State Similarity Based Approach for Improving Performance in RL [*]

Sertan Girgin^{1,2} and **Faruk Polat**^{1} and **Reda Alhajj**^{2,3}

Middle East Technical University^{1}
Dept. of Computer Engineering
{sertan,polat}@ceng.metu.edu.tr

University of Calgary^{2}
Dept. of Computer Science
{girgins,alhajj}@cpsc.ucalgary.ca

Global University^{3}
Dept. of Computer Science
rhajj@gu.edu.lb

Abstract

This paper employs state similarity to improve reinforcement learning performance. This is achieved by first identifying states with similar sub-policies. Then, a tree is constructed to be used for locating common action sequences of states as derived from possible optimal policies. Such sequences are utilized for defining a similarity function between states, which is essential for reflecting updates on the action-value function of a state onto all similar states. As a result, the experience acquired during learning can be applied to a broader context. Effectiveness of the method is demonstrated empirically.

1 Introduction

Reinforcement learning (RL) is the problem faced by an agent that must learn behavior through trial-and-error interactions with a dynamic environment, by gaining percepts and rewards from the world and taking actions to affect it. In most of the realistic and complex domains, the task that an agent tries to solve is composed of various subtasks and has a hierarchical structure formed by the relations between them [Barto and Mahadevan, 2003]. Each of these subtasks repeats many times at different regions of the state space. Although all instances of the same subtask, or similar subtasks, have almost identical solutions, without any (self) guidance an agent has to learn these solutions independently by going through similar learning stages again and again. This situation affects the learning process in a negative way, making it difficult to converge to optimal behavior in reasonable time.

The main reason of this problem is the lack of connections, that would allow to share solutions, between similar subtasks scattered throughout the state space. One possible way to build connections is to use *temporally abstract actions* (TAAs), which generalize primitive actions and last for a period of time [Sutton *et al.*, 1999; Dietterich, 2000; Barto and Mahadevan, 2003]. TAAs can be included in the problem definition, which requires extensive domain knowledge and becomes more difficult as the complexity of the problem increases, or be constructed automatically either by finding sub-goal states and generating corresponding options that solve them [Stolle and Precup, 2002; Menache *et al.*, 2002; Mannor *et al.*, 2004; Simsek *et al.*, 2005] or by identifying frequently occurring patterns over the state, action and reward trajectories [McGovern, 2002; Girgin *et al.*, 2006]. Since different instances of the same or similar subtasks would probably have different subgoals in terms of state representation, with subgoal based methods they will be discovered and treated separately. Although sequence based methods are capable of combining different instances of the same subtask, multiple abstractions may be generated for each of them.

TAA based methods try to solve the solution sharing problem inductively. Based on the fact that states with similar patterns of behavior constitute the regions of state space corresponding to different instances of similar subtasks, the notion of state equivalence can be used as a more direct mechanism of solution sharing. By reflecting experience acquired on one state to all similar states, connections between similar subtasks can be established implicitly. In fact, this reduces the repetition in learning, and consequently improves the performance. State equivalence is closely related with model minimization in Markov Decision Processes (MDPs), and various definitions exist. [Givan *et al.*, 2003] define equivalence of states based upon stochastic bisimilarity, such that two states are said to be equivalent if they are both action sequence and optimal value equivalent. Based on the notion of MDP homomorphism, [Ravindran and Barto, 2001; 2002] extended equivalence over state-action pairs, which as a result allows reductions and relations not possible in case of bisimilarity. Furthermore, they applied state-(action) equivalence to the options framework to derive more compact options without redundancies, called relativized options. While relativized options are defined by the user, instances are generated automatically. [Zinkevich and Balch, 2001] also addressed how symmetries in MDPs can be used to accelerate learning by employing equivalence relations on the state-action pairs; in particular for the multi-agent case, where permutations of features corresponding to various agents are prominent, but without explicitly formalizing them.

In this paper, following homomorphism notion, we propose a method to identify states with similar sub-policies without requiring a model of the MDP or equivalence relations, and show how they can be integrated into RL frame-

[*]This work was supported by the Scientific and Technological Research Council of Turkey under Grant No. 105E181(HD-7).

work to improve the learning performance. Using the collected history of states, actions and rewards, traces of policy fragments are generated and then translated into a tree form to efficiently identify states with similar sub-policy behavior based on the number of common action-reward sequences. Updates on the action-value function of a state are then reflected to all similar states, expanding the influence of new experiences. The proposed method can be treated as a meta-heuristic to guide any underlying RL algorithm. We demonstrate the effectiveness of this approach by reporting test results on two domains, namely various versions of the taxi cab and the room doorway problems. Further, the proposed method is compared with other RL algorithms, and a substantial level of improvement is observed on different test cases. Also, we present how the performance of out work compares with hierarchical RL algorithms, although the approaches are different. The tests demonstrate the applicability and effectiveness of the proposed approach.

