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 planning problems 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 propose to use first-order features to approximate the Q-function that generates a generalised policy. Empirical results for a service robot domain show that our method of online relational reinforcement learning is scalable to large-scale problems and enables transfer learning across 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 humans and objects as shown in Figure 1. The robot is required to attend to persons requiring assistance and complete any given task (e.g., delivering some items). The uncertainty in who needs assistance, what these task could be, and the number of persons and objects in the environment necessitates generalisation of learned knowledge from past observations in order to achieve a task in the new 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 planning problems. This restricts the practicality of RL methods. 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. Alternatively, an RL agent can be trained in a simulated environment then transferred to the real world—this is preferred as it is often costly and unsafe to train a robot in the real world.

We focus on online RL for problems represented with Relational Markov Decision Processes (RMDPs) which is a natural way to model interactions between robots and objects in its environment. 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+` [1], to incrementally construct sets of conjunctive state predicates as features to reduce the approximation errors. Since state predicates are ground over objects, problems with different objects or numbers of objects are represented by different sets of state predicates. Therefore, generalisation is not possible. Instead, we use conjunctive lifted state predicates, or first-order features, to approximate the Q-values of lifted actions rather than ground actions. The feature space for this first-order approximation is identical for all ground MDPs of a RMDP. Thus, generalisation and knowledge transfer is possible.

The paper makes two main contributions: (1) to represent the function approximation with first-order features which allows generalisation and transfer learning between different planning problems independent of the objects, number of objects, initial states, and goal states; and (2) to utilise contextual

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

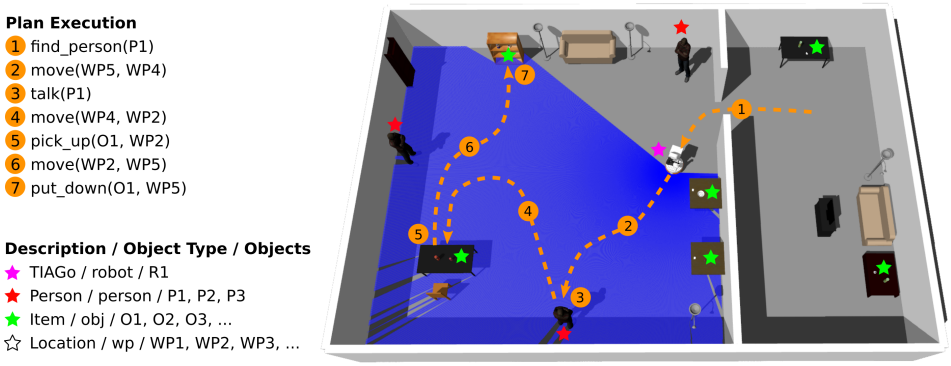

Figure 1: An environment in `Gazebo` for the `Service Robot` domain. A TIAGo fulfils a task from person $P1$ who requested assistance, by bringing an item $O1$ to a location $WP5$.

knowledge to introduce finer granularity in first-order approximations which result in lower approximation errors. We assume that the models are unknown except for the preconditions of actions (i.e., an illegal action will never be selected). Our objective 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 and present the necessary technical background. Next, we describe our method in detail. We then present an ablation study of our method on a service robot domain and demonstrate transfer learning between different problems and simulation environments. Lastly, we conclude and discuss future work.

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 [2, 3, 4] learn relational models and employ planning techniques with these models. Our work is a model-free relational RL method which performs online feature discovery using `iFDD+` [1]. Therefore, we focus on reviewing model-free methods.

Mausam [5] and Džeroski et al. [6] 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 [7] 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, rendering this work ill-suited in online RL where later observations yield new information which might necessitate the reconstruction of the trees. Ramon et al. [8] propose a tree restructuring operation but require statistics to be stored for every node. We use conjunctive features similar to `SVRRL` [9] 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` [10], 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 predicates as features and does not require any distance metric to be defined.

