International Journal of Machine Learning and Cybernetics (2023) 14:1543–1563
https://doi.org/10.1007/s13042-022-01713-5

ORIGINAL ARTICLE

Landmark based guidance for reinforcement learning agents under partial observability

Alper Demir[1] · Erkin Çilden[2] · Faruk Polat[3]

Received: 19 April 2022 / Accepted: 1 November 2022 / Published online: 16 November 2022

Abstract
Under partial observability, a reinforcement learning agent needs to estimate its true state by solely using its observation semantics. However, this interpretation has a drawback, which is called perceptual aliasing, avoiding the convergence guarantee of the learning algorithm. To overcome this issue, the state estimates are formed by the recent experiences of the agent, which can be formulated as a form of memory. Although the state estimates may still yield ambiguous action mappings due to aliasing, some estimates exist that naturally disambiguate the present situation of the agent in the domain. This paper introduces an algorithm that incorporates a guidance mechanism to accelerate reinforcement learning for partially observable problems with hidden states. The algorithm makes use of the landmarks of the problem, namely the distinctive and reliable experiences in the state estimates context within an ambiguous environment. The proposed algorithm constructs an abstract transition model by utilizing the landmarks observed, calculates their potentials throughout learning -as a mechanism borrowed from reward shaping-, and concurrently applies the potentials to provide guiding rewards for the agent. Additionally, we employ a known multiple instance learning method, diverse density, for automatically discovering landmarks before learning, and combine both algorithms to form a unified framework. The effectiveness of the algorithms is empirically shown via extensive experimentation. The results show that the proposed framework not only accelerates the underlying reinforcement learning methods, but also finds better policies for representative benchmark problems.

Keywords Diverse density · Landmark based guidance · Partial observability · Reinforcement learning

1 Introduction

Reinforcement Learning (RL) defines a machine learning paradigm where an agent tries to learn by interacting with its environment [53]. A reinforcement learning agent acts in an environment and tries to solve a task by using the rewards or punishments given. The learning capabilities of an agent are diminished under partial observability. Since the true states of the task are hidden from the agent, there is no guarantee of forming an optimal policy that can ensure the highest returns [6]. Under these circumstances, we can only expect an agent to exhibit *bounded rationality* given the limited sensations from the world.

In order to overcome the uncertainty of the environment, the agent can devise mechanisms; such as keeping track of eligibility trace marks on states to improve the convergence speed of a good policy, or keeping a state estimate to distinguish the true state it is in. Although the idea of eligibility traces seems simple, it is practical to implement and is shown to be useful to speed up the convergence of

The datasets generated during and/or analysed during the current study are available from the corresponding author on reasonable request.

✉ Alper Demir
alper.demir@ieu.edu.tr

Erkin Çilden
erkin.cilden@stm.com.tr

Faruk Polat
polat@ceng.metu.edu.tr

[1] Department of Computer Engineering, İzmir University of Economics, İzmir 35330, Turkey

[2] STM Defense Technologies Engineering and Trade Inc., Ankara 06530, Turkey

[3] Department of Computer Engineering, Middle East Technical University, Ankara 06531, Turkey

the underlying RL algorithm [37]. Faster propagation of the temporal difference error can improve convergence to a policy. However, there may be some tasks that no policy that is formed upon the pure observations can solve. In such a case, the agent has to transform the learning problem to a higher dimension by introducing a state estimation to overcome perceptual aliasing. By maintaining a sort of memory, it may distinguish one experience from another, leading to a more effective policy [27, 35, 43]. Yet, finding a suitable estimation to elevate or guarantee learning under partial observability is a challenge which is beyond the scope of this study.

Most of the time, fortunately, it is possible to generate state estimations that can be recognized as reliable parts of the agent's internal representation for an ambiguous environment. Such estimated states can clearly distinguish the true state of the problem, acting as *landmarks* for a partially observed domain. In general, a landmark is a unique experience that properly distinguishes the agent's current state in a partially observable setting, and we can exploit presence of landmarks to inform the agent about its progress. Relying on these landmarks, an agent can improve its learning performance for a problem with ambiguous observations.

In this study, we focus on tabular Reinforcement Learning with hidden states where an agent requires to form an estimated state from its previous experiences on the environment. We assume that the agent follows a state estimation mechanism in a discrete environment.

Our contributions are summarized as follows:

- We propose a guidance approach for RL with hidden states, Landmark Based Guidance (LBG), that requires the landmarks of the task. The core idea is that the inter-landmark transitions form a semi-Markov decision process and can be used to devise an abstract guidance mechanism by providing guiding rewards based on how valuable it is to visit each landmark in the task.
- We employ Diverse Density (DD) to identify landmarks in the set of estimated states. Diverse Density (DD) with our concept filtering mechanism is shown to work under such settings in our previous study.
- We present an end-to-end framework that operates on the set of estimated states to identify and utilize landmarks by combining DD and LBG during learning.

The experiments on several discrete benchmark problems demonstrate the improvement of our proposed framework on the learning speed.

The rest of the paper is organized as follows. Section 2 introduces the background knowledge for POMDPs with hidden states and reviews the related works. In Sect. 3, we clearly define the concepts of state estimation and landmark and describe Landmark Based Guidance with automatic landmark discovery. Section 3 also provides a complexity analysis of the overall method. Section 4 covers the experiments on several discrete domains. Finally, Sect. 5 gives a discussion and a conclusion to the study.

2 Background and related work

In this section, we provide the background necessary to build up the notions in the following chapters. The section introduces the environment models, explains well known learning methods on them and summarizes existing related work on the topics of the study.

2.1 Problem models

Two decision process model formalisms based on Markov Decision Processes (MDP) are used in this paper. Thus, we first recall that an MDP is a tuple $\langle S, A, T, R \rangle$, where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \rightarrow [0, 1]$ is a transition function, $R : S \times A \rightarrow \Re$ is a reward function [32]. Semantically, $T(s, a, s')$ represents the probability of landing at state s' after taking the action a in the state s, $R(s, a)$ provides the reward taken after employing the action a in the state s.

One of the models used in this work is the Semi-MDP (SMDP) which is an abstraction of MDP over time aiming to model transitions with stochastic time duration (i.e. an action can take more than one time step). It is a tuple $\langle S, A, T, R, F \rangle$, where the first four terms define an MDP and $F(t|s, a)$ denotes the probability that starting at s, action a completes within time t [5]. Obviously, MDP is a special form of SMDP with a step function having a jump at 1 [54]. Importance of SMDP is its ability to model temporal abstractions on an MDP so that improvements can be made both inside the abstracted actions and among the abstractions [54].

The other model is the Partially Observable MDP (POMDP), which is defined by a tuple $\langle S, A, T, R, \Omega, O \rangle$ defined by an MDP (S, A, T and R), a finite set of observations Ω, and an observation function $O : S \times A \rightarrow \Pi(\Omega)$. $O(s', a, o)$ represents the probability of getting the observation o after the agent takes the action a in the state s [33]. POMDP is a generalization of MDP that enables to model a partially observable environment.

2.2 Reinforcement learning

RL aims at learning which action is best for a learning agent by trying to form a model through interacting with the environment, utilizing perceptions and rewards based on its actions [53]. As the environmental dynamics are unknown, the agent needs to discover them during learning.

In RL point of view, an environment is modeled by one of the previous forms and the feedbacks provided to the agent is determined by the reward function of the model. The agent aims to map an action to every state so that it learns how to act in the environment.

Agent's behaviour is defined in terms of a *policy* $\pi : S \times A \rightarrow [0, 1]$ which defines how likely an action will be selected on a state. A learning agent aims to converge to an optimal policy π^* that maximizes the future discounted rewards from a state. Since the dynamics of the problem is hidden from the agent, the optimal policy needs to be converged by estimating the *value function* that describes the utility of being in a state. Following the idea, *temporal difference* (TD) [52] algorithms iteratively updates the value function during the interaction with the environment.

2.2.1 Eligibility traces

As a bridge between the one-step approach of TD methods and Monte Carlo methods, *eligibility traces* were introduced where the agent leaves decaying traces over the previous transitions and employs the value updates based on these traces.

The traces allow a reward to propagate through the previous transitions much faster, leading to a faster convergence. The algorithms that adapted this idea, like Q(λ) and Sarsa(λ), were shown to find good policies [37, 58], where λ represents the decay factor of the eligibility traces.

Q(λ) is the adaptation of Q-Learning algorithm that includes eligibility traces [58]. It keeps a decaying trace over the previously visited state-action pairs, representing their eligibility to the current temporal difference update. Although there are different versions of the algorithm, Watkins' Q(λ), which is also used in this study, resets the traces whenever the agent takes a non-greedy action, to keep the algorithm genuinely off-policy.

Sarsa(λ) is an online on-policy learning algorithm that also utilizes the eligibility trace mechanism [37]. It follows the same pattern of updating the regular Q values of state-action pairs with an error while leaving a trace over the previously visited state-action pairs. Unlike Q(λ), Sarsa(λ) uses the Q-value of the next state-action pair in the TD error and does not reset the eligibility traces upon non-greedy actions, since it is an on-policy learning algorithm.

The parameter $0 \leq \lambda \leq 1$, decays the eligibility traces, whereas they are reset at the end of each episode. While low λ values translate into applying the TD-error at a time step to more recent pairs, higher values of λ allow the algorithm to propagate it to more in past. Sarsa(λ) is shown to converge to a good policy on both fully and partially observable environments [37].

2.3 Reinforcement learning with hidden states

In a realistic context, the agent is not capable of gathering all information regarding the task, since its perception is somehow limited. A corresponding decision process model is formulated via partially observable MDPs, providing the agent with observations rather than states.

There are two interpretations for POMDPs in the literature. The first one considers the MDP structure is either known or estimated, i.e. the agent knows the set of states, the set of observations and the transition function, but does not know the observation function. In this interpretation, the agent can keep a probability distribution over the set of states, called the *belief state* [2], and update it with new observations from the environment by using Bayes rule. This approach gives rise to a RL research track called Bayesian Reinforcement Learning [15, 31, 48, 49, 56]. Some model-based studies aim to learn the environment dynamics to estimate the belief state [28] or to form policies over representations of histories [7, 55, 62].

The other interpretation, which we adopt in this study, constitutes a more realistic setting where there is no knowledge about the underlying semantics of the model and the agent perceives only observations and rewards from the environment. In fact, the agent is clueless about the limits of its sensations to represent the current state of the world.

In a POMDP with *hidden states* [43], the agent has to find a policy based only on the observations from the environment. According to this interpretation, the observation function O can map different states to the same observation, resulting in a problem called *perceptual aliasing* [59]. Perceptual aliasing makes it very difficult, sometimes even impossible, to solve the task especially when the optimal actions for these states are different and cannot be found relying on the same observation. In fact, it has been shown that the regular RL algorithms based on the most recent observation, such as Q-Learning [58], fail to converge to a good policy when the agent's perception is limited [51] because the task is no longer Markovian [3].

In such a partially observable environment, one way to learn is to estimate the true state by employing additional approaches. A state estimate is the agent's representation of the current state in the environment, and the agent aims to find a policy defined over the set of estimated states. It can be formed by a fixed length memory [35], by keeping the previous observation-action pairs. A better way is to extend the memory whenever it is required to form a good policy, leading to variable length memory approaches such as Utile Suffix Memory (USM), Nearest Sequence Memory (NSM) and U-Tree [43].

RL under partial observability is addressed with the advances in deep reinforcement learning methods. Recurrent neural networks are utilized to overcome non-Markovian domains [46, 50, 57, 61]. Such methods utilize Long Short-Term Memory (LSTM) units [27] to summarize the history of the agent. However, they are computationally expensive and reported to be sample-inefficient and too customized for an environment [24].

Finally, algorithms with eligibility traces such as Q(λ) and Sarsa(λ) were employed in POMDPs with hidden states, and shown to find good policies based on the estimated states [37, 58]. The state variable s in the update rules of the algorithms is replaced with the estimated state variable x as the required methodical transformation.

