algorithm 2 Independent Feature Selection

```
Require: feature set $\phi_{full}$, entropy increase threshold $\omega$
Compute $H_{\phi_{full}}$ by Equation (9)
for every feature $f$ in $\phi_{full}$ do
    $\phi_{-f} \longleftarrow \phi_{full} \setminus f$
    Compute $H_{\phi_{-f}}$ by Equation (9)
    $\Delta_f \longleftarrow H_{\phi_{-f}} - H_{\phi_{full}}$
end for
$\tilde{\phi}^* \longleftarrow \{f \in \phi_{full} | \Delta_f \geq \omega\}$
return $\tilde{\phi}^*$
```

The increased generality from using observations draws from the intuition that certain behaviours only make sense in the context of particular sensory features. If there is a high entropy in which actions should be taken in the context of a particular observation, then there are two possible reasons for this: 1) it may be the case that different tasks use this context differently, and so without conditioning action selection on the current task, there is no clear bias on which action to select, or 2) it may be that this observation is not informative enough to make the decision, providing scope for feature learning. This distinction can only be made in comparison to using the maximal feature set, as the most informative set of observation features. Without ground truth state information, this can only be ascertained through learning. If the observations are not informative enough, then this suggests that additional features would be useful, providing the agent with the opportunity for trying to acquire new features.

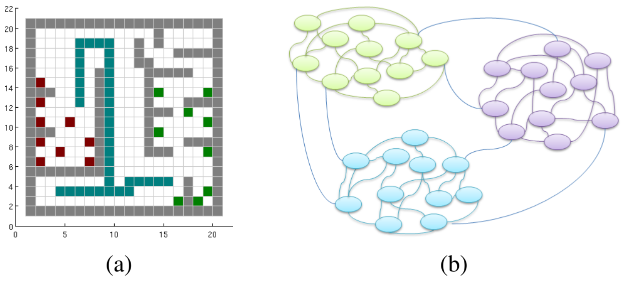
IV. Experiments

A. The Factory Domain

The first domain used in these experiments is the factory domain, which is an extended navigation domain involving a mobile manipulator robot placed on a factory floor. The factory layout consists of an arrangement of walls through which the robot cannot move, with some procurement and assembly points placed around the factory. Additionally there are express routes demarcated on the ground, which represent preferred paths of travel, corresponding to regions where collisions with other factory processes may be less likely. The domain used in these experiments is shown in Figure 2a.

The robot has an action set consisting of four movement actions (*North*, *South*, *East* and *West*), each of which move the robot in the desired direction, provided there is no wall in the destination position, a *Procure* action, and an *Assemble* action. *Procure* used at procurement point i, provides the robot with the materials required to build component i. *Assemble* used at assembly point i, constructs component i, provided the robot already possesses the materials required for component i.

A task is defined as a list of components which must be assembled by the robot. The domain has 9 components, and



(a)

(b)

Fig. 2: (a) Factory domain used in the first experiments. Grey cells are obstacles, white cells are free space, green cells are procurement points, red cells are assembly points, and cyan cells are express routes. (b) Caricature of the graph domain used in the final experiments, showing clusters of nodes with dense intra-cluster and sparse inter-cluster connections.

so this list can range in length from 1 to 9, giving a total of $2^9 - 1$ different tasks. Task rewards are defined as follows. All movement actions give a reward of -2, unless that movement results in the robot being on an express route, for a reward of -1. Collisions are damaging to the robot and so have a reward of -100. *Procure* at a procurement point for an item in the task definition which has not yet been procured gives a reward of 10. *Procure* anywhere else in the domain yields -10. *Assemble* at an assembly point for an item in the list which has already been procured but not assembled gives 10, and any other use of the *Assemble* action gives -10. Successful completion of the task gives 100 and the episode is terminated.

This domain provides a number of invariances which could be acquired by the robot. Locally, these include avoiding collisions with walls, preferring express routes to standard free cells, and not invoking a *Procure* or *Assemble* action unless at a corresponding location. As all tasks are defined as procuring and assembling a list of components, this additionally provides scope for learning that regardless of the components required, the robot should first move towards and within the region of procurement points until all components have been procured, after which it should proceed to the region of assembly points.

