First-Order Function Approximation for Transfer Learning in Relational MDPs

Jun Hao Alvin Ng[1,2], **Ronald P. A. Petrick**[1]
Edinburgh Centre for Robotics
[1]Department of Computer Science, Heriot-Watt University
[2]School of Informatics, University of Edinburgh
Edinburgh, Scotland, United Kingdom
{Alvin.Ng, R.Petrick}@hw.ac.uk

Abstract

Planning problems with a first-order structure can be modelled compactly with Relational Markov Decision Processes (RMDPs). If the model is unknown, value-based reinforcement learning methods can be used to solve these problems. The action-value function is approximated with features which are conjunctive ground state fluents. However, this approximation does not exploit the first-order structure of RMDPs and the generated policy can only solve a ground MDP of the RMDP. Our objective is to learn a generalised function approximation which induces a policy that can solve multiple ground MDPs. We achieve this by using conjunctive lifted state fluents as first-order features. This first-order approximation gives better generalisation but has a coarser granularity which can worsen performance. We propose the combination of first-order features and ground features to get both of their strengths. Empirical results for four domains show that our method could generalise over problems regardless of their scales and allow transfer learning.

Introduction

Intelligent agents capable of learning are desired for their adaptive behaviour and ability to solve planning problems given little prior information such as the underlying models specifying the dynamics between an agent and its environment. Reinforcement learning (RL) (Sutton and Barto 2018) is a powerful approach to tackle such problems: an agent acts in the environment and learns from the resulting observations to improve subsequent reasoning. Sample complexity, the amount of observations required to achieve near-optimal behaviour, is usually high in large scale planning problems. However, if the problems are structured, then generalisation could reduce sample complexity by applying learned knowledge to unseen situations similar to observed ones. The problem is abstracted in a relational representation and traditional RL methods, possibly with some modifications, are applied to the abstracted problem.

We focus on online RL for problems represented with Relational Markov Decision Processes (RMDPs). The action-value function, or Q-function, is approximated by projecting the state space into a lower dimensional space using a set of features. We utilise an existing online feature discovery algorithm, `iFDD+` (Geramifard, Dann, and How 2013), to incrementally construct sets of conjunctive state fluents as features to reduce the value approximation errors. Since state fluents are predicates ground over objects, problems with different objects or numbers of objects are represented by different sets of state fluents. Therefore, generalisation is not possible. Instead, we use conjunctive lifted state fluents, or first-order features, to approximate the Q-values of lifted actions rather than ground actions. The feature space for this relational or first-order approximation is identical for all ground MDPs constructed from a RMDP. Thus, generalisation and knowledge transfer is possible. However, the state abstraction increases: more states are partitioned into a region where they are considered the same for the purpose of value approximation. When this is not the case, performance can deteriorate.

This paper presents two main contributions. First, we represent the function approximation with first-order features which allows generalisation and transfer learning over different planning problems independent of the objects, number of objects, initial states, and goal states. Second, we combine the strengths of first-order approximation and ground approximation by considering a policy induced by their function approximations. We assume that preconditions are known for action selection and an illegal action will never be selected. The learning problem is to learn a first-order function approximation which induces a generalised policy that could solve multiple different planning problems.

The rest of the paper is organised as follows. First, we review related work on solving RMDPs and present the necessary technical background. Next, we describe our method in detail. We then present an ablation study of our method on four domains. Lastly, we conclude and discuss future work.

Related Work

Large scale planning problems can be solved efficiently by exploiting first-order representations such as RMDPs (Wang, Joshi, and Khardon 2008; Van Otterlo 2005). There are broadly two types of methods: those which assume the transition function and reward function are known, and those which do not. Our work is a model-free relational RL method which performs online feature discovery using

iFDD+ (Geramifard, Dann, and How 2013), and is more closely related to the latter. Morales (2003) defines abstract actions (r-actions) and abstract states (r-states) in a relational representation, then uses a modified Q-learning to learn policies based on the induced relational abstract state-action space. Some domain knowledge is required to define the r-states. Guestrin et al. (2003) samples ground MDPs of a RMDP and solves them with linear programming to learn a generalised, class-based value function; objects in the MDPs are classified into classes with supervised learning. Van Otterlo (2004) creates an abstract MDP of the underlying RMDP with the use of background knowledge, then solves the abstract MDP with modified Q-learning and prioritized sweeping. Mausam (2003) and Džeroski, De Raedt, and Driessens (2001) partition the state space into finer regions, each of which has a real-value representing the Q-value, using relational regression trees. Wu and Givan (2005) performs supervised learning of relational features, represented as decision trees, by adding features which correlate well to the Bellman error of value functions. Similarly, Croonenborghs et al. (2007) learns a relational CPD and a relational reward function online, represented with relational decision trees, and uses planning techniques with the learned models to provide better estimates of the Q-values. A decision tree is sensitive to the order of node splitting, rendering this work ill-suited in online RL where later observations yield new information which might necessitate the reconstruction of the trees. Ramon, Driessens, and Croonenborghs (2007) proposes a tree restructuring operation but requires statistics to be stored for every node.

We use conjunctive features similar to SVRRL (Sanner 2006) which has no such issues. SVRRL represents a value function with a relational naive Bayes net and learns both the values and structure of the network. It utilises a distance metric to generalise over handcrafted non-binary relational features. Likewise, RIB (Driessens and Ramon 2003), which uses instance-based learning where selected observed examples are stored, requires a distance metric to compute Q-values of unseen state-action pairs. Our work uses state fluents as binary features and does not require any distance metric to be defined. In these aspects, (Walker 2004) and (Wu and Givan 2007) are most similar to our work. The former selects features stochastically with the use of prior training data while we discover features online. The latter uses features with one free variable and approximates the value function while our features can have any number of free variables and we approximate the Q-function.

Preliminaries

Markov Decision Process (MDP). MDPs model fully-observable problems with uncertainty. A finite-horizon MDP is a tuple of the form $(\boldsymbol{S}, \boldsymbol{A}, T, R, s_0, H, \gamma)$ where $\boldsymbol{S}$ is a set of states, $\boldsymbol{A}$ is the set of actions, $T : \boldsymbol{S} \times \boldsymbol{A} \times \boldsymbol{S} \to [0, 1]$ is the transition function, $R : \boldsymbol{S} \times \boldsymbol{A} \to \mathbb{R}$ specifies rewards for performing actions, s_0 is the initial state, H is the planning horizon, and γ is the discount factor. We use symbols with boldface to indicate sets. MDPs are described explicitly by expressing T with a transition matrix for each action. For large state-action spaces, it is impractical to represent T with transition matrices. Factored MDPs (Boutilier, Dearden, and Goldszmidt 2000) represent a state s with a set of state variables $\boldsymbol{P}$ where $s = \{p_i\}_{i=1}^{|\boldsymbol{P}|}$. If the transition of p_i depends only on a small number of state variables $\bar{\boldsymbol{P}} \subset \boldsymbol{P}$, then T can be represented compactly by dynamic Bayesian networks (DBN). The conditional probability function (CPF) for p_i can be expressed as $T(p_i' \mid \boldsymbol{P}, a) = T(p_i' \mid \bar{\boldsymbol{P}}, a)$ where p_i' is the variable at the next time step and $a \in \boldsymbol{A}$.

