10

Controlled Use of Subgoals in Reinforcement Learning

Junichi Murata
Kyushu University
Japan

1. Introduction

Reinforcement learning (Kaelbling et al., 1996; Sutton & Barto, 1998) is a machine learning technique that automatically acquires a good action policy, i.e. a mapping from the current state to a good action to take, through trials and errors. A learning agent observes the state of its environment, chooses an action based on its current policy and executes the action. Responding to the action, the environment transitions to a new state, and a reword is given to the agent when applicable. The reward indicates how good or how bad the new state is, and the agent uses it to improve its policy so that it can obtain more rewards. Since reinforcement learning (abbreviated as RL hereafter) requires no other information, e.g. a model of environment, than the perceived states and rewards, it can be applied to a class of problems where the environment is complex or uncertain. The applications of RL include control of multi-legged robots (Kimura et al., 2001; Zennir at al., 2003), navigation of mobile robots (Millan, 1995), elevator hall call assignment (Crites & Barto, 1998; Kamal et al., 2005; Kamal and Murata, 2008) and board games (Tesauro, 1994). However, the learning agent must perform a large number of action trials in order to collect sufficient information about the unknown environment and organize it into a good policy. This takes a long time and therefore puts a limit on practical application of RL.

A number of techniques have been proposed to make RL faster. Most of them utilize *a priori* information on the target problem to compensate shortage of information and thus save the agent from a large number of trials. Smart and Kaelbling proposed using records of successful state-action pairs achieved by a human operator or a controller (Smart & Kaelbling, 2000). In the user-guided reinforcement learning of robots by Wang et al., a user teaches the robot a good action in real time (Wang et al., 2003). Driessens and Džeroski used a policy given by the human designer (Driessens & Džeroski, 2004). In all of the above research, the *a priori* information is in the form of good actions. Good states, however, are easier to find than good actions; you only need to find what the environment should be like but not how the agent can achieve that.

The technique described in this chapter focuses on *subgoals* as a form of good-state-related *a priori* information. A subgoal is a state or a subset of states that must be visited on the way from the initial state to the final goal state. With subgoals, the original problem can be divided into a set of small subproblems and this makes the learning faster. Use of subgoals in RL has been proposed by researchers, and it is closely related to (hierarchical) task partition and abstraction of actions. Singh used known subgoals to accelerate RL (Singh,

Source: Robotics, Automation and Control, Book edited by: Pavla Pecherková, Miroslav Flídr and Jindřich Duník,
ISBN 978-953-7619-18-3, pp. 494, October 2008, I-Tech, Vienna, Austria

1992). Wiering and Schmidhuber proposed HQ-Learning which learns subgoals to convert a POMDP (partially observable markov decision process) problem to a sequence of MDP problems (Wiering & Schmidhuber, 1996). HASSLE by Bakker and Schmidhuber learns subgoals for efficient RL (Bakker & Schmidhuber, 2004). A method was proposed by McGovern and Barto for discovering subgoals to create 'options' or temporally-extended actions (McGovern & Barto, 2001), and an improved method by Kretchmar et al. (Kretchmar et al., 2003). Subgoal information is either given by human designers/operators or acquired by the agent itself through learning. The idea of automatic acquisition of subgoals by learning (Wiering & Schmidhuber, 1996; McGovern & Barto, 2001; Kretchmar et al., 2003; Bakker & Schmidhuber, 2004) is very interesting. This learning itself, however, requires a considerable amount of time, and therefore is not very fascinating from the overall learning time point of view. On the other hand, it is not unrealistic that human designers/operators give the agents their *a priori* knowledge since there is usually some sort of *a priori* information available about the target problems. This kind of *a priori* information is usually ready before starting learning and does not require additional learning time. But, instead, there is a possibility that the humans' *a priori* knowledge is not perfect; it may contain errors and/or ambiguity.

In this chapter a technique is proposed that utilizes human-supplied subgoal information to accelerate RL. The most prominent feature of the proposed technique is that it has a mechanism to control use of subgoals; it keeps watching the effect of each subgoal on the action selection, and when it detects redundancy or harmfulness of a subgoal, it gradually stops using the subgoal. The technique is an extension of the one proposed by the author previously (Murata et al., 2007). Unlike the previous one, the redundancy and the harmfulness of subgoals are treated separately and thus finer control of use of subgoals is achieved. Moreover, after a sufficient number of learning iterations, we can tell which subgoal is useful and which is harmful by looking at the values of parameters that control use of subgoals.

These features are illustrated by simulation results using grid worlds with doors as example environments where the learning agent tries to find a path from the start cell to the goal cell. It is shown that use of exact subgoal information makes RL ten times faster in these rather simple problems. More acceleration can be expected in more complex problems. When a wrong subgoal is given and used without proper control, the agent can hardly find the optimal policy. On the other hand, with the proposed control mechanism enabled, the wrong subgoal is properly controlled, and its existence does not obstruct acquiring optimal actions. In this case, the learning is delayed, but the delay is not significant. It is also confirmed that the values of parameters introduced to control use of subgoals successfully display harmfulness/usefulness of subgoals. A wrong subgoal is not necessarily harmful; in some problem settings, it helps the agent find the optimal policy, and this is also properly reflected in the derived parameter values.

The remaining part of the chapter is organized as follows. In the next section, more descriptions will be given to the subgoals in RL, and a technique will be proposed to use them with proper control. In Section 3, several examples will be shown that illustrate the validity of the proposed method, which is followed by conclusions in Section 4.

2. Controlled use of subgoals in reinforcement learning

2.1 Subgoals in reinforcement learning problems

A subgoal is a state or a subset of states that must be visited on the way from the initial state to the final goal state. Since we assume subgoal information is provided by humans it should be more adequate to define a subgoal here as 'a state or a subset of states that *the human designer/operator thinks* must be visited on the way from the initial state to the final goal state', which implies that the subgoals can be erroneous.