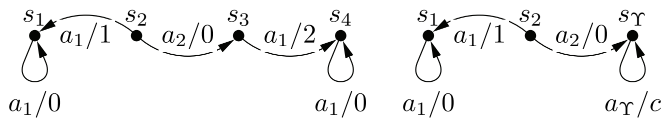
The rest of this paper is organized as follows: In Section 2, we briefly describe the standard RL framework of discrete time, finite MDPs, define state equivalence based on MDP homomorphisms, and show how they can be used to improve the learning process. Our approach to how similar states can be identified during the learning process is presented in Section 3. Experimental results are reported and discussed in Section 4. Section 5 is conclusions.

2 Background and Motivation

In this section, we briefly overview the background necessary to understand the material introduced in this paper. We start by defining MDP and related RL problem.

An MDP is a tuple $\langle S, A, \Psi, T, R\rangle$, where S is a finite set of states, A is a finite set of actions, $\Psi \subseteq S \times A$ is the set of admissible state-action pairs, $T : \Psi \times S \rightarrow [0, 1]$ is a state transition function such that $\forall (s, a) \in \Psi, \sum_{s' \in S} T(s, a, s') = 1$, and $R : \Psi \rightarrow \Re$ is a reward function. $R(s, a)$ is the *immediate* expected reward received when action a is executed in state s.[1] A (stationary) *policy*, $\pi : \Psi \rightarrow [0, 1]$, is a mapping that defines the probability of selecting an action in a particular state. The *value* of a state s under policy π, $V^{\pi}(s)$, is the expected infinite discounted sum of rewards that the agent will gain if it starts in state s and follows π. In particular, if the agent takes action a at state s and then follows π, the resulting sum is called the value of the state-action pair (s, a) and denoted $Q^{\pi}(s, a)$. The objective of an agent is to find an *optimal policy*, π^*, which maximizes the state value function for all states. Based on the experience in the form of sample sequences of states, actions, and rewards collected from on-line or simulated trial-and-error interactions with the environment, RL methods try to find a solution to an MDP by approximating optimal value functions. Detailed discussions of various approaches can be found in [Kaelbling *et al.*, 1996; Sutton and Barto, 1998; Barto and Mahadevan, 2003].

RL problems, in general, inherently contain subtasks that need to be solved by the agent. These subtasks can be represented by simpler MDPs that locally preserve the state transition and reward dynamics of the original MDP. Let $M = \langle S, A, \Psi, T, R\rangle$ and $M' = \langle S' \cup \{s_\Upsilon\}, A' \cup \{a_\Upsilon\}, \Psi', T', R'\rangle$ be two MDPs, such that s_Υ is an absorbing state with a single admissible action a_Υ which, once executed, transitions back to s_Υ with certainty. A surjection $h : \Psi \rightarrow \Psi' \cup \{(s_\Upsilon, a_\Upsilon)\}$ is a *partial MDP homomorphism* (pMDPH) from M to M', if for all states that do not map to s_Υ, state-action pairs that have the same image under h also have the same block transition behavior in M and the same expected reward[2]. Under h, M' is called the *partial homomorphic image* of M[3]. Two state-action pairs (s_1, a_1) and (s_2, a_2) in M are said to be *equivalent* if $h(s_1, a_1) = h(s_2, a_2)$; states s_1 and s_2 are equivalent if there exists a bijection $\rho : A_{s_1} \rightarrow A_{s_2}$ such that $\forall a \in A_{s_1},\ h(s_1, a) = h(s_2, \rho(a))$. The set of state-action pairs equivalent to (s, a), and the set of states equivalent to s are called the equivalence classes of (s, a) and s, respectively. [Ravindran and Barto, 2001] proved that if h above is a complete homomorphism, then an optimal policy π^* for M can be constructed from an optimal policy for M'[4]. This makes it possible to solve an MDP by solving one of its homomorphic images which can be structurally simpler and easier to solve. However, in general such a construction is not viable in case of pMDPHs, since the optimal policies would depend on the rewards associated with absorbing states.



(a) (b)

Figure 1: (a) Markov Decision Process M, and (b) its homomorphic image M'. Directed edges indicate state transitions. Each edge label denotes the action causing the transition between connected states and the associated expected reward.