Relational Markov Decision Process (RMDP). A RMDP $(\boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \mathcal{R}, s_0, H, \gamma)$ is a first-order representation of a factored MDP. $\boldsymbol{\mathcal{O}}$ is a set of objects, each associated with a type, $\boldsymbol{\mathcal{P}}$ is a set of state predicates over objects, $\boldsymbol{\mathcal{S}}$ is a set of all possible state specifications over $\boldsymbol{\mathcal{O}}$ and $\boldsymbol{\mathcal{P}}$, $\boldsymbol{\mathcal{A}}$ is the set of all possible instantiated actions, $\mathcal{T} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{S}} \to [0, 1]$ is the transition function, and $\mathcal{R} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \to \mathbb{R}$ is the reward function. Each predicate is applied over a type-consistent tuple of objects. Therefore, different ground MDPs can be constructed from a RMDP given different sets of objects $\boldsymbol{\mathcal{O}}$. If generalised policies for a RMDP can be found, then these policies can be used directly to solve any ground MDPs resulting from this RMDP.

Relational Dynamic Influence Diagram Language (RDDL). RDDL (Sanner 2010) is a planning language for describing RMDPs and is used in recent International Probabilistic Planning Competitions (IPPC) (Sanner 2011; Grzes, Hoey, and Sanner 2014). Semantically, RDDL describes DBNs extended with an influence diagram. RDDL domain and problem files are given as inputs to a planner. The domain file specifies object types, non-fluents, fluents, CPFs, and a reward function. Fluents (non-fluents) are state variables that change (do not change) with time. The problem file specifies objects, initial state, and values of non-fluents. A domain can have different problems. We refer to non-fluents, fluents, and predicates interchangeably unless necessary to differentiate between them. RDDL models parallel effects. In contrast, PPDDL (Younes and Littman 2004), another planning language, models transitions with correlated effects.

Reinforcement Learning (RL). When the transition function and reward function are not known, RL can be used for sequential decision-making. The sample complexity of an RL algorithm is the number of observations needed to achieve near-optimal results. The learning objective is to find a policy π which maximises $\sum_{t=0}^{H} \gamma^t r_t$ where r_t is the immediate reward received at time step t. π is generated from the Q-function where the Q-value is given as:

$$Q_H^\pi(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{H} \gamma^t r_t \,\middle|\, s_0 = s, a_0 = a \right]. \qquad (1)$$

Temporal Difference (TD) learning methods update their estimates of the Q-function given observations (s_t, a_t, r_t, s_{t+1}). TD(λ) methods average over n-step returns by using eligibility traces where λ is the trace decay. The update rule for the Q-function is given by:

$$Q_{t+1}(s_t, a_t) = Q_t(s_t, a_t) + \alpha \delta_t e_t(s_t, a_t), \qquad (2)$$

$$\delta_t = r_t + \gamma Q_t(s_{t+1}, a_{t+1}) - Q_t(s_t, a_t), \quad (3)$$

where α is the learning rate, $e_t(s,a)$ is the eligibility trace for (s,a), and δ_t is the TD error.

Linear Function Approximation. The Q-function can be approximated by linear functions of the form $Q_\theta(s,a) = \boldsymbol{\theta}^T \boldsymbol{\Phi}(s,a)$ to reduce the dimensions of the state spaces (Sutton and Barto 2018). The set of features $\boldsymbol{\Phi}(s,a)$ maps (s,a) to a vector of real numbers. The goal is to find the set of features and weights so that Q_θ closely approximates the optimal Q-function. The update rule for the $i\text{-}th$ component of the weight vector $\boldsymbol{\theta}$ at time step t is:

$$\theta_t^i = \theta_{t-1}^i + \alpha \delta_t \frac{\phi_i(s_t, a_t)}{||\boldsymbol{\Phi}(s_t, a_t)||_1}, \quad (4)$$

where the normalization factor is the $L1$ norm. $\boldsymbol{\Phi}$ (we omit s and a for brevity) partitions the state space into regions such that states in a region are considered the same for the purposes of approximating the Q-values. The coverage of a feature is the region of the state space for which the feature evaluates to 1 (or non-zero for non-binary features) (Geramifard et al. 2013b). Features with low coverage give fine granularity in approximation and thus have better accuracy than features with high coverage. This impacts the soundness of policies. On the other hand, features with high coverage offer better generalisation as learning is done over a smaller number of partitions of the state space.

Online RL with First-Order Approximation

In this work, we propose a **first-order linear function approximation** of the Q-function to induce a generalised deterministic policy which can directly solve multiple ground MDPs constructed from a RMDP. We use first-order features which are conjunctive lifted state fluents. New features are added incrementally to the set of features $\boldsymbol{\Phi}$ to reduce the approximation errors. A first-order feature must be fully ground in order to map (s,a) to a real value. However, there may exist several possible substitutions to ground a first-order feature. We utilise contextual knowledge and a selection criteria to reduce the set of possible substitutions. Compared to a **ground approximation**, where features are conjunctive ground state fluents, a first-order approximation has a coarser granularity which could deteriorate performance. We combine the strengths of both approximations in a **mixed approximation**. In this section, we elaborate on the aforementioned approaches and issues.

First-Order Function Approximation

A linear function approximation of the Q-function, Q_θ, can be used to evaluate the Q-values of actions in a problem with state-action space $\boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}}$ if the set of features $\boldsymbol{\Phi}$ maps (s,a) to a real value for every $s \in \boldsymbol{\mathcal{S}}$ and every $a \in \boldsymbol{\mathcal{A}}$. This implies that for every $\phi_f \in \boldsymbol{\Phi}$, $\phi_f \subseteq s \forall s \in \boldsymbol{\mathcal{S}}$. Q_θ can be applied to any problem with the state-action space $\boldsymbol{\mathcal{S}}' \times \boldsymbol{\mathcal{A}}'$ where $\boldsymbol{\mathcal{S}}' \subseteq \boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{A}}' \subseteq \boldsymbol{\mathcal{A}}$. This poses two limitations. First, transfer learning from a small scale problem to a large scale problem is not possible; this is a common motivation for transfer learning as learning is more efficient in a smaller state-action space. Second, non-fluents are invariants of a state. Thus, two problems with different non-fluents have non-overlapping state spaces and generalisation between them is not possible. Given these limitations, we propose a first-order approximation with a set of first-order features which is independent of the objects in $\boldsymbol{\mathcal{O}}$ and the ground non-fluents.