A subgoal divides the original problem into two subproblems with itself as the boundary: one subproblem where the agent is to find a policy that leads the agent/environment from the initial state to the subgoal state, and the other subproblem from the subgoal to the final goal state. Each subproblem is smaller than the original problem, and therefore it is easier for the agent to reach the goal of the subproblem by random actions, which results in a smaller number of trials to find a good policy. In this way, with subgoals, the original problem can be divided into a set of small subproblems, and this makes the learning faster. Imagine, for example, that you have an assistant robot in your office which learns its action policy by RL. Suppose that you order the robot to deliver a document to your colleague in an office upstairs. The robot tries to find a path from your office to the colleague's office by random wandering. But it will take a very long time to find the office upstairs by chance. Alternatively, you can give an additional instruction 'take an elevator', which means a subgoal 'being in an elevator' is given. With this instruction, the robot will first try to find an elevator and then try to find a path from the elevator to your colleague's office. This will be much easier to accomplish. The instruction 'take an elevator' can be valid in other situations like fetching a document from your colleague's office. In this sense, subgoals can be general information that is useful in different but similar problems. However, your robot may go downstairs even if it successfully arrives at the subgoal 'being in an elevator'. Or your building may have two or more elevators. Or even it may happen that you forget that your colleague has recently moved to an office on your floor. Subgoals can be a good guiding information but they may contain ambiguity and errors when given by humans.

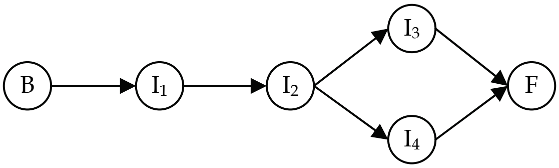


Fig. 1. Subgoals and their structure

Structure of subgoals can be represented by a directed graph as shown in Fig. 1 where B denotes the initial state and F the (final) goal state, and I_i, i=1, …, 4 stand for the subgoals. This figure indicates that, to reach the final goal F, the subgoal I_1 must be visited first, next the subgoal I_2, and then either I_3 or I_4 must be visited. Subgoals I_3 and I_4 are in parallel implying either subgoal must be visited, and I_1 and I_2 are in series indicating the subgoals must be visited in this order. In the following, terms *upstream subgoals* and *downstream subgoals* will be used. *Upstream subgoals* of a subgoal I_i are those subgoals closer to the initial state B than the subgoal I_i in the directed graph, and *downstream subgoals* of subgoal I_i are those closer to the final goal state F. In Fig. 1, subgoals I_1 is an upstream subgoal of subgoal I_2, and I_3 and I_4 are downstream subgoals of I_2.

Subgoals are presumably *good* states and the learning agent must be aware of this. Here, the goodness of a subgoal is represented and conveyed to the agent as a reward. This reward is not the real reward given by the environment but a *virtual* reward given by the agent itself. The received virtual rewards must be used to guide the action selection through value functions. We have to be careful because the subgoals may contain errors and ambiguity and also because an already achieved subgoal becomes no longer useful. A wrong or useless subgoal can mislead the agent in the wrong direction or does not play any significant role in policy update. If a subgoal turns out to be harmful, we have to stop using it in learning. If a subgoal is found to be redundant, it is better to cease to use it. Therefore, we need to treat the real reward and the virtual reward associated with each subgoal separately so that we can control use of each subgoal independently. Besides, by dealing with harmfulness and redundancy separately, we can detect harmful subgoals that have been mistakenly included by human designer/operator.

2.2 Controlled use of subgoals

Let us assume that we have n subgoals I_1, …, I_n. The agent gives a virtual reward $r_{i,t}$ with respect to subgoal I_i to itself at time t which takes a positive value when state at t is in subgoal I_i and is equal to zero otherwise. The real reward given by the environment at time t is denoted by r_t. To treat these real and virtual rewards separately, let us define a distinct action-value function based on each of them as follows:

$$Q(s_t, a_t) = \sum_{k=0}^{\infty} \gamma^k r_{t+1+k}, \tag{1}$$

$$Q_i(s_t, a_t) = \sum_{k=0}^{\infty} \gamma^k r_{i,t+1+k}, \quad i = 1, \cdots, n, \tag{2}$$

where s_t and a_t are state and action at time t, respectively, and $\gamma \in (0,1)$ is discount factor. Before learning, these action-value functions are initialized as zero, and then updated by the standard Q-learning:

$$Q(s_t, a_t) := (1-\alpha)Q(s_t, a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1}, a)\right], \tag{3}$$

$$Q_i(s_t, a_t) := (1-\alpha)Q_i(s_t, a_t) + \alpha\left[r_{i,t+1} + \gamma \max_a Q_i(s_{t+1}, a)\right], \quad i = 1, \cdots, n, \tag{4}$$

where operator ':=' means substitution and $\alpha > 0$ is learning rate.

Usually, when a subgoal is achieved, the agent can concentrate on seeking the next subgoal and then the next. For example, in Fig. 1, when subgoal I_1 is achieved, the agent only need to try to find subgoal I_2, and when this is also achieved, subgoal I_3 or I_4 is the next target. In this way, the target subgoal is switched from one to another. In other words, the original problem is hierarchically divided into distinct subproblems. However this is relevant only when the given subgoal information is correct. In the situations of our interest, the subgoal information is not perfect, and therefore the next subgoal can be a wrong subgoal. If I_2 is a wrong subgoal, it will be better, once I_1 is achieved, to seek I_3 or I_4 than I_2. In some cases, the order of subgoals may be wrong. In Fig. 1., for example, subgoal I_2 may be actually the first subgoal to be achieved before I_1. Therefore, we cannot employ the hierarchical problem partition. We must consider not the *next* subgoal only but *all* the subgoals in learning. In accordance with the above discussion, the actions are selected based on the following composite action-value function that consists of the action-value function calculated from the real rewards and all the action-value functions computed from the virtual rewards:

$$Q^A(s,a) = Q(s,a) + \sum_{i=1}^{n} c_i d_i Q_i(s,a), \tag{5}$$

where suffix t is dropped for simplicity. Positive coefficients c_i and d_i, i = 1, …, n are introduced to control use of subgoals. Coefficient c_i is used specifically to control use of subgoal when it is redundant while d_i is for regulating subgoal when it is harmful. They are initialized to 1.0, i.e. at the beginning, all the virtual rewards are considered as equally strongly as the real reward in action selection. Actual action is derived by applying an appropriate exploratory variation such as ε-greedy and softmax to the action that maximizes $Q^A(s,a)$ for the current state s. Therefore, learning of $Q(s,a)$ by equation (3) is an *off-policy* learning, and its convergence is assured just like the ordinary Q-learning on the condition that all state-action pairs are visited infinitely often. However, our interest is not in the convergence of $Q(s,a)$ for all state-action pairs but in avoiding visiting unnecessary state-action pairs by appropriately controlled use of subgoals.

When a subgoal I$_i$ is found to be either redundant or harmful, its corresponding coefficient c_i or d_i is decreased to reduce its contribution to the action selection.

A subgoal I$_i$ is redundant in state s when the optimal action in state s towards this subgoal I$_i$ is identical to the optimal action towards the final goal F or towards another subgoal I$_j$, $j \in R_i$, where R_i is the set of suffixes of subgoals that are reachable from subgoal I$_i$ in the directed graph. In other words, subgoal I$_i$ is redundant if, without help of the subgoal, the agent can find the optimal action that leads to the final goal or a downstream subgoal of subgoal I$_i$ which is closer to the final goal and thus more important. Let us define $\widetilde{Q}(s,a)$ as a sum of $Q(s,a)$ and those $Q_j(s,a)$ associated with the downstream subgoals of subgoal I$_i$,

$$\widetilde{Q}_i(s,a) = Q(s,a) + \sum_{j \in R_i} c_j d_j Q_j(s,a). \tag{6}$$

Then the optimal action in state s towards the downstream subgoals and the final goal is given by

$$\widetilde{a}_i^*(s) = \arg\max_a \widetilde{Q}_i(s,a), \tag{7}$$

and the optimal action towards subgoal I$_i$ in state s by

$$a_i^*(s) = \arg\max_a Q_i(s,a). \tag{8}$$

The relationship between subgoals and action-value functions is illustrated in Fig. 2. If $Q_i(s,a)$ or $\widetilde{Q}_i(s,a)$ is zero or negative for any a, it means that sufficient positive real rewards or sufficient virtual rewards associated with I$_j$, $j \in R_i$ have not been received yet and that the *optimal actions* given by equations (7) and (8) are meaningless. So, we need the following preconditions in order to judge redundancy or harmfulness of a subgoal in state s:

$$\exists a, Q_i(s,a) > 0 \quad \text{and} \quad \exists a, \widetilde{Q}_i(s,a) > 0\,. \tag{9}$$

Now, we can say that subgoal I*ᵢ* is redundant in state *s* when the following holds:

$$a_i^*(s) = \tilde{a}_i^*(s). \tag{10}$$

When subgoal I_i is found to be redundant in state *s*, its associated coefficient c_i is reduced by a factor $\beta \in (0,1)$:

$$c_i := \beta c_i. \tag{11}$$

Coefficient c_i is not set to zero at once because we have found that subgoal I_i is redundant in this particular state *s* but it may be useful in other states. Note that another coefficient d_i is kept unchanged in this case. Although the composite action-value function $Q^A(s,a)$ used for the action selection includes the terms related *upstream* subgoals of subgoal I_i, we do not consider them in reducing c_i. The upstream subgoals are less important than subgoal I_i. Preconditions (9) mean that subgoal I_i has already been achieved in the past trials. Then, if subgoal I_i and any of the less important subgoals play the same role in action selection, i.e. either of them is redundant, then it is the coefficient associated with that less important upstream subgoal that must be decreased. Therefore the redundancy of subgoal I_i is checked only against its *downstream* subgoals.

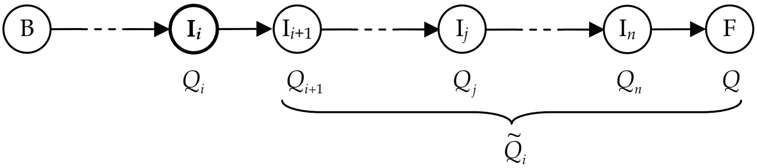


Fig. 2. Relationship between subgoals and action-value functions

A subgoal I_i is harmful in state *s* if the optimal action towards this subgoal is different from the optimal action towards the final goal or towards another subgoal I_j, $j \in R_i$, i.e. the action towards subgoal I_i contradicts with the action towards the final goal or a downstream subgoal. This situation arises when the subgoal is wrong or the agent attempts to go back to the subgoal seeking more virtual reward given there although it has already passed the subgoal. Using $a_i^*(s)$ and $\tilde{a}_i^*(s)$ above, we can say a subgoal I_i is harmful in state *s* if

$$a_i^*(s) \neq \tilde{a}_i^*(s), \tag{12}$$

and the preconditions (9) are satisfied. When a subgoal is judged to be harmful in state *s*, its associated coefficient d_i is reduced so that the subgoal does less harm in action selection. In this case coefficient c_i remains unchanged. Let us derive a value of d_i that does not cause the conflict (12). Such value of d_i, denoted by d_i^o, must be a value such that the action selected by maximizing $c_i d_i^o Q_i(s,a) + \tilde{Q}_i(s,a)$ does not differ from the action selected by $\tilde{Q}_i(s,a)$ only. So, the following must hold for state *s*,

$$\arg\max_a \left[c_i d_i^o Q_i(s,a) + \tilde{Q}_i(s,a)\right] = \arg\max_a \tilde{Q}_i(s,a). \tag{13}$$