In these aspects, [11] and [12] 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. Morales [13] 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. Some domain knowledge is required to define the r-states. Guestrin et al. [14] 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 with supervised learning. Van Otterlo [15] creates an abstract MDP of the underlying RMDP with the use of background knowledge then solves the abstract MDP

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 $\boldsymbol{w}$

with modified Q-learning and prioritized sweeping. 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) [16] 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, H is the planning horizon, and γ is the discount factor. Each predicate is applied over a type-consistent tuple of objects. Multiple ground MDPs, each representing a planning problem, can be constructed from a RMDP with a different $\boldsymbol{\mathcal{O}}$. A **generalised policy** for a RMDP is a policy $\pi : s \rightarrow a$ which directly solves any ground MDP of the RMDP.

Linear Value Functions. The Q-function $Q(s, a)$ represents the expected value, or Q-value, of executing an action a in state s. It 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 $\phi(s, a)$ 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 1 or non-zero for non-binary features [17]. 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 for a RMDP which can be used to directly solve any of its ground MDPs. We present a relational reinforcement learning method as shown in Algorithm 1. We learn a first-order linear function approximation of the Q-function, $\widetilde{Q}$, which induces a generalised policy $\pi_{\widetilde{Q}}$. 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 [18] with replacing eligibility traces for each feature [19]. Using `iFDD+` [1] for online feature discovery, new features are added to $\boldsymbol{\Phi}$ if their relevances exceed the user-defined threshold ξ (see Section 4.2). In the remainder of this section, we elaborate on Algorithm 1.

4.1 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. Let $\widetilde{Q}^1$ be an approximation for a ground MDP with state-action space $\boldsymbol{\mathcal{S}}^1 \times \boldsymbol{\mathcal{A}}^1$ and ϕ_i^1 is 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 ground MDP (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 predicate 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, the feature space of $\widetilde{Q}^1$ does not lie within the state-action space of the second MDP. Thus, $\widetilde{Q}^1$ cannot induce a policy $\pi_{\widetilde{Q}^1}$ which solves the second MDP unless $\boldsymbol{\mathcal{S}}^1 = \boldsymbol{\mathcal{S}}^2$ and $\boldsymbol{\mathcal{A}}^1 = \boldsymbol{\mathcal{A}}^2$. However, the two MDPs are trivially identical with possibly different initial states and goal states. Thus, $\pi_{\widetilde{Q}^1}$ cannot be considered as a generalised policy. This motivates our use of a **first-order approximation** independent of $\boldsymbol{\mathcal{S}}$, $\boldsymbol{\mathcal{A}}$, and $\boldsymbol{\mathcal{O}}$ (because $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{A}}$ are defined over $\boldsymbol{\mathcal{O}}$).

We denote a function approximation which uses ground state predicates for features as a **ground approximation**, and one which uses lifted state predicates for features as a first-order approximation. 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 (or lifted actions if using first-order approximation). In a first-order approximation, we instantiate a set of first-order features for each lifted action $\hat{a}$ instead of every ground action[1], 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 $\mathtt{a}(X, Y)$[2]. 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.

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 this domain, 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 ($person$) 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 ($\mathtt{find}(person)$). 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 of two possible types: (1) bring an item obj to location wp ($\mathtt{goal_object_at}(obj, wp)$) which requires the actions $\mathtt{pick_up}(obj, wp)$ and $\mathtt{put_down}(obj, wp)$, and (2) deliver obj to $person$ ($\mathtt{goal_object_with}(obj, person)$) which requires the action $\mathtt{give}(obj, person)$. TIAGo can also take an item from a person ($\mathtt{take}(obj, person)$).

Example 1. *In a ground approximation, we use the predicates* $\bar{\boldsymbol{\mathcal{P}}} = \{\mathtt{need_assistance}(P_1)$, ..., $\mathtt{need_assistance}(P_m)\}$*, where* P_i *are objects of the type* $person$*, as features (i.e.,* $\bar{\boldsymbol{\mathcal{P}}} \subset \boldsymbol{\Phi}_a \forall a \in \boldsymbol{\mathcal{A}}$*). In a first-order approximation, these predicates could be lifted to different first-order features for each lifted action:* $\phi_i^{talk} = \mathtt{need_assistance}(person)$ *for* $\mathtt{talk}(person)$ *and* $\phi_i^{move} = \mathtt{need_assistance}(\star person)$ *for* $\mathtt{move}(pos_1, pos_2)$*. The number of first-order features does not increase with the number of* $person$ *objects; contrast this with ground approximation where* $|\bar{\boldsymbol{\mathcal{P}}}| = m$ *number of* $person$ *objects. This demonstrates the scalability of first-order approximation and its independence from* $\boldsymbol{\mathcal{O}}$*. Because* `move` *does not have a variable of type* $person$*, the object is substituted with a free variable* $\star person$ *in* ϕ_i^{move} *and can be ground to* P_1*, ...,* P_m*. Conversely, the*

[1] We use ˆ to denote a lifted predicate or a set of lifted predicates.

