Generalised Task Planning with First-Order Function Approximation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Real world robotics often operates in uncertain and dynamic environments. Therefore, generalisation over different scenarios is of practical interest. In the absence of a model, value-based reinforcement learning can be used to learn a goal-directed policy. Typically, the interaction between robots and the objects in the environment exhibit a first-order structure. We introduce first-order, or relational, features to represent an approximation of the Q-function so that it can induce a generalised policy. Empirical results for a service robot domain show that our online relational reinforcement learning method is scalable to large-scale problems and enables transfer learning between different problems and simulation environments with dissimilar transition dynamics.

Keywords: task planning, relational reinforcement learning, transfer learning

1 Introduction

Adaptation to different scenarios is essential for real world robotic applications. As a motivating example, consider a service robot interacting with people and objects as shown in Figure 1. The robot is required to attend to a person who requires assistance and complete the tasks given by the person. The uncertainty in who needs assistance, what these task could be, and the number of people and objects in the environment necessitates generalisation of learned knowledge from past observations in order to complete new but similar tasks in a different environment. If the model of the interaction between the robot and its environment is unknown due to its complexity, then value-based reinforcement learning (RL) methods can be used to solve the planning problems. The amount of observations required to achieve near-optimal behaviours, or sample complexity, is usually prohibitive, especially in large-scale problems. This restricts the practicality of RL methods in applications where data collection is expensive. A solution is to utilise transfer learning which can reduce sample complexity by leveraging the knowledge learned in small-scale problems where learning can be more efficient, or in simulated environments where data collection is cheap and safe.

Interactions between robots and objects in its environment often exhibit a first-order structure. We model planning problems with a first-order structure with Relational Markov Decision Processes (RMDPs). Relational reinforcement learning (RRL) learns in a first-order representation rather than a propositional representation and can achieve knowledge transfer without the need of a mapping between problems [1]. We propose a model-free, value-based RRL method where 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+` [2], to incrementally add conjunctive state predicates as features to reduce the approximation error. Since state predicates are ground over objects, problems with different objects or numbers of objects have different sets of state predicates, or feature spaces, and generalisation is not possible. Instead, we use lifted state predicates, or first-order features, to approximate the Q-values of lifted actions rather than ground actions. By learning in a abstract state-action space which is common across all problems of the same domain (**related problems**), knowledge transfer is possible.

Our contribution is threefold. First, we propose a method to generate a first-order feature space automatically given a RMDP without the use of a model, expert knowledge, or training data. Second,

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

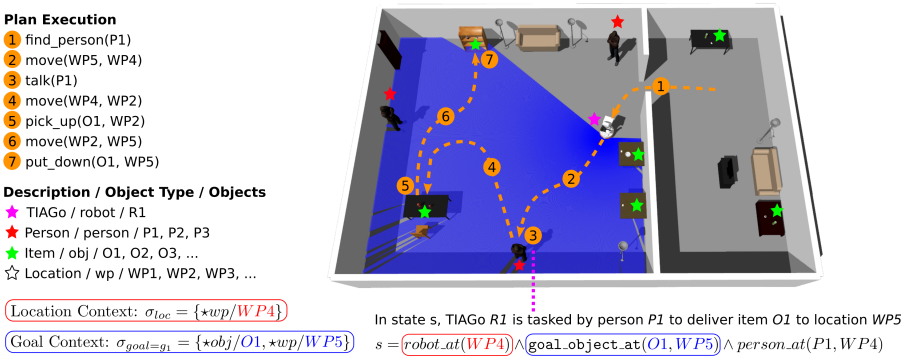

Figure 1: An environment in `Gazebo` for the `Service Robot` domain. A TIAGo fulfils a task from person $P1$ who requested assistance, by bringing an item $O1$ to a location $WP5$.

we implement a online, model-free RRL method which learns a first-order linear function approximation of the Q-function given an initial set of first-order features. The approximated Q-function induces a generalised policy which allows transfer learning between related problems independent of the objects, number of objects, initial states, and goal states. Third, we introduce the concept of contextual knowledge to reduce the granularity in the first-order function approximation, improve plan optimality, and reduce computational cost. We evaluate our method empirically on randomised problems of a service robot domain and in different simulation environments.

2 Related Work

Exploitation of the first-order structure of planning problems could reduce the sample complexity of RL methods. There are broadly two types of methods: model-based and model-free. Model-based methods such as [3, 4, 5, 6] learn relational models and employ planning techniques. Our work is a model-free RRL method. Therefore, we focus on reviewing model-free learning methods. Mausam and Weld [7] and Džeroski et al. [8] 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 [9] perform supervised learning of relational features, represented as decision trees, by adding features which correlate well to the Bellman error of value functions. A decision tree is sensitive to the order of node splitting and is ill-suited in online RL where later observations yield new information which might necessitate the reconstruction of the trees. Ramon et al. [10] propose a tree restructuring operation but require statistics to be stored for every node. Croonenborghs et al. [11] learn relational options using the relational regression trees from [8] which classify if a ground action is an action for the policy of a relational option. We use conjunctive features similar to `SVRRL` [12] which avoids the issue faced by decision trees. `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` [13], which uses instance-based learning where selected observed examples are stored, requires a domain-specific distance metric to compute Q-values of unseen state-action pairs. Our work uses state predicates as features and does not require any distance metric to be defined. In these aspects, [14] and [15] are most similar to our work. The former samples features from a set of features with the use of prior training data while we discover features online. The latter restricts features to have at most one free variable and approximates the value function while our features can have any number of free variables and we approximate the Q-function. Other works such as [8, 10, 11, 13] do not consider free variables. Instead, only objects in the arguments of the goal predicate and action are considered which restricts the applicability of their methods to problems with simple goals.

