Potential Based Reward Shaping for Hierarchical Reinforcement Learning

Yang Gao, Francesca Toni
Department of Computing, Imperial College London
{y.gao11,f.toni}@imperial.ac.uk

Abstract

Hierarchical Reinforcement Learning (HRL) outperforms many 'flat' Reinforcement Learning (RL) algorithms in some application domains. However, HRL may need longer time to obtain the optimal policy because of its large action space. Potential Based Reward Shaping (PBRS) has been widely used to incorporate heuristics into flat RL algorithms so as to reduce their exploration. In this paper, we investigate the integration of PBRS and HRL, and propose a new algorithm: *PBRS-MAXQ-0*. We prove that under certain conditions, PBRS-MAXQ-0 is guaranteed to converge. Empirical results show that PBRS-MAXQ-0 significantly outperforms MAXQ-0 given good heuristics, and can converge even when given misleading heuristics.

1 Introduction

Hierarchical Reinforcement Learning (HRL) algorithms incorporate hierarchical planning into classic Reinforcement Learning (RL). HRL algorithms exploit *temporal abstraction*, where decisions are not required at each step, but rather invoke the execution of temporally extended activities which follow their own policies until termination, to combat the *curse of dimensionality* (the exponential growth of the computational requirement for obtaining the optimal polices with the size of state space) faced by RL algorithms [2].

MAXQ [6] is one of the most widely used HRL algorithms. Unlike some other HRL algorithms, e.g. *Options* [17] and *Hierarchies of Abstract Machines* [16], which treat the original problem as a whole, MAXQ decomposes the whole problem into a hierarchy of sub-problems, and solves the whole problem by invoking solutions for sub-problems. By doing so, solutions of multiple sub-problems can be learnt simultaneously [2]. Although HRL algorithms facilitate people to give high-level domain knowledge to instruct RL, the knowledge that can be encoded in HRL is 'coarse-grained', in the sense that very detailed instructions can hardly be represented in the form of sub-task hierarchies (see Section 6 for details). Furthermore, because MAXQ allows a sub-problem to invoke some other sub-problems' solutions in any order and for any number of times, considerable exploration is needed to learn the best order to invoke available activities even in relatively small-scale problems [6].

Incorporating prior knowledge into RL algorithms has proven be to effective in improving RL's convergence speed. There have been multiple approaches for incorporating prior knowledge about the dynamic of the environment into MAXQ [11; 4]. However, in some problems, people may not have this prior knowledge but, instead, can provide some heuristics about which actions should (not) be performed in certain states [18]. These instructions have proven to be very useful in reducing exploration in *flat*[1] RL algorithms [15; 18]. However, to the best of our knowledge, there exist no techniques for incorporating heuristics into MAXQ.

In this paper, we propose an algorithm that incorporates *Potential Based Reward Shaping* (PBRS) into MAXQ. PBRS has been widely used in flat RL algorithms as an effective technique to reduce exploration and to accelerate learning [15; 18; 1]. PBRS gives extra rewards to actions recommended by domain experts, so as to encourage the agent to choose these actions. By doing so, PBRS can guide the exploration of RL algorithms without altering their original optimal policies. We propose a novel algorithm: *PBRS-MAXQ-0*, and prove that it is guaranteed to converge under certain conditions, regardless of the extra rewards being used. Empirically, we show that: (i) given reasonable heuristics, our approach can converge significantly faster than standard MAXQ-0 algorithm, and has competitive performances compared to other state-of-the-art MAXQ-based techniques; and (ii) even when given misleading heuristics, our algorithm is still able to converge (after a sufficiently long time of learning).

2 Related Work

Some work has been devoted to incorporating MAXQ with other RL techniques so as to improve its convergence speed. Jong and Stone [11] incorporated R-MAX [3] with MAXQ and proposed R-MAXQ. For each sub-task (either composite or primitive), a model is built and updated in the R-MAX style. The use of R-MAX guarantees that R-MAXQ can converge in finite-time. R-MAXQ provides a method to more systemically explore polices, but does not allow for using prior knowledge to reduce exploration. Cao and Ray [4]

[1] A RL algorithm is flat iff it is not a HRL algorithm.

proposed an algorithm for incorporating Bayesian priors into MAXQ. They defined priors on primitive actions' models and on task pseudo rewards, but did not construct models for composite sub-tasks (the purpose is to reduce the complexity of the model). Empirical results showed that by using 'good' priors (i.e. the posteriors learnt in previous experiments), their approach can converge faster than MAXQ-0. However, both of these approach do not allow for incorporating heuristics into MAXQ, and both are much more computationally expensive than MAXQ-0 (see table 1 in [4]).

Our work assumes that the heuristics and their extra rewards are given a priori by domain experts. Research has been devoted to obtaining the heuristics and their extra rewards in PBRS, e.g. [10; 8; 7]. We also assume that the task hierarchy is given. Other work has explored learning MAXQ hierarchies in different settings, e.g. [13].

3 Background

We first introduce the Semi-Markov Decision Process (SMDP), and then provide a brief description of MAXQ and PBRS, the techniques we integrate.