B. Results with State Action Priors

The results in Figure 3, which compares the performance per episode of a learning agent using a set of different priors, demonstrates that using action priors reduces the cost of the initial phase of learning, which is largely concerned with coarse scale exploration. In this case, the loss incurred in the early episodes of learning is dominated by the fact that the agent explores in the wrong directions, rather than performs invalid or ill-placed actions. This figure also shows comparative performance of Q-learning with uniform priors (i.e. "standard" Q-learning), as well as with two different hand specified priors; an "expert" prior and an "incorrect" prior.

The "expert" prior is defined over the state space, to guide the agent towards the procurement area of the factory if the agent has any unprocured items, and to the assembly area otherwise. This prior was constructed by a person, who was required to specify the best direction for each state in the domain. We note that this prior is tedious to specify by hand, as it involves an expert specifying preferred directions of motion for the entire state space of the agent (number of states in the factory × number of different item configurations). Note that, although the performance is very similar, this prior does not perform as well as the learnt prior, likely due to a perceptual bias on behalf of the expert's estimation of optimal routing. We also compare to an "incorrect" prior. This is the same as the expert prior, but we simulate a critical mistake in the understanding of the task: when the agent has unprocured items, it moves to the assembly area, and otherwise to the procurement area. This prior still provides the agent with an improvement in the initial episodes of uniform priors, as it contains some "common sense" knowledge including not moving into walls, moving away from the start location, etc. Q-learning is still able to recover from this error, and ultimately learn the correct solution.

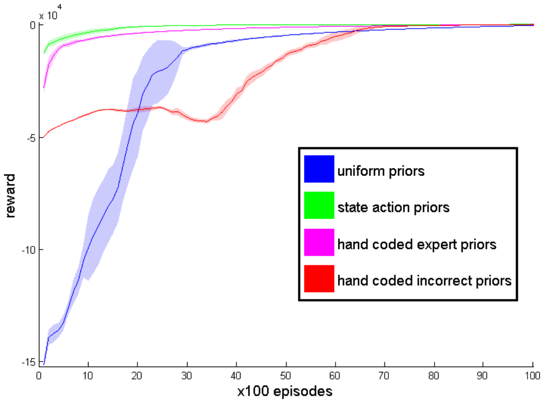


Fig. 3: Comparative performance between Q-learning with uniform priors, with state based action priors learned from 35 random tasks, and with two different pre-specified priors (see text for details), averaged over 15 runs. Each task was to assemble 4 components, selected uniformly at random. The shaded region represents one standard deviation.

Figure 4 shows the speed up advantage in learning a set of $N = 40$ tasks, starting from scratch and then slowly accumulating the prior from each task, against learning each task from scratch. This case is for a simple version of the task, which involved procuring and assembling a single item. As a result, all task variants are likely to have been encountered by the time the agent is solving the final tasks.

On the other hand, Figure 5 shows that a similar effect can be observed for the case of a more complicated task, requiring the assembly of 4 randomly selected items. In this case, even by the time the learning agent has accumulated a prior composed from 40 tasks, it has only experienced a small fraction of the possible tasks in this domain. Despite this, the agent experiences very similar benefits to the single item case.

C. Results with Observation Action Priors

In order to demonstrate the effect of using the observation action priors, we present a modification of the factory domain. Recall that as state action priors define distributions over each state of the domain, they cannot be robustly ported between

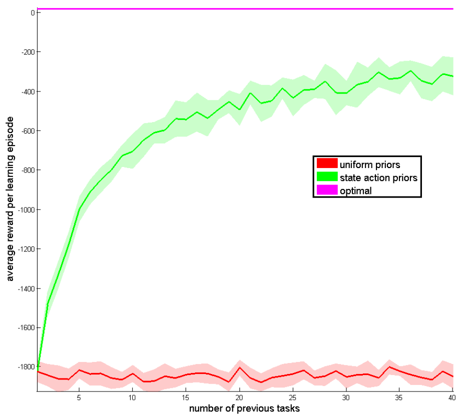


Fig. 4: Comparative performance between Q-learning with uniform priors, and with state based action priors accumulated from an increasing number of tasks (0 to 40). These curves show the average reward per episode averaged over 10 runs, where the task was to assemble 1 random component. The "optimal" line is average performance of an optimal policy. The shaded region represents one standard deviation.