[2] We use lowercase (uppercase) letters to represent variables (objects).

bound variable $person$ in ϕ_i^{talk} is ground according to the variable binding of $\texttt{talk}(person)$ *(e.g., for* $\texttt{talk}(P3)$*, $person$ is ground to $P3$).*

4.2 Online Feature Discovery

In Section 4.1, we describe the initialisation of $\mathbf{\Phi}$ where each feature is a state predicate. More **complex features**, measured by the number of constituent predicates, introduce a finer granularity in the partitioning of the state space which could reduce the approximation errors. We describe line 6 of Algorithm 1 here. We utilise `iFDD+`, a model-free, online feature discovery algorithm which adds conjunctive state predicates to $\mathbf{\Phi}$ to reduce the approximation error. A set of candidate features $\mathbf{\Phi}_a^c$ consists of features which are the conjunction of any two features (its **parent features**) in $\mathbf{\Phi}_a$. We denote the generation of $\mathbf{\Phi}_a^c$ from $\mathbf{\Phi}_a$ as $\mathbb{P}(\mathbf{\Phi}_a)$. `iFDD+` iteratively adds candidate features to $\mathbf{\Phi}_a$ 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 $\mathbf{\Phi}_a$, this in turn generates new and more complex 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, 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). First, the substitution σ_a is applied. Any state predicate with at least one free variable is not fully ground. The set of possible substitutions $\boldsymbol{\sigma}_{all}$ consists of every combination of objects for the free variables. The number of substitutions for a first-order feature $\phi_i(s,a)$ and for $\mathbf{\Phi}_{\hat{a}}$ are:

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

respectively, where $\boldsymbol{v}$ is the set of free variables in ϕ_i and $|v|$ is the number of objects of the same type as v. The selection of a substitution determines the set of ground features, the Q-values, the step update of $\boldsymbol{w}$, and the set of active features for feature discovery. Clearly, the choice of a substitution has a great influence on performance. We refer to this problem as the **grounding ambiguity**. One possible solution is to treat a free variable like an existential quantifier where $\phi_i(s,a)$ is true if there exists a substitution in $\boldsymbol{\sigma}_{all}$ which makes it true. However, 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 discuss this in detail in Section 4.4. To remedy this, we propose the use of contextual knowledge to reduce the number of objects which a free variable can be grounded to. We introduce two forms of contextual knowledge which require trivial domain knowledge: goal and location. The number of substitutions for free variables are reduced by removing from $\boldsymbol{\sigma}_{all}$ any substitution which conflicts with $\boldsymbol{\sigma}_{goal}$ (goal context) and σ_{loc} (location context).

Goal Context. In a planning problem, goals $\boldsymbol{\mathcal{G}}$ 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 in planning problems. The objects in a state predicate $g \in \boldsymbol{\mathcal{G}}$ are used for grounding free variables of the same type. We consider only active goals; a goal is active if it is known and has not been achieved (e.g., in `SR`, a goal or task is known only after a person has given it). 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 [20] but is not the same as additive rewards as defined in [21] where each goal is assumed to contribute uniformly and additively to the reward. In our work, $\phi_i = 1$ 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, and (2) `G1` uses only one goal at a time. The metric used by `GA` will be outlined in a later part of this section. `G1` selects the next goal in an ordering of active goals, $\bar{\boldsymbol{\mathcal{G}}}$, 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 tasks or goals are represented by the predicates* `goal_object_with` *and* `goal_object_at`. *For a particular planning problem, let the goals be* $g_1 = \texttt{goal_object_with}(O1, P1)$ *(item* $O1$ *needs to be delivered to person* $P1$*) and* $g_2 = \texttt{goal_object_at}(O2, WP2)$ *(item* $O2$ *needs to be at location* $WP2$*). The goal contextual grounding are* $\sigma_{goal=g_1} = \{\star obj/O1, \star person/P1\}$ *and* $\sigma_{goal=g_2} = \{\star obj/O2, \star wp/WP2\}$.