For example, consider the deterministic MDP M given in Fig. 1(a). If the discount factor, γ, is set as 0.9, then optimal policy at s_2 is to select action a_2; M' given in Fig. 1(b) is a partial homomorphic image of M[5] and one can easily determine that $Q^*(s_2, a_1) = 1$ and $Q^*(s_2, a_2) = 9 * c$, where c is the expected reward for executing action a_Υ at the absorbing state. Accordingly, the optimal policy for M' at s_2 is to select a_1 if $c < 1/9$, and a_2 otherwise, which is different than the optimal policy for M. This example demonstrates that unless reward functions are chosen carefully, it may not be possible to solve a given MDP by employing the solutions of its partial homomorphic images. However, if equivalence classes of state-action pairs and states are known, they can be used to speed up the learning process. Let (s, a) and (s', a') be two equivalent state-action pairs in a given MDP M based on the partial homomorphic image M' of M under the surjection h.

[1] s, a is used instead of (s, a), wherever possible.

[2] $h((s, a)) = (f(s), g_s(a))$, where $f : S \rightarrow S' \cup \Upsilon$ is a surjection and $\{g_s : A_s \rightarrow A'_{f(s)} | s \in S\}$ is a set of state dependent surjections.

[3] For a formal definition see [Ravindran and Barto, 2001; 2002]

[4] $\forall a \in g_s^{-1}(a'), \pi^*(s, a) = \pi^*_{M'}(f(s), a')/|g_s^{-1}(a')|$

[5] Under h defined as $f(s_1) = s_1, f(s_2) = s_2, f(s_3) = f(s_4) = s_\Upsilon$, $g_{\{s_1,s_2\}}(a_1) = a_1, g_{s_2}(a_2) = a_2, g_{\{s_3,s_4\}}(a_1) = a_\Upsilon$

Suppose that while learning the optimal policy, the agent selects action a at state s which takes it to state t with an immediate reward r, such that t is mapped to a non-absorbing state under h. Let Ω be the set of states in M which are accessible from s' by taking action a', i.e. probability of transition is non-zero, and are mapped to the same state in M' as t under h. By definition, for any $t' \in \Omega$, the probability of experiencing a transition from s' to t' upon taking action a' is equal to that of from s to t upon taking action a. Also, since (s, a) and (s', a') are equivalent, we have $R(s, a) = R(s', a')$, i.e., both state-action pairs have the same expected reward, and since t is mapped to a non-absorbing state, independent of the reward assigned to the absorbing state, they have the same policy. Therefore, for any state $t' \in \Omega$, $o = \langle s', a', r, t' \rangle$ can be regarded as a virtual experience tuple and $Q(s', a')$ can be updated similar to $Q(s, a)$ based on observation o, i.e., pretending as if action a' transitioned the agent from state s' to state t' with an immediate reward r.

The method described above assumes that equivalence classes of states and corresponding pMDPHs are already known. If such information is not available prior to learning, then, for a restricted class of pMDPHs, it is still possible to identify states that are similar to each other with respect to state transition and reward behavior based on the history of events, as we show in the next section. Experience gathered on one state, then, can be reflected to similar states to improve the performance of learning.

3 Finding Similar States

For the rest of the paper, we will restrict our attention to *direct* pMDPHs in which, except the absorbing state and its associated action, the action sets of an MDP and its homomorphic image are the same and there is an identity mapping on the action component of state-action tuples[6]. Let $M = \langle S, A, \Psi, T, R \rangle$ be a given MDP. Starting from state s, a sequence of states, actions and rewards $\sigma = s_1, a_2, r_2, \ldots, r_{n-1}, s_n$, such that $s_1 = s$ and each a_i, $2 \leq i \leq n-1$, is chosen by following a policy π is called a π-*history* of s with length n. $AR_\sigma = a_1 r_1 a_2 \ldots a_{n-1} r_{n-1}$ is the action-reward sequence of σ, and the restriction of σ to $\Sigma \subseteq S$, denoted by σ_Σ, is the longest prefix $s_1, a_2, r_2, \ldots, r_{i-1}, s_i$ of σ, such that for all $j = 1..i, s_j \in \Sigma$ and $s_{i+1} \notin \Sigma$. If two states of M are equivalent under a direct pMDPH h, then the set of images of π-histories restricted to the states that map to non-absorbing states under h, and consequently, the list of associated action-reward sequences are the same [7]. This property of equivalent states leads to a natural approach to calculate the similarity between any two states based on the number of common action-reward sequences.