Considering equation (7), the above equation (13) holds when

$$c_i d_i^o Q_i(s,a) + \widetilde{Q}_i(s,a) \le c_i d_i^o Q_i(s,\widetilde{a}_i^*(s)) + \widetilde{Q}_i(s,\widetilde{a}_i^*(s)) \tag{14}$$

is satisfied for all a. Then, by straightforward calculation, the value of d_i^o that assures the above inequality (14) is derived as

$$d_i^o = \min_{a \in A_i(s)} \frac{1}{c_i} \frac{\widetilde{Q}_i(s,\widetilde{a}_i^*(s)) - \widetilde{Q}_i(s,a)}{Q_i(s,a) - Q_i(s,\widetilde{a}_i^*(s))}, \quad A_i(s) = \left\{ a \,\middle|\, Q_i(s,a) > Q_i(s,\widetilde{a}_i^*(s)) \right\}. \tag{15}$$

In equation (15) we restrict actions to those belonging to set $A_i(s)$. This is because for actions which satisfy inequality $Q_i(s,a) \le Q_i(s,\widetilde{a}_i^*(s))$, inequality (14) naturally holds for any d_i since $c_i d_i > 0$ and $\widetilde{Q}_i(s,a) \le \widetilde{Q}_i(s,\widetilde{a}_i^*(s))$ from the definition of $\widetilde{a}_i^*(s)$ in equation (7). Now d_i is slightly reduced so that it approaches d_i^o by a fraction of δ_i:

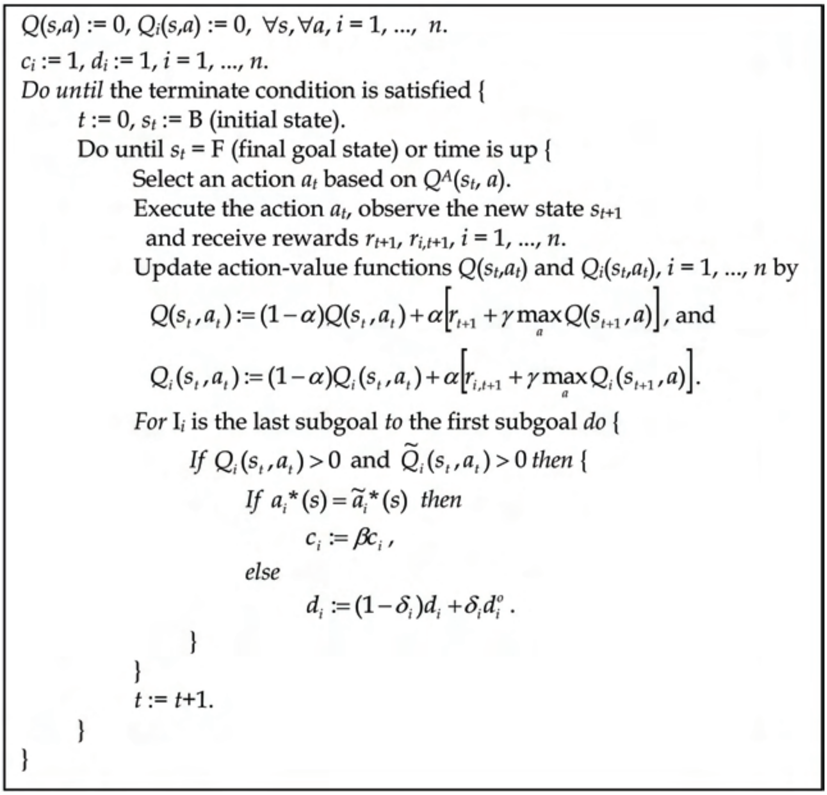
$$d_i := (1-\delta_i) d_i + \delta_i d_i^o, \tag{16}$$

where δ_i is a small positive constant. There is a possibility that the original value of d_i is already smaller than d_i^o. In that case, d_i is not updated. Coefficient d_i is not reduced to d_i^o at once. We have observed a conflict among the subgoal I_i and a downstream subgoal (or the final goal itself), and it seems that we need to reduce the coefficient d_i for subgoal I_i to solve the conflict. The observed conflict is genuine on the condition that the action-value functions Q_i, Q_j, $j \in R_i$ and Q used to detect the conflict are sufficiently correct (in other words they are well updated). Therefore, in the early stage of learning, the observed conflict can be non-authentic. Even if the conflict is genuine, there is a situation where d_i is not to be reduced. Usually a downstream subgoal of subgoal I_i is more important than I_i, and therefore the conflict must be resolved by changing the coefficient associated with the subgoal I_i. However, when the downstream subgoals are wrong, reducing the coefficient associated with the subgoal I_i is irrelevant. These possibilities of non-genuine conflict and/or wrong downstream subgoals demand a cautious reduction of d_i as in equation (16). Moreover, to suppress possible misleading by wrong downstream subgoals, parameter δ_i is set smaller for upstream subgoals because a subgoal located closer to the initial state has a more number of downstream subgoals and therefore is likely to suffer from more undesirable effect caused by wrong subgoals.

Because update of d_i depends on *downstream* coefficients c_j and d_j, $j \in R_i$ contained in $\widetilde{Q}_i$, the update is done starting with the last subgoal namely the subgoal closest to the final goal to the first subgoal that is the closest to the initial state.

The overall procedure is described in Fig. 3. Action-values Q and Q_i are updated for s_t and a_t, and then it is checked if these updates have made the subgoal I_i redundant or harmful. Here the action-values for other state-action pairs remain unchanged, and thus it suffices that the preconditions (9) are checked for s_t and a_t only.