3.1 Semi-Markov Decision Process

Before introducing SMDP, we describe the Markov Decision Process (MDP), as the two are closely related. A MDP is a tuple (S, A, P, R, γ), where S is the set of states, A is the set of actions, $P(s'|s,a)$ is the *transition probability* of moving from state s to s' by performing action a, $R(s'|s,a)$ gives the immediate reward received when action a is executed in state s, moving to state s', and $\gamma \in [0,1]$ is the discount factor. The goal of planning in a MDP is to find a policy $\pi : S \to A$, specifying for each state the action to take that maximises the discounted sum of future rewards. The *value function* $V^\pi(s)$ represents the expected discounted sum of rewards that will be received by following π starting in state s:

$$V^\pi(s) = E[r_t + \gamma r_{t+1} + \gamma^2 r_{t+2} \cdots | s_t = s, \pi],$$

where r_t is the immediate reward received in time-step t. The value function satisfies the *Bellman equation* for a fixed policy π:

$$V^\pi(s) = \sum_{s'} P(s'|s,\pi(s))[R(s'|s,\pi(s)) + \gamma V^\pi(s')].$$

For an optimal policy π^*, the value function satisfies:

$$V^*(s) = \max_a \sum_{s'} P(s'|s,a)[R(s'|s,a) + \gamma V^*(s')].$$

SMDP is a generalisation of MDP in which the actions can take variable amounts of time to complete. We restrict our attention to discrete-time SMDP, and let the random variable τ denote the number of time steps action a takes in state s. The transition probability function P is extended to become the joint distribution of the result state s' and the number of time steps τ when action a is performed in state s: $P(s', \tau|s, a)$. Similarly, the reward function is extended to $R(s', \tau|s, a)$. Therefore, in SMDP, the value function is:

$$V^\pi(s) = \sum_{s',\tau} P(s', \tau|s, \pi(s))[R(s', \tau|s, \pi(s)) + \gamma^\tau V^\pi(s')],$$

and the optimal value function V^* satisfies:

$$V^*(s) = \max_a \sum_{s',\tau} P(s', \tau|s, a)[R(s', \tau|s, a) + \gamma^\tau V^*(s')].$$

3.2 The MAXQ-0 Algorithm

MAXQ decomposes the overall value function for a policy into a collection of value functions for individual sub-tasks. A core MDP M is decomposed into a set of sub-tasks $\{M_0, M_1, \cdots, M_n\}$, forming a hierarchy with M_0 the root sub-task, i.e. solving M_0 solves M. Actions taken to solve M_0 may be *primitive* actions or policies that solve other *composite* sub-tasks, which can in turn invoke primitive actions or policies. The sub-tasks and primitive actions into which a sub-task M_i is decomposed are called the *children* of M_i, and M_i is called the *parent* of its children. The children sub-tasks of a composite sub-task are allowed to be invoked in any order and any number of times. In most works on MAXQ (see, e.g. [6; 12]), primitive sub-tasks take exactly one time step, so the time spent in finishing a composite sub-task M_i depends on the number of primitive sub-tasks performed in M_i. The structure of the MAXQ decomposition is concisely summarised by the *task graph* [6].

A composite sub-task M_i consists of three components: i) the *sub-task policy*, π_i, specifying which children sub-task of M_i should be invoked in a specific state; ii) a *termination predicate*, partitioning the state set of the core MDP into S_i, the set of states that M_i can execute, and T_i, the set of states where M_i terminates; and iii) a *pseudo-reward function*, assigning reward values to the states in T_i. In this work, for simplicity, me let all pseudo-rewards be 0, but in Section 4.2 we will describe that our algorithm can be extended to the MAXQ algorithms with non-zero pseudo rewards.

Given a hierarchical decomposition $\{M_0, \cdots, M_n\}$ of M, a *hierarchical policy* is $\pi = \{\pi_0, \cdots, \pi_n\}$, with π_i the policy of M_i. Let $V^\pi(i,s)$ denote the cumulative reward of performing sub-task M_i in state s following π, where i is a shorthand for M_i. $V^\pi(i,s)$ is formally defined as follows:

$$V^\pi(i,s) = \begin{cases} \sum_{s'} P(s'|s,i)R(s'|s,i) & i \text{ primitive}, \\ Q^\pi(i,s,\pi_i(s)) & i \text{ composite}. \end{cases} \quad (1)$$

When M_i is a primitive action, $V^\pi(i,s)$ is the expected one-step reward of performing action M_i in state s; otherwise, $V^\pi(i,s)$ is $Q^\pi(i,s,\pi_i(s))$, which is the expected return for M_i of performing sub-task $M_{\pi_i(s)}$ in state s and then following π until M_i terminates, such that:

$$Q^\pi(i,s,\pi_i(s)) = V^\pi(\pi_i(s), s) + C^\pi(i,s,\pi_i(s))$$

where the *completion function* $C^\pi(i,s,\pi_i(s))$, defined as

$$C^\pi(i,s,\pi_i(s)) = \sum_{s',\tau} P_i^\pi(s', \tau|s, \pi_i(s))\gamma^\tau V^\pi(i, s')$$

gives the expected return for completing sub-task M_i after sub-task $M_{\pi_i(s)}$ terminates. P_i^π is the transition probability in M_i given a policy π. The three equations above provide a recursive way to write the value function given a hierarchical policy. The greedy policy with respect to $Q^\pi(i,s,a)$ is the optimal policy for the SMDP corresponding to M_i: $\pi_i^*(s) = \text{argmax}_a\, Q(i,s,a)$.