Some works require extensive domain knowledge to define specific representations to facilitate efficient learning. Morales [16] 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 abstract state-action space. Konidaris and Barto [17] learns a shaping reward in the action-space, represented by sensations which are common and hold the same semantics for all problems, to accelerate learning in another problem. In a later work [18], options in the agent-space are learned and transferred.

Algorithm 1: RRL with Online Feature Discovery `iFDD+`

Input: RMDP $(\boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \mathcal{R}, s_0, H, \gamma)$, Discovery threshold ξ

1 Initialise: $\boldsymbol{\Phi} \leftarrow$ initialise_features$(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{A}})$; $\boldsymbol{w} \leftarrow \mathbf{0}$; $t \leftarrow 0$; $\widetilde{Q} := \boldsymbol{w}^T \boldsymbol{\Phi}$

2 **for** $t = 1$ ***to*** H ***and*** s_t *is not terminal state* **do**

3 $\quad a_t \leftarrow \pi_{\widetilde{Q}}(s_t)$, $\hat{a}_t \leftarrow \text{lift}(a_t)$

4 $\quad s_{t+1}, r_t, \Delta_t \leftarrow$ Execute action a_t in simulation environment or real-world

5 $\quad$ Compute temporal difference (TD) error $\delta_t \leftarrow r_t + \gamma \max_{a \in \boldsymbol{\mathcal{A}}} \widetilde{Q}(s_{t+1}, a) - \widetilde{Q}(s_t, a_t)$; Update $\boldsymbol{w}$

6 $\quad \eta_t(\phi_i) \leftarrow \frac{\left|\sum_{j=0, \phi_i(s_j, \hat{a}_j)=1}^{t} \delta_j\right|}{\sqrt{\sum_{j=0, \phi_i(s_j, \hat{a}_j)=1}^{t} 1}} \forall \phi_i \in \mathbb{P}(\boldsymbol{\Phi}_a)$; **if** $\eta_t(\phi_i) > \xi$ **then** append ϕ_i to $\boldsymbol{\Phi}_a$ and 0 to $\boldsymbol{w}$

Van Otterlo [19] creates an abstract MDP of the underlying RMDP with the use of background knowledge, then solves the abstract MDP with modified Q-learning or prioritized sweeping. Sharma et al. [20] perform case-based learning and quantify the state similarity by the Euclidean distance between a pair of states which are represented by real-valued state variables. These state variables are handcoded and can describe any state in all problems for their domain of interest. The Q-values of an action is a weighted sum of the Q-values of matching cases, weighted by how similar these cases are to the current state. Our work uses trivial domain knowledge for contextual grounding which we posit are known in most, if not all, planning problems anyway.

3 Preliminaries

Relational Markov Decision Process (RMDP). Factored Markov Decision Processes (MDPs) [21] represent a state s with a set of state predicates $\boldsymbol{\mathcal{P}}$ where $s = \{p_i\}_{i=1}^{|\boldsymbol{\mathcal{P}}|}$ and $p_i \in \boldsymbol{\mathcal{P}}$. We use symbols with boldface to indicate sets. If the transition of p_i depends only on a small number of state predicates $\bar{\boldsymbol{\mathcal{P}}} \subset \boldsymbol{\mathcal{P}}$, then the transition function $\mathcal{T}$ can be represented compactly; for a predicate p_i, $\mathcal{T}(p_i' \,|\, \boldsymbol{\mathcal{P}}, a) = \mathcal{T}(p_i' \,|\, \bar{\boldsymbol{\mathcal{P}}}, a)$ where p_i' is the predicate at the next time step and a is an action. A Relational Markov Decision Process (RMDP) is a first-order representation of a factored MDP given by the tuple $(\boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{A}}, \mathcal{T}, \mathcal{R}, s_0, H, \gamma)$. $\boldsymbol{\mathcal{O}}$ is a set of objects, each associated with a type, $\boldsymbol{\mathcal{P}}$ is defined over $\boldsymbol{\mathcal{O}}$, $\boldsymbol{\mathcal{S}}$ 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, $\mathcal{T} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{S}} \rightarrow [0, 1]$, $\mathcal{R} : \boldsymbol{\mathcal{S}} \times \boldsymbol{\mathcal{A}} \rightarrow \mathbb{R}$ is the reward function, s_0 is the initial state, H is the planning horizon, and γ is the discount factor. Each predicate is applied over a type-consistent tuple of objects. Different planning problems of a domain can be constructed where their RMDPs have different $\boldsymbol{\mathcal{O}}$, initial states, and goal states. A **generalised policy** is a policy $\pi : s \rightarrow a$ which directly solves any problem of a domain.