Location Context. Mobile robots move in an environment and interact with objects in their vicinity. Following this observation, location context substitutes the free variable with the current location of the robot. 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 predicate* $\texttt{robot_at}(wp)$ *represents the location of a robot. In a state where* $\texttt{robot_at}(WP1)$ *is true,* $\sigma_{loc} = \{\star wp/WP1\}$. *Assume that there are two features* $\phi_1 = \texttt{person_at}(\star person, \star wp) \wedge \texttt{robot_at}(\star wp) \wedge \texttt{goal_object_with}(\star obj, \star person)$*, and* $\phi_2 = \texttt{robot_at}(\star wp) \wedge \texttt{goal_object_at}(\star obj, \star wp)$. *Following Example 2, we can apply* $\sigma_{goal=g_1}$ *and* σ_{loc} *without any conflict to fully ground* ϕ_1 *to* $\texttt{person_at}(P1, WP1) \wedge \texttt{robot_at}(WP1) \wedge \texttt{goal_object_with}(O1, P1)$. *The resulting feature queries if the robot is at the same location as* $P1$ *who* $O1$ *should be delivered to. On the other hand,* $\sigma_{goal=g_1}$ *and* σ_{loc} *are conflicting: the former grounds* $\star wp$ *to* $WP2$ *and the latter to* $WP1$. *However, they can still be applied together if* σ_{loc} *take precedence:* $\sigma_{loc,goal=g_1} = \{\star obj/O2, \star wp/WP1\}$. *Applying this on* ϕ_2*, the resulting feature* $\texttt{robot_at}(WP1) \wedge \texttt{goal_object_at}(O2, WP1)$ *queries if the robot is at the same location as the goal location of* $O2$.

Maximisation Metric. We propose to select $\boldsymbol{\sigma}_M \subset \boldsymbol{\sigma}_{all}$ such that a metric M is maximised: $\boldsymbol{\sigma}_M = \{\arg\max_{\sigma \in \boldsymbol{\sigma}_{all}} M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big)\}$ where $\boldsymbol{\Phi}^{\sigma}_{\hat{a}}$ is the grounding of $\boldsymbol{\Phi}_{\hat{a}}$ with σ. This method is applied after contextual grounding if it is used. We consider $M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big) = \widetilde{Q}(s, a)$ following the motivation of optimistic Q-learning [22]. However, this could introduce instability to an online learning algorithm. In each time step, the weights of active features in $\boldsymbol{\Phi}^{\sigma}_{\hat{a}}$ are updated (line 5 of Algorithm 1). This changes the Q-values which in turn could change $\boldsymbol{\Phi}^{\sigma}_{\hat{a}}$ thereby invalidating the weight update. Thus, we define $M\big(\boldsymbol{\Phi}^{\sigma}_{\hat{a}}\big) = \sum_{\phi_i \in \boldsymbol{\Phi}^{\sigma}_{\hat{a}}} \big(|\phi_i|\big)^2$ where active complex features are preferred. `GA` also uses this definition of M to select goals for contextual grounding. If some domain knowledge or a model is available, `GA` could use heuristics for M (i.e., select nearest goal).

4.4 Granularity of First-Order Approximations

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 predicate $\mathrm{p}(x, y)$ can be ground in $|x||y|$ ways. Since $\widehat{\boldsymbol{\mathcal{S}}}$, $\widehat{\boldsymbol{\mathcal{P}}}$, and $\widehat{\boldsymbol{\mathcal{A}}}$ are independent of $\boldsymbol{\mathcal{O}}$, 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 $\sum_{a \in \boldsymbol{\mathcal{A}}} |\boldsymbol{\Phi}(a)| = 2^{2|\boldsymbol{\mathcal{P}}|} \times |\boldsymbol{\mathcal{A}}|$ 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 $|\widehat{\boldsymbol{\mathcal{A}}}| \leq |\boldsymbol{\mathcal{A}}|$ (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 **greedy actions** (i.e., actions with the maximal Q-values) but some of them are in fact non-optimal actions. The coarse granularity limits the type of problems a first-order approximation can be applied in. Because first-order approximations do not explicitly consider every ground state predicates, it is suited for problems with factored transitions and reward functions and with independent goals.