Based on this recursive definition, Dietterich [6] proposes the *MAXQ-0* algorithm to learn the hierarchical policy from sample trajectories. MAXQ-0 calls itself recursively to descend through the hierarchy, eventually executing primitive actions. When it returns from each call, it updates the completion function for the appropriate sub-task, with discounting determined by the number of primitive actions executed. Dietterich proves that by using an *ordered greedy in the limit with infinite exploration* (OGLIE) action-selection policy, MAXQ-0 is guaranteed to converge to the (hierarchical) optimal policy.

3.3 Potential-Based Reward Shaping

Potential-Based Reward Shaping (PBRS) is proposed by Ng et al. [15] as the difference of some potential function Φ over the current state s and the next state s'. By integrating PBRS into MDP, the value function following policy π becomes:

$$\hat{V}^{\pi}(s) = E[\sum_{t}^{\infty} \gamma^t (r_t + F(s_t, s_{t+1}) | s_t = s, \pi], \quad (2)$$

where $F(s_t, s_{t+1}) = \gamma\Phi(s_{t+1}) - \Phi(s_t)$. We can see that most potential values in Eq. (2) cancel each other out, and the only remaining potential value is $-\Phi(s)$, i.e. $V^{\pi}(s) - \Phi(s) = \hat{V}^{\pi}(s)$. Ng et al. have proven that the optimal policy wrt. a PBRS-augmented MDP is also optimal wrt. the original MDP. However, in PBRS, since the potential values are only based on states, these values cannot provide instructions on which actions are more promising in some state. To tackle this problem, Wiewiora et al. [18] extended classical PBRS to the case of shaping functions based on both states and actions: $\Phi(s, a)$, and propose the *Look-ahead Advice* (LA).[2] They proved that by using LA, arbitrary potential values can be incorporated into RL without altering its optimal policy. In HRL, however, potential values that only consider action a and state s are not sufficient: within different parent sub-task i, the goodness of performing a at s can be different. Hence, in HRL, we need potential values that also involve the parent sub-task, so as to give more accurate instructions to the action-selection process. This work is an extension of LA, such that potential values are in the form of $\Phi(i, s, a)$, where i is a composite sub-task, s is a state and a is a child of i.

4 MAXQ with PBRS

In this section, we present our main contribution: the *PBRS-MAXQ-0* algorithm. We first give rules for integrating potential values into V and C values, and describe the relation between the PBRS-augmented V and C functions and the original V and C functions. Then we present PBRS-MAXQ-0 and prove its soundness. Due to the space limit, we only sketch the proof of one theorem[3]. Throughout this section, let $H_M = \{M_0, \cdots, M_n\}$ be a hierarchical decomposition of the core MDP M, and let i range over all sub-tasks in H_M.

[2]They have also proposed a technique called *look-back advice*, but this is not theoretically sound. So here we only focus on LA.

[3]The full proofs can be found at www.doc.ic.ac.uk/~yg211/sup_mat_ijcai.pdf.

4.1 Integrating Potential Values into MAXQ

Recall that, in classic PBRS, potential values are integrated into value functions in a form of the ensuing state's potential value minus the current state's potential value (see Eq. (2)). In HRL, we integrate the potential values into value functions in a similar way. To be more specific, given a fixed hierarchical policy $\pi = \{\pi_0, \cdots, \pi_n\}$, let $\tilde{Q}^{\pi}(i, s, a) = \tilde{V}^{\pi}(a, s) + \tilde{C}^{\pi}(i, s, a)$ where

$$\tilde{V}^{\pi}(i, s) = \begin{cases} \sum_{s'}[P(s'|s, i)R(s'|s, i)] & i \text{ primitive}, \\ \tilde{Q}^{\pi}(i, s, \pi_i(s)) & i \text{ composite}. \end{cases} \quad (3)$$

When M_i does not terminate at state s:

$$\tilde{C}^{\pi}(i, s, a) = \sum_{s', \tau} P_i^{\pi}(s', \tau | s, a) \cdot 1_{S_i}(s') \cdot \gamma^{\tau}[V^{\pi}(\pi_i(s'), s') + \tilde{C}^{\pi}(i, s', \pi_i(s')) + \Phi(i, s', \pi_i(s'))] - \Phi(i, s, a), \quad (4)$$

where V^{π} is the function defined in Eq. (1); otherwise (i.e. when M_i terminates at state s), $\tilde{C}^{\pi}(i, s, a) = 0$, where a can be any child of i. Note that $1_{S_i}(s')$ in Eq. (4) is an indicator function, and S_i is the set of states where sub-task M_i does not terminate (see Section 3.2). From Eq. (4), we can see that, in HRL, potential values are directly integrated into C values, because composite sub-tasks' V values in HRL can be recursively constructed from C values (see Eq. (3)).