Given any two states s and s' in S, let $\Pi_{s,i}$ be the set of π-histories of state s with length i, and $\varsigma_i(s, s')$ calculated as

$$\varsigma_i(s, s') = \frac{\sum_{j=1}^{i} |\{\sigma \in \Pi_{s,j} | \exists \sigma' \in \Pi_{s',j}, AR_\sigma = AR_{\sigma'}\}|}{\sum_{j=1}^{i} |\Pi_{s,j}|}$$

be the ratio of the number of common action-reward sequences of π-histories of s and s' with length up to i to the number of action-reward sequences of π-histories of s with length up to i. $\varsigma_i(s, s')$ will be close to 1, if s' is similar to s in terms of state transition and reward behavior; and close to 0, in case they differ considerably from each other. Note that, even for equivalent states, the action-reward sequences will eventually deviate and follow different courses as the subtask that they are part of ends. As a result, for i larger than some threshold value, ς_i would inevitably decrease and no longer be a permissible measure of the state similarity. On the contrary, for very small values of i, such as 1 or 2, ς_i may over estimate the amount of similarity since number of common action-reward sequences can be high for short action-reward sequences. Also, since optimal value of i depends on the subtasks of the problem, to increase robustness, it is necessary to take into account action-reward sequences of various lengths. Therefore, the maximum value or weighted average of $\varsigma_i(s, s')$ over a range of problem specific i values, k_{min} and k_{max}, can be used to combine the results of evaluations and approximately measure the degree of similarity between states s and s', denoted by $\varsigma(s, s')$. Once $\varsigma(s, s')$ is calculated, s' is regarded as equivalent to s if $\varsigma(s, s')$ is over a given threshold value $\tau_{similarity}$. Likewise, by restricting the set of π-histories to those that start with a given action a in the calculation of $\varsigma(s, s')$, similarity between state-action pairs (s, a) and (s', a) can be measured approximately.

If the set of π-histories for all states are available in advance, then equivalent states can be identified prior to learning. However, in general, the dynamics of the system is not known in advance and consequently such information is not accessible. Therefore, using the history of observed events (i.e. sequence of states, actions taken and rewards received), the agent must incrementally store the π-histories of length up to k_{max} during learning and enumerate common action-reward sequences of states in order to calculate similarities between them. For this purpose, we propose an auxiliary structure called *path tree*, which stores the prefixes of action-reward sequences of π-histories for a given set of states. A path tree $P = \langle N, E \rangle$ is a labeled rooted tree, where N is the set of nodes, such that each node represents a unique action-reward sequence; and $e = (u, v, \langle a, r \rangle) \in E$ is an edge from u to v with label $\langle a, r \rangle$, indicating that action-reward sequence v is obtained by appending a, r to u, i.e., $v = uar$. The root node represents the empty action sequence. Furthermore, each node u holds a list of $\langle s, \xi \rangle \in S \times \mathcal{R}$ tuples, stating that state s has one or more π-histories starting with

[6] $g_s(a) = a$, for every s that maps to a non-absorbing state.

[7] Let (s, a) and (s', a') be two state-action pairs that are equivalent to each other based on partial homomorphic image $M' = \langle S' \cup \{s_\Upsilon\}, A \cup \{a_\Upsilon\}, \Psi', T', R' \rangle$ of $M = \langle S, A, \Psi, T, R \rangle$ under direct pMDPH h. Consider a π-history of state s of length n, and let $\Sigma_h \subseteq S$ be the inverse image of S' under h, and $\sigma_{\Sigma_h} = s_1, a_2, r_2, \ldots, r_{k-1}, s_k, k \leq n$ be the restriction of σ on Σ_h. The image of σ_{Σ_h} under h, denoted by $F(\sigma_{\Sigma_h})$, is obtained by mapping each state s_i to its counterpart in S', i.e., $F(\sigma_{\Sigma_h}) = h(s_1), a_2, r_2, h(s_2), a_3, \ldots, r_{k-1}, h(s_k)$. By definition, $F(\sigma_{\Sigma_h})$ is a π-history of $h(s)$ in M', and since s and s' are equivalent, there exists a π-history σ' of s' such that the image of its restriction to Σ_h under h is equal to $F(\sigma_{\Sigma_h})$, i.e., $F(\sigma'_{\Sigma_h}) = F(\sigma_{\Sigma_h})$; and furthermore $AR_{\sigma'_{\Sigma_h}} = AR_{\sigma_{\Sigma_h}}$.