Algorithm 1: Initialise a set of first-order features for a lifted action

```
1 Function GET_LIFTED_FEATURES(P, A_â):
2     Φ_â = ∅
3     for a ∈ A_â do
4         Φ_a ← GET_GROUND_FEATURES(P, a)
5         Φ_â ← Φ_â ∪ LIFT(Φ_a, σ_a)
6     return QUANTIFIED(Φ_â)
```

We instantiate a set of first-order features for each lifted action $\hat{a}$, $\boldsymbol{\Phi}_{\hat{a}}$, instead of every ground action[1]; otherwise, Q_θ will not be independent of $\boldsymbol{\mathcal{O}}$ because ground actions are defined over $\boldsymbol{\mathcal{O}}$. This is shown in Algorithm 1. The inputs are the set of state fluents $\boldsymbol{\mathcal{P}}$ and the set of ground actions for $\hat{a}$, $\boldsymbol{\mathcal{A}}_{\hat{\boldsymbol{a}}}$. For each ground action $a \in \boldsymbol{\mathcal{A}}_{\hat{\boldsymbol{a}}}$, a set of ground features $\boldsymbol{\Phi}_a$ is initialised (`GET_GROUND_FEATURES` in line 4). In this work, this is the set of every state fluent and non-fluent, as well as their negation. Each feature $\phi_f \in \boldsymbol{\Phi}_a$ is lifted in accordance with the substitution of a, σ_a (`LIFT` in line 5). For example, $\sigma_a = \{?arg_1/x, ?arg_2/y\}$ grounds $\mathtt{a}(?arg_1, ?arg_2)$ to $\mathtt{a}(x,y)$[2]. A feature might be partially lifted if it contains objects which are not in the arguments of a. $\boldsymbol{\Phi}_{\hat{a}}$ is the union of these partially or fully lifted features for each $a \in \boldsymbol{\mathcal{A}}_{\hat{\boldsymbol{a}}}$ (line 5). Remaining objects in $\boldsymbol{\Phi}_{\hat{a}}$ are substituted with **free variables** to yield a set of fully lifted, or first-order, features (`QUANTIFIED` in line 6). In contrast with bound variables which are ground with objects in the arguments of actions, free variables can be ground with any object of the same type.

$\boldsymbol{\Phi}_{\hat{a}}$ is independent of $\boldsymbol{\mathcal{O}}$ and the number of objects in a problem. Thus, a first-order approximation can be applied to different problems even with different scales. A first-order feature must be fully ground before it can map (s,a) to a real value. First, the substitution σ_a is applied. The features are not fully ground if there are free variables. The set of possible substitutions for $\boldsymbol{\Phi}_{\hat{a}}$, $\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}$, consists of every combination of objects for the free variables. We discuss the grounding of first-order features in a later part of this section.

Example 1 (Recon domain)**.** We use the `Recon` (`RC`) domain from the IPPC (Sanner 2011) as an example. In `RC`, an agent moves in a 2-dimensional grid with the action $\mathtt{move}(?pos)$ where there is a base, some hazard, and objects (represented by the variable $?obj$) in different grid positions $?pos$. The agent ($?agent$) is equipped with three different tools ($?tool$) which it can use on objects with the action $\mathtt{useToolOn}(?agent, ?tool, ?obj)$ to different effects if the

[1] We use $\hat{}$ to denote a lifted fluent or a set of lifted fluents.

[2] We use $?arg$ to represent a variable of type arg.

agent and the object ($?obj$) are at the same position. A tool can be damaged (damaged($?tool$)) if the agent is at a hazard or is adjacent to it. The probability of getting a good reading from a damaged tool is lower. The agent can repair a tool if it is at the base with the action repair($?agent, ?tool$). The position of the agent is represented by the state fluent agentAt($?agent, ?pos$) and the position of an object, which does not change, is represented by the non-fluent OBJECT_AT($?obj, ?pos$)[3]. In a problem instance of RC with $n \times n$ grid, there are n^2 objects of type pos and the agent can be in n^2 locations which are represented by n^2 ground state fluents of agentAt($?agent, ?pos$). These are used as features in a ground approximation. Thus, a ground approximation is dependent on $\mathcal{O}$ and the number of objects in $\mathcal{O}$. This does not allow generalisation to another problem instance with a different grid size.

Example 2 (First-order approximation)**.** Following Example 1, the n^2 ground state fluents of agentAt($?agent, ?pos$) are represented by two first-order features, agentAt($?agent, ?pos$) and agentAt($?agent, !pos$), for the action move($?pos$) regardless of the size of the grid[4]. Thus, a first-order approximation is not dependent on $\mathcal{O}$ and the number of objects in $\mathcal{O}$, and generalisation to another problem instance with a different grid size is possible.

Learning Conjunctive Features

The set of first-order features is initialised with Algorithm 1 where each feature is a lifted state fluent. This is often inadequate in approximating the optimal Q-function. More complex features, which are conjunctive state fluents, can be added to $\mathbf{\Phi}$ to reduce the approximation errors. For this purpose, we utilise iFDD+ (Geramifard, Dann, and How 2013), a model-free, online feature discovery algorithm. We provide iFDD+ with an initial set of features. For a first-order approximation, this is generated with Algorithm 1. For a ground approximation, this is generated with line 4 in Algorithm 1 (GET_GROUND_FEATURES). Then, iFDD+ adds candidate features to $\mathbf{\Phi}$ if their cumulative approximation errors, or relevances, exceed a user-defined threshold ξ. Since the set of candidate features, $\mathbf{\Phi}^c$, consists of features which are the conjunction of any two features (its **parent features**) in $\mathbf{\Phi}$, this in turn introduces increasingly complex features to $\mathbf{\Phi}^c$. The relevance of a feature ϕ_f at time step t is:

$$\eta_t(\phi_f) = \frac{\left| \sum_{i=0,\phi_f(s_i,a_i)=1}^{t} \delta_i \right|}{\sqrt{\sum_{i=0,\phi_f(s_i,a_i)=1}^{t} 1}}. \tag{5}$$

Equation 5 is based on the relation of iFDD+ to orthogonal matching pursuit algorithms. We refer readers to (Geramifard et al. 2013a) for further details. The features are binary: $\phi_f(s,a) = 1$ if ϕ_f is active in s and 0 otherwise. A feature is active if it is true in the state and is not a parent feature of any active feature.

[3]We use lowercase (uppercase) letters for the names of fluents (non-fluents).

[4]We use $!pos$ to denote a free variable of type pos.