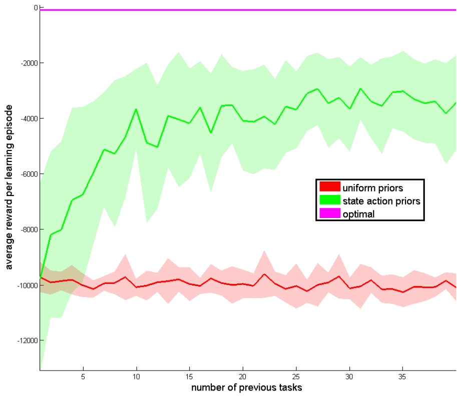


Fig. 5: Comparative performance between Q-learning with uniform priors, and with state based action priors accumulated from an increasing number of tasks (0 to 40). These curves show the average reward per episode averaged over 10 runs, where the task was to assemble 4 random components. The "optimal" line is average performance of an optimal policy. The shaded region represents one standard deviation.

similar domains. On the other hand, as observation action priors are based on local features rather than global state information, this information is more portable, albeit possibly less informative. The modified factory domain provides a test-bed to demonstrate this point.

The modified domain stipulates that the factory floor layout changes for each different task. This corresponds to either the learning agent moving between different factories, or the fact that the obstacles, procurement and assembly points may be mobile and change with some regularity (e.g. whenever a new delivery is made). Each factory floor consists of a 3×3 lattice of zones, each of which is 6×6 cells. There is an outer wall, and walls in between every two zones, with random gaps in some (but not all) of these walls, such that the entire space is connected. Additionally, each zone contains randomly placed internal walls, again with connectivity maintained. Two zones are randomly chosen as procurement zones, and two zones as assembly zones. Each of these chosen zones has either four or five of the appropriate work points placed at random. Examples of this modified domain are shown in Figure 6.

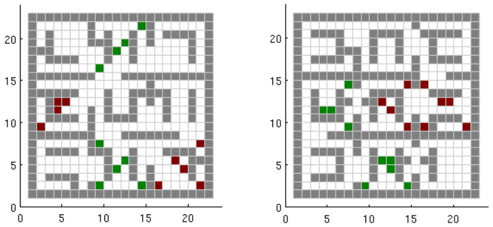


Fig. 6: Two instances of the modified factory domain. Grey cells are obstacles, white cells are free space, green cells are procurement points, and red cells are assembly points. The procurement and assembly points count as traversable terrain.

D. Feature Selection

This modified domain has a different layout for every task. As a result, every task instance has a different transition function T. This is in contrast to the original factory domain, where each task differed only in reward function R. State based action priors can therefore not be expected to be as useful as before. We thus use observation priors, and although the choice of observation features remains open, we discuss four particular feature sets.

Figure 7 demonstrates the improvement obtained by using observation priors over state priors in this modified domain. Note here that the state priors still provide some benefit, as many of the corridor and wall placings are consistent between task and factory instances. Figure 7 shows the effect of four different observation priors:

- ϕ_1: two elements – the type of terrain occupied by the agent, and a ternary flag indicating whether any items need to be procured or assembled.
- ϕ_2: four elements – the types of terrain of the cells adjacent to the cell occupied by the agent.
- ϕ_3: six elements – the types of terrain of the cell occupied the agent as well as the cells adjacent to that, and a ternary flag indicating whether any items need to be procured or assembled. Note that $\phi_3 = \phi_1 \cup \phi_2$.
- ϕ_4: ten elements – the terrain of the 3×3 cell grid around the agent's current position, and a ternary flag indicating whether any items need to be procured or assembled.

As can be seen, all four observation priors contain information relevant to the domain, as all provide an improvement over the baselines. There is however a significant performance difference between them, motivating the idea to use the priors for feature selection as discussed in Section III-B.

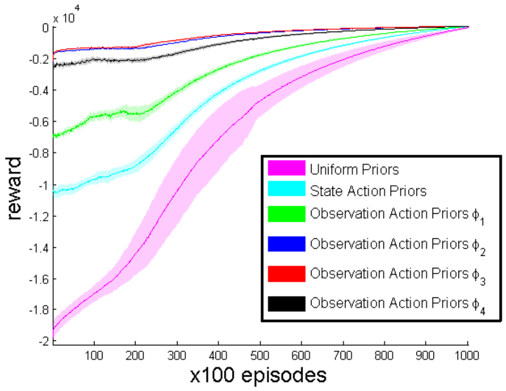


Fig. 7: Comparative performance in the modified factory domain between Q-learning with uniform priors, with state based action priors, and with four different observation based action priors: ϕ_1, ϕ_2, ϕ_3 and ϕ_4. These curves show the average reward per episode averaged over 10 runs, where the task was to assemble 4 random components. In each case the prior was obtained from 80 training policies. The shaded region represents one standard deviation.