Each of coefficients c_i, i= 1, …, n represents non-redundancy of its associated subgoal, while d_i reflects harmlessness of the subgoal. All of coefficients c_i eventually tend to zero as the learning progresses since the agent does not need to rely on any subgoal once it has found an optimal policy that leads the agent and environment to the final goal. On the other hand, the value of d_i depends on the property of its associated subgoal; d_i remains large if its corresponding subgoal is not harmful while d_i associated with a harmful subgoal decreases to zero. Therefore, by inspecting the value of each d_i when the learning is complete, we can find which subgoal is harmful and which is not.



```
$Q(s,a) := 0,\ Q_i(s,a) := 0,\ \forall s, \forall a,\ i = 1, \ldots, n.$
$c_i := 1,\ d_i := 1,\ i = 1, \ldots, n.$
Do until the terminate condition is satisfied {
    $t := 0,\ s_t :=$ B (initial state).
    Do until $s_t =$ F (final goal state) or time is up {
        Select an action $a_t$ based on $Q^A(s_t, a)$.
        Execute the action $a_t$, observe the new state $s_{t+1}$
          and receive rewards $r_{t+1}, r_{i,t+1},\ i = 1, \ldots, n$.
        Update action-value functions $Q(s_t,a_t)$ and $Q_i(s_t,a_t)$, $i = 1, \ldots, n$ by
          $Q(s_t,a_t) := (1-\alpha)Q(s_t,a_t) + \alpha\left[r_{t+1} + \gamma \max_a Q(s_{t+1},a)\right]$, and
          $Q_i(s_t,a_t) := (1-\alpha)Q_i(s_t,a_t) + \alpha\left[r_{i,t+1} + \gamma \max_a Q_i(s_{t+1},a)\right]$.
        For $\mathrm{I}_i$ is the last subgoal to the first subgoal do {
            If $Q_i(s_t,a_t) > 0$ and $\tilde{Q}_i(s_t,a_t) > 0$ then {
                If $a_i^*(s) = \tilde{a}_i^*(s)$ then
                    $c_i := \beta c_i$,
                else
                    $d_i := (1-\delta_i)d_i + \delta_i d_i^o$.
            }
        }
        $t := t+1$.
    }
}
```

Fig. 3. Learning procedure

3. Examples

The proposed technique is tested on several example problems where an agent finds a path from the start cell to the goal cell in grid worlds. The grid worlds have several doors each of which requires a fitting key for the agent to go through it as shown in Fig. 4. The agent must pick up a key to reach the goal. Therefore having a key, or more precisely having just picked up a key, is a subgoal. The state consists of the agent's position (x-y coordinates) and which key the agent has. The agent can move to an adjacent cell in one of four directions (north, south, east and west) at each time step. When the agent arrives at a cell where a key exists, it picks up the key. Key 1 opens door 1, and key 2 is the key to door 2. The agent receives a reward 1.0 at the goal cell F and also a virtual reward 1.0 at the subgoals. When it selects a move to a wall or to the boundary, a negative reward –1.0 is given and the agent stays where it was. An episode ends when the agent reaches the goal cell or 200 time steps have passed.

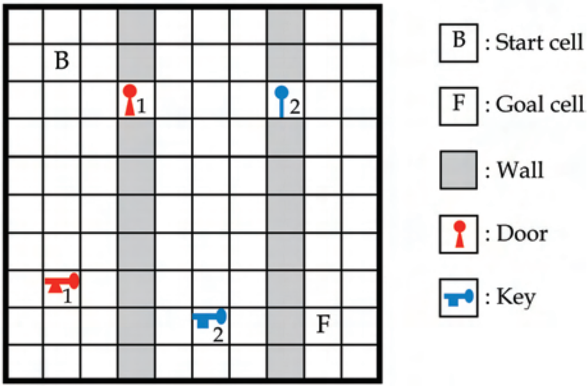


Fig. 4. Grid world 1

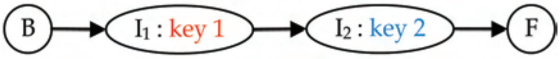


Fig. 5. Subgoal structure of grid world 1

3.1 Effect of use of correct subgoals

The subgoals in the example depicted in Fig. 4 can be represented by a directed graph shown in Fig.5. In RL, the first arrival at the goal state must be accomplished by random actions because the agent has no useful policy yet. Since the agent has to collect two keys to go through the two doors in this example, it takes a large number of episodes to arrive at the final goal by random actions only. Here we are going to see how much acceleration of RL we will have by introducing correct subgoals.

Q-learning is performed with and without taking the subgoals into consideration. The parameters used are as follows: discount factor γ=0.9, learning rate α=0.05, β in equation (11) is 0.99, and decreasing rates δ_i of coefficient d_i is 0.005 for subgoal I_1 and 0.01 for I_2. Softmax action selection is used with ‘temperature parameter’ being 0.1.

The numbers of episodes required for the agent to reach the goal for the first time by greedy action based on the learnt Q^A (i.e. the action that maximizes Q^A) and the numbers of episodes necessary to find an optimal (shortest) path to the goal are listed in Table 1. These are averages over five runs with different pseudo random number sequences. The table indicates that consideration of the correct subgoals makes the learning more than ten times faster in this small environment, which verifies the validity of introducing correct subgoals to accelerate RL. Also more acceleration can be expected for larger or more complex environments.

	Number of episodes	
	First arrival at the goal	Finding an optimal path
Without subgoals	11861.0	13295.8
With subgoals	1063.2	1068.0

Table 1. Numbers of episodes required before achieving the goal (grid world 1)

3.2 Effect of controlled use of subgoals

Now let us turn our attention to how the control of use of subgoals by coefficients c_i and d_i works. Here we consider another grid world shown in Fig. 6 where key 1 is the only correct key to the door and key 2 does not open the door. We apply the proposed method to this problem considering each of subgoal structures shown in Fig. 7. In this figure, subgoal structure (a) is the exact one, subgoal structure (b) has a wrong subgoal only, subgoal structures (c) and (d) have correct and wrong subgoals in series and subgoal structure (e) has correct and wrong subgoals in parallel. The same values are used as in the previous subsection for the parameters other than δ_i. For a single subgoal in (a) and (b) δ_1 is set to 0.01, for series subgoals in (c) and (d) δ_1=0.005 and δ_2=0.01 are used, and for the parallel subgoals in (e) 0.01 is used for both δ_1 and δ_2.