Contextual Grounding

The set of first-order features $\mathbf{\Phi}$ can be ground in several possible ways if at least one of its features ϕ_f have one or more free variables. The number of possible substitutions for ϕ_f and for $\mathbf{\Phi}_{\hat{a}}$ are:

$$|\boldsymbol{\sigma}_{\phi_f}| = \prod_{v \in \boldsymbol{v}_{\phi_f}} |v|, \tag{6}$$

$$|\boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}| = \prod_{\phi_f \in \mathbf{\Phi}_{\hat{a}}} |\boldsymbol{\sigma}_{\phi_f}|, \tag{7}$$

respectively, where $\boldsymbol{v}_{\phi_f}$ is the set of free variables in ϕ_f and $|v|$ is the number of objects of the same type as the free variable v. If we impose a constraint that a free variable must be substituted with the same object for every feature in $\mathbf{\Phi}_{\hat{a}}$, then the number of possible substitutions is:

$$|\boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}| = \prod_{v \in \boldsymbol{v}_{\mathbf{\Phi}_{\hat{a}}}} |v| \tag{8}$$

where $\boldsymbol{v}_{\mathbf{\Phi}_{\hat{a}}} = \bigcap_{\phi_f \in \mathbf{\Phi}_{\hat{a}}} \boldsymbol{v}_{\phi_f}$. The selection of a substitution is crucial as it influences the Q-values, the step update of $\boldsymbol{\theta}$ (see Equation 4), and feature discovery—we refer to this issue as the **grounding ambiguity**.

Example 3 (Number of groundings)**.** In RC, $|\boldsymbol{\sigma}_{\phi_f}| = |obj| \times |pos|$ for ϕ_f = OBJECT_AT($!obj, !pos$). In a problem with 5 objects and 4×4 grid, for $\hat{a}$ = useToolOn($?agent, ?tool, ?obj$), $|\boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}| \sim 450 \times 10^{15}$ (for Equation 7) or 288 (for Equation 8).

We can consider every possible substitutions $\boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}$ by applying an element-wise logical OR operation on the set of vectors of real numbers resulting from the grounding of $\mathbf{\Phi}_{\hat{a}}$ with σ ($\mathbf{\Phi}_{\hat{a}}^{\sigma}$) for every $\sigma \in \boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}$. Semantically, a first-order feature ϕ_f is true in a state s if there exists a substitution which makes it true in s. However, ϕ_f would then be true in many states (e.g., in RC, pictureTaken($!obj$) will be true if the agent has taken a picture of any object). This gives a function approximation with coarse granularity which can deteriorate performance. We discuss this in detail in the *Granularity of First-Order Approximation* section. To remedy this, we propose the use of contextual knowledge to eliminate substitutions in $\boldsymbol{\sigma}_{\mathbf{\Phi}_{\hat{a}}}$ which conflict with substitutions due to contextual knowledge. Two substitutions are conflicting if they substitute a variable with different objects. We introduce two forms of contextual knowledge which require trivial domain knowledge to obtain: goal and location.

Goal Context. Goals $\boldsymbol{\mathcal{G}}$ are represented by state fluents which can be trivially determined from either the reward function or the definition of terminal states, both of which are commonly assumed to be known in RL problems. We consider every unachieved goal g for contextual grounding. Then, the set of substitutions due to goal context is defined as $\boldsymbol{\sigma}_{goal} = \bigcup_{g \in \boldsymbol{\mathcal{G}}} \sigma_{goal=g}$ where $\sigma_{goal=g}$ is the contextual grounding due to g. We assume that achieved goals do not affect the policy in the present and future. The Q-values then represent the expected values of actions for achieving these goals. This is similar to the goal-associated Q-function in

(Veeriah, Oh, and Singh 2018) but is not the same as additive rewards as defined in (Sanner and Boutilier 2012) where each goal is assumed to contribute uniformly and additively to the reward.

Example 4 (Determine state fluents for goal context)**.** In RC, a reward of 20 (-20) is obtained for taking a good (bad) picture of an object. The state fluent $\texttt{pictureTaken}(?obj)$ represents the fact that the picture of the object $?obj$ has been taken. A terminal state is reached if the agent has taken pictures of all objects. The goals are the set of ground state fluents for $\texttt{pictureTaken}(?obj)$ for every object of type obj. Following Example 3, using $\sigma_{goal=\texttt{pictureTaken}(o1)} = \{!obj/o1\}$, $|\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}|$ is reduced from 288 to 48.

Location Context. In some domains, agents move in an environment and a state fluent represents the location of the agent. If the problem has a factored transition function, as is the case in RMDPs, we can assume that the agent can only interact with objects in its vicinity. Following this assumption, location context substitutes a free variable with the current location of the agent. Location context can be used with the goal context if there is no conflict (i.e., they substitute free variables of different types) and this queries if the agent is at the same location as the goal of interest.

Example 5 (Determine state fluents for location context)**.** In RC, the agent $a1$ uses tools on objects which are at the same location. If the agent is at $pos1$ (i.e., $\texttt{agentAt}(a1, pos1)$ is true), then the contextual grounding due to location context is $\sigma_{loc} = \{!pos/pos1\}$. The first-order features $\texttt{OBJECT\_AT}(?obj, !pos)$ and $\texttt{BASE}(!pos)$ are grounded as $\texttt{OBJECT\_AT}(?obj, pos1)$ and $\texttt{BASE}(pos1)$ (true if the base is at $pos1$), respectively. They query if the agent and the object, or base, are at the same location. It is of no interest whether the object or the base is elsewhere. Following Example 3, $|\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}|$ is reduced from 288 to 18. Combined with $\sigma_{goal=\texttt{pictureTaken(o1)}} = \{!obj/o1\}$, $\texttt{OBJECT\_AT}(!obj, !pos)$ is ground to $\texttt{OBJECT\_AT}(o1, pos1)$. This assumption exploits the factored state space where the transition of a state fluent (e.g., $\texttt{pictureTaken}(o1)$) depends on a small number of state fluents (e.g., $\texttt{agentAt}(a1, pos1)$ and $\texttt{OBJECT\_AT}(o1, pos1)$) which are determined with contextual knowledge. If both goal context and location context are used, then $|\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}| = 3$ (i.e, the number of tools, $|!tool| = 3$).