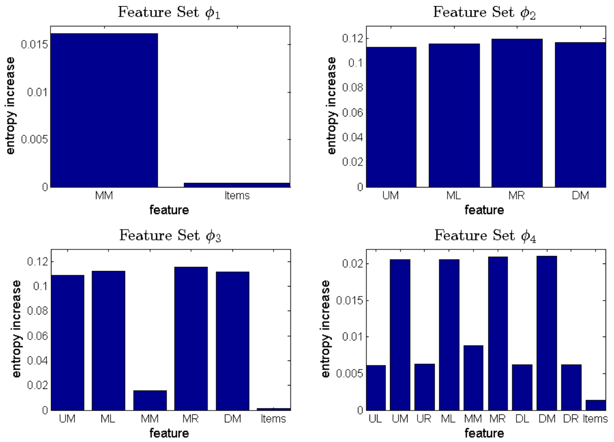


Fig. 8: Feature importance of each feature in the four different observation based action priors: ϕ_1, ϕ_2, ϕ_3 and ϕ_4. The spatial features are labeled as relative positions YX, with $Y \in \{(U)p, (M)iddle, (D)own\}$ and $X \in \{(L)eft, (M)iddle, (R)ight\}$. The feature $Items$ is a flag, indicating if the agent still needs to assemble or procure any items.

Surprisingly, Figure 7 shows that the most beneficial feature set is ϕ_3, with ϕ_2 performing almost as well. The fact that the richest feature set, ϕ_4, did not outperform the others seems counterintuitive. The reason for this is that using these ten features results in a space of $4^9 \times 3$ observations, rather than the $4^5 \times 3$ of ϕ_3. This factor of 256 increase in the observation space means that for the amount of data provided, there were too few samples to provide accurate distributions over the actions in many of the observational settings.

We next identify the set of the most useful features using Algorithm 2, by iteratively removing the features which contribute the least to reducing the entropy of the action priors. Recall that when posed as an optimisation problem in Equation (10), the term $c|\phi|$ was used as a regulariser to control for the effect of having too large a feature set: as seen in ϕ_4 in Figure 7. These results, shown in Figure 8, indicate that the relative importance for the ten features (all of which are present in ϕ_4) is consistent across the four feature sets. As expected, the ϕ_4 results indicate that the values of the cells diagonally adjacent to the current cell occupied by the agent are not important, as they are at best two steps away from the agent.

Surprisingly, neither the value of the cell occupied by the agent, nor the current state of the assembly carried by the agent are considered relevant. Consider the current state of assembly: this is already a very coarse variable, which only tells the agent that either a procurement or an assembly is required next. This is very local information, and directly affects only a small handful of the actions taken by the agent. Now consider the cell currently occupied by the agent. This indicates whether the agent is situated above an assembly or procurement point. Again, this is only useful in a small number of scenarios. Note that these features are still useful, as shown by the performance of ϕ_1 relative to state based priors or uniform priors.

What turns out to be the most useful information is the contents of the cells to the North, South, East and West of the current location of the agent. These provide two critical pieces of information to the agent. Firstly, they mitigate the negative effects that would be incurred by moving into a location occupied by a wall. Secondly, they encourage movement towards procurement and assembly points. These then constitute the most valuable features considered in our feature sets. This observation is confirmed by the fact that ϕ_2 performs very similarly to ϕ_3.

E. Human Elicited Priors

A benefit of action priors is that they need not all be learnt from policies executed by the same agent. In this way, the priors can be accumulated from a number of different agents operating in the same space. Furthermore, trajectories can be demonstrated to the learning agent, perhaps by a human teacher, as a means of training. Figure 9 illustrates how human elicited action priors can be used to improve learning performance on a new task. In this case, solution trajectories were provided by a human for 40 randomly selected assembly tasks in the modified factory domain. It can be seen that the learning agent performs comparably to having full prior policies, even though only trajectories were provided.