2.4 Diverse density

With limited observability, it becomes important to notice and make use of distinctive information during learning. A viable method is to identify which state visit made an episode successful or unsuccessful according to the observation it yields. Since an observation does not inherit the state features required for dissociation, the identification must be on the episode level. For this task, we argue that a method, namely Diverse Density, is suitable since it has shown to be effective for hidden-state POMDPs [8].

Diverse Density (DD) is originally proposed to address multiple-instance (MI) problems where the bag of instances, but not the instances, are labeled individually [40]. DD forms two types of bags as positive and negative where a positive bag is required to contain at least one positive instance and a negative bag contains only negative ones. Using these bags, the algorithm aims to identify the target concept c_t that leads to the such classification of positive and negative bags.

The original algorithm uses the term concept as an abstract notion that is learned by using the bags. Under RL setting, a concept can be a state in fully observable problems, or an observation or an experience of observations and actions in partially observable domains.

Diverse density is used for online subgoal discovery [34] where the positive-negative classification is applied to the episodes in a goal-oriented MDP [44]. According to their implementation, if an episode ends with a goal state, it is considered as positive; otherwise as negative. With this idea, McGovern *et al.* proposed a method that calculates the diverse density value of a state and they argue that a state with a higher DD value, makes its episode a successful (positive) one and should play an important role in reaching to a goal state. Such states must correspond to a bottleneck or a subgoal state. In their study, Diverse Density is shown to work on the problem of finding subgoals in an MDP online [44].

DD is also used in partially observable problems for discovering landmarks formed only by pure observations [8]. Demir et al. have analyzed why well-known subgoal identification methods, such as graph-based algorithms, fail to identify landmarks in a partially observable setting. The study showed that a multiple-instance algorithm, Diverse Density, is more fit for the online landmark discovery task and proposed a modified version of DD, namely DD with concept filtering (DDCF), that is capable of identifying landmarks in a hidden-state POMDP. Additionally, it improves the speed of identification and removes the necessity of prior information on the domain dynamics.

2.5 Reward shaping

Reward shaping has been used to introduce additional rewards for the agent so that the learning process is further improved. RL with reward shaping operates on the new reward function R' where $R' = R + F$ and F represents the shaping reward. Ng et al. showed that policy invariance with reward shaping can be guaranteed in an MDP by proposing a potential based reward shaping (PBRS) approach where the arbitrary potential function Φ is defined for each state and the shaping reward function is formed as $F(s, a, s') = \gamma\Phi(s') - \Phi(s)$ [14, 47]. PBRS is further extended with potential based advice by forming the potential function with the actions as $\Phi(s, a)$ [60].

Plan based reward shaping uses a STRIPS plan where each state maps to an abstract state and the current step in the plan is used for the potential of a state [19, 21]. Efthymiadis et al. introduced knowledge revision to plan based reward shaping when the plan is inconsistent or wrong [17]. When a hierarchy exists in the task and it is known beforehand, it is shown that reward shaping approach can be formulated into MAX-Q, a well known hierarchical RL (HRL) algorithm, and can outperform its predecessor [18].

PBRS, on the other hand, can be applied for both model-free and model-based RL [1], and its effects are further analyzed. Grzes et al. employed parameter analysis and argued that PBRS should conform several conditions in order to form a consistent advice to the agent [19, 22]. Devlin et al. also showed that dynamic reward shaping, where the potential function is not fixed, can maintain the guarantees of policy invariance [13]. More recently, Grzes stated that the potential value of any terminal state must be zero in order to keep the policy guarantees of PBRS [20] and Marom et al. proposed an algorithm that decays the effect of shaping rewards by experience so that any possible convergence problem can be avoided [39]. Reward shaping idea also found itself a place in multi-agent RL and it has been shown that potential based reward shaping does not alter the Nash equilibria [4, 10–12, 38].

Automatic learning of the potential function turns out to be an interesting problem and also gained attention. Marthi proposed an algorithm that completely solves an abstract model with macro actions formed by sampling from the task and using the value function of this abstract model as the potential function [41]. Grzes et al. adapted a similar abstraction idea which learns the values of the abstract states by value iteration while acting in the environment [23].

Although most related work focused on MDP models, there are very few that take partial observability into account [42]. In [16], PBRS is applied to online POMDP planning so that the shaping reward function is defined via the belief states of the agent.

3 Providing guiding rewards based on landmarks

The problem of perceptual aliasing can be handled by allowing the agent to keep experiences in memory. This way, the agent can form a state estimation that may provide distinction over aliased states. Although finding a distinctive form of state estimation is challenging, such an estimation may still provide room for learning and contain hints for the agent.

In this section, we further dive into the definition of state estimation. However, how to design a good state estimation method is out of the scope of this paper, so we assume that the state estimation method is defined and is available to the agent. We provide a formal definition for the state estimates that can clearly distinguish a true state and propose a method which makes use of these dependable estimations to improve learning speed. Finally, we attack the problem of automatic identification of those estimates on-line to complete the overall learning framework.

3.1 State estimation

An agent can keep a state estimate x by using its previous experiences in an environment. A straightforward example can be the past k observations and actions, forming the estimated state $x_t = o_{t-k}a_{t-k}...o_{t-1}a_{t-1}o_t$ for the time step t. Regardless of the state estimate's structure, the set of estimated states X, then, becomes the plane that the agent forms its policy upon, i.e. the policy is now formed as $\pi : X \times A \rightarrow [0, 1]$. As a natural result, the function $M : X \rightarrow \mathcal{P}(S)$ emerges and determines the mapping between the two sets (Fig. 1) where $\mathcal{P}$ represents power set notation. Note that, M is not a one-to-one function because two different state estimates can map to the same set of true states.

If $\forall x \in X, |M(x)| = 1$, there is no ambiguity and all the state estimates point to distinct true states. But this is almost impossible for real-life problems since finding a state estimation method to clearly disambiguate all states of the environment is very challenging. Alternatively, one may not need to distinguish all the states, for problems where it is enough

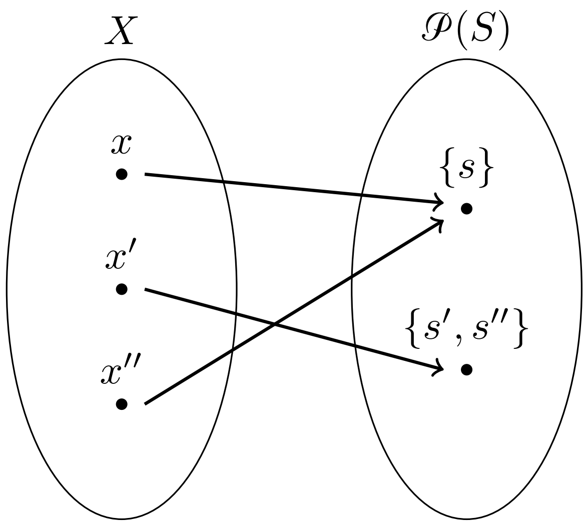


Fig. 1 An example mapping from the set of estimated states X to the power set of true states S in a POMDP with hidden states

for a solution policy to distinguish the estimates yielding different optimal actions. In fact, a perfect scenario would be to have a smaller set of state estimates abstracting over the states with the same optimal action. Methods like USM [43] somewhat aim to achieve such abstract representations by determining the distinctions based on the return distributions, where a branch in a USM tree can be considered as an estimated state.

3.2 Landmarks

It naturally follows for us to assume that not all of the sensations from the environment have ambiguity. Usually there are few observations that the agent can depend on. Some experiences may correspond to specific states and remove the uncertainty about the current state in the state space. Once we have such estimated states, the agent can make use of their "distinctiveness" in the sense of uniqueness, to improve learning.

For such estimated states, we adopt the term "landmark" having the basic meaning of "an event that is marking an important stage or turning point" and put a formal definition for Reinforcement Learning setting as in Definition 1.

Definition 1 The state estimate $x \in X$ is a **landmark** if

$$P(s_t = s | x_t = x) = 1,$$
$$\text{and}$$
$$\forall s' \neq s, P(s_t = s' | x_t = x) = 0.$$

Definition 1 states that a *landmark* is an estimated state, which is mapped to only one true state, so that there is no ambiguity on the current true state for the agent when its state estimate yields the corresponding landmark. By Bayes rule:

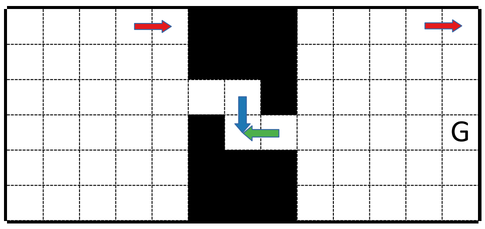
$$\forall s' \neq s, P(s_t = s' | x_t = x) = \frac{P(x_t = x | s_t = s') \cdot P(s_t = s')}{P(x_t = x)} = 0$$

Since this definition is meaningful when there exists a $x \in X$, $P(x_t = x) = 0$ is unlikely. Then, either $\forall s' \neq s, P(s_t = s') = 0$ or $P(x_t = x | s_t = s') = 0$ should be true. Therefore, we conclude that there is no other state that can be represented with this estimate x, that is x can only map to one true state.

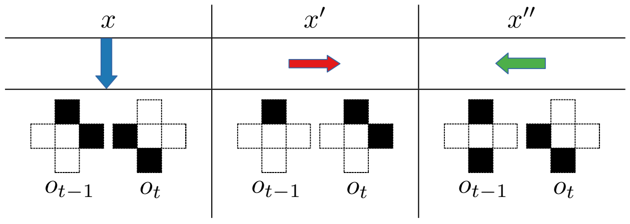
Considering the function $M : X \rightarrow \mathcal{P}(S)$, a state estimate x is a landmark if and only if $|M(x)| = 1$. In the example of Fig. 1, the estimated states x and x'' are landmarks since $M(x) = \{s\}$ and $M(x'') = \{s\}$ but x' is not since $M(x') = \{s', s''\}$. Note that the definition does not require a one-to-one mapping, the agent may have more than one representative for the same true state as in the s case, which is estimated by x and x''. Although they may be redundant estimations of the same state, they still are landmarks. Moreover, Definition 1 does not restrict the structure of the estimated states as long as the representations map to only one true state. Moreover, the definition is for a state estimate, not the true state that it maps to.

Our definition aligns with the one of "sufficient statistics of history" [30] and extends the one proposed for the true states that yield unique observations [29]. In the latter study, a state $s \in S$ is a *landmark state* if $\exists o \in \Omega$ such that $O(o|s) = 1$ and $\forall s' \neq s, O(o|s') = 0$. This definition is restricted to problems where there are observations uniquely representing corresponding true states. With perceptual aliasing, however, the agent may still need to keep a memory to form state estimations. For such a case, even though there are no unique observations for a state, a state estimation can still be capable of representing a unique experience, forming a landmark according to our definition. Thus it focuses on the estimated states instead of observations and provides a wider perspective.

Let us clarify the landmark concept with an example which is given in Fig. 2. The perceptions of the agent are limited to the existence of a wall in the neighboring grid cell in four compass directions, and only the goal state provides a unique observation. For the rest, every observation is yielded by at least two states. In case of pure observations, there is only one landmark, which is the goal state marked as G. To overcome ambiguity, let's assume the agent aims to learn a policy on the estimations formulated as $x_t = o_{t-1}o_t$. That is, it keeps the previous observation in memory to form its estimated state at time t. Table 2b shows state estimations formed accordingly where single observations are ordered from left to right following their temporal order. In this setting, having x as shown in Table 2b can only happen after the agent takes a transition indicated by the blue arrow.



(a) 2D grid world domain



(b) Example state estimations with $x_t = o_{t-1}o_t$

Fig. 2 Example 2D grid world domain **a** where each cell represents a state, the agent can take four navigational actions and gets observations based on the presence of a wall in those directions. The goal state is marked as G. Example state estimations **b** where the form of the estimation is $x_t = o_{t-1}o_t$. The observation sketches are ordered from left to right according to the time they are observed and the estimated states x, x', x'' correspond to the experiences shown with blue, red and green arrows, respectively

Although single observations do not remove ambiguity separately (as they can be observed for three states of the environment), the given estimation identifies the location of the agent unambiguously, therefore, forms a landmark. Similarly, the agent can only have x'' as shown in Table 2b after taking the green transition. Both landmarks x and x'' correspond to the same state, parallel to Fig. 1. However, the state estimation x' is not a landmark since such a transition in red can end in two different states of the problem.