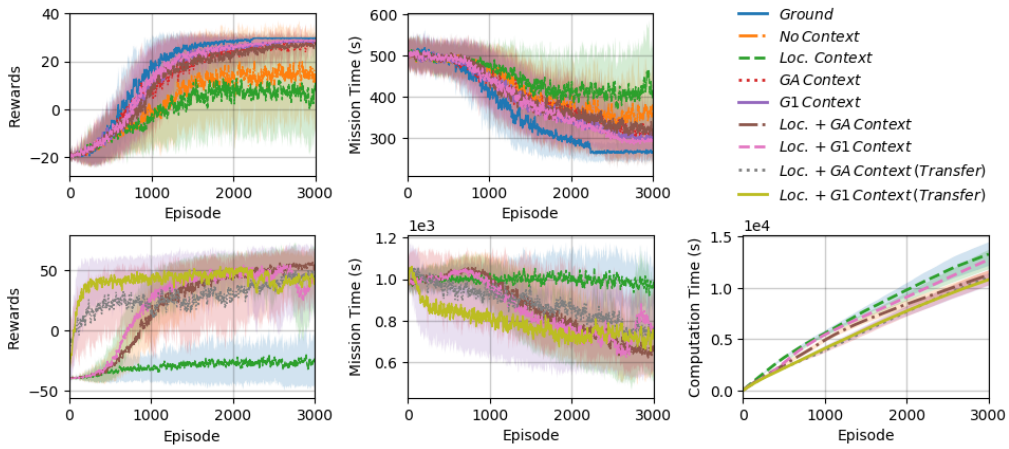

Figure 2: Results for P_{small} (top row) and P_{large} (bottom row) using `RDDLSim` as a simulator. The performances of first-order approximation with different combinations of contextual grounding are compared with ground approximation ($Ground$). Results for transfer learning ($Transfer$) from P_{small}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

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}). Each experiment is conducted on a single core of AMD Opteron 6376 and with at most 4 GB of RAM allowed.

Experimental Setup. In P_{small} (P_{large}), $\boldsymbol{\mathcal{O}}$ contains three (six) items, one person (three people), and five (ten) locations and $|\boldsymbol{\mathcal{S}}| \times |\boldsymbol{\mathcal{A}}| = 2^{56} \times 65$ ($2^{216} \times 264$); 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 by the people. Executing an action gives a reward of -1 while completing each goal gives 20. We tested our work in two simulation environments: `RDDLSim` [23] and `ROS` (Robot Operating System) with the `ROSPlan` [24, 25] 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 persons it encountered along the way, and might not be in the same location as the person since it could detect persons from a distance. Lastly, in `RDDLSim`, picking up and putting down an item always succeeds while these actions might fail in `ROS`.

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). It was observed that the mission time reduces asymptotically while the reward increases. We attempted a reward of $-\Delta$, however, the learned policy prefers actions with small Δ (e.g., $\texttt{talk}(person)$) over completing the goals. Hence, we used a reward of -1 for executing an action which causes the policy to minimise the number of actions executed; this seems to suffice in reducing the mission time.

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 $\boldsymbol{\sigma}$ 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., 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 novel use of first-order features to approximate the Q-function which is learned online under the framework of relational reinforcement learning. Free variables in the first-order features caused ambiguity in how they should be ground. The choice of substitution influences the policy and online feature discovery, and is critical to performance. We resolve this problem by grounding free variables with contextual knowledge which reduces the granularity in function approximation. Empirical results showed that our method is scalable to large-scale problems by keeping memory usage and computational cost tractable, and can generalise across planning problems of different scales and across different simulation environments. In future work, we plan to investigate other forms of contextual knowledge to ground free variables and to consider temporal information such as duration or time constraints in the function approximation.

References

[1] 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.

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

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

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

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

[7] 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.

[8] 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.

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

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

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

[12] 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.

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

[14] 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.

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

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

[17] 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.

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

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

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

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

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

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

[24] 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.

[25] 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.