The human supplied trajectories were not assumed to be optimal, but the human was assumed to be familiar enough with the task such that the loss incurred when compared to an optimal solution was bounded. This demonstrates that our action prior approach works even when the training data is suboptimal. We also note here that our method can be easily applied to a case where a number of human demonstrators supply trajectories. Any trajectories which are estimated to be poorer in quality could be weighted by a smaller value of $w(\pi)$, in Equation (3). Practically, a weight could be computed for each human which indicates the competence level of that human at the task, and all trajectories supplied by that human

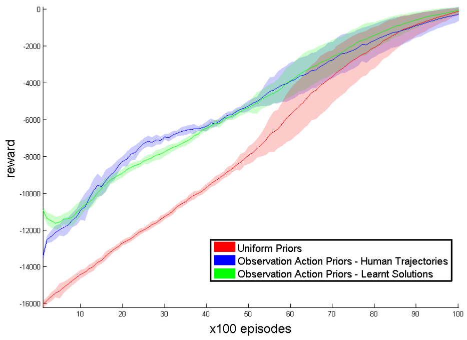


Fig. 9: Comparative performance in the modified factory domain between Q-learning with uniform priors, with *human elicited* observation based action priors, and with self trained observation based action priors. These curves show the average reward per episode averaged over 20 runs, where the task was to assemble 4 random components. In each case the prior was obtained from 40 training policies. The shaded region represents one standard deviation.

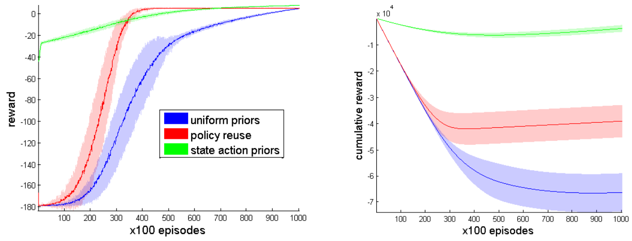
would be weighted accordingly. This approach would be useful for example in a crowdsourcing framework [21].

These results suggest that action priors can feasibly be obtained by human demonstration, and used to model human preferences in such a way as to guide the learning of an autonomous agent on a similar task.

F. The Graph Domain

We now introduce the graph domain, to show comparative performance on a domain with a very large action set. A random graph with 1,000 nodes is constructed, with 10 clusters of 100 nodes each. Intra-cluster connections are relatively dense, with any edge between two nodes in the same cluster existing with probability 0.1. On the other hand, inter-cluster connections are sparse, with a probability of 0.005 of two nodes being connected. A task is defined as the agent having to reach a particular node. The actions available to the agent are to attempt a move to any of the 1,000 nodes: this will succeed only if an edge exists between those nodes (with a reward of -1), and otherwise fail with a reward of -5. Reaching the goal node terminates the episode with a reward of 10. This domain is caricatured in Figure 2b.

In this large action set domain, as well as comparing performance between learning with action priors and with uniform priors, we compare to the use of probabilistic policy reuse (PRQ) [22]. This represents a state-of-the-art example of the family of methods which transfer knowledge by selecting the best of a library of policies to seed learning in the new task. PRQ couples Q-learning with a library of learnt policies (we use the same 20 optimal policies used in constructing the action priors) and uses these policies for exploration, keeping probabilities over the policy library to indicate task similarity and so potential for reuse. Because of the cluster structure of the graph, we expect this method to perform well, as there are representatives of each cluster in the library, which could guide the new policy into the vicinity of its desired goal. However, from the results presented in Figure 10, we see while that PRQ benefits from greatly accelerated convergence over vanilla Q-learning, the knowledge it reuses essentially points the learning algorithm in the right direction. Even if a policy from the library guides the agent to a node in the same cluster as the true goal, this does not provide the agent with advice for navigating around the cluster.



(a)

(b)

Fig. 10: Comparative results on the graph domain between Q-learning with uniform priors, Q-learning with state action priors, and probabilistic policy reuse, showing (a) per episode reward, and (b) cumulative reward. Results are averaged over 20 random tasks.

In contrast, the information encapsulated by the action priors lacks the individual goal knowledge of a single policy in the library, but instead represents the domain invariances, which in this case are the graph edges. The lower performance observed in the initial episodes of Figure 10a is then the result of essentially navigating between different potential goal nodes, rather than learning the inherent structure of the domain.

V. Priors in Humans

As was shown in Section IV, action priors can provide substantial benefits to learning and decision making, through their ability to prune and prioritise action choices in a context dependent manner. This guides the search process towards solutions that are more likely to be useful, given past experience.

Making good decisions requires evaluating the effects of decisions into the future, so as to estimate the value of each choice. However, this is an expensive search process, as the number of possible futures is exponential in the number of choices to be made, with a branching factor given by the number of actions which can be taken at each point in time.