It is natural to think that the agent will have a set of landmarks $\mathcal{L}$ in its representation of the world, the set of estimated states X. These landmarks pinpoint distinct true states in an environment, making them dependable estimations in the agent's internal representation of an uncertain environment. Therefore, their presence can be further utilized to inform the agent while learning. In the following subsection, we propose a method to apply additional rewards based on landmarks to guide the agent towards reward peaks.

Note that Definition 1 describes landmarks as a relation between the agent's state estimations and the model's terms. But in a hidden-state POMDP, the agent does not know the dynamics of the model, so it cannot know which of its state estimations are landmarks by using the probabilities in the definition. The agent should instead discover them online during the early stages of learning with an additional mechanism. For the automatic landmark discovery task, we argue that Diverse Density is a promising candidate.

Since a landmark can be visited when the agent is located in a singular true state, we expect it to be seen less frequently, compared to an aliased estimated state mapping to multiple states. Besides, visiting a useful landmark, on the way to a goal state, should be relatively rare in successful episodes of the agent, making the diverse density of such a landmark high compared to the others. Thus, we believe that DD is a suitable candidate for automatic landmark discovery task. In this study, we couple DD with applying guiding rewards to lead to a better learning under partial observability.

3.3 Landmark based guidance

Landmarks, by definition, are dependable parts of an uncertain situation. Under partial observability, they carry reliable information. The interpretation of Definition 1 pinpoints a sole location of the estimated state space corresponding to its mapping and carries the *potential* of a single state to achieve the goal, unlike ambiguous estimations corresponding to multiple states. That is, they have the same characteristics as their underlying true state in terms of learning. If the agent performs a series of primitive (atomic) transitions from a low-potential landmark to a high-potential one, this abstract move can be rewarded, utilizing the landmarks to inform the agent about its progress. Such a reward may be very useful in partially observable problem domains with delayed or sparse reward mechanisms [9].

One may simply propose to provide a bonus upon reaching a landmark. However, a bonus under partial observability may cause the agent to get stuck in a landmark. Since a visit at a landmark will be rewarded, the agent may eventually prefer to choose those actions rather than explore the uncertain regions enough. Therefore, the agent should be rewarded based on whether its actions orient it towards transitions with high rewards or not, suggesting a routine control on the differences between the potentials of visited landmarks.

We argue that application of additional rewards for problems with hidden states is meaningful if the transitions among the landmarks are taken into account only. Since a potential value assigned to an ambiguous state estimate is unreliable, providing an additional reward to a transition with an ambiguous state estimate will not be beneficial in a partially observable environment. It might encourage the agent to take an action at a state, based on the potential value of its current ambiguous state estimate, and the action might thus mislead the agent away from a goal state. Instead, assuming that the state transitions between two landmarks form a temporal abstraction, one can define a meta-level transition among two landmarks at an abstract level. This meta-level transition may correspond to a series of primitive actions in the underlying model, but the additional reward is provided only when these actions complete a transition between two landmarks.

Note that this approach does not align with the idea of the potential based reward shaping and benefit from the guarantees of it, since it does not shape the reward in each transition and changes the return of the sequence of states the agent follows. On the other hand, policy invariance is obviously not a concern in a non-Markovian environment since an optimal, or even a “good enough” deterministic memoryless policy is not guaranteed at all [36]. Due to these differences, we avoid to use the term “reward shaping”.

In order to apply additional rewards, one needs to calculate the potentials of the landmarks. Our work is influenced by the abstraction and value iteration ideas of [23] where an abstraction is created over the set of MDP states and an online learning method for the potentials of these abstract states is proposed. We make use of this idea for the value iteration approach on the abstract model similar to their proposal, where the set of landmarks form an abstract model and the agent executes abstract actions lasting over one time step between the landmarks. However, our method differentiates in that we assume the problem model is a POMDP with hidden state interpretation, and we do not make further abstractions over the set of observations. We follow their value iteration approach on the abstract model of landmarks since it better aligns with the reward mechanism of the problem, unlike other heuristics. The abstract model is in fact an SMDP [54] and we will call it *Landmark-SMDP*. Figure 3 depicts how the Landmark-SMDP of the `6Rooms` domain looks like, also sketching which true problem state each landmark corresponds to.

Definition 2 A *Landmark-SMDP* is a SMDP whose set of states S is constructed by the landmarks $\mathcal{L}$ of the formed state estimation set X.

The Landmark-SMDP is inherently composed of less number of states compared to the set of estimated states of the POMDP since usually most states are perceptually aliased in a partially observable environment. Therefore, the potentials of the landmarks can be easily calculated with value iteration and then used to calculate internal rewards in the underlying POMDP while performing actions in the environment.

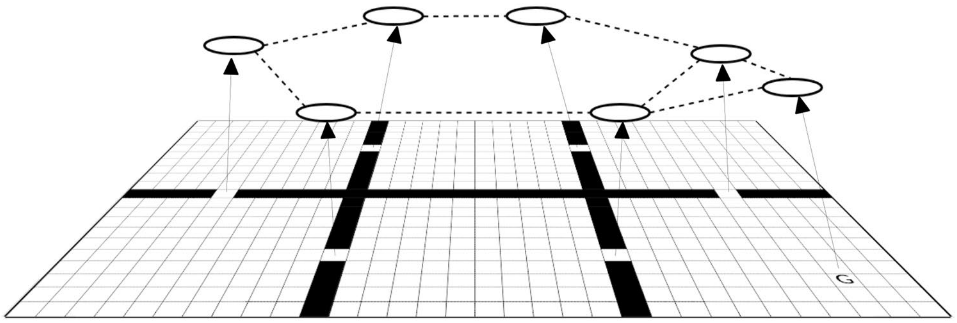


Fig. 3 An illustration of an abstract model for a sample grid world domain (`6Rooms`, see Fig. 5a for the original sketch). The landmarks correspond to the doorways and the goal state is marked with *G*, where circles and dashed lines represent the landmarks and transitions between landmarks in the abstract SMDP model. Note that, depending on the observation semantics and state estimation model, the formal definition of the natural "doorway" landmarks may vary

Algorithm 1 Reinforcement Learning with Landmark Based Guidance

```
Require: $\gamma, \mathcal{L}, \alpha_v, \gamma_v$
1:  $\forall \ell \in \mathcal{L}, V(\ell) = 0, t = 0$
2:  $\ell = \emptyset, \tau = \emptyset, \ell' = \emptyset, \tau' = \emptyset$
3:  $H = \emptyset$
4:  if $x_t \in \mathcal{L}$ then
5:      $\ell = x_t, \tau = t$
6:  end if
7:  repeat
8:      Observe transition $\langle x_t, a_t, r_t, x_{t+1} \rangle$
9:      if $a \neq \emptyset$ then
10:         $H = H \cup \langle x_t, a_t, r_t, x_{t+1} \rangle$          ▷ Save the transition if necessary
11:     end if
12:     $f_t = 0$
13:     if $x_{t+1} \in \mathcal{L}$ then ▷ Check if the reached estimated state is a landmark
14:         $\ell' = x_{t+1}$
15:         $\tau' = t + 1$
16:         if $\ell \neq \emptyset$ then                ▷ Check if there is a previous landmark
17:             $f_t = V(\ell') - V(\ell)$
18:             $n = \tau' - \tau$
19:             $\hat{r} = r_\tau + \gamma_v \cdot r_{\tau+1} + ... + \gamma_v^{n-1} \cdot r_{\tau'-1}$          ▷ by using $H$
20:             $V(\ell) = (1 - \alpha_v) \cdot V(\ell) + \alpha_v \cdot (\hat{r} + \gamma_v^n \cdot V(\ell'))$
21:         end if
22:         $\ell = \ell', \tau = t + 1$
23:         $\ell' = \emptyset, \tau' = \emptyset$
24:         $H = \emptyset$
25:     end if
26:     Learn by using $R'(x_t, a_t, x_{t+1}) = r_t + \mathbf{1}_{f_t > 0} \cdot f_t$
27:     $t = t + 1$
28: until episode ends
```

The overall algorithm for Landmark Based Guidance (LBG) is given in Algorithm 1. The algorithm assumes that the set of landmarks $\mathcal{L}$ is given. It requires as input the learning rate α_v and the discount rate γ_v for the Landmark-SMDP value iteration. The algorithm starts with initialization of the value function V of the Landmark-SMDP, the time index t, the previous and the current landmark variables ℓ and ℓ', the previous and the current time variables τ and τ' to keep the time that ℓ and ℓ' are seen and the short history H (Lines 1–3). Upon deciding that the initial estimated state is a landmark, ℓ and τ are initialized by the estimated state and the current time respectively (Lines 4–5). Then comes the familiar observation-action loop, where the agent interacts with its environment and observes transitions between the estimated states x_t and x_{t+1}. Meanwhile, the algorithm keeps track of a transition in H to check if there is a previously observed landmark, in order to calculate the discounted sum of rewards to be used in the Line 18.

Since the precondition to provide guiding rewards through the abstract model, it is checked whether or not the agent arrives at a landmark. If it does, the current landmark ℓ' and the current landmark time τ' are set (Lines 13–14). If there is a landmark previously observed, this means it is possible to provide the additional reward, which is calculated in Line 16.

Following the internal reward calculation, the algorithm determines the sum of discounted rewards, gathered between the previous landmark ℓ and the current landmark ℓ' by using H and makes a value update on ℓ (Line 18), where n represents the number of steps taken between the two (Line 17).

Afterwards, the previous landmark variables are shifted with the current landmark variables (Line 20) and the current landmark variables and the history H are reset (Lines 21–22). Finally, the updated reward is fed to the underlying learning algorithm, and the agent continues to interact with the environment. Here, the internal reward is used only if it is positive (Line 24). As the agent aims to learn a policy over an "uncertain state estimation set", application of a punishment for attaining a certain estimated state might not be beneficial. Instead, rewarding correctly is enough by itself to guide the agent towards transitions yielding higher rewards.

3.4 Automatic landmark discovery

Although Algorithm 1 requires the set of landmarks in advance, for a more realistic problem setting, the agent has no prior knowledge about which state estimates are true and unambiguous. In order to increase the autonomy in this respect, the agent should discover the landmark set during learning.

Our previous study [8] shows that DD, when coupled with a concept filtering mechanism (DDCF), is capable to identify unique observations under partial observability. It is coupled with SarsaLandmark algorithm to discover landmarks caused by single observations automatically and to improve learning speed for RL with hidden states. Former performance of DDCF on automatic landmark discovery makes it a promising candidate for the wider definition of landmarks. However, our previous study is limited to landmarks that contain single observations and does not extend to problems that the agent may form landmarks including its memory.

The DD idea is based on identification of the unique points in successful histories. It requires a bag level classification to have successful and unsuccessful episodes. Afterwards, it checks the instances that are uniquely seen on positive bags but not on negative ones. By definition, a landmark maps to only one state, unlike other ambiguous state estimations that yielded by visits to different states. Hence, in a successful episode, visiting an ambiguous state estimation is more likely than visiting a landmark, causing a useful landmark to be observed more *diversely* in an episode, yet more *densely* on successful ones.

With this motivation, one can employ DD, especially the DDCF variant, to identify the landmarks during learning to form an overall framework that can run in a realistic RL setting. Here, every instance and concept is a state estimation, and a bag is an episode throughout the policy execution for the problem. The outline of DDCF is follows:

- After each episode, DDCF classifies it according to a success criteria (for example, reaching to the goal state under a step threshold) as either a positive or a negative bag.
- In order to calculate diverse density values for each estimated state, DDCF algorithm requires the shortest path distances between each pair.
- After the episode classification, DDCF updates its graph of interactions between estimated states.
- Based on this graph, the distance between estimated states is calculated.
- DDCF further employs a concept filtering with a metric called *congestion ratio* to filter out the redundant candidates for landmarks.
- Using the distances, DDCF calculates DD value for each estimated state in this filtered set.
- Next, it employs the static filter of the original algorithm to avoid identifying estimated states close to a goal state.
- As a last step, for each estimated state with a peak diverse density value, DDCF increases the relative running average and if an estimated state passes the threshold, it is marked as a landmark.