Selecting a Substitution. Besides the use of contextual knowledge, we propose a selection method to consider $\boldsymbol{\sigma}_M \subseteq \boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}$ for grounding $\boldsymbol{\Phi}_{\hat{a}}$ such that a metric M is maximised:

$$\boldsymbol{\sigma}_M = \{\arg\max_{\sigma \in \boldsymbol{\sigma}} M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big)\}. \tag{9}$$

If contextual grounding is used, it is applied before this selection method. We consider two possible definitions of M:

$$M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big) = \boldsymbol{\theta}^T \boldsymbol{\Phi}^{\sigma}(s,a) = Q_{\theta}(s,a), \tag{10}$$

$$M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big) = \sum_{\phi_f \in \boldsymbol{\Phi}^{\sigma}_{\hat{a}}} \phi_f(s,a)|\phi_f|^2, \tag{11}$$

where $|\phi_f|$ is the number of state fluents in ϕ_f. Equation 10 uses the Q-value for M and is an optimistic evaluation of $Q_{\theta}(s,a)$ following the motivation of optimistic Q-learning (Sutton and Barto 2018). However, this could introduce instability to an online learning algorithm. Since $\boldsymbol{\sigma}_M$ depends on the Q-values (or weights $\boldsymbol{\theta}$), it might change after $\boldsymbol{\theta}$ is updated with Equation 4. This not only invalidates the weight update but is also unstable as the TD error could increase rather than decrease after the weight update. Therefore, we consider the number of active and complex features, defined in Equation 11, for M.

Granularity of First-Order Approximation

A first-order approximation gives a coarser granularity than a ground approximation which can cause its performance to deteriorate. The lifted state-action space, $\widehat{\boldsymbol{\mathcal{S}}} \times \widehat{\boldsymbol{\mathcal{A}}}$, is often much smaller than the ground state-action space—a lifted fluent $\texttt{p}(?arg_1, ?arg_2)$ can be ground in $|arg_1||arg_2|$ ways. The maximum number of features is $\sum_{a \in \boldsymbol{\mathcal{A}}} |\boldsymbol{\Phi}_a| = 2^{2|\boldsymbol{\mathcal{P}}|} \times |\boldsymbol{\mathcal{A}}|$ in a ground approximation, and $\sum_{\hat{a} \in \widehat{\boldsymbol{\mathcal{A}}}} |\boldsymbol{\Phi}_{\hat{a}}| = 2^{2|\widehat{\boldsymbol{\mathcal{P}}}|} \times |\widehat{\boldsymbol{\mathcal{A}}}|$ in a first-order approximation. Since $|\widehat{\boldsymbol{\mathcal{P}}}| \leq |\boldsymbol{\mathcal{P}}|$ and $|\widehat{\boldsymbol{\mathcal{A}}}| \leq |\boldsymbol{\mathcal{A}}|$, a first-order approximation uses fewer features to partition the state space[5]. Therefore, the size of the partitions (or granularity) in a first-order approximation must be larger than a ground approximation. Furthermore, the use of OR increases the coverage of a first-order feature because there only needs to exist a grounding amongst several possible groundings to make the feature true. The coarse granularity limits the type of problems a first-order approximation can be applied in. It is suited for problems with factored transitions and reward functions and with independent goals. Since features are binary, it is not suited for problems which require real-valued features. This issue is mitigated in a ground approximation due to the inclusion of every ground state fluent as features which serves as implicit counting.

Example 6 (Inter-dependent goals)**.** If the goals in RC are to take pictures of objects in a particular order, then the first-order feature $\texttt{pictureTaken}(!obj)$ cannot represent the set of objects with pictures taken whereas the ground features (e.g., $\texttt{pictureTaken}(o1)$, $\texttt{pictureTaken}(o2)$, etc.) could. In other words, a first-order approximation partitions states in which at least one object has its picture taken into a region while a ground approximation partitions each of these states separately. Goal context is crucial in reducing the number of objects considered for grounding $!obj$ for a finer granularity. This drawback of first-order approximation is resolved if there is a non-fluent which specifies the dependence of goals (e.g., $\texttt{order}(?obj_1, ?obj_2)$). This is the case in the Academic Advising domain (Guerin et al. 2012) (refer to the *Experiments* section for details).

Example 7 (Counting features)**.** Following Example 6, a binary first-order feature $\texttt{pictureTaken}(!obj)$ cannot function as a counting feature (i.e., return the number of objects which has its picture taken). While a ground approximation also uses binary features, it performs an implicit counting—since every ground state fluent of $\texttt{pictureTaken}(?obj)$ are

[5]They are equal if there is one object for each type.

features, if n objects have their pictures taken, then n of these state fluents are true.

A coarse granularity in the function approximation can cause **plateaus** which are regions of the state space where there are multiple actions with equal maximal Q-values but not all of them are optimal. A random selection among these actions is resorted to as a tiebreaker. A straightforward remedy to plateaus is to include non-fluents as first-order features. They are unnecessary in a ground approximation since the value of a ground non-fluent does not change. However, a lifted non-fluent can be evaluated to either true or false depending on its grounding. These features are often crucial in determining which action of $\boldsymbol{\mathcal{A}}_{\hat{a}}$ to select.

Example 8 (Plateau)**.** Let the state s_1 be a situation in RC where the agent is at $pos2$ and needs to move to $pos3$ to take a picture of an object $o1$. There are no other objects. Then, $Q_\theta(s_1, \texttt{move}(pos3))$ should be larger than $Q_\theta(s_1, \texttt{move}(pos1))$. The non-fluent $\texttt{OBJECT\_AT}(o1, pos3)$ is required as a first-order feature $\texttt{OBJECT\_AT}(!obj, ?pos)$; otherwise, there would be a plateau as $Q_\theta(s_1, \texttt{move}(pos3)) = Q_\theta(s_1, \texttt{move}(pos1))$.

Example 8 illustrates the importance of first-order features with free variables. When combined with contextual grounding, these features give a finer granularity of the function approximation. The inclusion of non-fluents does not entirely resolve plateaus. If $o1$ is more than one grid position away from the agent, there is no first-order feature or ground feature which can represent this fact.

Mixed Approximation

By considering both first-order approximation and ground approximation, hereafter referred to as a mixed approximation, we can obtain the respective strengths of each representation (i.e, better generalisation and finer granularity). We propose two ways to utilise a mixed approximation:

$$Q_\theta^{mixed}(s,a) = Q_\theta^{gnd}(s,a) + Q_\theta^{fo}(s,a), \tag{12}$$

$$Q_\theta^{mixed}(s,a) = \begin{cases} Q_\theta^{fo}(s,a), & \text{if } episode \leq SW \\ Q_\theta^{gnd}(s,a), & \text{otherwise,} \end{cases} \tag{13}$$

where Q_θ^{gnd} $\left(Q_\theta^{fo}\right)$ is the ground (first-order) approximation, and SW is a parameter which sets the episode from which the policy switches from considering Q_θ^{fo} to Q_θ^{gnd}. By combining the Q-functions in Equation 12, we can resolve the plateaus due to a first-order approximation; while the Q-values from Q_θ^{fo} are equal for multiple actions, those from Q_θ^{gnd} are unlikely (because each ground action has its own weight components unlike in first-order approximation) and can serve as a tiebreaker. We denote the policies generated by Equation 12 as π_{sum} and by Equation 13 as π_{switch}. Online learning of Q_θ^{gnd} and Q_θ^{fo} are done concurrently and independently (i.e., weight updates and feature discovery).