We refer to $\tilde{Q}^{\pi}$, $\tilde{V}^{\pi}$ and $\tilde{C}^{\pi}$ as the *PBRS-augmented functions* for MAXQ. Now we show that the original C^{π} values can be easily obtained from PBRS-augmented $\tilde{C}^{\pi}$ values.

Proposition 1. Given a MAXQ decomposition H_M and its fixed hierarchical policy $\pi = \{\pi_0, \cdots, \pi_n\}$, for any composite sub-task M_i and any state s, $\tilde{C}^{\pi}(i, s, \pi_i(s)) = C^{\pi}(i, s, \pi_i(s)) - \Phi(i, s, \pi_i(s))$.

We can see that the relation between $\tilde{C}^{\pi}$ and C^{π} is similar to the relation between $\hat{V}^{\pi}$ and V^{π} given in Section 3.3. Since $\tilde{V}^{\pi}$ values can be constructed from $\tilde{C}^{\pi}$ values, given Proposition 1, we can obtain the relation between V^{π} and $\tilde{V}^{\pi}$ as follows:

Proposition 2. Given a MAXQ decomposition H_M and its fixed hierarchical policy $\pi = \{\pi_0, \cdots, \pi_n\}$, for any sub-task $M_{a^0}, 0 \leq a^0 \leq n$, and any state s:

- if M_{a^0} is a primitive action, $\tilde{V}^{\pi}(a^0, s) = V^{\pi}(a^0, s)$;
- if M_{a^0} is a composite sub-task, suppose in a sample trajectory following π, $\pi_{a^0}(s) = a^1, \cdots, \pi_{a^{m-1}}(s) = a^m, m > 0$, and a^m is a primitive action, then $\tilde{V}^{\pi}(a^0, s) = V^{\pi}(a^0, s) - \sum_{j=1}^{m} \Phi(a^{j-1}, s, a^j)$.

Proposition 2 suggests that V^{π} values can be obtained by adding $\tilde{V}^{\pi}$ and the sum of all potential values in a sample trajectory. Next, we will propose an algorithm which not only naturally integrates potential values into MAXQ-0, but is also guaranteed to converge to the optimal policy.

4.2 The PBRS-MAXQ-0 Algorithm

The PBRS-MAXQ-0 algorithm is given in Alg. 1. Here, $\alpha \in \mathbb{R}, \alpha \in [0, 1]$ is the learning rate parameter. Note that all V values appearing in Alg. 1 are the original V values (Eq. (1)). Also, function **Evaluate** returns two values, and in line

28, we only use its first returned value. Note that $\tilde{C}$ values are updated (line 15) according to Eq. (4), and we reconstruct the original C values from $\tilde{C}$ values (line 30) according to Proposition 1; as a result, we can see that the first returned value of function **Evaluate** is the original V value (line 31), according to Proposition 2.

Algorithm 1 The PBRS-MAXQ-0 algorithm.

```
1: function PBRS-MAXQ-0(Sub-task i, State s)
2: Seq := <> /*initialise seq as an empty list*/
3: if i is a primitive action then
4:    execute i, receive reward r and observe next state s'
5:    V_{t+1}(i,s) := (1 − α_t(i)) · V_t(i,s) + α_t(i) · r
6:    push s onto the beginning of seq
7: else
8:    while sub-task i does not terminate at state s do
9:       choose an action a according to a BOGLIE policy
10:      ChildSeq := PBRS-MAXQ-0(a, s)
11:      observe next state s'
12:      V_t(a*, s'), a* := Evaluate(i, s')
13:      N := 1
14:      for each s in ChildSeq do
15:         C̃_{t+1}(i,s,a) := (1 − α_t(i)) C̃_t(i,s,a) + α_t(i){γ^N [V_t(a*,s') + C̃_t(i,s',a*) + Φ(i,s',a*)] − Φ(i,s,a)}
16:         N := N + 1
17:      end for
18:      append ChildSeq onto the front of Seq
19:      s := s'
20:   end while
21: end if
22: return Seq

23: function Evaluate(Sub-task i, State s)
24: if i is a primitive action then
25:    return V_t(i,s)
26: else
27:    for all child sub-task a of i do
28:       V_t(a,s) := Evaluate(a,s)[0]
29:    end for
30:    a* := argmax_a [V_t(a,s) + C̃_t(i,s,a) + Φ(i,s,a)]
31:    return < V_t(a*,s) + C̃_t(i,s,a*) + Φ(i,s,a*), a* >
32: end if

33: /*Main Programme*/
34: initialise all V and C̃ values arbitrarily
35: initialise Φ values
36: PBRS-MAXQ-0(root sub-task 0, starting state s_0)
```

Now we prove that PBRS-MAXQ-0 converges to the optimal policy. Recall that, in MAXQ-0, by using an OGLIE policy (see Section 3.2), π^* is guaranteed to be obtained. We propose the *biased OGLIE* (BOGLIE) policy as follows:

Definition 1. Given a MDP M and its MAXQ decomposition H_M, potential values are integrated into MAXQ according to PBRS-MAXQ-0. A BOGLIE policy is an action selection policy such that:

- each child sub-task is executed infinitely often in every state that is visited infinitely often;
- in the limit (i.e. after infinite long time of learning), for any composite sub-task a^0 and state s, the policy selects the *biased-greedy child* a^1 of a^0 at state s:

$$\underset{a^1 \in A_{a^0}(s)}{\operatorname{argmax}} [\tilde{Q}_t(a^0, s, a^1) + \sum_{j=1}^{m} \Phi(a^{j-1}, s, a^j)]$$

  with probability 1, where $A_{a^0}(s)$ is the set of available child of a^0 at state s, a^j ($j \in \{1, \cdots, m\}$) is the biased-greedy child of a^{j-1}, and a^m is a primitive action, where $m \in \mathbb{N}^*$ is the number of biased-greedy descendants of a^0 in state s; and
- for each composite sub-task i and state s, there is a fixed order for sub-tasks in $A_i(s)$ s.t. the policy breaks ties in favour of the sub-task that appears earliest in the order.

Note that $\tilde{Q}_t(i,s,a) = \tilde{V}_t(a,s) + \tilde{C}_t(i,s,a)$, where t is the number of updates. Also, note that the action returned by function **Evaluate** (line 31) is a biased-greedy child. Let a *biased ϵ-greedy policy* be a policy such that in each state s and for any composite sub-task i, the biased-greedy child of i is selected with a probability of $1 - \epsilon$, and a random child of i is selected with a probability of ϵ. This policy is an approximated BOGLIE policy. Now we prove that PBRS-MAXQ-0 is guaranteed to converge with a BOGLIE policy.

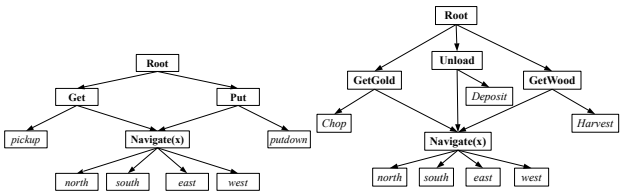
Theorem 1. Given a MDP M and its MAXQ decomposition $H_M = \{M_0, \cdots, M_n\}$, suppose all immediate rewards and all potential values are bounded. If $\alpha_t(i) > 0$ is a sequence of constants for sub-task i s.t.

$$\lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t(i) = \infty \qquad \text{and} \qquad \lim_{T\to\infty} \sum_{t=1}^{T} \alpha_t^2(i) < \infty,$$

then with probability 1, algorithm PBRS-MAXQ-0 converges to π^*, the unique hierarchical optimal policy for M consistent with H_M.

Proof. (Sketch.) Without loss of generality, we prove this theorem by showing that the learning dynamics of PBRS-MAXQ-0 with all $\tilde{C}$ values initialised as 0 is the same as that of MAXQ-0 with all C values initialised as their corresponding Φ values, i.e. $\tilde{C}_t(i,s,a) - \tilde{C}_0(i,s,a) = C_t(i,s,a) - C_0(i,s,a)$ where $\tilde{C}_0(i,s,a) = 0, C_0(i,s,a) = \Phi(i,s,a)$. We prove this inductively. The base case is to show that when $t = 0$, this equation holds, which is trivially true. To prove the inductive case, we need to prove that given any composite sub-task i and state s, i's greedy child (i.e. the child that was greedily selected by an OGLIE policy) at s wrt. C values is the same as i's biased-greedy child at s wrt. $\tilde{C}$ values. This is also proved by induction, starting from composite sub-tasks with only primitive children and moving towards the other composite sub-tasks. The key point in proving this is to show that function **Evaluate** in MAXQ-0 and PBRS-MAXQ-0 selects the same child given the same parent i and state s.

Since MAXQ-0 is guaranteed to converge given the conditions presented in this theorem, PBRS-MAXQ-0 is also guaranteed to converge, and $\tilde{C}^*(i,s,a) = C^*(i,s,a) - \Phi(i,s,a)$.



(a) Taxi (b) Resource Collection

Figure 1: Task graphs. Composite and primitive sub-tasks are in boldface and italic, respectively.

Since **Evaluate** in PBRS-MAXQ-0 and MAXQ-0 select the same child, the optimal policies of PBRS-MAXQ-0 and MAXQ-0 are the same. □

Note that all theoretical analysis until now is based on the setting that all pseudo rewards are 0. However, we can see that potential values and pseudo rewards are independent of each other: intuitively, potential values represent the goodness of performing a child sub-task in a parent sub-task, whereas pseudo rewards indicate whether a terminal state is desirable or not; furthermore, as for the scope, pseudo rewards can only be used in one level and only in learning, whereas potential values are used throughout the whole MAXQ hierarchy and not only used in learning, but also in execution (in the BOGLIE policy). Hence, all results given above also apply to MAXQ with non-zero pseudo rewards.