This is a problem shared by humans. When a person is faced with any planning problem, or any decision, this cannot be solved by evaluating all possible sequences of actions, for the aforementioned reason. It is thus assumed that some automatic pruning of human decision trees occur, and negative rewards (or punishments) seem to be particularly effective at inhibiting behaviours [16]. There is evidence to suggest that this pruning is Pavlovian in nature, particularly when exposed to a large negative feedback, and that this pruning is task independent. There is also much evidence to suggest that humans rely heavily on prior knowledge for decision making, although that process is not always conscious [17].

When making decisions which require a search through large and rich problem spaces, humans thus seem to select a subset of the valid choices for further consideration. This is supported by long-standing results in psychology. For example, the game of chess has been used as a vehicle for exploring the theory of chunking [5]. A chunk refers to a recognisable set of features, which in this context is a local pattern on the chess board. Various experiments were run on players of differing skill levels, testing their ability to recall and reconstruct sequences of briefly observed board positions, which were either drawn from real games or random. These results supported the conclusion that the more skilled a player, the larger the repertoire of such patterns they have stored in memory (estimated to be of the order of well over 50,000 for expert players). Having additional information associated with these chunks would account for the ability of an expert to notice advantageous moves "at a glance" – essentially pruning away poor action choices. As a result, recognising chunks reduces the need for look-ahead search.

In this way, perceptual cues are easily recognised, and in turn they trigger actions, which could be thought of as an *intuition* of the current situation [18]. Unconscious early perceptual processes learn the relationships between the visual elements of a scene under inspection. These are invoked fast and in parallel, and act as a pre-processing step for later stages of cognition [19]. Template theory [20] extends this idea, by incorporating the idea of templates, which allow for open parameters to be present in recognisable patterns. Templates provide further semantic information which may be helpful in making action selection, again pruning and biasing the search process. It has been conjectured in multiple studies that "experts use the rapid recognition of complex patterns, mediated by perceptual templates, in order to efficiently constrain and guide their search for good moves" [19].

This interpretation corresponds to the semantics of action priors, allowing a search process to prioritise exploration based on recommended actions for a given context. Our approach also hypothesises and demonstrates how these priors may be gathered from experience. The overall effect is that if one does not know what action to take, given uncertainty in the task, it could be chosen according to the action prior, which represents common sense and intuition in the domain[5]. This bias can be used in instantaneous action selection to cause no immediate harm in the absence of more task specific knowledge.

VI. Related Work

The idea of learning priors over behaviours and using these to bias search is an important theme in various contexts. Notably, in problems where the policies are drawn from some parametric policy space, policy priors can be learnt over the space and used to generalise knowledge [23]. Importantly, in our work we do not assume a known model over policy space. In another thread, action pruning over the *entire* domain has been studied [24], rather than the case we consider which treats each state or observation individually. This approach has the effect of pruning away actions that are always harmful throughout the domain, but the pruning is not context-specific. Other work has explored incorporating a heuristic function into the action selection process to accelerate reinforcement learning [25], but this does not address the acquisition of these prior behaviours. That approach is also sensitive to the choice of values in the heuristic function, and requires setting additional parameters.

Our interpretation of action priors as distilling domain specific knowledge from a set of task instance specific behaviours is similar to the idea of dividing a problem (or set thereof) into an agent space and a problem space [26]. The agent space refers to commonalities between the problems, whereas the problem space is specific to each task. This formulation involves learning a reward predictor which, in a sense, can be used to guide action selection.

Where our approach reduces learning time by biasing and restricting the search over action space, similar benefits have been found by only searching over limited aspects of the *state* space, particularly in relational planning problems. Examples include reasoning only in terms of the subset of objects that are relevant for current planning purposes (relevance grounding) [12], or using variables to stand in for the objects relevant to the current actions (deictic references) [27]. This is similar to the way in which pruning is used in search, but we prune based on the expected utility of the action, estimated from its utility in the optimal policies for a set of previous tasks.