Transfer Learning. Transfer learning is an attractive option to deal with large scale problems which require extensive amounts of exploration before reaching some meaningful states where rewards are observed (Wu and Givan 2010). We leverage on Q_θ^{fo} learned in a small scale problem to solve large scale problems. We focus our experiments on the following four modes of mixed approximation:

1. Learn Q_θ^{fo} from scratch, use π_{sum},
2. Transfer Q_θ^{fo} and keep it unchanged, use π_{switch},
3. Transfer Q_θ^{fo} and keep it unchanged, use π_{sum}, and
4. Transfer Q_θ^{fo} and update it online, use π_{sum}.

In all of these modes, Q_θ^{gnd} is learned online from scratch since transfer learning is not possible for ground approximation. (1) does not perform transfer learning and learns Q_θ^{fo} online from scratch. In (2), we transfer a learned Q_θ^{fo} and use it as an exploratory policy to acquire meaningful observations to train Q_θ^{gnd}, then switch to Q_θ^{gnd} when it is a sufficiently accurate enough approximation. (3) is similar to (2) but Q_θ^{fo} and Q_θ^{gnd} are considered by the policy in every episode. (4) differs from (2) and (3) in that Q_θ^{fo} continues to be updated in the current problem.

Experiments

Our experiments support two claims of our work: (1) our proposed first-order approximation is able to solve large scale problems comparable with the baseline, the ground approximation, and (2) a mixed approximation inherits the strengths of a first-order approximation and a ground approximation. We refer to the ground approximation, first-order approximation, and mixed approximation as Q_θ^{gnd}, Q_θ^{fo}, and Q_θ^{mixed}, respectively. Results are averaged over 10 independent runs and the (one) standard deviations are represented by shaded areas in the figures. $\boldsymbol{\theta}$, $\boldsymbol{\Phi}$, $\boldsymbol{\Phi}^c$, and relevances of $\boldsymbol{\Phi}^c$ are updated across episodes while no information is exchanged between runs. For transfer learning, the aforementioned information learned in small scale problems is transferred to large scale problems. We used ϵ-$greedy$ with a linearly decaying ϵ. The parameters used are $\epsilon = 1$ for experiments without transfer learning and 0.2 otherwise, $\alpha = 0.3, \gamma = 0.9, \lambda = 0.95$, and $\xi = 3$. H ranges from 30 to 50 depending on the scale of the problem. Experiments are conducted on an Intel Xeon E5-2660 v3 2.60 GHz with 8 cores and 32 GB of RAM.

Domains. We consider four domains: Recon (RC), Triangle Tireworld (TT) (Little and Thiebaux 2007), Academic Advising (AA) (Guerin et al. 2012), and Robot Inspection (RI). RC, TT, and AA are domains used in recent IPPCs (Sanner 2011; Grzes, Hoey, and Sanner 2014). We solve large scale problems but also require small scale problems for transfer learning. For RC and RI, a randomised problem is used in each run.

We use a modified version of RC: the actions `up`, `down`, `left`, and `right` are replaced with the action $\texttt{move}(?pos)$. This allows separate weights for each ground action of `move`.

We used RC3[6] and RC6 and the size of their state-action spaces are $2^{42} \times 28$ and $2^{55} \times 38$, respectively.

In TT, a vehicle moves in a grid environment to reach a goal location (i.e., only one goal). There is a probability of 0.5 of getting a flat tire when moving. The tire needs to be replaced with a spare tire; if there isn't one, a dead end is reached. The vehicle can load a spare tire if it doesn't have one and there is one at its current location. We used TT3 and TT6 and the size of their state-action spaces are $2^{33} \times 242$ and $2^{59} \times 814$, respectively.

In AA, a student has to pass some required courses. The passing rate of a course $course1$ depends on the number of prerequisites $course2$, defined by the non-fluent $\texttt{PREREQ}(course1, course2)$, the student has passed. A cost is given at each step for taking or retaking a failed course, and for each required course which the student has not passed. We used AA3 and AA5 and the size of their state-action spaces are $2^{30} \times 16$ and $2^{40} \times 21$, respectively.

RI is a newly introduced domain in this paper. A robot has to survey a location where objects are at before it can inspect them. A one-time reward of 20 per inspected object is given when the robot transmits at the communication tower (i.e., it can achieve multiple goals at once). When the robot inspects an object, there is a probability of its camera losing calibration (0.15) which reduces the probabilities of success for surveying and inspection from 1 to 0.2 and 0.9, respectively. The robot can return to the docking station and calibrate its camera. Unlike RC, the robot can move directly between any pair of locations in a step. We used RI-S (small scale) and RI-L (large scale) and the size of their state-action spaces are $2^{23} \times 27$ and $2^{29} \times 36$, respectively.

Free Variables and Contextual Grounding. Figure 1 shows the ablation results for Q_θ^{fo} with various configurations compared with Q_θ^{gnd}. First, we examine if features with free variables are necessary by eliminating them from $\mathbf{\Phi}$. For TT, these features are unnecessary; the performance without free variables are tied with those using location context or goal context. In RC6 and RI-L, the performances without free variables are the worst. This emphasises the importance of free variables in decreasing the granularity of first-order approximation.

Next, we look into the efficacy of contextual grounding. For RC6, the experiments for Q_θ^{fo} with only goal context are omitted due to high computational costs incurred in dealing with a large set of possible substitutions for the first-order features. In general, location context (if applicable) combined with goal context gives the best performance. Since TT does not require features with free variables, contextual grounding has minimal influence on performance though including them does not worsen performance. Q_θ^{fo} outperforms Q_θ^{gnd} in RI-L and RC-6 but fares worse in TT6 and AA5. This is because TT and AA have plateaus. In TT, if the vehicle moves along a shorter path, there are locations farther down the path where there are no spare tires and dead ends are unavoidable. Due to the increased abstraction in Q_θ^{fo}, there are no first-order features which represent the availability of spare tires in locations other than the vehicle's current location and adjacent locations.