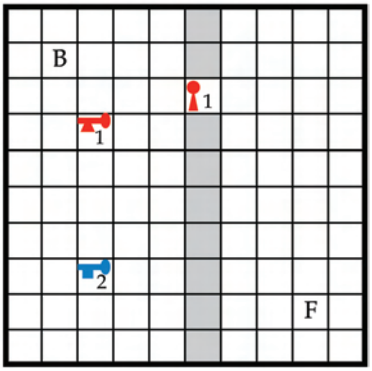
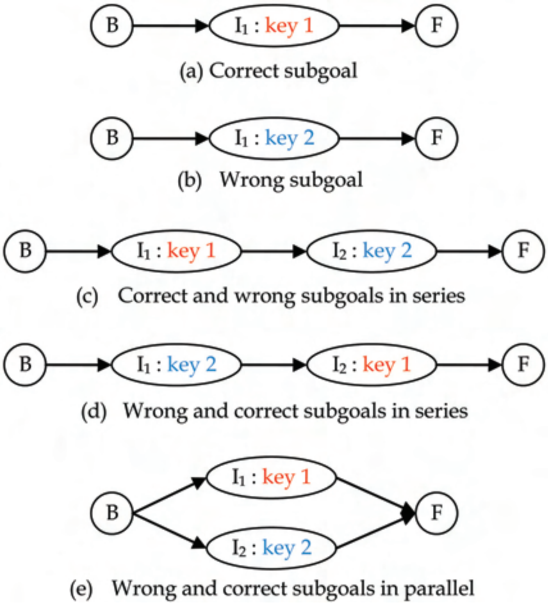


Fig. 6. Grid world 2 with a correct key and a wrong key



(a) Correct subgoal

(b) Wrong subgoal

(c) Correct and wrong subgoals in series

(d) Wrong and correct subgoals in series

(e) Wrong and correct subgoals in parallel

Fig. 7. Possible subgoal structures for grid world 2

The numbers of episodes before the first arrival at the goal and before finding an optimal path are shown in Table 2 together with the values of coefficients d_i after learning and the ratio of d_i for the correct subgoal (d_{correct}) to d_i for the wrong subgoal (d_{wrong}) where available. All of these are averages over five runs with different pseudo random number sequences.

Subgoals used in learning	Number of episodes		Coefficients d_i after learning		
	First arrival at the goal	Finding an optimal path	For correct subgoal (d_{correct})	For wrong subgoal (d_{wrong})	$d_{\text{correct}} / d_{\text{wrong}}$
None	99.4	103.2	-	-	-
Correct	76.2	79.0	2.61×10^{-1}	-	-
Wrong	139.0	206.8	-	9.79×10^{-5}	-
Correct & wrong in series	116.6	180.0	3.48×10^{-1}	3.52×10^{-5}	1.30×10^{7}
Wrong & correct in series	87.4	97.8	4.15×10^{-2}	7.06×10^{-3}	1.37×10^{5}
Correct & wrong in parallel	116.8	163.4	9.85×10^{-2}	2.21×10^{-4}	1.03×10^{8}

Table 2. Numbers of episodes required before achieving the goal (grid world 2)

With the exact subgoal information given, the agent can reach the goal and find the optimal path faster than the case without considering any subgoal. When a wrong subgoal is provided in place of / in addition to the correct subgoal, the learning is delayed. However, the agent can find the optimal path anyway, which means that introducing a wrong subgoal does not cause a critical damage and that the proposed subgoal control by coefficients c_i and d_i works well. Finding the optimal path naturally takes more episodes than finding any path to the goal. The difference between them is large in the cases where wrong subgoal information is provided. This is because the coefficient associated with the wrong subgoal does not decay fast enough in those cases. The preconditions (9) for reducing the coefficient demand that the subgoal in question as well as at least one of its downstream subgoals have been already visited. Naturally the subgoals closer to the initial state in the state space (not in the subgoal structure graph) are more likely to be visited by random actions than those far from the initial state. In this grid world, the correct key 1 is located closer to the start cell than the wrong key 2 is, and therefore the correct subgoal decays faster and the wrong subgoal survives longer, which causes more delay in the learning.

Coefficients d_i are used to reduce the effect of harmful subgoals. Therefore, by looking at their values in Table 2, we can find which subgoal has been judged to be harmful and which has not. Each of the coefficients d_i for the correct subgoals takes a value around 0.1 while each of those for the wrong subgoals is around 10^{-4}. Each ratio in the table is larger than 10^5. Thus the coefficients d_i surely reflect whether their associated subgoals are harmful or not. In Table 2, the coefficient for the wrong subgoal in the case of 'wrong and correct subgoals in series' is 7.06×10^{-3} and is not very small compared with the value of 4.15×10^{-2} for the correct subgoal. This has been caused by just one large coefficient value that appeared in one of the five runs. Even in this run, the learning is successfully accomplished just like in other runs. If we exclude this single value from average calculation, the average coefficient value for this subgoal is around 10^{-6}.

To confirm the effect of subgoal control, learning is performed with the coefficient control disabled, i.e. both of c_i and d_i are fixed to 1.0 throughout the learning. In the case that the correct subgoal is given, the result is the same as that derived with the coefficient control. However, in other four cases where a wrong subgoal is given, the optimal path has not been found within 200000 episodes except for just one run in the five runs. Therefore, simply giving virtual rewards to subgoals does not work well when some wrong subgoals are included. When either c_i or d_i is fixed to 1.0 and the other is updated in the course of learning, similar results to those derived by updating both coefficients are obtained, but the learning is delayed when wrong subgoal information is provided. In composite action-value function Q^A used in action selection, each action-value function Q_i associated with subgoal I_i is multiplied by a product of c_i and d_i. The product decreases as the learning proceeds, but its speed is slow when either c_i or d_i is fixed. A large product of c_i and d_i makes the 'attractive force' of its corresponding subgoal strong, and the agent cannot perform a bold exploration to go beyond the subgoal and find a better policy. Then harmfulness of a subgoal cannot be detected since the agent believes that visiting that subgoal is a part of the optimal path and does not have another path to compare with in order to detect a conflict. Therefore, coefficient c_i must be reduced when its associated subgoal is judged to be redundant to help agent to explore the environment and find a better policy. The above results and observation verifies that the proper control of use of subgoals is essential.