Algorithm 1 Q-learning with equivalent state update.

```
1: Initialize $Q$ arbitrarily (e.g., $Q(\cdot,\cdot) = 0$)
2: Let $T$ be the path tree initially containing root node only.
3: repeat
4:     Let $s$ be the current state
5:     repeat                                                ▷ for each step
6:         Choose and execute $a$ from $s$ using policy derived from
           $Q$ with sufficient exploration. Observe $r$ and the next state $s'$;
           append $\langle s, a, r, s' \rangle$ to observation history.
7:         $\Delta Q(s,a) = \alpha(r + \gamma \max_{a' \in A_{s'}} Q(s',a') - Q(s,a))$
8:         for all $(t, a_t)$ similar to $(s, a)$ do
9:             Let $t'$ be a state similar to $s'$.
10:            $\Delta Q(t,a_t) = \alpha(r + \gamma \max_{a_{t'} \in A_{t'}} Q(t',a_{t'}) - Q(t,a_t))$
11:        end for
12:        $s = s'$
13:    until $s$ is a terminal state
14:    Using the observation history, generate the set of $\pi$-histories
       of length up to $k_max$ and add them $T$.
15:    Traverse $T$ and update eligibility values of the tuples in each
       node; prune tuples with eligibility value less than $\xi_{threshold}$.
16:    Calculate $\varsigma$ and determine similar states $s, s'$ such that
       $\varsigma(s,s') > \tau_{similarity}$.
17:    Clear observation history.
18: until a termination condition holds
```

action sequence σ_u. ξ is the *eligibility value* of σ_u for state s, representing its occurrence frequency. It is incremented every time a new π-history for state s starting with action sequence σ_u is added to the path tree, and gradually decremented otherwise. A π-history $h = s_1 a_2 r_2 \ldots r_{k-1} s_k$ can be added to a path tree by starting from the root node, following edges according to their label. Let $\hat{n}$ denote the active node of the path tree, which is initially the root node. For $i = 1..k-1$, if there is a node n such that $\hat{n}$ is connected to n by an edge with label $\langle a_i, r_i \rangle$, then either ξ of the tuple $\langle s_1, \xi \rangle$ in n is incremented or a new tuple $\langle s_1, 1 \rangle$ is added to n if it does not exist, and $\hat{n}$ is set to n. Otherwise, a new node containing tuple $\langle s_1, 1 \rangle$ is created, and $\hat{n}$ is connected to this node by an edge with label $\langle a_i, r_i \rangle$. The new node becomes the active node.

After each episode, or between two termination conditions such as reaching reward peaks in case of non-episodic tasks, using the consecutive fragments of the observed sequence of states, actions and rewards, a set of π-histories of length up to k_{max} are generated and added to the path tree using the procedure described above. Then, the eligibility values of tuples in the nodes of the tree are decremented by a factor of $0 < \xi_{decay} < 1$, called *eligibility decay rate*, and tuples with eligibility value less than a given threshold, $\xi_{threshold}$, are removed to keep the tree of manageable size and focus the search on recent and frequently used action-reward sequences. Using the generated path tree, $\varsigma(s,s')$ can be calculated incrementally for all $s, s' \in S$ by traversing it in breadth-first order and keeping two arrays $\kappa(s)$ and $K(s,s')$. $\kappa(s)$ denotes the number of nodes containing s, and $K(s,s')$ denotes the number of nodes containing both s and s' in their tuple lists. Initially $\varsigma(s,s')$, $K(s,s')$, and $\kappa(s)$ are set to 0. At each level of the tree, the tuple lists of nodes at that level are processed, and $\kappa(s)$ and $K(s,s')$ are incremented accordingly. After processing level i, $k_{min} \leq i \leq k_{max}$, for every s, s' pair that co-exist in the tuples list of a node at that level, $\varsigma(s,s')$ is compared with $K(s,s')/\kappa(s)$ and updated if the latter one is greater. Note that, since eligibility values stored in the nodes of the path tree is a measure of occurrence frequencies of corresponding sequences, it is possible to extend the similarity function by incorporating eligibility values in the calculations as normalization factors. This would improve the quality of the metric and also result in better discrimination in domains with high degree of non-determinism, since the likelihood of the trajectories will also be taken into consideration. In order to simplify the discussion, we opted to omit this extension. Once ς values are calculated, equivalent states, i.e., state pairs with ς greater than $\tau_{similarity}$, can be identified and incorporated into the learning process. The proposed method applied to Q-learning algorithm is presented in Alg. 1.