Linear Function Approximation. The Q-function estimates the expected return, or Q-value $Q(s, a)$, of executing an action a in state s and following a policy thereafter. The Q-function can be represented with some form of parametric function approximation such as a linear combination of features: $\widetilde{Q} = \sum_{i=1}^{k} w_i \phi_i$ where $\boldsymbol{\Phi} = \{\phi_1, \ldots, \phi_k\}$ is a set of features and $\boldsymbol{w} = \{w_1, \ldots, w_k\}$ is a set of scalar weights. A feature ϕ_i maps (s, a) to a scalar. The learning objective is to find the set of features and weights such that $\widetilde{Q}$ closely approximates the optimal Q-function. A policy π induced by the optimal Q-function maximises the total discounted reward $\sum_{t=0}^{H} \gamma^t r_t$ where r_t is the reward observed at time step t. A feature 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 non-zero [22]. 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. Conversely, features with high coverage offer better generalisation as learning is done over a smaller number of partitions of the state space.

4 Online Learning with First-Order Linear Function Approximation

Our objective is to learn a generalised policy which can be used to directly solve every problem of the same domain. We present our online RRL method in Algorithm 1 which learns a linear function approximation of the Q-function, $\widetilde{Q}$, where the features $\boldsymbol{\Phi}$ are first-order (initialised in line 1). $\widetilde{Q}$ is

defined over the abstract state space (represented by the first-order features) and lifted action space which are the same in all related problems. This allows transfer learning where $\widetilde{Q}$ is learned in a problem and transferred directly to solve another problem. Thus, the policy $\pi_{\widetilde{Q}}$ induces by $\widetilde{Q}$ is a generalised policy. At each time step t (line 2), the robot executes an action a_t given by $\pi_{\widetilde{Q}}$ (lines 3 and 4) and observes the successor state s_{t+1}, reward r_t, and duration of execution Δ_t. The TD error is computed and the weights $\boldsymbol{w}$ are updated accordingly (line 5) with a TD learning method. We used Double Q-learning [23] with replacing eligibility traces for each feature [24]. Using `iFDD+` [2] for online feature discovery, new features, which are the conjunction of two existing features in $\boldsymbol{\Phi}$, are added to $\boldsymbol{\Phi}$ if their relevances η_t exceed the user-defined threshold ξ (see Section 4.2). This reduces the approximation errors of $\widetilde{Q}$ by introducing a finer granularity in $\widetilde{Q}$. In the remainder of this section, we elaborate on Algorithm 1.

4.1 Generalisation with First-Order Features

A feature ϕ_i is a set of conjunctive state predicates or a state predicate and evaluates to 1 if it is satisfied in a state, otherwise it evaluates to 0. We denote a function approximation which uses ground state predicates for features as a **ground approximation**. Let $\widetilde{Q}^1$ be a ground approximation for a RMDP with state-action space $\boldsymbol{\mathcal{S}}^1 \times \boldsymbol{\mathcal{A}}^1$ and ϕ_i^1 be one of its features. Evaluating the value of $\phi_i^1 = p_m^1 \wedge \ldots \wedge p_n^1 \in \boldsymbol{\mathcal{S}}^1$ is straightforward. However, in another RMDP (i.e., a different planning problem) with state-action space $\boldsymbol{\mathcal{S}}^2 \times \boldsymbol{\mathcal{A}}^2$, ϕ_i^1 cannot be evaluated if any of its constituent state predicates p_m^1, ..., p_n^1 is not in $\boldsymbol{\mathcal{S}}^2$. Extending this further, $\widetilde{Q}^1(s,a)$ does not exist if $s \in \{\boldsymbol{\mathcal{S}}^2 - \boldsymbol{\mathcal{S}}^1\}$ or $a \in \{\boldsymbol{\mathcal{A}}^2 - \boldsymbol{\mathcal{A}}^1\}$. In other words, $\boldsymbol{\mathcal{S}}^2 \times \boldsymbol{\mathcal{A}}^2$ does not lie within the feature space of $\widetilde{Q}^1$. Thus, $\widetilde{Q}^1$ cannot induce a policy for the second RMDP unless $\boldsymbol{\mathcal{S}}^1 = \boldsymbol{\mathcal{S}}^2$ and $\boldsymbol{\mathcal{A}}^1 = \boldsymbol{\mathcal{A}}^2$. However, the two RMDPs are trivially identical with possibly different initial states and goal states. We are interested in transfer learning between related problems with different state-action spaces. This motivates our use of a **first-order approximation** which is independent of $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{A}}$, and $\boldsymbol{\mathcal{O}}$.