5 Experiments

We implement MAXQ-0 and PBRS-MAXQ-0 in two widely used applications for MAXQ: the *Fickle Taxi* problem and the *Resource Collection* problem, to compare their performances. By performing experiments in these applications, we attempt to answer the following questions: (i) given reasonable potential values, whether PBRS-MAXQ-0 can converge faster than MAXQ-0; (ii) given misleading potential values, whether the PBRS-MAXQ-0 can finally converge; and (iii) whether PBRS-MAXQ-0's performance is comparable or even better than other state-of-the-art approaches.

We first describe the detailed experimental settings of these two domains in Section 5.1 and 5.2, and present and discuss the learning performances in Section 5.3. In both domains, we use biased ϵ-greedy (see Section 4.2) for PBRS-MAXQ-0, and use standard ϵ-greedy for MAXQ-0. All experimental settings we use are the same as those in [4] (except for the learning parameters α, ϵ and γ, whose values are not provided in [4]). In all experiments and for all algorithms, we have $\gamma = 1$. All results presented in this section are averaged over 100 independent experiments, and the lighter colour areas in Figure 2 and 3 represent 95% confidence intervals.

5.1 Fickle Taxi Problem

The Fickle Taxi problem is an episodic RL problem consisting of a 5×5 grid world with four landmarks, labelled (R)ed, (G)reen, (B)lue and (Y)ellow. In the beginning of each episode, the taxi is in a random square, a passenger is in a random landmark and wants to be transported to another (randomly chosen) landmark . The taxi has to navigate to the passenger's starting position, pick up the passenger, go to the destination and put down the passenger there, and then an episode ends. When the passenger is picked and one square away from its starting position, it has 30% chance to change its destination.

There are six primitive actions available for the taxi: i) four navigation actions that move the taxi one square in the indicated direction with probability 0.7 and in each perpendicular direction with probability 0.15: *north*, *south*, *east* and *west*; ii) the *pickup* action, which transfers the passenger into the taxi and iii) the *putdown* action, which puts down the passenger at the square in which the taxi is. Hitting wall is a no-op and results in no additional rewards. The task graph we use is presented in Fig. 1(a). The state variables consist of the position of the taxi, the status of the taxi (empty or occupied) and the source and destination of the passenger. The agent gets a reward of $+20$ for completing a task, -1 for each action and an additional reward of -10 when the taxi performs *pickup* and *putdown* illegally.

We test the performances of three MAXQ implementations: MAXQ-0, PBRS-MAXQ-0 with reasonable heuristics (denoted by PBRS-good) and with misleading heuristics (denoted by PBRS-bad). The potential values $\Phi(i, s, a)$ used in PBRS-good are as follows:

- When sub-task a is *pickup*, $\Phi(i, s, a) = +10$ if the passenger has not been picked up and the taxi is in the passenger's position; otherwise, $\Phi(i, s, a) = -10$.
- When sub-task a is *putdown*, $\Phi(i, s, a) = +10$ if the passenger has been picked up and the taxi is in the passenger's destination position; otherwise, $\Phi(i, s, a) = -10$.
- When sub-task a is **Navigate(**x**)**, where x can be any landmark, $\Phi(i, s, a) = +10$ if:
  - i is **Get**, the passenger has not been picked up and x is the passenger's position; or
  - i is **Put**, the passenger has been picked up and x is the passenger's destination.

  In all other situations, $\Phi(i, s, a) = -10$.
- When a is *north*, *south*, *east* or *west*, the parent sub-task must be **Navigate(**x**)**, where x can be any landmark. We let $\Phi(i, s, a) = +5$ if performing a reduces the Manhattan distance to x, otherwise, $\Phi(i, s, a) = -5$.

The potential values used in PBRS-bad are the opposite of those used in PBRS-good in each state. Note that the heuristics used in PBRS-good contain imperfect instructions: for example, recommending the agents to choose the navigation action that reduces the Manhattan distance may not be the best action in some states, because the agent may be stopped by a wall. By using the imperfect instructions in PBRS-good and misleading instructions in PBRS-bad, we can more comprehensively investigate the convergence property and robustness of PBRS-MAXQ-0.

For simplicity, we let all sub-tasks use the same α value (i.e. for any sub-task i, $\alpha(i)$ in Alg. 1 is the same). The initial values and the decreasing rates (in brackets) of α and ϵ are listed in Table 1. After each episode, we let $\alpha_{t+1} = \alpha_t \times r_\alpha$ and $\epsilon_{t+1} = \epsilon_t \times r_\epsilon$ where r_α and r_ϵ are the decreasing rates for α and ϵ, respectively. Note that these parameters are selected

Table 1: The learning parameters used in Taxi problem.

	MAXQ-0	PBRS-good	PBRS-bad
α	1.0 (0.999)	1.0 (0.999)	1.0 (0.999)
ϵ	0.5 (0.9)	0.5 (0.9)	1.0 (0.99)

to maximise the convergence speed of each algorithm.