When DDCF finishes, the identified landmarks can be used to provide guiding rewards to the agent. However, these two processes should not be executed concurrently, since a newly

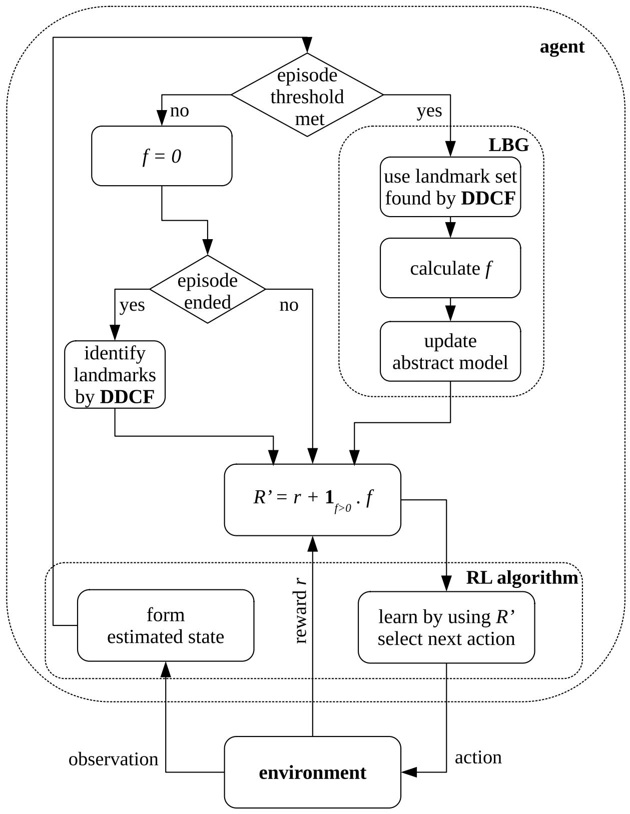


Fig. 4 The workflow of a RL agent with DDCF and LBG extensions

identified landmark may disrupt the value iteration and initiate misguiding rewards. That is why we first employ DDCF to find a set of landmarks to be used to guide the agent. After a certain episode threshold is met, the Landmark-SMDP is formed by using the identified landmarks and is used by LBG to introduce guiding rewards.

The overall framework is depicted in Fig. 4. After each interaction with the environment, the underlying RL algorithm forms the estimated state using the new observation and reward. The estimated state is then fed to the proposed extension. First, the algorithm checks for an episode threshold. Failure means this is the first stage of identifying landmarks with DDCF and there is no guiding reward yet. If this transition ends the episode, DDCF algorithm is employed as stated before. If there is a newly identified landmark, it is added to the set of identified landmarks. Alternatively, upon passing the episode threshold, the algorithm goes into stage two of LBG. Together with the landmarks identified in the first stage, LBG calculates the guiding reward *f* and updates its Landmark-SMDP by altering the potentials via Algorithm 1. The guiding reward is added to the immediate reward to be used by the underlying reinforcement learning algorithm.

As a use case scenario, consider the training of an autonomous agent (such as a robot, vehicle etc.) within a certain real, or more realistically, a virtual environment. Consider that limitations of the agent's sensors are represented as the observation space, while the previous experiences and the current observation forms the agent's estimated state for that time. The first phase of the process focuses on the automatic identification of the landmarks within the context of the estimated states. By using DDCF, the agent extracts the landmarks in the estimated state space, while RL algorithm has already started. The second phase consists of the standard exploration-exploitation steps of the RL mechanism, augmented with an abstract model representing the Landmark-SMDP, again within the context of the estimated states. The abstract model is used to guide the agent by using the calculated potential of a landmark reached recently, by means of eventually directing the agent to landmarks with high potential. This way, the agent uses the landmarks as "checkpoints" on the way to the goal, consequently requiring less exploration, resulting in faster convergence. As a result, the training of the agent would require less time and resource.

3.5 Complexity analysis of the framework

In this section, we analyze the computational overhead of the proposed framework over the underlying learning mechanism. We provide the proofs for the theorems concerning the time complexity of each part of the framework and formulate an overall analysis.

Lemma 1 *Let E_i be a sequence of transitions occurred in the episode number i, X_i be the set of observed state estimations, T_i be the set of transitions between them after the i^{th} episode and the operator $||$ represent the cardinality of a sequence or a set. Then, Diverse Density with Concept Filtering runs in $O(|E_i| + |X_i|^2 + |X_i| \cdot |T_i|)$ at the end of an episode i of the main Reinforcement Learning loop.*

Proof In order to identify the set of landmarks, DDCF employs a search for the state estimations with peak diverse density values at the end of an episode. When the episode ends, it is traversed to count the occurrence of each state estimation and the counts are kept in a hash map with $|X_i|$ keys where X_i is the set of unique state estimations after the episode i. This operation can run in $O(|E_i|)$ as the iteration needs to count every instance in the episode.

The modified version of the algorithm [8] keeps an unweighted and undirected graph of transitions between the state estimations, in order to calculate the distances between pairs and employs filtering mechanism. After each step, the newly observed transition is added to the graph, so cumulatively, forming the graph takes $O(|X_i| + |T_i|)$ where there are $|X_i|$ number of vertices and $|T_i|$ number of edges after the i^{th} episode. Additionaly, updating the graph requires recalculating of the shortest path distances. By employing Breadth-first Search from each state estimation, the distance matrix can be constructed in $O(|X_i| \cdot (|X_i| + |T_i|)) = O(|X_i|^2 + |X_i| \cdot |T_i|)$ where the graph is represented as an adjacency list.

The algorithm advances by classifying the new episode either as successful or unsuccessful. This classification takes $O(|E_i|)$ time by checking whether the last reward is the highest of all throughout the episode or not.

In order to apply concept filtering, our algorithm [8] requires a metric, namely congestion ratio $CR^{(k)}(x)$, to be calculated for each state estimation x. First, the algorithm finds the degree and bridging coefficient $BC(x)$ of each node in the graph in $O(|T_i|)$ in the adjacency list representation by iterating over all edges. For calculating k-clustering coefficient $CC^{(k)}(x)$ of a node, it traverses a row of the distance matrix and finds k-neighborhood of a node in $O(|X_i|)$. Finding the number of edges in k-neighborhood of a node takes $O(|T_i|)$ by checking the existence of an edge between the nodes within the neighborhood. Calculation of $CC^{(k)}(x)$ for all nodes takes $O(|X_i| \cdot (|X_i| + |T_i|))$. Hence, calculating $CR^{(k)}(x)$, takes $O(|T_i| + |X_i| \cdot (|X_i| + |T_i|)) = O(|X_i|^2 + |X_i| \cdot |T_i|)$ for all nodes. Deciding that a node has the peak value among its direct neighbors again requires an iteration over the set of edges, consuming $O(|T_i|)$ time, therefore the time complexity of this step is determined by $O(|X_i|^2 + |X_i| \cdot |T_i|)$.

Following the formation of filtered set of state estimations X_f, the algorithm employs a diverse density search among them. The diverse density value of a state estimation can be calculated iteratively, so at the end of each episode, only the last episode's value should be calculated and added. Since the counts for each episode are stored in a hash map having $|X_i|$ number of keys, the search takes $O(|X_f| \cdot |X_i|)$ time. The final steps of the algorithm, which involve the running average threshold check and static filter application, are employed for the set of peak valued state estimations X_p, hence yield $O(|X_p|)$ complexity.

To wrap up, at the end of each episode, DDCF takes $O(|E_i| + |X_i|^2 + |X_i| \cdot |T_i| + |E_i| + |X_i|^2 + |X_i| \cdot |T_i| + |X_f| \cdot |X_i| + |X_p|)$. Since $|X_p| \leq |X_f| \leq |X_i|$, this results in $O(|E_i| + |X_i|^2 + |X_i| \cdot |T_i|)$ where E_i is the sequence of transitions in the episode i, X_i and T_i are the sets of observed state estimations and the transitions between them, at the end of episode i, respectively. □

Lemma 2 *Let n be the number of episodes for which DDCF is employed, $|\hat{E}|$ be the average length of an episode, X_n be the set of observed state estimations and T_n be the set of transitions between them after the n^{th} episode. Then, the identification of the landmark set by DDCF takes $O(n \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n|)$ time.*

Proof Assume that DDCF is employed for n episodes. Since the graph grows with new episodes, $|X_i| \leq |X_{i+1}|$ and $|T_i| \leq |T_{i+1}|$. Due to big-Oh notation, the time complexity of the graph operations is dominated by the final size of the graph. Moreover, the algorithm still needs to parse each episode history. So, assuming $|\hat{E}|$ is the average length of an episode, using Lemma 1, the identification of the landmark set takes $O(n \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n|)$ when DDCF is run for n episodes. □

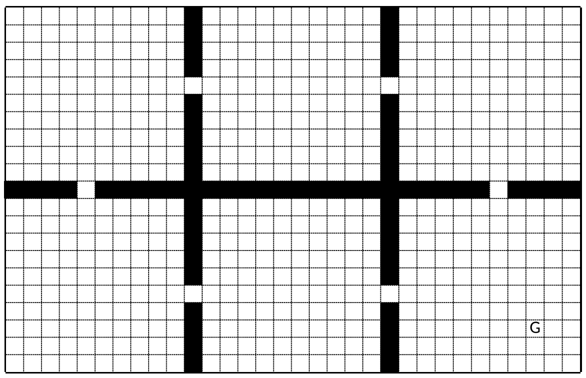
Theorem 1 *Let n be the number of episodes for which DDCF is employed, m be the number of episodes for which overall framework is executed, $|\hat{E}|$ be the average length of an episode, X_n be the set of observed state estimations and T_n be the set of transitions between them after the n^{th} episode. Then, the time overhead of the framework is $O(m \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n|)$.*

Proof Let DDCF be executed for n episodes to identify the landmark set. Then, by Lemma 2, it takes $O(n \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n|)$ time. Following the formation of the landmark set, LBG is started. At each time step, calculation of the guiding reward and a value iteration for the potentials takes place at constant time as the accumulated discounted rewards can be iteratively calculated. Hence, these operations take $O(|\hat{E}|)$ assuming that $|\hat{E}|$ is the average length of an episode. Since LBG is employed after DDCF, it executes for $m - n$ episodes. Therefore, time complexity of the whole framework is $O(n \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n| + (m - n) \cdot |\hat{E}|) = O(m \cdot |\hat{E}| + |X_n|^2 + |X_n| \cdot |T_n|)$, given that DDCF runs for n episodes and the framework runs for m episodes. □

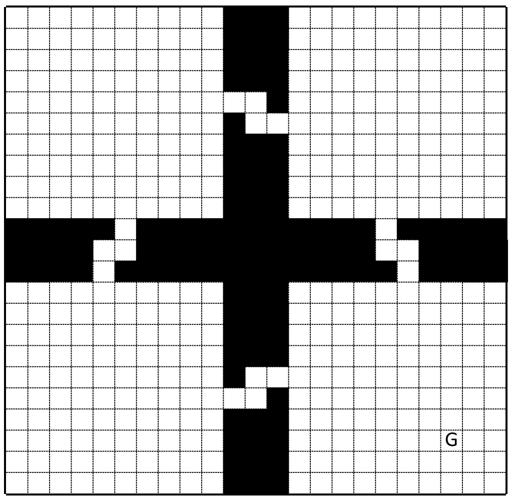
4 Experiments

In the experiments, we addressed the following questions: (1) Is DDCF capable of identifying complex landmarks? (2) Does LBG provide a speed gain in the learning by having the set of landmarks either given in advance or identified online? (3) How does λ effect the learning performance of LBG with methods using eligibility traces? (4) How does the coverage of landmarks affect the learning speed of LBG?