In a ground approximation, the features for each ground action a, $\boldsymbol{\Phi}_a$, are initialised as the set of every state predicate and their negation; $\boldsymbol{\Phi} := [\boldsymbol{\Phi}_{a_1}, ..., \boldsymbol{\Phi}_{a_m}]$ where m is the number of ground actions. We use $\hat{}$ to denote a lifted predicate or a set of lifted predicates and lowercase (uppercase) letters to represent variables (objects). In a first-order approximation, we instantiate a set of first-order features for each lifted action $\hat{a}$ instead of every ground action. Otherwise, $\widetilde{Q}$ will not be independent of $\boldsymbol{\mathcal{O}}$ because ground actions are defined over $\boldsymbol{\mathcal{O}}$. $\hat{a}$ can be ground with the variable binding $\sigma_a = \{x/X, y/Y\}$ to a ground action $\mathtt{a}(X,Y)$. For every $\hat{a} \in \widehat{\boldsymbol{\mathcal{A}}}$, we initialise $\boldsymbol{\Phi}_{\hat{a}}$ (line 1 of Algorithm 1) with the following steps:

1. We denote the set of ground actions for $\hat{a}$ as $\boldsymbol{\mathcal{A}}_{\hat{a}}$. The initial set of ground features for every ground action $a \in \boldsymbol{\mathcal{A}}_{\hat{a}}$ are lifted in accordance with σ_a where objects in the arguments of a are substituted with their corresponding **bound variables**.
2. $\boldsymbol{\Phi}_{\hat{a}}$ is the union of the features from step 1. Substitute remaining objects with **free variables**. We use $\star obj$ to denote a free variable of type obj.

This procedure will generate the same set of first-order features for every related problem. Since $\widehat{\boldsymbol{\mathcal{A}}}$ is also the same, our first-order approximation has the same representation in all related problems. A first-order feature has to be fully grounded in order to evaluate its value in a state. In contrast with bound variables which are ground with objects in the arguments of actions, free variables can be ground to any object of the same type. We discuss the grounding of free variables in Section 4.3.

Service robot domain. We introduce a novel domain, `Service Robot` (`SR`), in this paper. A `Gazebo` environment for `SR` is illustrated in Figure 1. In `SR`, a TIAGo robot has to assist some people. The robot navigates from location pos_1 to pos_2 with the action $\mathtt{move}(pos_1, pos_2)$. A person can request assistance (represented by the state predicate $\mathtt{need_assistance}(person)$) which is a probabilistic exogenous event. The robot does not initially know where the people are located and has to find them with the action $\mathtt{find}(person)$. It then moves to the person and talks to them ($\mathtt{talk}(person)$). If the person has requested assistance, the robot would receive some tasks (or goals) of two possible types: (1) bring an item obj to location wp (represented by the state predicate $\mathtt{goal_object_at}(obj, wp)$) which requires the actions $\mathtt{pick_up}(obj, wp)$ and $\mathtt{put_down}(obj, wp)$,

and (2) deliver obj to $person$ ($\texttt{goal_object_with}(obj, person)$) which requires the action $\texttt{give}(obj, person)$. TIAGo can also take an item from a person ($\texttt{take}(obj, person)$).

Example 1. *In a ground approximation, we use the predicates* $\bar{\boldsymbol{\mathcal{P}}} = \{\texttt{need_assistance}(P_1), ..., \texttt{need_assistance}(P_m)\}$*, where* P_i *are objects of the type* $person$*, as features (i.e.,* $|\boldsymbol{\mathcal{P}}|$ *number of features). In a first-order approximation, these predicates are lifted to a first-order feature. For the lifted action* $\texttt{talk}(person)$*,* $\phi_i = \texttt{need_assistance}(person)$*. To evaluate* ϕ_i*, it has to be grounded. The bound variable* $person$ *is substituted according to the variable binding of* `talk` *(e.g., for* $\texttt{talk}(P3)$*,* $person$ *is substituted with* $P3$*). For* $\texttt{move}(pos_1, pos_2)$*, the first-order feature is* $\phi_j = \texttt{need_assistance}(\star person)$*; because* `move` *does not have an argument of type* $person$*,* P_i *is substituted with a free variable* $\star person$*.* ϕ_j *could be grounded by substituting* $\star person$ *to any object of type* $person$*. The number of first-order features does not increase with the number of* $person$ *objects unlike the ground approximation. This demonstrates that the first-order approximation does not scale with the number of objects and is independent of* $\boldsymbol{\mathcal{O}}$*.*

4.2 Online Feature Discovery

In Section 4.1, we describe the initialisation of $\boldsymbol{\Phi}$ where each feature is a lifted state predicate. There could be large approximation errors in $\widetilde{Q}$ as each of these feature has a high coverage. Hence, adding conjunctive state predicates, which have lower coverage, to $\boldsymbol{\Phi}$ could reduce the approximation error. The problem of feature discovery is to determine which features to add. We utilise `iFDD+`, a model-free, online feature discovery algorithm (line 6 of Algorithm 1). A set of candidate features $\boldsymbol{\Phi}_a^c$ consists of features which are the conjunction of any two features (its **parent features**) in $\boldsymbol{\Phi}_a$. We denote the generation of $\boldsymbol{\Phi}_a^c$ from $\boldsymbol{\Phi}_a$ as $\mathbb{P}(\boldsymbol{\Phi}_a)$. `iFDD+` iteratively adds candidate features to $\boldsymbol{\Phi}_a$ at each time step if their cumulative approximation errors, or relevances, exceed the discovery threshold ξ. The relevance of a candidate feature is updated at each time step by accumulating the TD error only if the candidate feature is active. A feature is active if it is true in the state and is not a parent feature of any active feature. The weight of a newly added feature is equals to the sum of the weights of its parents. When a feature ϕ_i is added to $\boldsymbol{\Phi}_a$, this in turn generates new candidate features with ϕ_i as one of its parent features.