5.2 Resource Collection Problem

The Resource Collection problem we consider is a 25×25 grid world, in which the agent can *harvest* wood and *chop* gold from forests and goldmines, respectively, and *deposit* them at a town hall. The goal is to reach a predefined quota of wood and gold. There are two goldmines and two forests in the grid world. In each goldmine (forest), two portions of gold (wood) are contained. Note that the agent can only carry one portion of gold or wood at each moment. The quota for both gold and wood are three portions.

There are seven primitive actions available: i) four navigation actions that move the agent one square in the indicated direction with probability 0.7 and in the other directions with probability 0.1: *north*, *south*, *east* and *west*; ii) the *chop* and *harvest* action, which collects one portion of gold or wood if the agent is adjacent to a goldmine or a forest; and iii) the *deposit* action, which deposits the item the agent is carrying (if any) in the town hall. The task graph is presented in Fig. 1(b). The state variable we use consists of the position of the agent, what the agent is carrying, how much gold/wood remains in a goldmine/forest, how much gold/wood is still needed to meet the quota and whether the agent is adjacent to a goldmine/forest. The agents get a reward of $+50$ for meeting the gold/wood quota, a reward of -1 for each action and an additional reward of -1 for performing *chop*, *harvest* or *deposit* illegally (note that *deposit* is legal only when the agent is carrying some resources and in the town hall).

We implement the same three algorithms as in the Taxi problem. The potential values $\Phi(i, s, a)$ used in PBRS-good are as follows:

- When sub-task a is **Navigate(x)**, (where x can be the position of any goldmine, forest or the town hall), $\Phi(i, s, a) = +5$ if:
  - i is **GetGold** and x is a non-empty goldmine, or
  - i is **GetWood** and x is a non-empty forest.

  Otherwise, $\Phi(i, s, a) = -5$.
- When a is *north*, *south*, *east* or *west*, the parent sub-task must be **Navigate(x)**, where x can be any goldmine, forest or the town hall. We let $\Phi(i, s, a) = +2$ if performing a reduces the Manhattan distance to x; otherwise, $\Phi(i, s, a) = -2$.

The potential values used in PBRS-bad are the opposite of those used in PBRS-good. The learning parameters used in each algorithm are listed in Table 2, and they are selected to maximise the convergence speed of each algorithm.

5.3 Learning Performances and Discussions

The learning performances of all three implementations in the Fickle Taxi and Resource Collection problems are given in Fig. 2 and 3, respectively. We observe that, in both domains, PBRS-good significantly outperforms standard MAXQ-0 and

Table 2: Learning parameters used in Resource Collection.

	MAXQ-0	PBRS-good	PBRS-bad
α	0.3 (0.996)	0.1 (1)	0.3 (0.999)
ϵ	1.0 (0.99)	0.01 (1)	1.0 (0.999)

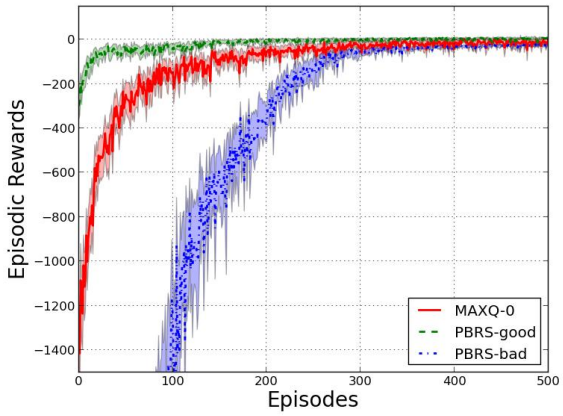


Figure 2: Performances in the Fickle Taxi problem.

PBRS-bad, in terms of both initial performance and convergence speed. PBRS-bad is also able to converge, although it converges significantly slower than the other two algorithms. Also, we find that to make PBRS-bad converge faster, α and ϵ values should be initialised as big values and decrease at slower rates (as indicated in Table 1 and 2), so as to ensure that more policies can be explored. By comparing PBRS-good with other state-of-the-art HRL algorithms in these two domains (see Fig. 1 in [4]), we find that the performance of PBRS-good is similar to that of Bayesian HRL with good priors[4], and better than all the other approaches' performances, including Bayesian Q-Learning and R-MAXQ [11]. These results indicate that: (i) given reasonable heuristics, PBRS-MAXQ-0 is able to converge significantly faster than MAXQ-0; (ii) given misleading heuristics, PBRS-MAXQ-0 is still able to converge, by using carefully tuned learning parameters; and (iii) PBRS-MAXQ-0's performance is competitive with other state-of-the-art MAXQ-based algorithms given 'good' heuristics. However, the model-based MAXQ approaches are computationally much more expensive than MAXQ-0 (see Table 1 in [4]), whereas PBRS-MAXQ-0 only introduces negligible computational overhead. Therefore, when heuristics are available, we believe that PBRS-MAXQ-0 is a more efficient technique to be used.

6 Conclusion

In this paper, we investigate the integration of two popular RL techniques: PBRS and MAXQ. We propose the PBRS-MAXQ-0 algorithm, and theoretically prove that by using PBRS-MAXQ-0, arbitrary potential values can be integrated

[4]The 'good priors' used in [4] are actually model posterior computed in previous experiments. Hence, it is difficult to directly compare the 'quality' of prior knowledge used in our work and in [4].