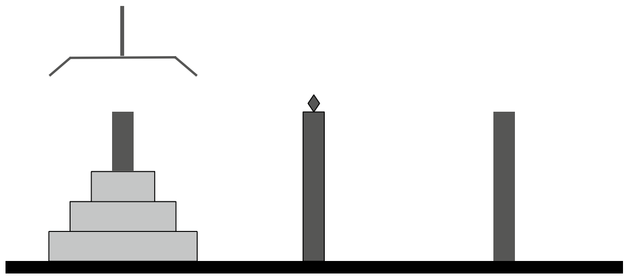
In order to answer these questions, two well known RL algorithms with eligibility traces were used as underlying learning algorithms, namely Q(λ) and Sarsa(λ) [37, 58], due to their effectiveness in POMDPs with hidden states. Watkins' Q(λ) version is used for Q(λ) implementation in order to keep it truly off-policy. Note that our guidance mechanism can be adapted to different learning algorithms and our focus for this study is to show that it can provide an improvement



(a) 6Rooms



(b) 4Rooms4Hallways



(c) ToH3

Fig. 5 Sketches of the domains used in the experiments for Landmark Based Guidance. The goal states are marked with G in the grid world domains

Table 1 Details of the domains used in the experiments for Landmark Based Guidance

Problem	$\|S\|$	$\|A\|$	$\|\Omega\|$	Action Noise	References
`6Rooms`	564	4	43	Yes	[45]
`4Rooms4Hallways`	374	4	12	Yes	[44]
`ToH3`	161	4	31	No	[26]

over the underlying algorithms with no specific bound to them.

For the first question, we analyzed the results of DDCF in terms of precision and recall of the discovered landmark set. For the second one, the aforementioned algorithms are tested in terms of the number of steps to reach to a goal state and success rates of the final policies against two LBG variants: (1) LBG with the landmarks provided in advance, (2) LBG coupled with DDCF for automatic landmark identification. Experiments were carried out using different forms of state estimation since higher level state representations are required for some domains in order for the problem to be solvable to some extent, i.e. the solution set to include a relatively good policy. Note that, the aim of the experimentation is not to find the best state estimation, but to show that LBG can be helpful on estimated state formulations of various complexity levels.

The effect of trace decay is tested with different values of λ in terms of speed gain over the underlying algorithm. Finally, the last query is addressed by providing subsets of landmarks differing in coverage to LBG in advance to observe the effect on learning performance.

The following sections include the experimental results and their discussion of several problems.

4.1 Problem domains

We carried out experimentation for three domains, two of which (`6Rooms` and `4Rooms4Hallways`) are grid world domains, and one (`ToH3`) is a puzzle. The sketches and the size characteristics of the problems are shown in Fig. 5 and Table 1, respectively. These problems are selected since they possess different observation semantics that build up landmarks in various state estimation forms. Although these forms do not completely overcome the ambiguity in the problem, they form landmarks that can be used to guide the agent.

`6Rooms` and `4Rooms4Hallways` are navigational tasks where the agent can take four actions as *north*, *east*, *south*, *west*. Actions are stochastic, resulting in the intended direction with 0.95 probability and either left or right of the intended direction with 0.025 probability.

In `6Rooms`, the agent's observations are formed according to its distance to the walls in four compass directions (one step from the wall, two steps from the wall, closer to the wall in this direction than the other, further from the wall in this direction than the other). Moreover, the doorways of `6Rooms` provide unique observations. Conversely, `4Rooms4Hallways` limits the perceptions to the presence of a wall in the next cell in four directions. For both navigational domains, the agent starts at the north-west room and aims to reach to uniquely observable goal state (marked as G in Fig. 5a and b) where each regular action is receives

–0.01 punishment and upon reaching the goal, the agent is rewarded by 1.

ToH3 is a classical puzzle that contains *m* rods and *n* disks where initially, the disks are placed either on the left-most or the middle rod in the increasing size order. The goal of the agent is to move the disks one at a time so that eventually they are placed on the right-most rod with the same order. In this version of the problem, the agent has an arm that can employ four deterministic actions as *left*, *right*, *pick up* and *put down*. An action yields a reward of 10 upon reaching to the uniquely visible goal state, and no reward or punishment is provided otherwise. ToH3 contains 3 disks and 3 rods where the middle rod has an additional –distinguishing– shape that is visible by the agent with a probability of 0.8. The agent's perception is limited to the contents of the rod that the arm is currently on: whether or not the current rod is the middle one and whether or not the arm is holding a disk. Because of this partial observation semantics, the agent has to keep some sort of memory to form a good policy for solving the task.

This study focuses on the domains that may lead to landmarks in different forms of state estimations. 6Rooms, on the one hand, contains landmarks within the observation space since the doorways provide unambiguous observations. On the other hand, although the observations 4Rooms4Hallways are extremely ambiguous, by maintaining a memory the agent can form unique estimations. Recalling the example in Fig. 2, the two step observation sequence where an observation with a north-east wall boundary is followed by an observation with a south-west wall boundary is possible for only one estimated state, while these two observations are individually yielded by three distinct states each. Finally, for ToH3, the agent may need to maintain an even more complex memory, including the action between the previous observation and the current one, creating landmarks as shown in Fig. 6.

Note that our experiments contain problems generating landmarks in key points of learning on different levels of state estimation, in order to illustrate the positive effect of our method. However, complex forms of state estimations may yield landmarks which do not need to be visited in order to reach a goal state. For example, a transition like in Fig. 6d occurs due to a redundant action, making the agent stay in the same state. This observation-action-observation sequence is a landmark since it is a unique experience, but it is not a useful one and certainly does not correspond to a state that needs to be visited.

As it can be seen in Fig. 7, the observation transition semantics do not align with the state space structure of the problems. Moreover, Table 2 shows how the size of the estimated state set X and the size of the landmark set $\mathcal{L}$ alter by different forms of state estimation formulations. In order to further understand how ambiguous the estimated states are, we propose a metric called the *aliasing ratio*, which is defined as,

$$\kappa = \frac{\sum_{x \in X} |M(x)|}{|X|} \tag{1}$$

where $|M(x)|$ represents the number of true states that the state estimate x corresponds to. Note that, this metric should not be interpreted as a measure of difficulty of the problem. Instead, we can argue it indicates the uncertainty of the observation semantics. As can be seen in Table 2, a form of memory can decrease the aliasing ratio for all of the problems, yet including the transition action between observations is only useful for ToH3 due to non-determinism on the outcomes of actions in other problems.

4.2 Settings

Q(λ) and Sarsa(λ) algorithms are compared to their versions with LBG and the coupling of DDCF and LBG, in terms of the number of steps to reach the goal state. The main experiments are performed with different forms of state estimation for 6Rooms, 4Rooms4Hallways and ToH3 as $x_t = o_t$, $x_t = o_{t-1}o_t$ and $x_t = o_{t-1}a_{t-1}o_t$, respectively. As baselines, optimal MDP policy performance and deep RL approaches with LSTM units, namely A2C [46] and ACKTR [61], are also plotted.

In the experiments, an episode starts at a randomly selected initial state of the problem and ends either when the agent reaches a goal state or after 5000 steps are taken. The results are averaged over 50 experiments, each of which took 25000 episodes.

For the learning parameters, we used the values that showed the best performance in the original studies. Consequently, $\lambda = 0.9$ is used for the main experimentation. For both RL algorithms and all the domains, $\alpha = 0.01, \gamma = 0.9$ are used, and an ϵ-greedy action selection method is employed with ϵ starting at 0.2 and linearly decaying down to 0.0001 throughout an experiment (until the last episode).

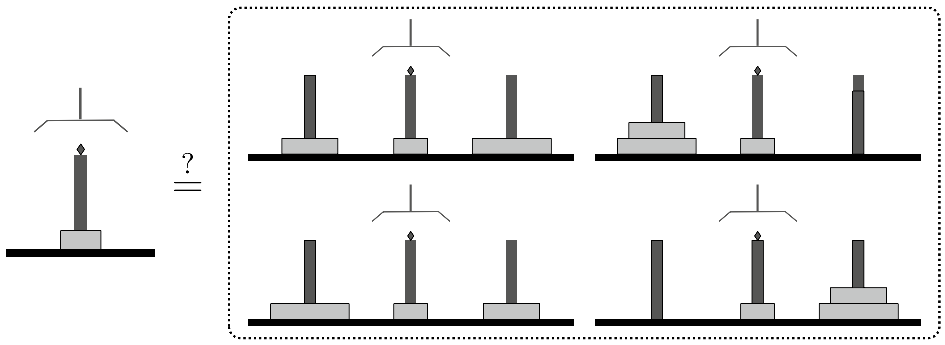
The value iteration of the Landmark-SMDP used $\gamma_v = 0.99$ and $\alpha_v = 0.05$ in all the domains, which give the best results for LBG. For LBG, we assumed the landmark set $\mathcal{L}$ is provided beforehand, that is, the agent can perfectly sense whether or not it is in a landmark. For the coupling of DDCF and LBG, we let DDCF to run for 2000 episodes, then employed LBG with the identified set of landmarks. For DDCF, we used $\theta = 10.0$ and $\lambda_{DD} = 0.95$ where an episode is considered successful if it ends with a peak reward and lasts shorter than a determined step threshold. We used a static filter of 2 steps in 6Rooms and 4Rooms4Hallways where it is set to 3 for ToH3 problem. Moreover, we employed concept filtering by setting $k = 2$ in the congestion ratio metric.

We used Stable Baselines implementations [25] for A2C and ACKTR algorithms. Both of the methods had a network of 5 fully connected layers having 32 neurons per layer. They used 256 sampled experiences in each update with the learning rate of 0.0001. We selected the number of hidden neurons for the LSTM as 32. For ACKTR, we had the weight for the entropy loss as 0.01, the weight for the loss on the value function as 0.25 and gradient clipping for Kullback-Leibler as 0.01. The rest of the parameters are left as their default.

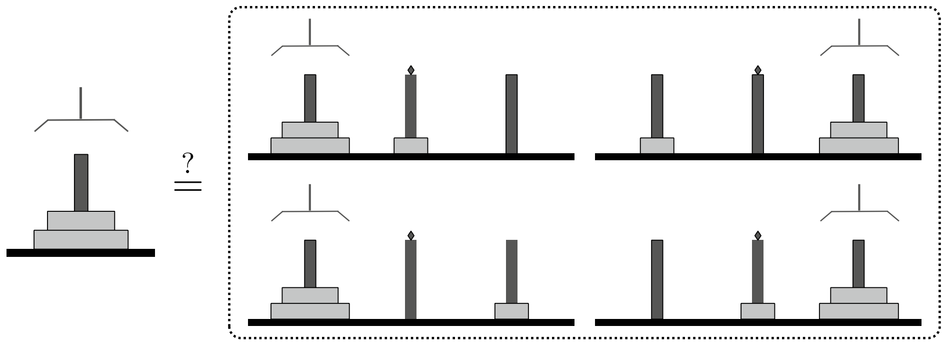
4.3 Learning performances

For the main experiments, we reported the average number of steps taken to reach the goal state with the 95% bootstrapped confidence intervals. The agent used different forms

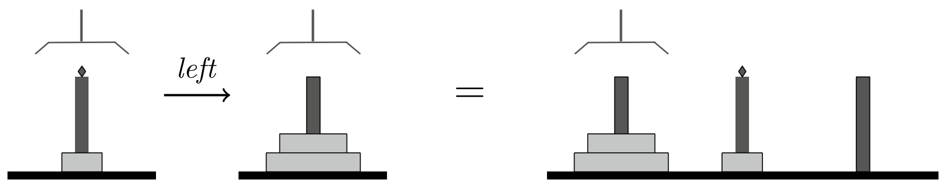
Fig. 6 A simple example in `ToH3` problem where observation of only one rod may correspond to multiple configurations, but a transition from one to another with *left* action is corresponds to a single estimated state, creating a landmark formed as $x_t = o_{t-1}a_{t-1}o_t$



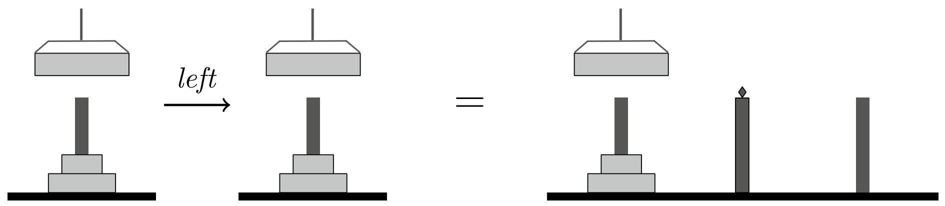
(a) An observation on the middle rod with the smallest disk, corresponding to four different states.