4.3 Contextual Grounding of Free Variables

To evaluate the values of first-order features in a state, they must be fully ground. This is required in Algorithm 1 to determine the Q-values (lines 3 and 5) and the active features and candidate features (line 6). The grounding influences the step update of $\boldsymbol{w}$ (line 5) and feature discovery (line 6). Bound variables are substituted with objects of the same type in the arguments of a ground action (see Example 1). Features with at least one free variable are not fully ground. The set of possible substitutions for these features consists of every combination of objects for the free variables. The number of substitutions for a first-order feature ϕ_i and for $\boldsymbol{\Phi}_{\hat{a}}$ are:

$$|\boldsymbol{\sigma}_{\phi_i}| = \prod_{v \in \boldsymbol{v}} |v|, \text{ and } |\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}| = \prod_{\phi_i \in \boldsymbol{\Phi}_{\hat{a}}} |\boldsymbol{\sigma}_{\phi_i}|, \tag{1}$$

respectively, where $\boldsymbol{\sigma}_{\phi_i}$ ($\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}$) is the set of possible substitutions for ϕ_i ($\boldsymbol{\Phi}_{\hat{a}}$), $\boldsymbol{v}$ is the set of free variables in ϕ_i and $|v|$ is the number of objects of the same type as v. If we consider every substitution (i.e., to treat a free variable like an existential quantifier) where ϕ_i is true in s if there exists a substitution in $\boldsymbol{\sigma}_{\phi_i}$ which makes it true, then there would be many such states (e.g., in Example 1, $\texttt{need_assistance}(\star person)$ will be true if any person needs assistance). Thus, the first-order approximation has a coarse granularity which can deteriorate performance. We propose two methods to resolve this. First, we use **contextual knowledge** to reduce the number of objects which a free variable can be grounded to (i.e., reduce $|v|$ in Equation 1). Any substitution which conflicts with the substitution due to contextual knowledge is removed from $\boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}$. Two substitutions conflict if they substitute a free variable with a different object. We introduce two forms of contextual knowledge which require trivial domain knowledge: goal and location. Second, after contextual knowledge is applied, we select $\boldsymbol{\sigma}_M \subseteq \boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}$ such that a metric M is maximised: $\boldsymbol{\sigma}_M = \{\arg\max_{\sigma \in \boldsymbol{\sigma}_{\boldsymbol{\Phi}_{\hat{a}}}} M\big(\boldsymbol{\Phi}_{\hat{a}}^{\sigma}\big)\}$ where $\boldsymbol{\Phi}_{\hat{a}}^{\sigma}$ is the grounding of $\boldsymbol{\Phi}_{\hat{a}}$ with σ. We define $M\big(\boldsymbol{\Phi}_{\hat{a}}^{\sigma}\big) = \sum_{\phi_i \in \boldsymbol{\Phi}_{\hat{a}}^{\sigma}} \phi_i(s,a)(|\phi_i|)^2$ where $|\phi_i|$ is the number of state predicates in ϕ_i. This maximises the number of active, complex features. The choice of M must be dependent on the state only to preserve the Markovian property of RMDPs.

Goal Context. The goals $\mathcal{G}$ in a planning problem are state predicates 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. The objects in $g \in \mathcal{G}$ are used for grounding free variables of the same type: $\sigma_{goal=g} = \{\star x/X, \ldots, \star z/Z\}$ where $g = \mathtt{p}(X, \ldots, Z)$. We consider only active goals; a goal is active if it is known and has not been achieved (e.g., in SR, a goal is known only after a person has given a task). We assume that achieved goals are inconsequential for decision-making in the present and future. The Q-values after applying goal context represent the expected values of actions for achieving the goal. This is similar to the goal-associated Q-function in [25] but is not the same as additive rewards as defined in [26] where each goal is assumed to contribute uniformly and additively to the reward. $\phi_i = 1$ if there exists a g such that $\sigma_{goal=g}$ results in ϕ_i being true; the value does not change even if more of such goals exist.

A goal typically gives a contextual grounding which is in conflict with another goal. We propose two goal selection schemes: (1) GA uses every goal with the maximal metric M, and (2) G1 uses only one goal at a time. G1 selects the next goal in an ordering of active goals, $\bar{\mathcal{G}}$, when the currently selected goal is achieved. The order in $\bar{\mathcal{G}}$ could affect the optimality and soundness of the policy. In domains where there is a sequence to achieving goals, the choice of $\bar{\mathcal{G}}$ is crucial. We determine $\bar{\mathcal{G}}$ from the state trajectories observed in the previous episode. A partial order $g_i < g_j$ is added if g_i is achieved before g_j. $\bar{\mathcal{G}}$ is randomly shuffled at the start of an episode such that every partial order is satisfied. If every goal is achieved, then the total order is known though it might not be optimal. However, due to exploration in RL, the order of goals achieved might not be the same as $\bar{\mathcal{G}}$. Thus, a more optimal $\bar{\mathcal{G}}$ could still be found.