Options in hierarchical reinforcement learning are defined as temporally extended actions with initiation sets where they can be invoked, and termination conditions [28], [29]. Although there are similarities between learning the initiation sets of options and action priors, they are distinct in that an initiation set defines where the option *can physically be instantiated*, whereas an action prior describes regions where the option is *useful*. For example, while pushing hard against a door may always be possible, this level of force would be damaging to a glass door, but that choice would not be ruled out by options. Consequently, action priors not only augment options, but are beneficial when using large sets of options to mitigate the negative impact of exploration. Similarly, action priors are related to the idea of learning *affordances* [30], being action possibilities provided by some environment. These are commonly modelled as properties of objects, and can be learnt from experience (see e.g. [31]). Again, the ambitions of action priors are subtly different to that of affordances, as affordances model the notion of where an action *can* be applied, whereas an action prior describes whether or not it *should* be applied. Ideally, action priors should be applied over action sets which arise as the result of affordance learning.

One may alternatively reuse experience by decomposing a task into a set of subcomponents, learning optimal policies for these elements, and then piecing them together [32], possibly applying transforms to increase generality [33]. Action priors differ by discovering a subset of reasonable behaviours in each perceptual state, rather than one optimal policy, and thus can be used for a variety of different tasks in the same domain, although the policies must still be learned. As a result, action priors are complementary to this decomposition approach.

Novel Contributions: The proposed action priors approach thus builds on existing attempts to formalise the way in

[5] We do note that this notion of common sense is Markovian in nature, as it does not consider temporally extended behaviours.

which past experience can be generalised and applied to new situations, where the tasks differ. To the best of our knowledge, this is the first work to quantify the utility of each action in every state across a repertoire of different tasks, and use this to guide exploration and learning in new tasks.

VII. Conclusion

We introduce the concept of action priors, as distributions over the action set of an agent, conditioned on either state or observations received. This provides a mechanism whereby an agent can, over the course of a lifetime, accumulate general knowledge about a domain in the form of local behavioural invariances. This is an important form of knowledge transfer, as it allows for a decision making agent to bias its search process towards previously useful choices.

As alluded to in Section V, pruning action possibilities is useful in other decision making paradigms. The same principle demonstrated here in the context of reinforcement learning could thus be applied to planning, and the action priors could be used to either prune certain actions or provide a preference list for depth-first search. In either case, this should guide the search toward solutions that have been previously encountered, allowing for focused search biased by experience. Provided there is structure to the behaviours in the domain, the result is that paths to reach a goal (either in the case of motion planning or classical planning) could be discovered faster through more directed search. This additionally adds an element of safety to the exploration process. As such, this principle has far-reaching applications in real-world domains.

We have shown that not only are action priors useful in practice, but they also draw many parallels to the way in which humans prune action choices automatically when making decisions. Hopefully further work in this direction can serve to strengthen this connection of accelerating learning between human and artificial decision making.

Acknowledgements

Subramanian Ramamoorthy acknowledges the support of the European Commission through SmartSociety Grant agreement no. 600854, under the programme FOCAS ICT-2011.9.10. The authors gratefully acknowledge the helpful and insightful comments made by the four anonymous reviewers, which greatly improved the quality of this paper.

References

[1] S. Koos, A. Cully, and J.-B. Mouret, "High resilience in robotics with a multi-objective evolutionary algorithm," *Genetic and evolutionary computation conference companion*, pp. 31–32, 2013.

[2] S. Thrun, "Is learning the n-th thing any easier than learning the first?," *Advances in Neural Information Processing Systems*, pp. 640–646, 1996.

[3] M. E. Taylor and P. Stone, "Transfer Learning for Reinforcement Learning Domains: A Survey," *Journal of Machine Learning Research*, vol. 10, pp. 1633–1685, 2009.

[4] I. Havoutis and S. Ramamoorthy, "Motion planning and reactive control on learnt skill manifolds," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1120–1150, 2013.

[5] H. A. Simon and W. G. Chase, "Skill in Chess: Experiments with chess-playing tasks and computer simulation of skilled performance throw light on some human perceptual and memory processes," *American Scientist*, vol. 61, pp. 394–403, July-August 1973.

[6] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. The MIT Press, 1998.

[7] C. J. Watkins and P. Dayan, "Q-Learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.

[8] B. S. Rosman, "Learning Domain Abstractions for Long Lived Robots", *Ph.D. Dissertation*, The University of Edinburgh, 2014.

[9] B. S. Rosman and S. Ramamoorthy, "What good are actions? Accelerating learning using learned action priors," *International Conference on Development and Learning and Epigenetic Robotics*, November 2012.

[10] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial Intelligence*, vol. 101, pp. 99–134, May 1998.