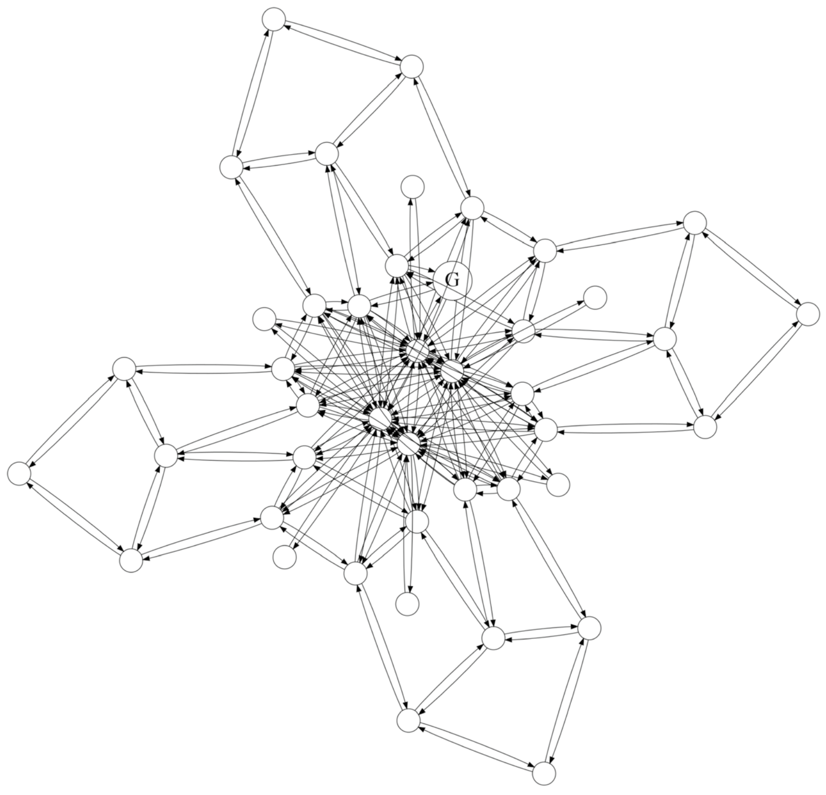
(b) An observation on a rod with the disks except the smallest one, corresponding to four different states.



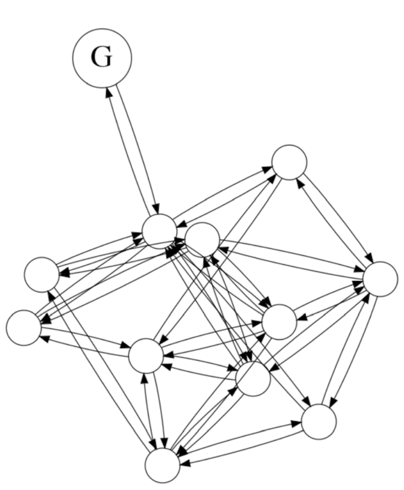
(c) The landmark generated by the transition between the observations via *left* action, mapping to only one state.



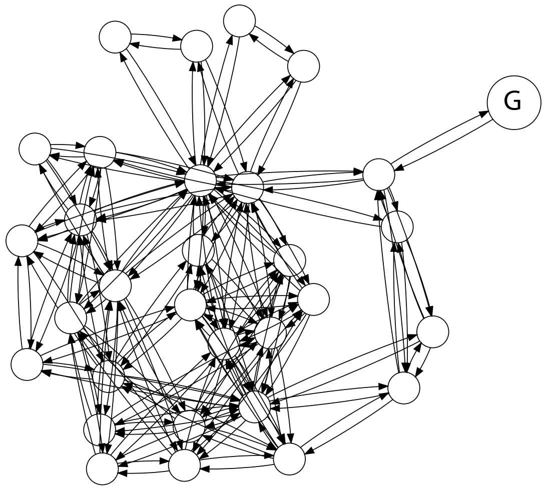
(d) An unnecessary landmark generated by staying in the same state via *left* action, mapping to only one state.



(a) 6Rooms



(b) 4Rooms4Hallways



(c) ToH3

Fig. 7 Observation transition graphs of the domains. The goal states are labeled as G

Table 2 Details of the estimated state space with different state estimation forms where κ is the aliasing ratio

State Estimate	6Rooms			4Rooms4Hallways			ToH3		
	$\|X\|$	$\|\mathcal{L}\|$	κ	$\|X\|$	$\|\mathcal{L}\|$	κ	$\|X\|$	$\|\mathcal{L}\|$	κ
$x_t = o_t$	43	7	14.09	12	1	34.66	31	5	6.87
$x_t = o_{t-1}o_t$	305	125	6.28	69	21	12.58	251	26	3.75
$x_t = o_{t-1}a_{t-1}o_t$	1010	418	6.23	244	76	12.85	442	190	2.64

of state estimation so that we can show the performance of our proposed method with different forms of landmarks.

In Table 3, the identification performance of DDCF + LBG is given when the agent used state estimates as $x_t = o_t$, $x_t = o_{t-1}o_t$ and $x_t = o_{t-1}a_{t-1}o_t$ for `6Rooms`, `4Rooms4Hallways` and `ToH3` respectively. As stated earlier, DDCF was run for the first 2000 episodes to find the landmarks in the task. In case of accuracy, DDCF is shown to perform well on different levels of state estimation. It resulted in noise free sets of landmarks in almost all the problems. On the contrary, the table shows DDCF does not cover all the landmarks. This behavior originates from the focus of DDCF on the landmarks that are most useful for success. Especially in `ToH3`, there are 179 landmarks in the form $x_t = o_{t-1}a_{t-1}o_t$ (Table 2), yet only a few of them cause the highest DD values and are enough for guiding the agent.

Figure 8 shows the learning performances in `6Rooms`. Learning a policy based on the pure observations ($x_t = o_t$) seems to work in this domain. Although it is not optimal, all the algorithms can find a good policy that can lead the agent towards high rewards. It is clear from the figure that LBG dramatically improves the learning performance of both Q(λ) and Sarsa(λ) since it guides the agent to the goal state from the beginning of the learning with the landmark set provided beforehand. Likewise, the agent's performance improves when the set of landmarks found by DDCF is fed to LBG and the guidance starts. Around the episode 2000, the number of steps to the goal state start to decrease significantly. This shows that the landmark set is unnecessary in advance, and it can be found online to gain learning performance. Although DDCF does not discover all the landmarks, the found ones suffice for improving the learning speed of the baseline algorithms. Moreover, LBG outperforms both A2C and ACKTR that show to be sample-inefficient during learning.

For `4Rooms4Hallways` domain, keeping a memory may be required in order to observe some sort of learning. In Fig. 9, it is shown that keeping the previous observation in the estimated state form ($x_t = o_{t-1}o_t$) can improve the performance of the baseline algorithms, Q(λ) and Sarsa(λ). In this level of state estimation, Fig. 10 shows a similar result. Both baseline algorithms gives a slow learning sign where Sarsa(λ) performs better compared to Q(λ). In contrast, LBG improves both of the underlying algorithms and also outperforms A2C and ACKTR. The number of steps to reach the goal state significantly drops when our guidance method is employed with the agent being directed towards it in the early stages of learning. Moreover, DDCF + LBG has a similar immediate effect on the learning performance after the landmarks are discovered and the guidance begins around episode 2500. This shows that the overall algorithm is also helpful when more complex landmarks are present.

Figure 11 shows that the baseline algorithms cannot learn by using the single observations from the environment ($x_t = o_t$), yet including the actions between the previous and the current observations seems to make the agent solve the

Table 3 Landmark identification performance of DDCF under different learning algorithms. Values are given with their lower and upper bound of confidence intervals

Problem	Q(λ)		Sarsa(λ)	
	Precision	Recall	Precision	Recall
`6Rooms`	1.000 (1.000, 1.000)	0.631 (0.586, 0.674)	1.000 (1.000, 1.000)	0.703 (0.657, 0.743)
`4Rooms4Hallways`	1.000 (1.000, 1.000)	0.402 (0.383, 0.427)	1.000 (1.000, 1.000)	0.412 (0.388, 0.442)
`ToH3`	0.827 (0.798, 0.861)	0.056 (0.053, 0.060)	0.784 (0.754, 0.815)	0.051 (0.047, 0.055)

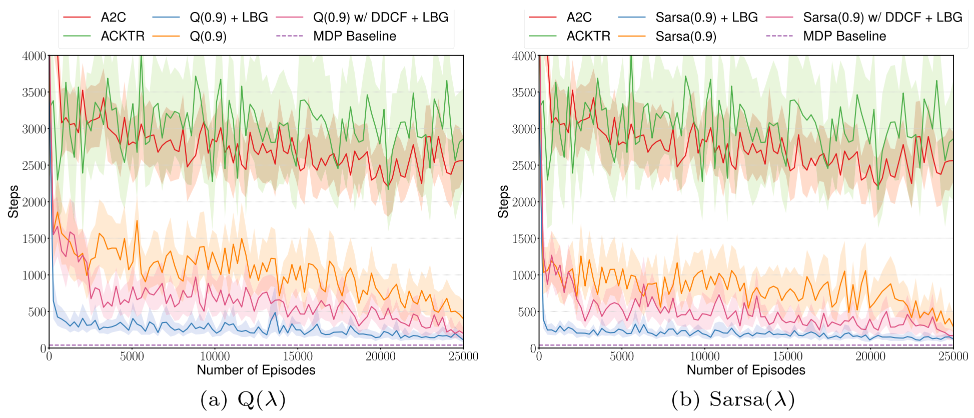


Fig. 8 Average number of steps taken to reach the goal state in `6Rooms` domain where the state estimation has the form of $x_t = o_t$. The dashed line represents the best value from the MDP version of the problem and shaded areas are the 95% bootstrapped confidence intervals

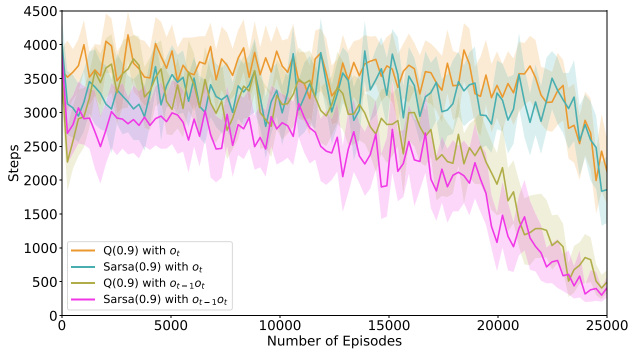


Fig. 9 Average number of steps taken to reach the goal state in 4Rooms4Hallways domain with different state estimation forms. Shaded areas are the 95% bootstrapped confidence intervals

task in ToH3. It can be seen in Fig. 12, all of the algorithms except A2C and ACKTR almost converge to the best value taken from the MDP version of the problem when the state estimation is formed as $x_t = o_{t-1}a_{t-1}o_t$. A2C and ACKTR, on the other hand, struggle to learn, similarly to the other experiments. Again, landmark based guidance led the agent towards the best policy much sooner. The agent with LBG reached the goal state earlier, causing the episode to be much shorter. Although DDCF's coverage on the landmarks is low, its improvement over the baseline algorithms is significant. This proves that not all the landmarks are to be used to achieve good guidance.

Additionally, we have run the final policies learned by different methods greedily on the domains, in order to test the success of the learned policies, with no supporting mechanism. The agent followed a final greedy policy for 50 trials of 5000 steps, and each trial is considered successful if it ends with a goal state. Figure 13 shows the average success percentages of the final policies from 50 experiments for each domain with their corresponding state estimation levels. On one hand, LBG leads to better policies for all the problems where the complete algorithm of DDCF + LBG also improves the underlying learning algorithms. On the other hand, in ToH3 domain, all the algorithms can find successful policies.