Example 2. *In SR, the goals are represented by the state predicates* $\mathtt{goal_object_with}(obj, person)$ *and* $\mathtt{goal_object_at}(obj, wp)$. *Suppose that the goals are to deliver item $O1$ to location $WP5$ and item $O2$ to person $P1$. The former is denoted as $g_1 = \mathtt{goal_object_at}(O1, WP5)$ and the latter as $g_2 = \mathtt{goal_object_with}(O2, P1)$. The goal g_1 involves the objects $O1$ with type obj and $WP5$ with type wp, and the corresponding contextual grounding is $\sigma_{goal=g_1} = \{\star obj/O1, \star wp/WP5\}$. Likewise, $\sigma_{goal=g_2} = \{\star obj/O2, \star person/P1\}$.*

Location Context. Mobile robots move in an environment and can only interact with objects in their vicinity. Following this observation, location context substitutes the free variable with the current location of the robot: $\sigma_{loc} = \{\star wp/WP\}$ if the robot is at location WP of type wp. Location context can be used with the goal context if there is no conflict (i.e., they substitute free variables of different types)—this queries if the robot is at the same location as the goal of interest. If there is a conflict, then location context takes precedence.

Example 3. *In SR, the state predicate* $\mathtt{robot_at}(wp)$ *represents the location of a robot. If the robot is at $WP4$, then the contextual grounding is $\sigma_{loc} = \{\star wp/WP4\}$. Following Example 2, we can apply $\sigma_{goal=g_2}$ and σ_{loc} together without any conflict to give the contextual grounding $\sigma_{loc,goal=g_2} = \{\star wp/WP4, \star obj/O2, \star person/P1\}$. On the other hand, $\sigma_{goal=g_1}$ and σ_{loc} are conflicting as they ground $\star wp$ to $WP5$ and to $WP4$, respectively. They can still be applied together by letting either one takes precedence (i.e., ignore the grounding of the other one). If location context takes precedence, then $\sigma_{loc,goal=g_1} = \{\star obj/O1, \star wp/WP4\}$.*

4.4 Granularity of First-Order Approximations

The lifted state-action space, $\widehat{\mathcal{S}} \times \widehat{\mathcal{A}}$, is often much smaller than the ground state-action space—a lifted predicate $\mathtt{p}(x, y)$ can be ground in $|x||y|$ ways. Since $\widehat{\mathcal{S}}$, $\widehat{\mathcal{P}}$, and $\widehat{\mathcal{A}}$ are independent of $\mathcal{O}$, the lifted state-action space does not increase with the scale of the planning problem which makes first-order approximations scalable to large-scale problems (see Example 1). However, the downside is a coarser granularity in approximation. The maximum number of features is $\sum_{a \in \mathcal{A}} |\Phi(a)| = 2^{2|\mathcal{P}|} \times |\mathcal{A}|$ in a ground approximation, and $\sum_{\hat{a} \in \widehat{\mathcal{A}}} |\Phi_{\hat{a}}| = 2^{2|\widehat{\mathcal{P}}|} \times |\widehat{\mathcal{A}}|$ in a first-order approximation. Since $|\widehat{\mathcal{P}}| \leq |\mathcal{P}|$ and $|\widehat{\mathcal{A}}| \leq |\mathcal{A}|$ (they are equal if there is one object for each type), a first-order approximation uses fewer features to partition the state space. Therefore, the size of the partitions (or granularity) must be larger than a ground approximation. Furthermore, free variables in a first-order feature increase its coverage because there only needs to exist a substitution amongst several possible substitutions to make the feature true. This deteriorates performance if there are plateaus in the policy induced. Plateaus are regions of the state space where there are multiple non-optimal actions with the maximal Q-value. The coarse granularity limits the type of problems a first-order

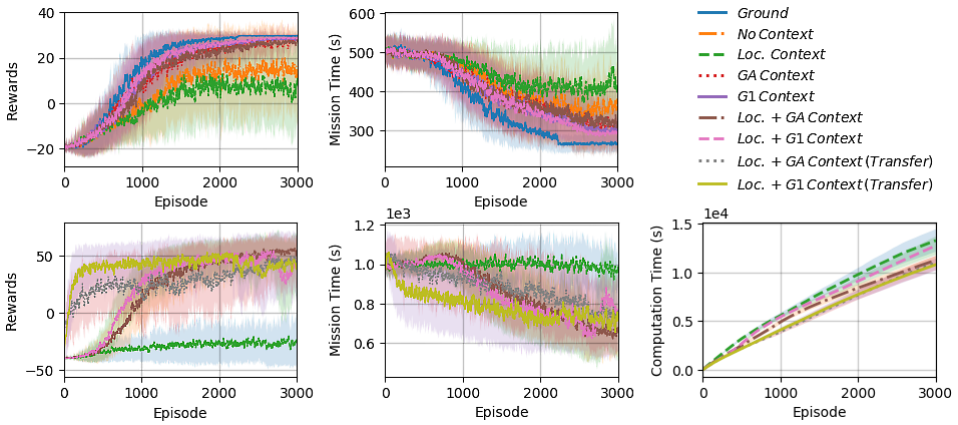

Figure 2: Results for P_{small} (top row) and P_{large} (bottom row) using `RDDLSim` as a simulator. The performances of first-order approximation with different combinations of contextual grounding are compared with ground approximation ($Ground$). Results for transfer learning ($Transfer$) from P_{small} to P_{large} are included as well. The shading represents one standard deviation.

approximation can be applied in. Because first-order approximations do not consider every ground state predicates, it is suited for planning in factored MDPs with independent goals.