[11] N. K. Jong and P. Stone, "State Abstraction Discovery from Irrelevant State Variables," *International Joint Conference on Artificial Intelligence*, pp. 752–757, August 2005.

[12] T. Lang and M. Toussaint, "Relevance Grounding for Planning in Relational Domains," *European Conference on Machine Learning*, 2009.

[13] B. S. Rosman and S. Ramamoorthy, "A Multitask Representation using Reusable Local Policy Templates," *AAAI Spring Symposium Series on Designing Intelligent Robots: Reintegrating AI*, 2012.

[14] B. R. Donald, "On information invariants in robotics," *Artificial Intelligence*, vol. 72, no. 1, pp. 217–304, 1995.

[15] B. Rosman, "Feature selection for domain knowledge representation through multitask learning," *International Conference on Development and Learning and Epigenetic Robotics (ICDL)*, 2014.

[16] Q. J. Huys, N. Eshel, E. O'Nions, L. Sheridan, P. Dayan, and J. P. Roiser, "Bonsai trees in your head: how the Pavlovian system sculpts goal-directed choices by pruning decision trees," *PLoS computational biology*, vol. 8, no. 3, p. e1002410, 2012.

[17] M. Strevens, *Tychomancy*. Harvard University Press, 2013.

[18] H. A. Simon, "What is an "explanation" of behavior?," *Psychological Science*, vol. 3, no. 3, pp. 150–161, 1992.

[19] M. Harre, T. Bossomaier, and A. Snyder, "The Perceptual Cues that Reshape Expert Reasoning," *Scientific Reports*, vol. 2, July 2012.

[20] F. Gobet and H. A. Simon, "Templates in chess memory: A mechanism for recalling several boards," *Cognitive psychology*, vol. 31, no. 1, pp. 1–40, 1996.

[21] G. V. de la Cruz, B. Peng, W. S. Lasecki and M. E. Taylor, "Generating Real-Time Crowd Advice to Improve Reinforcement Learning Agents," *Proceedings of the AAAI Workshop on Learning for General Competency in Video Games*, 2015.

[22] F. Fernandez and M. Veloso, "Probabilistic policy reuse in a reinforcement learning agent," *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, 2006.

[23] D. Wingate, N. D. Goodman, D. M. Roy, L. P. Kaelbling, and J. B. Tenenbaum, "Bayesian Policy Search with Policy Priors," *International Joint Conference on Artificial Intelligence*, 2011.

[24] A. A. Sherstov and P. Stone, "Improving Action Selection in MDP's via Knowledge Transfer," *AAAI*, pp. 1024–1029, 2005.

[25] R. A. C. Bianchi, C. H. C. Ribeiro, and A. H. R. Costa, "Heuristic Selection of Actions in Multiagent Reinforcement Learning," *International Joint Conference on Artificial Intelligence*, pp. 690–695, 2007.

[26] G. D. Konidaris and A. G. Barto, "Autonomous shaping: Knowledge transfer in reinforcement learning," *Proceedings of the 23rd International Conference on Machine Learning*, pp. 489–496, 2006.

[27] H. M. Pasula, L. S. Zettlemoyer, and L. P. Kaelbling, "Learning symbolic models of stochastic domains," *Journal of Artificial Intelligence Research*, vol. 29, pp. 309–352, May 2007.

[28] D. Precup, R. S. Sutton, and S. Singh, "Theoretical results on reinforcement learning with temporally abstract options," *European Conference on Machine Learning*, 1998.

[29] M. Pickett and A. G. Barto, "PolicyBlocks: An Algorithm for Creating Useful Macro-Actions in Reinforcement Learning," *International Conference on Machine Learning*, pp. 506–513, 2002.

[30] J. J. Gibson, *The Ecological Approach to Visual Perception*. Lawrence Erlbaum Associates, Inc., 2nd ed., 1986.

[31] J. Sun, J. L. Moore, A. Bobick, and J. M. Rehg, "Learning Visual Object Categories for Robot Affordance Prediction," *The International Journal of Robotics Research*, vol. 29, pp. 174–197, February/March 2010.

[32] D. Foster and P. Dayan, "Structure in the Space of Value Functions," *Machine Learning*, vol. 49, pp. 325–346, 2002.

[33] B. Ravindran and A. G. Barto, "Relativized Options: Choosing the Right Transformation," *Proceedings of the Twentieth International Conference on Machine Learning*, 2003.