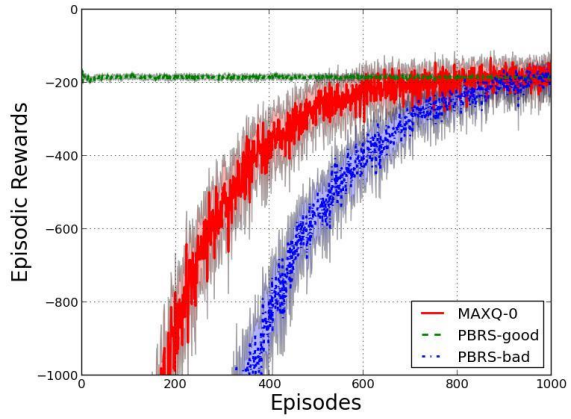


Figure 3: Performances in the Resource Collection problem.

into MAXQ without altering its original optimal policies. We empirically test the effectiveness of PBRS-MAXQ-0 in two widely used HRL testbeds. Empirical results suggest that PBRS-MAXQ-0 is an effective, efficient and robust method for incorporating heuristics into HRL.

Our work improves the granularity of domain knowledge that can be integrated into HRL: although MAXQ provides domain experts with a method to instruct RL by defining subtask hierarchies, very detailed instructions can hardly be represented by MAXQ. For example, in the Taxi problem (see Section 5.1), the instruction 'choose the action that reduces the Manhattan distance to the goal' can hardly be represented by MAXQ hierarchies, although this instruction can significantly improve the learning speed. Our work allows people to integrate finer-grained domain knowledge into MAXQ, while maintaining the convergence property of MAXQ.

We view model-based prior knowledge and heuristics as complementary information for improving learning speed of RL. Thus, an immediate future work is to incorporate our technique with model-based MAXQ approaches like R-MAXQ [11] and Bayesian HRL [4]. Also, since both PBRS [5] and MAXQ [9; 14] have been successfully used in multi-agent learning, applying PBRS-MAXQ-0 to multi-agent problems is also worth further investigation. Furthermore, since there have been some works automatically proposing potential values for classic PBRS [10; 8; 7], similar techniques should also be explored for PBRS-MAXQ-0.

Acknowledgement

We thank Marc Deisenroth and anonymous reviewers for their helpful comments on earlier versions of this paper.

References

[1] J. Asmuth, M. Littman, and R. Zinkov. Potential-based shaping in model-based reinforcement learning. In *Proc. of AAAI*, 2008.

[2] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *DISCRETE EVENT DYN S*, 13:343–379, 2003.

[3] R. I. Brafman and M. Tennenholtz. R-Max - a general polynomial time algorithm for near-optimal reinforcement learning. *JMLR*, 2:213–231, 2003.

[4] F. Cao and S. Ray. Bayesian hierarchical reinforcement learning. In *Proc. of NIPS*, pages 73–81, 2012.

[5] S. Devlin and D. Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Proc. of AAMAS*, pages 225–232, 2011.

[6] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *JAIR*, 13:227–303, 2000.

[7] Y. Gao and F. Toni. Argumentation accelerated reinforcement learning for cooperative multi-agent systems. In *Proc. of ECAI*, 2014.

[8] Y. Gao, F. Toni, and R. Craven. Argumentation-based reinforcement learning for robocup soccer keepaway. In *Proc. of ECAI*, 2012.

[9] M. Ghavamzadeh, S. Mahadevan, and R. Makar. Hierarchical multi-agent reinforcement learning. *JAAMAS*, 13(2):197–229, 2006.

[10] M. Grzes and D. Kudenko. Plan-based reward shaping for reinforcement learning. In *Proc. of IEEE Conf. 'Intelligent Systems'*, 2008.

[11] N. Jong and P. Stone. Hierarchical model-based reinforcement learning: R-Max + MAXQ. In *Proc. of ICML*, 2008.

[12] R. Makar and S. Mahadevan. Hierarchical multi-agent reinforcement learning. In *Proc. of AAMAS*, 2001.

[13] N. Mehta, S. Ray, P. Tadepalli, and T. Dietterich. Automatic discovery and transfer of MAXQ hierarchies. In *Proc. of ICML*, pages 648–655. ACM, 2008.

[14] N. Mehta and P. Tadepalli. Multi-agent shared hierarchy reinforcement learning. In *ICML Workshop on Rich Representations for Reinforcement Learning*, page 45. Citeseer, 2005.

[15] A. Ng, D. Harada, and S. Russell. Policy invariance under reward transformations: theory and application to reward shaping. In *Proc. of ICML*, pages 278–287, 1999.

[16] R. Parr and S. Russell. Reinforcement learning with hierarchies of machines. In *Proc. of NIPS*, 1997.

[17] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and SMDPs: A framework for temporal abstraction in reinforcement learning. *AIJ*, 112:181–211, 1999.

[18] E. Wiewiora, G. Cottrell, and C. Elkan. Principled methods for advising reinforcement learning agents. In *Proc. of ICML*, 2003.