5 Experiments and Results

We demonstrate empirically that our work (1) learns a generalised and goal-directed policy, (2) enables transfer learning from small-scale problems (P_{small}) to large-scale problems (P_{large}) and between different simulation environments, and (3) reduces memory and computational costs. Results are averaged over 10 independent runs where a different, randomised problem is used. We used ϵ-$greedy$ policy with a linearly decaying ϵ over episodes. The parameters used are $\epsilon_{initial} = 1$ for experiments without transfer learning and 0.2 otherwise, $\alpha = 0.3$ (learning rate), $\gamma = 0.9$, $\lambda = 0.7$ (decay rate for eligibility traces), and $\xi = 3$. H is 20 (40) for P_{small} (P_{large}). We assume that the preconditions of actions are known (i.e., an illegal action will never be executed).

Experimental Setup. In P_{small} (P_{large}), $\mathcal{O}$ contains three (six) items, one person (three people), and five (ten) locations and $|\mathcal{S}| \times |\mathcal{A}| = 2^{56} \times 65$ ($2^{216} \times 264$); the probability of people needing assistance is 0.5 (0.5, 0.3, and 0.0 for three people). Each person gives two tasks. The problems are randomised in the locations of items and in the tasks given. Executing an action gives a reward of -1 and completing a goal gives 20. We tested our work in two simulation environments: `RDDLSim` [27] and `ROS` (Robot Operating System) with the `ROSPlan` [28, 29] package. Both simulators return an observation for executing an action in a state (line 4 of Algorithm 1). We augment the observation from `RDDLSim` with Δ, approximated as the mean of the observed Δ in `ROS` injected with normally distributed noise (20% of a standard deviation). We built a `Gazebo` environment (see Figure 1) for a more realistic simulation relative to `RDDLSim`. Firstly, in `RDDLSim`, the robot loses localisation when moving with a probability of 0.1; in `ROS`, it loses localisation when the covariance of the pose estimation from adaptive Monte-Carlo localization exceeds a threshold. Secondly, in `RDDLSim`, the robot always succeeds in finding a person and ends up in the same location as the person. In `ROS`, the robot follows an exploration path to find a person. It might find the intended person and/or other people it encountered along the way, and might not be in the same location as the person since it could detect people from a distance. Lastly, in `RDDLSim`, picking up and putting down an item always succeeds while these actions might fail in `ROS`. Each experiment in `RDDLSim` is conducted on a single core of AMD Opteron 6376 and with at most 4 GB of RAM allowed.

Ablation Study. We conducted an ablation study for contextual grounding in first-order approximation and benchmark them against the ground approximation. `RDDLSim` was used. The results are shown in Figure 2. We measure performance by the total undiscounted reward received in each episode, and by the total duration of plan execution (mission time). Executing an action incurs a re-

ward of -1 which could result in a learned policy which minimises the number of actions executed; this seems to suffice in reducing the mission time as it was observed that the mission time reduces asymptotically while the reward increases. `G1` and `Loc.+G1` (combination of location context and `G1`) generally outperforms `GA` and `Loc.+GA`. This is because the goals in `SR` are independent of each other and considering one goal at a time reduces $\boldsymbol{\sigma}_{\Phi_{\hat{a}}}$ and coarseness of granularity more than `GA` does. In P_{small}, using no context or only location context has the worst performances. Due to our modelling choice to include wp in arguments of actions (e.g., $\texttt{pick_up}(obj, wp)$), features with $\star wp$ are unnecessary to approximate the Q-values of some actions—we opt against using domain knowledge to prune these features. Nevertheless, location context is crucial in reducing the memory and computational cost; experiments for P_{large} cannot be completed without location context as it takes too long. The computation time for P_{large} is further reduced when goal context is also used. The computation time for P_{small} are comparable for all runs and are thus omitted. Although ground approximation has the best performance in P_{small}, it could not be used in P_{large} as the memory usage exceeds the 4 GB RAM limit. This highlights the poor scalability of ground approximation—for P_{small}, the ground approximation has 6894 ± 2.7 (one standard deviation) features while the various configurations of first-order approximation have 280 to 316 features.

Transfer Learning. In Figure 2, we show the results of transfer learning where $\widetilde{Q}$ trained in P_{small} ($\widetilde{Q}^{P_{small}}$) is used, and continues to be updated, in P_{large}. The contextual grounding used in P_{small} must also be used in P_{large} for transfer learning to work. We consider only `Loc.+G1` or `Loc.+GA` as these are computationally tractable. The former outperforms the latter for the aforementioned reasons. The boost in performance due to transfer learning is evident in episode 1. The asymptotic performance without transfer learning approaches that of transfer learning as expected. However, for `Loc.+GA`, transfer learning performed asymptotically poorer than without transfer learning. This is because `GA` considers multiple goals for contextual grounding. Thus, it generalises poorly from P_{small}, which has two goals, to P_{large}, which has four goals.