4 Experiments

We applied the similar state update method described in Section 3 to Q-learning and compared its performance with different RL algorithms on two test domains: a six-room maze[8] and various versions of the taxi problem[9] (Fig. 2(a)). Also, its behavior is examined under various parameter settings, such as maximum length of π-histories and eligibility decay rate. In all test cases, the initial Q-values are set to 0, and ϵ-greedy action selection mechanism, where action with maximum Q-value is selected with probability $1-\epsilon$ and a random action is selected otherwise, is used with $\epsilon = 0.1$. The results are averaged over 50 runs. Unless stated otherwise, the path tree is updated using an eligibility decay rate of $\xi_{decay} = 0.95$ and an eligibility threshold of $\xi_{threshold} = 0.1$, and in similarity calculations the following parameters are employed: $k_{min} = 3$, $k_{max} = 5$, and $\tau_{similarity} = 0.8$.

Fig. 2(b) shows the total reward obtained by the agent until goal position is reached in six-room maze problem when equivalent state update is applied to Q-learning and Sarsa(λ) algorithms. Based on initial testing, we used a learning rate and discount factor of $\alpha = 0.125$, and $\gamma = 0.90$, respectively. For the $Sarsa(\lambda)$ algorithm, λ is taken as 0.98. State similarities are calculated after each episode. As expected, due to backward reflection of received rewards, Sarsa(λ) converges much faster compared to Q-learning. The learning curve of

[8]The agent's task is to move from a randomly chosen position in the top left room to the goal location in the bottom right room using primitive actions that move the agent to one of four neighboring cells. Each action is non-deterministic and succeeds with probability 0.9, or moves the agent perpendicular to the desired direction with probability 0.1. The agent only receives a reward of 1 when it reaches the goal location and a negative reward of -0.01 otherwise.

[9]The agent tries to transport a passenger between predefined locations on a $n \times n$ grid world using six primitive actions: move to one of four adjacent squares, *pickup* or *dropoff* the passenger. Should a move action cause the agent hit to wall/obstacle, the position of the agent will not change. The movement actions are non-deterministic and with 0.2 probability agent may move perpendicular to the desired direction. An episode ends when the passenger is successfully transported and the agent receives a reward of $+20$. There is an immediate negative reward of -10 if pickup or drop-off actions are executed incorrectly, and -1 for any other action.