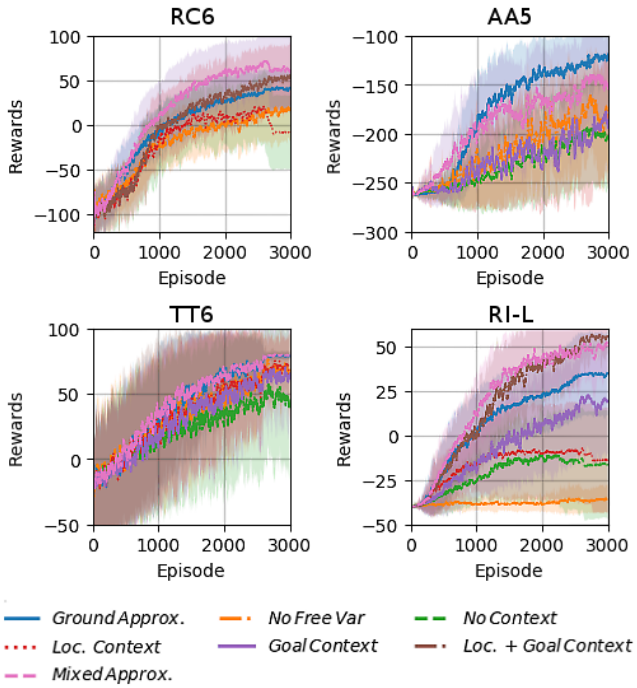

Figure 1: Results for an ablation study for first-order approximation without free variables (*No Free Var*), without contextual grounding (*No Context)*, and with different combinations of contextual grounding. Location context (*Loc. Context*) is not applicable in AA5. Goal context and location context cannot be used together in TT6. Mixed approximation combines the best performing configuration of first-order approximation for each domain with ground approximation. Performance is benchmarked against the ground approximation.

In AA, goals could be dependent on other goals—the success of passing a course depends on the number of prerequisites passed. This cannot be represented by binary features (see Example 7). Furthermore, the use of OR coupled with the failure of iFDD+ to learn certain conjunctive features caused a coarse granularity which worsens performance. The conjunctive feature $\phi_f = \texttt{PREREQ}(?course, !course) \wedge \texttt{PROGRAM\_REQUIREMENT}(!course) \wedge \neg\texttt{passed}(!course)$ represents the fact that (1) $?course$ is a prerequisite of $!course$, (2) $!course$ needs to be passed (i.e., a goal), and (3) $!course$ has not been passed. $!course$ is ground to the same $course$ object which means "$?course$ is a prerequisite of an unachieved goal". If ϕ_f has not been learned yet, $!course$ can be ground to different objects in each of the three constituent features of ϕ_f; these features represent "$?course$ is a prerequisite of at least one course, at least one course needs to be passed, and at least one course has not

[6]IPPC domains have numbered problem instances where a larger number typically indicates a larger-scale problem. We denote a problem instance # of a domain DOMAIN with the shorthand DOMAIN#.

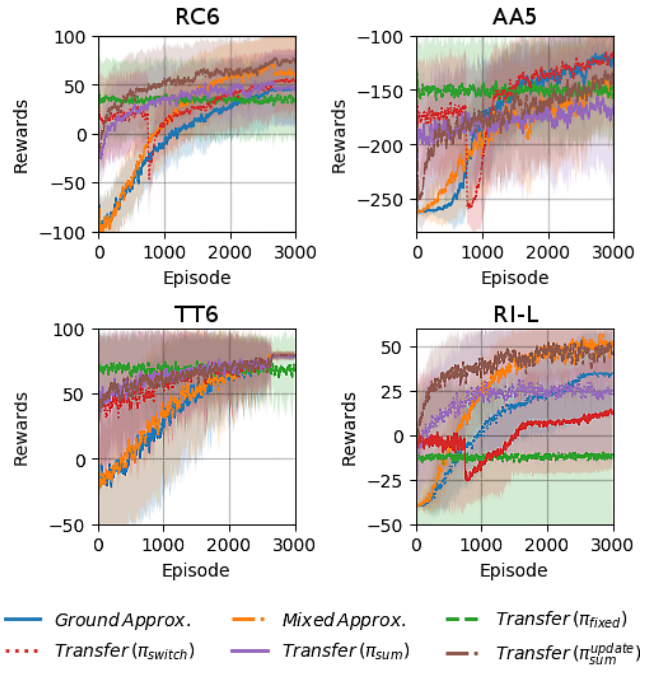

Figure 2: Results for transfer learning using first-order approximation where the Q-function learned in a small scale problem is used: (1) without further learning (π_{fixed}), (2) with π_{switch}, (3) with π_{sum}, (4) and with π_{sum}^{update} where the transferred Q-function continues to be updated in the large scale problems. For (2), (3), and (4), a ground approximation is learned from scratch. Results for ground approximation and mixed approximation without transfer learning are provided as baselines.

been passed" which is less useful in deciding which course to take.

Mixed Approximation. The performance for mixed approximation is shown in Figure 1. For each domain, we selected the best performing configuration of first-order approximation and combined it with the ground approximation. As discussed previously, there are plateaus in `AA`, `RC`, and `TT`. Q_θ^{mixed} utilises a ground approximation as a tiebreaker to resolve plateaus. In all three domains, this resulted in improved performance over Q_θ^{fo}. In particular, Q_θ^{mixed} has the best performance in `RC6` due to the combined strengths of generalisation and fine granularity. In `RI-L`, there are no plateaus and thus, Q_θ^{mixed} is comparable to Q_θ^{fo} but still outperforms Q_θ^{gnd}.

Generalisation and Transfer Learning. We used the first-order approximations learned in small scale problems (i.e., `RC3`, `AA3`, `TT3`, and `RI-S`), denoted as Q_θ^{small}, to solve large scale problems (i.e., `RC6`, `AA5`, `TT6`, and `RI-L`). The parameters and the number of episodes are the same in both small and large scale problems except that exploration (ϵ) is reduced in the latter. Crucially, contextual grounding used in the large scale problems must be the same as in the small scale problems; we used location context (if applicable) and goal context for all experiments. Figure 2 shows the results. We included the results for Q_θ^{gnd} and Q_θ^{mixed} from Figure 1 for ease of comparison.

We experimented with four modes of transfer learning. (1) We use a greedy policy generated from Q_θ^{small} which is kept unchanged (π_{fixed}). Since no learning takes place, the performance does not improve over episodes and are outperformed by Q_θ^{gnd} and Q_θ^{mixed} in later episodes. While a first-order approximation provides generalisation, some learning is still required to adapt to the new and unseen problems. This motivates the remaining three modes of transfer learning. We kept Q_θ^{small} unchanged, learned Q_θ^{gnd} online, and either (2) switch from an ϵ-$greedy$ policy induced by Q_θ^{small} to one induced by Q_θ^{gnd} (π_{switch}) at episode 750 (i.e., $SW = 750$), or (3) use an ϵ-$greedy$ policy induced by the sum of the Q-values from Q_θ^{small} and Q_θ^{gnd} (π_{sum}). In `RC6`, `AA5`, and `RI-L`, the rewards decreased rapidly at episode 750 which suggests that SW should be set larger. Nevertheless, (2) outperforms (1) asymptotically in all problems. To do away with finetuning SW, we use π_{sum} in (3) which avoided the sharp decrease in reward. In (4), we use the same setup as in (3) but allow Q_θ^{small} to be updated so that it can improve its generalisation and adapt to the current problem. In `RC6` and `RI-L`, (4) has the best performance. (2) has the best performance in `AA5` because first-order approximation does not perform well in `AA`; the switch to consider only Q_θ^{gnd} improved performance. In general, our mixed approximation for transfer learning outperforms the baselines, Q_θ^{gnd} and Q_θ^{mixed} without transfer learning, significantly initially, but have comparable asymptotic performance as expected.