Lastly, we evaluate the generalisation from `RDDLSim` simulations to `ROS` simulations. We used ten randomised problems for P_{small} and P_{large} which are different from the ones used in `RDDLSim`. A greedy policy induced by $\widetilde{Q}^{P_{small}}$ using `Loc.+G1` can directly solve P_{small} but not P_{large}. Therefore, we trained $\widetilde{Q}^{P_{small}}$ further in 500 episodes of P_{large} using `RDDLSim` with a modified transition function (i.e., the robot's location does not change when finding a person) to solve P_{large}. Without further training, the policy could fail to instruct the robot to move to a person after finding that person. The results are shown in Table 1 alongside the results from `RDDLSim` (i.e., Figure 2) for ease of comparison. For P_{small}, the performance in `RDDLSim` is better with smaller variances which is expected since the simulations do not consider real-world dynamics and are more optimistic. Surprisingly, the results for P_{large} are comparable for both simulation environments. Thus, we have demonstrated that first-order approximations can generalise over different simulation environments.

Table 1: Results for transfer learning from `RDDLSim` to `ROS` simulations aggregated over 100 runs. This is compared with results from `RDDLSim` (`Loc.+G1` for P_{small} and `Loc.+G1 (Transfer)` for P_{large}).

	Reward	**Mission Time (s)**
P_{small} (`ROS`)	22.7 ± 10.6	303.7 ± 86.7
P_{small} (`RDDLSim`)	28.6 ± 0.9	289.8 ± 30.3
P_{large} (`ROS`)	45.2 ± 21.0	607.3 ± 116.4
P_{large} (`RDDLSim`)	46.0 ± 24.7	648.6 ± 122.3

6 Conclusion

We propose a model-free, online RRL algorithm which learns an approximation of the Q-function with first-order features that generalises across different problems of the same domain. Free variables in the first-order features caused ambiguity in how they should be substituted. The choice of a substitution influences the learning and is critical to performance. We resolve this problem by utilising contextual knowledge to substitute free variables. Empirical results showed that our method is scalable to large-scale problems by keeping memory usage and computational cost tractable, and can generalise across problems of different scales and across different simulation environments. In future work, we plan to investigate other forms of contextual knowledge and to consider temporal information such as duration or time constraints.

References

[1] M. E. Taylor and P. Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

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

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

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

[5] T. Lang, M. Toussaint, and K. Kersting. Exploration in relational domains for model-based reinforcement learning. *JMLR*, 13:3725–3768, 2012.

[6] D. Martínez, G. Alenya, T. Ribeiro, and C. Torras. Relational reinforcement learning for planning with exogenous effects. *JMLR*, 18(1):2689–2732, 2017.

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

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

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

[10] J. Ramon, K. Driessens, and T. Croonenborghs. Transfer Learning in Reinforcement Learning Problems Through Partial Policy Recycling. In *Proc. ECML*, volume 4701, pages 699–707, 09 2007. doi:10.1007/978-3-540-74958-5_70.

[11] T. Croonenborghs, K. Driessens, and M. Bruynooghe. Learning relational options for inductive transfer in relational reinforcement learning. In *Proc. International Conference on Inductive Logic Programming*, pages 88–97, 2007.

[12] S. Sanner. Online feature discovery in relational reinforcement learning. In *Proc. Open Problems in Statistical Relational Learning Workshop (SRL-06)*, 2006.

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

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

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

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

[17] G. Konidaris and A. Barto. Autonomous shaping: Knowledge transfer in reinforcement learning. In *Proc. ICML*, pages 489–496, 2006.

[18] G. D. Konidaris and A. G. Barto. Building portable options: Skill transfer in reinforcement learning. In *Proc. IJCAI*, volume 7, pages 895–900, 2007.

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

[20] M. Sharma, M. Holmes, J. Santamaría, A. Irani, C. Jr, and A. Ram. Transfer learning in real-time strategy games using hybrid cbr/rl. In *Proc. IJCAI*, pages 1041–1046, 01 2007.

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

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

[23] H. v. Hasselt, A. Guez, and D. Silver. Deep Reinforcement Learning with Double Q-Learning. In *Proc. AAAI*, page 2094–2100, 2016.

[24] S. Singh, R. Sutton, and P. Kaelbling. Reinforcement Learning with Replacing Eligibility Traces. *Machine Learning*, 22, 11 1995. doi:10.1023/A:1018012322525.

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

[26] S. Sanner and C. Boutilier. Practical Linear Value-Approximation Techniques for First-order MDPs. In *Proc. UAI*, page 409–417, 06 2012.

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

[28] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, and M. Carreras. ROSPlan: Planning in the robot operating system. In *Proc. ICAPS*, 2015.

[29] G. Canal, M. Cashmore, S. Krivic, G. Alenyà, D. Magazzeni, and C. Torras. Probabilistic Planning for Robotics with ROSPlan. In *Proc. TAROS*, pages 236–250. Springer, 7 2019.