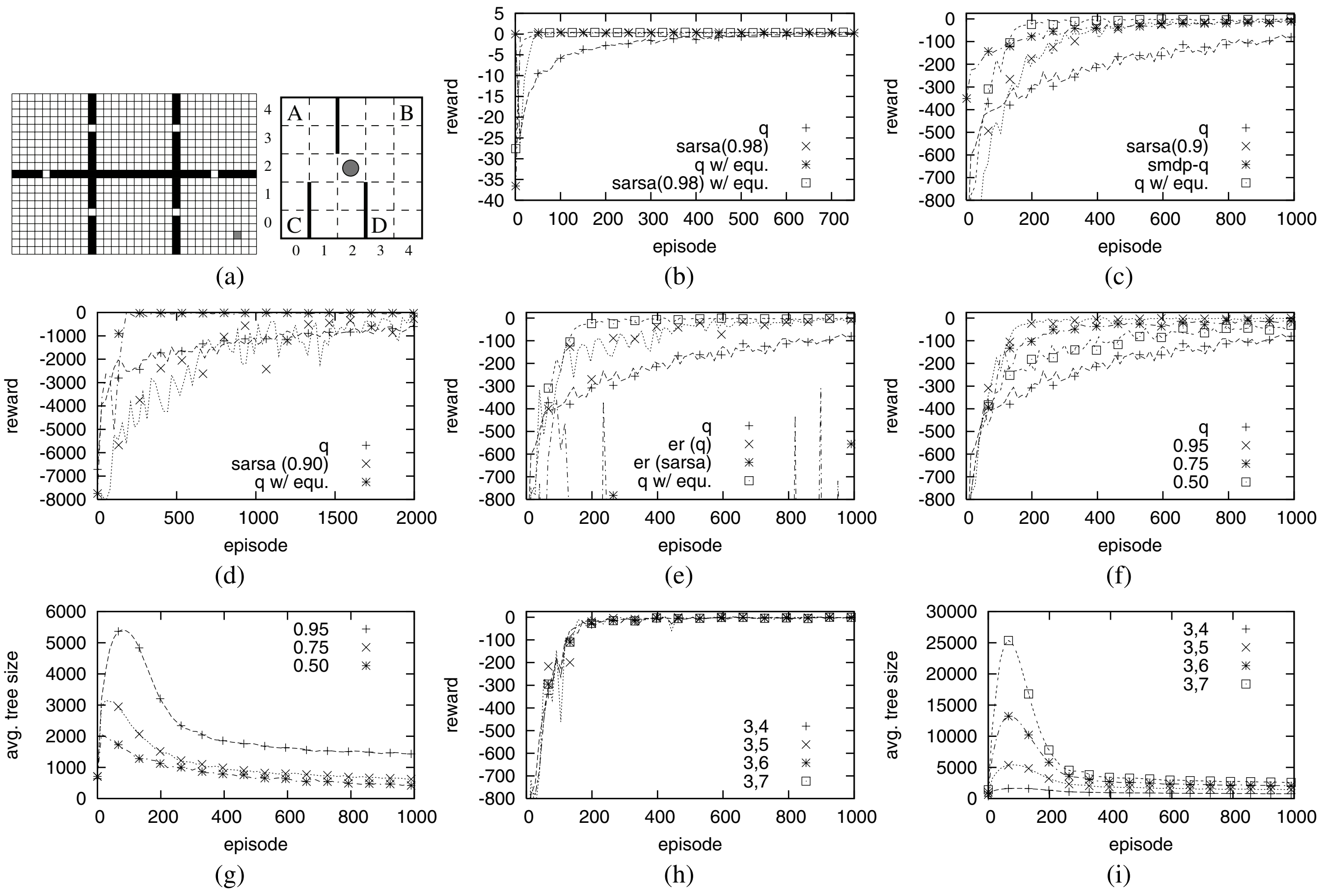


Figure 2: (a) Six-room maze and 5×5 Taxi problems, (b) Results for the six-room maze problem; Results for the (c) 5×5 and (d) 12×12 taxi problems; (e) Experience replay with the same number of state updates; Effect of ξ_{decay} and $k_{min,max}$ on 5×5 taxi problem: (f,h) Reward obtained, and (h,i) average size of the path tree.

Q-learning with equivalent state update indicates that, starting from early states of learning, the proposed method can effectively utilize state similarities and improve the performance considerably. Since convergence is attained in less than 20 episodes, the result obtained using Sarsa(λ) with equivalent state update is almost indistinguishable from that of Q-learning with equivalent state update.

The results of the corresponding experiments in 5×5 taxi problem showing the total reward obtained by the agent is presented in Fig. 2(c). The initial position of the taxi agent, location of the passenger and its destination are selected randomly with uniform probability. α is set to 0.05, and λ is taken as 0.9 in Sarsa(λ) algorithm. State similarities are computed every 5 episodes starting from the 20^{th} episode in order to let the agent gain experience for the initial path tree. Similar to the six-room maze problem, Sarsa(λ) learns faster than regular Q-learning. In SMDP Q-learning [Bradtke and Duff, 1994], in addition to primitive actions, the agent can select and execute hand-coded options, which move the agent from any position to one of predefined locations using minimum number of steps. Although SMDP Q-learning has a very steep learning curve in the initial stages, by utilizing abstractions and symmetries more effectively Q-learning with equivalent state update performs better in the long run. The learning curves of algorithms with equivalent state update reveals that the proposed method is successful in identifying similar states which leads to an early improvement in performance, which allows the agent to learn the task more efficiently. The results for the larger 12×12 taxi problem presented in Fig. 2(d) demonstrate that the improvement becomes more evident as the complexity of the problem increases.

The proposed idea of using state similarities and then updating similar states leads to more state-action value, i.e., Q-value, updates per experience. It is known that remembering past experiences and reprocessing them as if the agent repeatedly experienced what it has experienced before, which is called *experience replay* [Lin, 1992], speed up the learning process by accelerating the propagation of rewards. Experience replay also results in more updates per experience. In order to test whether the gain of the equivalent state update method in terms of learning speed is simply due to the fact that more Q-value updates are made or not, we compared its performance to experience replay using the same number of updates. The results obtained by applying both methods to regular Q-learning algorithm on the 5×5 taxi problem are presented in Fig. 2(e). Even though the performance of learning improves with experience replay, it falls short of Q-learning with equivalent state update. This indicates that in addition to number of updates, how they are determined is also important. By reflecting updates in a semantically rich manner based on the similarity of action-reward patterns, rather than neutrally as in experience replay, the proposed method turns out to be more efficient. Furthermore, in order to apply experience replay, the state transition and reward formulation of a problem should not change over time as past experiences may no longer be relevant otherwise. The dynamic nature of the sequence tree allows the proposed method to handle such situations as well.