4.4 Analysis on λ

The strength of the baseline RL algorithms used in this study comes from the eligibility trace mechanism they incorporate. Thus, the effect of eligibility traces over the learning performance is an important aspect to analyze. λ controls the decay of the eligibility traces, as well as how far back in past the guiding reward can propagate. In this section, the experiments are carried out for Sarsa(λ) and Sarsa(λ) w/ LBG for several λ values. 6Rooms domain is selected for testing since it has distant landmarks so that the length of the trace can provide more advantage on the learning speed.

Table 4 shows the results averaged over 50 experiments. As expected, the learning performance of Sarsa(λ) decreases with the decrease of λ. Leaving a trace over the past transitions makes the agent converge to a good policy much faster. Alternatively, we can see that additional guidance provided by LBG can have a better effect when λ is high. When the agent reaches a landmark, LBG rewards the actions taken

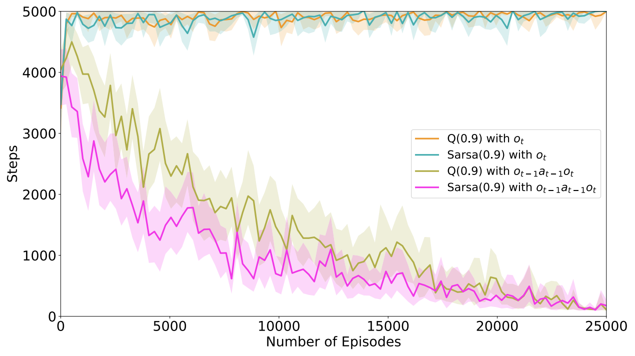


Fig. 11 Average number of steps taken to reach the goal state in ToH3 domain with different state estimation forms. Shaded areas are the 95% bootstrapped confidence intervals

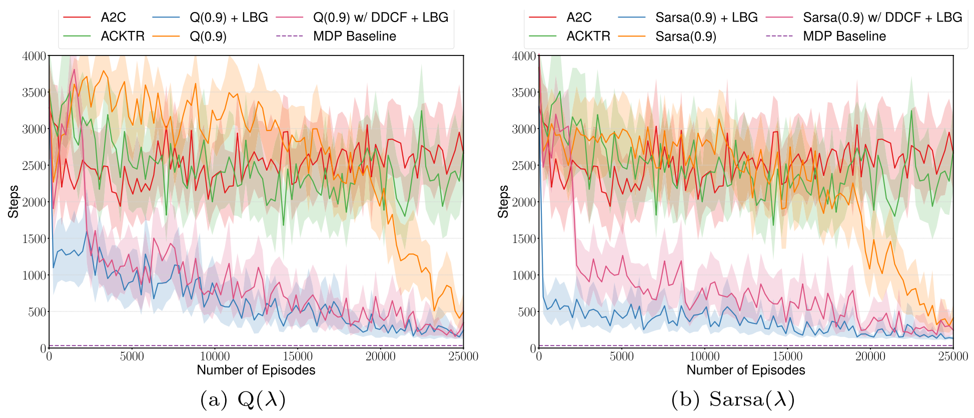


Fig. 10 Average number of steps taken to reach the goal state in 4Rooms4Hallways domain where the state estimation has the form of $x_t = o_{t-1}o_t$. The dashed line represents the best value from the MDP version of the problem and shaded areas are the 95% bootstrapped confidence intervals

if the abstract transition lands in a landmark with a higher value. When λ is high, the set of past estimated states that are affected by this guiding reward becomes larger. We can conclude that LBG further strengthens the benefits of invoking an eligibility trace mechanism.

4.5 Analysis on guidance with subsets of landmarks

We further analyzed the learning performance of Landmark Based Guidance when only a random subset of the whole landmark set is provided to the agent. This way, we can demonstrate the effect of the landmark set on the guidance method.

At the beginning of each experiment, we took a random subset of landmarks and ran LBG with this subset. We experimented on 6 different versions of Sarsa(λ) algorithm guided by 0%, 20%, 40%, 60%, 80% 100% of landmarks. In the experiments, we used `4Rooms4Hallways` domain with the state estimation $x_t = o_{t-1}o_t$ where there are 21 landmarks in this form (Table 2).

Figure 14 shows the number of steps taken to the goal averaged over 50 experiments. As expected, knowing a bigger subset of the landmarks leads to a better learning performance since the agent can depend on more landmarks and rewarded more frequently. Additionally, we compared these results to the performance of DDCF + LBG where the method scored 0.41 recall, which is nearly 40% of the landmarks. It can be seen in Fig. 15 that the performance of DDCF + LBG matches to having a random subset of 80%, even though it had a late start on guidance due to the episode threshold. This supports our claim that DDCF is more focused on landmarks that are useful for learning the task at hand.

5 Conclusion

In this study, a landmark based guidance (LBG) approach is applied to the partially observable problem setting, where the states are hidden from the agent. The proposed method makes use of landmarks to baseline a potential function for introducing additional rewards. LBG approach argues that a RL agent can achieve a better solution by means of additional internal rewards upon completion of a transition

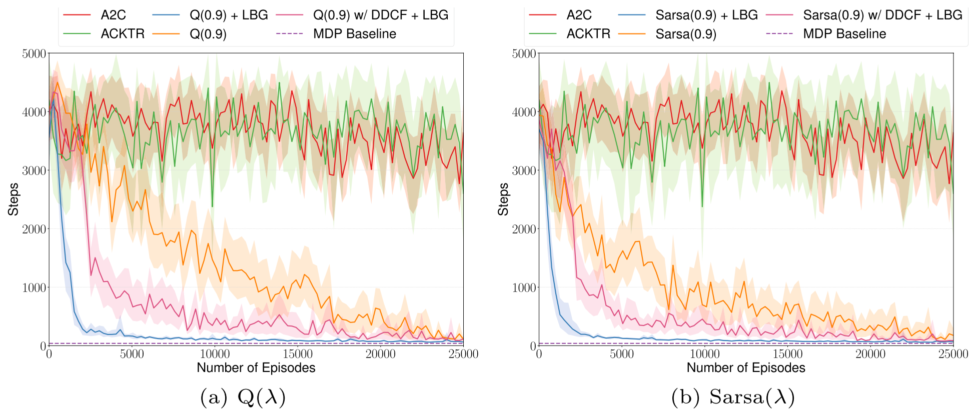


Fig. 12 Average number of steps taken to reach the goal state in `ToH3` domain where the state estimation has the form of $x_t = o_{t-1}a_{t-1}o_t$. The dashed line represents the best value from the MDP version of the problem and shaded areas are the 95% bootstrapped confidence intervals

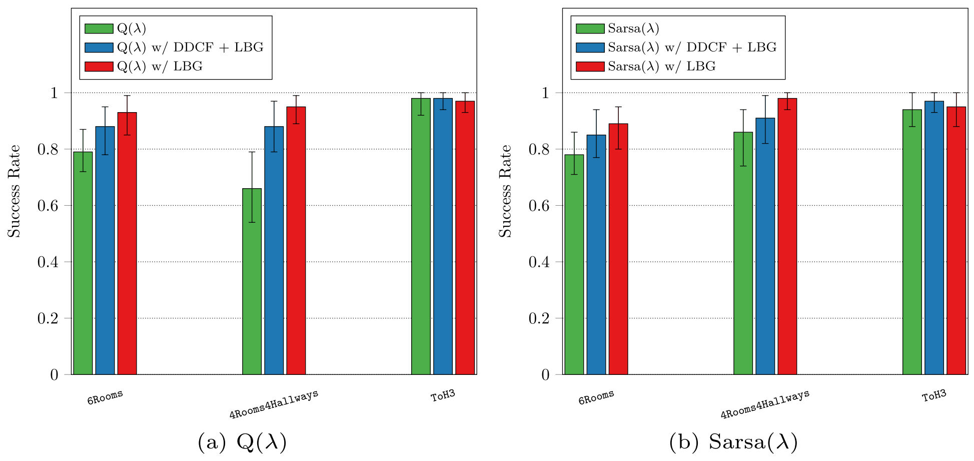


Fig. 13 The success rate of the final greedy policies to reach the goal states in all of the domains

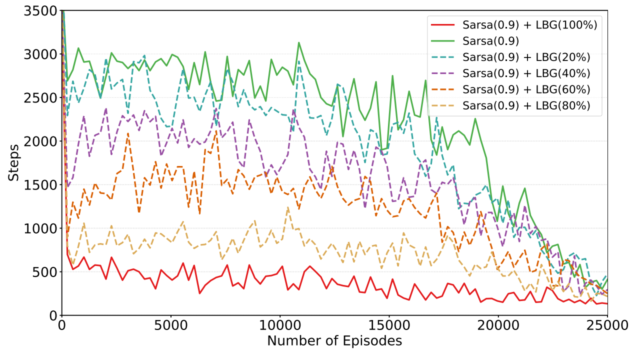


Fig. 14 Average number of steps to goal in `4Rooms4Hallways` when LBG is fed with the different subsets of the landmark set. The percentage of the provided subsets to LBG are given in parentheses, Sarsa(0.9) is given as a baseline without LBG and the confidence intervals are omitted for better view

Table 4 The number of steps to reach the goal state averaged over episodes for various λ values in `6Rooms` domain. Gain is calculated as the ratio of the difference in the number of steps over the number of steps taken by Sarsa(λ)

λ	Sarsa(λ)	Sarsa(λ) w/ LBG	Gain
0	1969.03	1601.11	18.6%
0.1	1935.62	1458.96	24.6%
0.3	1727.27	1106.59	35.9%
0.5	1457.33	710.56	51.2%
0.7	1186.17	404.71	65.8%
0.9	804.95	202.86	74.7%

among two landmarks at the abstract level. Experiments on several problems show LBG can significantly improve the learning performance of well known off-policy and on-policy learning algorithms, like Q(λ) and Sarsa(λ). LBG can be further coupled with DDCF to discover the landmarks during learning, removing the necessity to provide the landmark set beforehand. DDCF+LBG not only outperformed the literature baselines, but also showed similar performance with the pre-defined landmarks setting, in terms of learning speed.

The study shows that landmarks usually exist and can be identified at different levels of state estimation forms. Their natural presence can be utilized by guiding with additional rewards. Both the landmarks and their potentials can be found online, and DDCF+LBG is an algorithm that combines discovery and usage of landmarks. On one hand, an estimated state set without any landmarks may be considered nondistinctive since having no landmarks shows that the agent's internal representations are still ambiguous, mapping to multiple states with possibly different optimal actions. Such a state estimation approach must be improved to clear out ambiguity in order for DDCF+LBG to identify and utilize them to guide the agent. On the other hand, in the case where the state estimation successfully identifies each state of the domain and each estimated state is a landmark, DDCF will focus on key landmarks, acting as bottlenecks, and those landmarks will guide the agent with LBG. Although finding such state estimation methods is quite challenging, our approach can still speed up the learning process.

As the space of estimated states expands, the number of natural landmarks also increase. This allows DDCF+LBG to scale up easily since DDCF will identify the most useful landmarks and LBG will help the agent towards reaching a goal state in a bigger estimated state space.

As a follow-up work to Landmark Based Guidance, one can experiment with algorithms that devise their state estimations during learning, rather than having a fixed form at the beginning. Methods like USM extend the memory whenever necessary, causing a set of estimated states of different sizes. The proposed framework of DDCF+LBG is still a fit candidate to work under those circumstances. DDCF can pick the landmarks among the set of estimated states, and LBG can fuse them to provide guiding rewards.

Our study on tabular discrete Reinforcement Learning with hidden states can be extended to continuous environments after a proper discretization process is applied to clearly describe the estimated states. This way, DDCF+LBG can be couple with state-of-the-art deep reinforcement learning methods, which we leave as a future work.