Conclusions and Future Work

We represented a linear approximation of the action-value function with first-order features to achieve generalisation over different planning problems independent of the objects, number of objects, and initial and goal states. Free variables in first-order features are crucial in giving a finer granularity in the function approximation but caused ambiguity in how they should be ground. We proposed to ground them by considering contextual knowledge and introduced goal and location context. The strengths of ground and first-order approximations can be achieved by considering them together in a mixed approximation. Empirical results for four benchmark domains show that our method reduces the sample complexity and can generalise across different planning problems, enabling transfer learning. In future work, we plan to consider richer forms of first-order features such as universal variables or counting features. This may allow wider and more challenging problems to be solved.

Acknowledgements

This work was partially funded by the EPSRC ORCA Hub (http://orcahub.org/) under grant number EP/R026173/1.

References

Boutilier, C.; Dearden, R.; and Goldszmidt, M. 2000. Stochastic dynamic programming with factored representations. *Artificial intelligence* 121(1-2): 49–107.

Croonenborghs, T.; Ramon, J.; Blockeel, H.; and Bruynooghe, M. 2007. Online Learning and Exploiting Relational Models in Reinforcement Learning. In *Proc. IJCAI*, 726–731.

Driessens, K.; and Ramon, J. 2003. Relational instance based regression for relational reinforcement learning. In *Proc. ICML*, 123–130.

Džeroski, S.; De Raedt, L.; and Driessens, K. 2001. Relational reinforcement learning. *Machine Learning* 43(1-2): 7–52.

Geramifard, A.; Dann, C.; and How, J. P. 2013. Off-policy learning combined with automatic feature expansion for solving large MDPs. In *Proc. 1st Multidisciplinary Conference on Reinforcement Learning and Decision Making*, 29–33.

Geramifard, A.; Walsh, T.; Roy, N.; and How, J. 2013a. Batch iFDD: A Scalable Matching Pursuit Algorithm for Solving MDPs. In *Proc. UAI*.

Geramifard, A.; Walsh, T. J.; Tellex, S.; Chowdhary, G.; Roy, N.; and How, J. P. 2013b. *A Tutorial on Linear Function Approximators for Dynamic Programming and Reinforcement Learning*, volume 6. Now Foundations and Trends. doi:10.1561/2200000042.

Grzes, M.; Hoey, J.; and Sanner, S. 2014. International Probabilistic Planning Competition (IPPC) 2014. In *Proc. ICAPS*.

Guerin, J.; Hanna, J. P.; Ferland, L.; Mattei, N.; and Goldsmith, J. 2012. The Academic Advising Planning Domain. In *Proc. ICAPS Workshop on the International Planning Competition*.

Guestrin, C.; Koller, D.; Gearhart, C.; and Kanodia, N. 2003. Generalizing Plans to New Environments in Relational MDPs. In *Proc. IJCAI*, 1003–1010. Morgan Kaufmann Publishers Inc.

Little, I.; and Thiebaux, S. 2007. Probabilistic planning vs. replanning. In *Proc. ICAPS Workshop on the International Planning Competition: Past, Present and Future*.

Mausam, D. S. W. 2003. Solving Relational MDPs with First-Order Machine Learning. In *Proc. ICAPS Workshop on Planning Under Uncertainty And Incomplete Information*.

Morales, E. 2003. Scaling Up Reinforcement Learning with a Relational Representation. In *Proc. Workshop on Adaptability in Multi-agent System*.

Ramon, J.; Driessens, K.; and Croonenborghs, T. 2007. Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling. In *Proc. ECML*, volume 4701, 699–707. doi:10.1007/978-3-540-74958-5_70.

Sanner, S. 2006. Online Feature Discovery in Relational Reinforcement Learning. In *Proc. Open Problems in Statistical Relational Learning Workshop (SRL-06)*.

Sanner, S. 2010. Relational Dynamic Influence Diagram Language (RDDL): Language Description. Http://users.cecs.anu.edu.au/ ssanner/IPPC_2011/RDDL.pdf.

Sanner, S. 2011. ICAPS 2011 International Probabilistic Planning Competition (IPPC). URL http://users.cecs.anu.edu.au/~ssanner/IPPC_2011/.

Sanner, S.; and Boutilier, C. 2012. Practical Linear Value-Approximation Techniques for First-order MDPs. In *Proc. UAI*, 409–417.

Sutton, R. S.; and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Van Otterlo, M. 2004. Reinforcement Learning for Relational MDPs. In *Proc. Machine Learning Conference of Belgium and the Netherlands*.

Van Otterlo, M. 2005. A Survey of Reinforcement Learning in Relational Domains. Technical report, CTIT Technical Report Series.

Veeriah, V.; Oh, J.; and Singh, S. 2018. Many-Goals Reinforcement Learning. *arXiv preprint arXiv:1806.09605* .

Walker, T. 2004. Relational Reinforcement Learning via Sampling the Space of First-Order Conjunctive Features. In *Proc. ICML-04 Workshop on Relational RL*.

Wang, C.; Joshi, S.; and Khardon, R. 2008. First Order Decision Diagrams for Relational MDPs. *Journal of Artificial Intelligence Research* 31: 431–472.

Wu, J.; and Givan, R. 2005. Feature-Discovering Approximate Value Iteration Methods. In Zucker, J.; and Saitta, L., eds., *Abstraction, Reformulation and Approximation, Proceedings of the 6th International Symposium (SARA)*, volume 3607 of *Lecture Notes in Computer Science*, 321–331. doi:10.1007/11527862_25.

Wu, J.-H.; and Givan, R. 2007. Relational State-Space Feature Learning and Its Applications in Planning. In *AAAI Fall Symposium: Computational Approaches to Representation Change during Learning and Development*, 88.

Wu, J.-H.; and Givan, R. 2010. Automatic Induction of Bellman-Error Features for Probabilistic Planning. *JAIR* 38: 687–755. doi:10.1613/jair.3021.

Younes, H. L.; and Littman, M. L. 2004. PPDDL1.0: An extension to PDDL for expressing planning domains with probabilistic effects. Technical Report CMU-CS-04-162, Carnegie Mellon University.