In order to analyze how various parameter choices of the proposed method affect the learning behavior, we conducted a set of experiments under different settings. The results for various ξ_{decay} values are presented in Fig. 2(f,g). As the eligibility decay decreases, the number of π-histories represented in the path tree also decrease, and recent π-histories dominate over existing ones. Consequently, the less number of equivalent states can be identified and the performance of the method also converges to that of regular Q-learning. Fig. 2(h,i) shows how the length of π-histories affect the performance in the taxi domain. Different k_{max} values show almost indistinguishable behavior, although the path tree shrinks considerably as k_{max} decreases. Despite the fact that minimum and maximum π-history lengths are inherently problem specific, in most applications, k_{max} near k_{min} is expected to perform well as restrictions of pMDPHs to smaller state sets will also be pMDPHs themselves.

5 Conclusions

In this paper, we proposed and analyzed the interesting and useful characteristics of a tree-based approach that, during learning, identifies states similar to each other with respect to action-reward patterns, and based on this information reflects state-action value updates of one state to multiple states. Experiments conducted on two well-known domains highlighted the applicability and effectiveness of utilizing such a relation between states in the learning process. The reported test results demonstrate that experience transfer performed by the algorithm is an attractive approach to make learning systems more efficient. However, when the action set is very large, the parameters, such as eligibility decay, must be chosen carefully in order to reduce the computational cost and keep the size of the tree manageable. Also, instead of matching action-reward sequences exactly, they can be matched partially and grouped together based on their reward values (ex. ϵ neighborhood). This is especially important in cases where the immediate reward has a distribution. We are currently working on adaptation of the method to continuous domains, which also covers these issues. Furthermore, since the similarity between two states is a value which varies between 0 and 1, rather than thresholding to determine equivalent states and reflecting the whole experience, it is possible to make use of the degree of similarity, for example, by updating Q-values in proportion to their similarity. Future work will examine possible ways of extending and improving the proposed method to consider such information.

References

[Barto and Mahadevan, 2003] A. G. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems*, 13(4):341–379, 2003.

[Bradtke and Duff, 1994] S. J. Bradtke and M. O. Duff. Reinforcement learning methods for continuous-time markov decision problems. In *Advances in NIPS 7*, pages 393–400, 1994.

[Dietterich, 2000] T. G. Dietterich. Hierarchical reinforcement learning with the MAXQ value function decomposition. *Journal of Artificial Intelligence Research*, 13:227–303, 2000.

[Girgin *et al.*, 2006] S. Girgin, F. Polat, and R. Alhajj. Learning by automatic option discovery from conditionally terminating sequences. In *Proc. of the 17th ECAI*, 2006.

[Givan *et al.*, 2003] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in markov decision processes. *Artificial Intelligence*, 147(1-2):163–223, 2003.

[Kaelbling *et al.*, 1996] L. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[Lin, 1992] L.-J. Lin. Self-improving reactive agents based on reinforcement learning, planning and teaching. *Machine Learning*, 8(3-4):293–321, 1992.

[Mannor *et al.*, 2004] S. Mannor, I. Menache, A. Hoze, and U. Klein. Dynamic abstraction in reinforcement learning via clustering. In *Proc. of the 21st ICML*, pages 71–78, 2004.

[McGovern, 2002] A. McGovern. *Autonomous Discovery of Temporal Abstractions From Interactions With An Environment*. PhD thesis, University of Massachusetts Amherts, May 2002.

[Menache *et al.*, 2002] I. Menache, S. Mannor, and N. Shimkin. Q-cut - dynamic discovery of sub-goals in reinforcement learning. In *Proc. of the 13th ECML*, pages 295–306, 2002.

[Ravindran and Barto, 2001] B. Ravindran and A. G. Barto. Symmetries and model minimization in markov decision processes. Technical Report 01-43, University of Massachusetts, Amherst, 2001.

[Ravindran and Barto, 2002] B. Ravindran and A. G. Barto. Model minimization in hierarchical reinforcement learning. In *Proc. of the 5th SARA*, pages 196–211, London, UK, 2002. Springer-Verlag.

[Simsek *et al.*, 2005] O. Simsek, A. P. Wolfe, and A. G. Barto. Identifying useful subgoals in reinforcement learning by local graph partitioning. In *Proc. of the 22nd ICML*, 2005.

[Stolle and Precup, 2002] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Proc. of the 5th SARA*, pages 212–223, 2002.

[Sutton and Barto, 1998] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 1998.

[Sutton *et al.*, 1999] R. S. Sutton, D. Precup, and S. Singh. Between MDPs and semi-MDPs: a framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[Zinkevich and Balch, 2001] M. Zinkevich and T. R. Balch. Symmetry in markov decision processes and its implications for single agent and multiagent learning. In *Proc. of the 18th ICML*, pages 632–, 2001.