3.3 Effect of subgoals on problems with different properties

In the results shown in Table 2, the learning is not accelerated much even if the exact subgoal structure is given, and the results with wrong subgoal are not too bad. Those results of course depend on the problems to be solved. Table 3 shows the results for a problem where the positions of key 1 and key 2 are exchanged in grid world 2. Also the results for grid world 3 depicted in Fig. 8 are listed in Table 4. Here the correct and the wrong keys are located in the opposite directions from the start cell. The same parameter values are used in both examples as those used in the original grid world 2. The values in the tables are again averages over five runs with different pseudo random number sequences.

Subgoals used in learning	Number of episodes		Coefficients d_i after learning		
	First arrival at the goal	Finding an optimal path	For correct subgoal (d_{correct})	For wrong subgoal (d_{wrong})	$d_{\mathrm{correct}} / d_{\mathrm{wrong}}$
None	323.8	343.6	-	-	-
Correct	117.4	121.2	3.82×10^{-3}	-	-
Wrong	196.8	198.4	-	2.23×10^{-2}	-
Correct & wrong in series	188.2	189.6	6.84×10^{-3}	9.42×10^{-3}	2.53×10^{1}
Wrong & correct in series	117.2	126.8	4.37×10^{-5}	3.80×10^{-1}	9.47×10^{-4}
Correct & wrong in parallel	100.0	100.6	2.08×10^{-2}	9.57×10^{-3}	5.14

Table 3. Numbers of episodes required before achieving the goal (grid world 2 with keys exchanged)

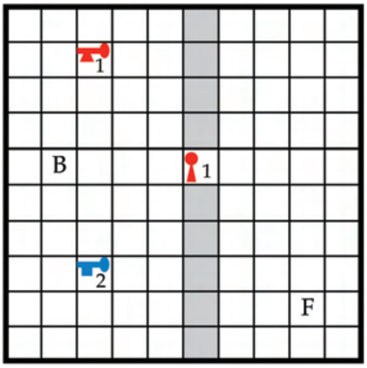


Fig. 8. Grid world 3 with two keys in opposite directions

By exchanging the two keys in grid world 2, the problem becomes more difficult than the original because the correct key is now far from the start cell. So, without subgoals, the learning takes more episodes, and introduction of subgoals is more significant than before as shown in Table 3. The wrong key is located on the way from the start cell to the correct key, and although picking up the wrong key itself has no useful meaning, the wrong subgoal guides the agent in the right direction towards the correct subgoal (correct key). Therefore the wrong subgoal information in this grid world is wrong but not harmful; it is even helpful in accelerating the learning as shown in Table 3. Also, since it is not harmful, coefficients d_i corresponding to the wrong subgoals remain large after the learning.

Subgoals used in learning	Number of episodes		Coefficients d_i after learning		
	First arrival at the goal	Finding an optimal path	For correct subgoal (d_{correct})	For wrong subgoal (d_{wrong})	$d_{\mathrm{correct}} / d_{\mathrm{wrong}}$
None	153.8	155.8	-	-	-
Correct	95.8	99.6	4.80×10^{-2}	-	-
Wrong	150.4	309.8	-	1.08×10^{-4}	-
Correct & wrong in series	170.2	346.8	1.84×10^{-3}	9.83×10^{-5}	7.32×10^{1}
Wrong & correct in series	107.2	109.0	2.01×10^{-3}	1.22×10^{-4}	4.98×10^{1}
Correct & wrong in parallel	106.4	226.0	6.04×10^{-3}	4.75×10^{-4}	1.05×10^{6}

Table 4. Numbers of episodes required before achieving the goal (grid world 3)

In contrast, the wrong key in grid world 3 lies in the opposite direction from the correct key. So, this wrong subgoal has worse effect on the learning speed as shown in Table 4. Here the coefficients d_i for the wrong subgoals are smaller than those for the correct subgoals.

For grid worlds 2 and 3, the actual subgoal structure is that shown in Fig. 7. (a). To investigate the performance of the proposed method on problems with parallel subgoals, key 2 in grid world 2 is changed to a key 1. So the environment now has two correct keys, and the actual subgoal structure is just like Fig. 7. (e) but both the keys are correct. Five different subgoal structures are considered here: 'near subgoal', 'far subgoal', 'near and far subgoals in series', 'far and near subgoals in series' and 'near and far subgoals in parallel' where 'near subgoal' denotes the subgoal state 'picking up key near the start cell', and 'far subgoal' refers to the subgoal 'picking up the key far from the start cell'. Note that there is no wrong subgoal in this grid world. The results shown in Table 5 are similar to those already derived. Introduction of subgoal(s) makes the goal achievement faster, but in some subgoal settings, finding the optimal path is slow. The subgoal structure 'near and far subgoals in parallel' is the exact one, but this gives the worst performance in finding the optimal path in the table. In this problem, both the keys correspond to correct subgoals, but one (near the start cell) is more preferable than the other, and the less-preferable subgoal survives longer in this setting as described in Section 3.2. This delays the learning.