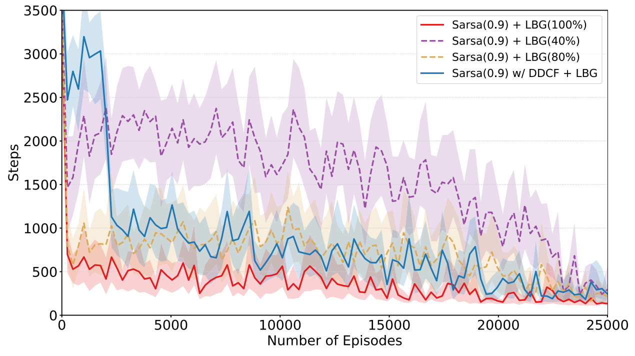


Fig. 15 Comparison of DDCF + LBG to LBG with the subsets of landmarks in terms of average number of steps to goal in `4Rooms4Hallways`. The percentage of the provided subsets are given in parentheses and shaded areas are the 95% bootstrapped confidence intervals

Acknowledgements Authors would like to thank Hüseyin Aydın for his support.

References

1. Asmuth J, Littman ML, Zinkov R (2008) Potential-based Shaping in Model-based Reinforcement Learning. In: Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, pp 604–609

2. Astrom KJ (1965) Optimal control of Markov processes with incomplete state information. J Math Anal Appl 10(1):174–205. https://doi.org/10.1016/0022-247X(65)90154-X
3. Aydın H, Çilden E, Polat F (2022) Using chains of bottleneck transitions to decompose and solve reinforcement learning tasks with hidden states. Future Generation Comput Syst 133:153–168. https://doi.org/10.1016/j.future.2022.03.016
4. Babes M, De Cote EM, Littman ML (2008) Social reward shaping in the prisoner's dilemma. In: Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems, pp 1389–1392
5. Bradtke SJ, Duff MO (1994) Reinforcement learning methods for continuous-time Markov decision problems. Adv Neural Info Process Syst 7:393–400
6. Cassandra AR (1998) Exact and approximate algorithms for partially observable markov decision processes. Dissertation, Brown University
7. Çilden E, Polat F (2015) Toward generalization of automated temporal abstraction to partially observable reinforcement learning. IEEE Trans Cybern 45(8):1414–1425. https://doi.org/10.1109/TCYB.2014.2352038
8. Demir A, Çilden E, Polat F (2019) Automatic landmark discovery for learning agents under partial observability. Knowl Eng Rev 34:E11. https://doi.org/10.1017/S026988891900002X
9. Demir A, Çilden E, Polat F (2019b) Landmark based reward shaping in reinforcement learning with hidden states. In: Proceedings of the 18th International Conference on Autonomous Agents and Multi Agent Systems, pp 1922–1924
10. Devlin S, Kudenko D (2011) Theoretical considerations of potential-based reward shaping for multi-agent systems. In: The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, pp 225–232
11. Devlin S, Kudenko D (2016) Plan-based reward shaping for multi-agent reinforcement learning. Knowl Eng Rev 31(1):44–58. https://doi.org/10.1017/S0269888915000181
12. Devlin S, Kudenko D, Grześ M (2011) An empirical study of potential-based reward shaping and advice in complex, multiagent systems. Adv Complex Syst 14(02):251–278. https://doi.org/10.1142/S0219525911002998
13. Devlin SM, Kudenko D (2012) Dynamic potential-based reward shaping. In: Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, pp 433–440
14. Dong Y, Tang X, Yuan Y (2020) Principled reward shaping for reinforcement learning via Lyapunov stability theory. Neurocomputing 393:83–90. https://doi.org/10.1016/j.neucom.2020.02.008
15. Duff MO (2002) Optimal learning: Computational procedures for Bayes-adaptive Markov decision processes. Dissertation, University of Massachusetts at Amherst
16. Eck A, Soh LK, Devlin S et al (2016) Potential-based reward shaping for finite horizon online POMDP planning. Auton Agents Multi-Agent Syst 30(3):403–445. https://doi.org/10.1007/s10458-015-9292-6
17. Efthymiadis K, Devlin S, Kudenko D (2016) Overcoming incorrect knowledge in plan-based reward shaping. Knowl Eng Rev 31(1):31–43. https://doi.org/10.1017/S026988891500017X
18. Gao Y, Toni F (2015) Potential based reward shapings for hierarchical reinforcement learning. In: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, pp 3504–3510
19. Grzes M (2010) Improving exploration in reinforcement learning through domain knowledge and parameter analysis. Dissertation, University of York
20. Grzes M (2017) Reward shaping in episodic reinforcement learning. In: Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, pp 565–573
21. Grzes M, Kudenko D (2008) Plan-based reward shaping for reinforcement learning. In: 2008 4th International IEEE Conference Intelligent Systems, IEEE, pp 10–22, https://doi.org/10.1109/IS.2008.4670492
22. Grzes M, Kudenko D (2009) Theoretical and empirical analysis of reward shaping in reinforcement learning. In: 2009 International Conference on Machine Learning and Applications, pp 337–344, https://doi.org/10.1109/ICMLA.2009.33
23. Grzes M, Kudenko D (2010) Online learning of shaping rewards in reinforcement learning. Neural Netw 23(4):541–550. https://doi.org/10.1016/j.neunet.2010.01.001
24. Henderson P, Islam R, Bachman P, et al (2018) Deep reinforcement learning that matters. In: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, pp 3207–3214, https://doi.org/10.1609/aaai.v32i1.11694
25. Hill A, Raffin A, Ernestus M, et al (2018) Stable baselines. https://github.com/hill-a/stable-baselines
26. Hinz AM (1989) The tower of Hanoi. L'Enseignement Mathématique 35:289–321. https://doi.org/10.5169/seals-57378
27. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735
28. Igl M, Zintgraf L, Le TA, et al (2018) Deep variational reinforcement learning for pomdps. In: Proceedings of the 35th International Conference on Machine Learning, pp 2117–2126
29. James MR, Singh SP (2009) Sarsalandmark: an algorithm for learning in pomdps with landmarks. In: Proceedings of The 8th International Joint Conference on Autonomous Agents and Multiagent Systems - vol 1, pp 585–591
30. James MR, Wolfe B, Singh SP (2005) Combining memory and landmarks with predictive state representations. In: Proceedings of The 19th International Joint Conference on Artificial intelligence, pp 734–739
31. Jaulmes R, Pineau J, Precup D (2005) Active learning in partially observable markov decision processes. In: 16th European Conference on Machine Learning Proceedings, LNCS, vol 3720, pp 601–608, https://doi.org/10.1007/11564096_59
32. Kaelbling LP, Littman ML, Moore AP (1996) Reinforcement learning: a survey. J Artif Intell Res 4:237–285. https://doi.org/10.1613/jair.301
33. Kaelbling LP, Littman ML, Cassandra AR (1998) Planning and acting in partially observable stochastic domains. Artif Intell 101(1–2):99–134. https://doi.org/10.1016/S0004-3702(98)00023-X
34. Li R, Cai Z, Huang T et al (2021) Anchor: The achieved goal to replace the subgoal for hierarchical reinforcement learning. Knowl-Based Syst 225(107):128. https://doi.org/10.1016/j.knosys.2021.107128
35. Lin LJ, Mitchell TM (1992) Memory approaches to reinforcement learning in non-markovian domains. Technical Report CMU-CS-92-138, Carnegie Mellon University
36. Littman ML (1994) Memoryless policies: theoretical limitations and practical results. In: From Animals to Animats 3: Proceedings of the third International Conference on Simulation of Adaptive Behavior, pp 238–245
37. Loch J, Singh SP (1998) Using eligibility traces to find the best memoryless policy in partially observable markov decision processes. In: Proceedings of the Fifteenth International Conference on Machine Learning, pp 323–331
38. Lu X, Schwartz HM, Givigi SN (2011) Policy invariance under reward transformations for general-sum stochastic games. J Artif Intell Res 41:397–406. https://doi.org/10.1613/jair.3384

39. Marom O, Rosman B (2018) Belief reward shaping in reinforcement learning. In: Proceedings of The Thirty-Second AAAI Conference on Artificial Intelligence, pp 3762–3769, https://doi.org/10.1609/aaai.v32i1.11741
40. Maron O, Lozano-Pérez T (1998) A framework for multiple-instance learning. Adv Neural Info Process Syst 10:570–576
41. Marthi B (2007) Automatic shaping and decomposition of reward functions. In: Proceedings of the 24th International Conference on Machine Learning, pp 601–608, https://doi.org/10.1145/1273496.1273572
42. Martinez C, Ramasso E, Perrin G et al (2020) Adaptive early classification of temporal sequences using deep reinforcement learning. Knowl-Based Syst 190(105):290. https://doi.org/10.1016/j.knosys.2019.105290
43. McCallum A (1996) Reinforcement learning with selective perception and hidden state. Dissertation, University of Rochester
44. McGovern A, Barto AG (2001) Automatic discovery of subgoals in reinforcement learning using diverse density. In: Proceedings of the Eighteenth International Conference on Machine Learning, pp 361–368
45. Menache I, Mannor S, Shimkin N (2002) Q-Cut—dynamic discovery of sub -goals in reinforcement learning. In: 13th European Conference on Machine Learning Proceedings, LNCS, vol 2430, pp 295–306, https://doi.org/10.1007/3-540-36755-1_25
46. Mnih V, Badia AP, Mirza M, et al (2016) Asynchronous methods for deep reinforcement learning. In: Proceedings of The 33rd International Conference on Machine Learning, pp 1928–1937
47. Ng AY, Harada D, Russell SJ (1999) Policy invariance under reward transformations: Theory and application to reward shaping. In: Proceedings of the Sixteenth International Conference on Machine Learning, pp 278–287
48. Ross S, Chaib-draa B, Pineau J (2007) Bayes-adaptive POMDPs. Adv Neural Info Process Syst 20:1225–1232
49. Ross S, Pineau J, Chaib-draa B et al (2011) A Bayesian approach for learning and planning in partially observable Markov decision processes. J Mach Learn Res 12(48):1729–1770
50. Schulman J, Wolski F, Dhariwal P, et al (2017) Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347
51. Singh SP, Jaakkola T, Jordan MI (1994) Learning without state-estimation in partially observable Markovian decision processes. Mach Learn Proc 1994:284–292. https://doi.org/10.1016/c2009-0-27542-8
52. Sutton RS (1988) Learning to predict by the methods of temporal differences. Mach Learn 3:9–44. https://doi.org/10.1007/BF00115009
53. Sutton RS, Barto AG (2018) Reinforcement learning: an introduction, 2nd edn. MIT Press
54. Sutton RS, Precup D, Singh S (1999) Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. Artif Intell 112(1–2):181–211. https://doi.org/10.1016/S0004-3702(99)00052-1
55. Toro Icarte R, Waldie E, Klassen T et al (2019) Learning reward machines for partially observable reinforcement learning. Adv Neural Info Process Syst 32:15523–15534
56. Vlassis N, Ghavamzadeh M, Mannor S, et al (2012) Bayesian reinforcement learning. In: Reinforcement Learning: State-of-the-Art. Springer Berlin Heidelberg, pp 359–386, https://doi.org/10.1007/978-3-642-27645-3_11
57. Wang Y, He H, Tan X (2020) Truly proximal policy optimization. In: Proceedings of The 35th Uncertainty in Artificial Intelligence Conference, pp 113–122
58. Watkins C (1989) Learning from delayed rewards. Dissertation, Cambridge University
59. Whitehead SD, Ballard DH (1991) Learning to perceive and act by trial and error. Mach Learn 7(1):45–83. https://doi.org/10.1023/A:1022619109594
60. Wiewiora E (2003) Potential-based shaping and q-value initialization are equivalent. J Artif Intell Res 19:205–208. https://doi.org/10.1613/jair.1190
61. Wu Y, Mansimov E, Grosse RB et al (2017) Scalable trust-region method for deep reinforcement learning using Kronecker-factored approximation. Adv Neural Info Process Syst 30:5285–5294
62. Zhang M, McCarthy Z, Finn C, et al (2016) Learning deep neural network policies with continuous memory states. In: 2016 IEEE International Conference on Robotics and Automation, pp 520–527, https://doi.org/10.1109/ICRA.2016.7487174