Subgoals used in learning	Number of episodes		Coefficients d_i after learning	
	First arrival at the goal	Finding an optimal path	For near subgoal	For far subgoal
None	106.0	109.6	-	-
Near	76.2	79.0	2.62×10^{-1}	-
Far	136.2	203.8	-	2.96×10^{-3}
Near & far in series	126.6	205.2	4.06×10^{-1}	1.15×10^{-5}
Far & near in series	84.6	95.0	2.78×10^{-2}	7.06×10^{-3}
Near & far in parallel	116.4	169.6	7.95×10^{-2}	2.21×10^{-4}

Table 5. Numbers of episodes required before achieving the goal (grid world 2 with two correct keys)

Introduction of subgoals usually makes goal achievement (not necessarily by an optimal path) faster. But, a wrong or less-preferable subgoal sometimes makes finding the optimal path slower than the case without any subgoals considered, especially when it occupies a position far from the initial state. However, the wrong subgoals do not cause critically harmful effect such as impractically long delay and inability of finding the goal at all thanks to the proposed mechanism of subgoal control. Also we can find the harmful subgoals by inspecting the coefficient values used for subgoal control. This verifies the validity of the proposed controlled use of subgoals in reinforcement learning.

4. Conclusions

In order to make reinforcement learning faster, use of subgoals is proposed with appropriate control of each subgoal independently since errors and ambiguity are inevitable in subgoal information provided by humans. The method is applied to grid world examples and the results show that use of subgoals is very effective in accelerating RL and that, thanks to the proposed control mechanism, errors and ambiguity in subgoal information do not cause critical damage on the learning performance. Also it has been verified that the proposed subgoal control technique can detect harmful subgoals.

In reinforcement learning, it is very important to balance *exploitation*, i.e. making good use of information acquired by learning so far in action selection, with *exploration*, namely trying different actions seeking better actions or policy than those already derived by learning. In other words, a balance is important between what is already learnt and what is to be leant yet. In this chapter, we have introduced subgoals as a form of *a priori* information. Now we must compromise among leant information, information yet to be learnt and *a priori* information. This is accomplished, in the proposed technique, by choosing proper values for β and δ_i that control use of *a priori* information through coefficients c_i and d_i as well as an appropriate choice of exploration parameter such as 'temperature parameter' used in softmax that regulates exploration versus exploitation. A good choice of parameters may need further investigations. However, this will be done using additional *a priori* information such as confidence of the human designer/operator in his/her subgoal information. Also a possible extension of the method is to combine it with a subgoal learning technique.

5. Acknowledgements

The author would like to acknowledge the support for part of the research by the Japan Society for the Promotion of Science, Grant-in-Aid for Scientific Research (C), 16560354, 2004-2006.

6. References

Bakker, B. & Schmidhuber, J. (2004). Hierarchical Reinforcement Learning with Subpolicies Specializing for Learned Subgoals, *Proc. 2nd IASTED Int. Conf. Neural Networks and Computational Intelligence*, pp. 125-130.

Crites, R.H. & Barto, A.G. (1998). Elevator Group Control Using Multiple Reinforcement Learning Agents, *Machine Learning*, Vol. 33, pp. 235-262.

Driessens, K. & Džeroski, S. (2004). Integrating Guidance into Relational Reinforcement Learning, *Machine Learning*, Vol. 57, pp. 271-304.

Kaelbling, L.P.; Litman, M.L. & Moor, A.W. (1996). Reinforcement Learning: A survey, *J. of Artificial Intelligence Research*, Vol. 4, pp. 237-285.

Kamal, M.A.S.; Murata, J. & Hirasawa, K. (2005). Elevator Group Control Using Multiagent Task-Oriented Reinforcement Learning, *IEEJ Trans. EIS*, Vol. 125, pp. 1140-1146.

Kamal, M.A.S. & Murata, J. (2008). Reinforcement learning for problems with symmetrical restricted states, *Robotics and Autonomous Systems*, to appear.

Kimura, H.; Yamashita, T. & Kobayashi, S. (2001), Reinforcement Learning of Walking Behavior for a Four-Legged Robot, *Proc. 40th IEEE Conf. Decision and Control*, pp. 411-416.

Kretchmar, R.M.; Feil, T. & Bansal, R. (2003). Improved Automatic Discovery of Subgoals for Options in Hierarchical Reinforcement Learning, *J. Computer Science & Technology*, Vol. 3, pp. 9-14.

McGovern, A. & Barto, A.G. (2001). Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density, *Proc. 18th Int. Conf. Machine Learning*, pp. 361-368.

Millan, J.D. (1995). Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot, *Robotics and Autonomous Systems*, Vol. 15, pp. 275-299.

Murata , J.; Ota, K. & Abe, Y. (2007). Introduction and Control of Subgoals in Reinforcement Learning, *Proc. IASTED Conf. Artificial Intelligence and Applications*, pp. 329-334.

Singh, S. (1992). The Efficient Learning of Multiple Task Sequences, In: *Advances in Neural Information Processing Systems* 4, pp. 251-258, Morgan Kauffman, San Mateo, USA.

Smart, W.D. & Kaelbling, L.P. (2000). Practical Reinforcement Learning in Continuous Spaces, *Proc. 17th Int. Conf. Machine Learning*, pp. 903-910.

Sutton, R.S. & Barto, A.G. (1998). *Reinforcement Learning, An Introduction*, A Bradford Book, The MIT Press, Cambridge, USA.

Tesauro, G.J. (1994). TD-gammon, a self-teaching backgammon program, archives master-level play, *Neural Computation*, Vol. 6, pp. 215–219.

Wang, Y.; Huber, M.; Papudesi, V.N. & Cook, D.J. (2003). User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment, *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp. 424-429.

Wiering, M. & Schmidhuber, J. (1996). HQ-Learning: Discovering Markovian Subgoals for Non-Markovian Reinforcement Learning, *Tech. Rep. IDSIA-95-96*.

Zennir, Y.; Couturier, P. & Temps, M.B. (2003). Distributed Reinforcement Learnig of a Six-Legged Robot to Walk, *Proc. 4th Int. Conf. Control and Automation*, pp. 896-900.