INTRODUCTION AND CONTROL OF SUBGOALS IN REINFORCEMENT LEARNING

Junichi Murata, Yasuomi Abe and Keisuke Ota
Department of Electrical and Electronic Systems Engineering
Kyushu University
Fukuoka, Japan
email: murata@ees.kyushu-u.ac.jp

ABSTRACT
Reinforcement learning (RL) can be applied to a wide class of problems because it requires no other information than perceived states and rewards to find good action policies. However, it takes a large number of trials before acquiring the optimal policy. In order to make RL faster, use of subgoals is proposed. Since errors and ambiguity are inevitable in subgoal information provided by human designers, a mechanism is proposed that controls use of subgoals. The method is applied to examples and the results show that use of subgoals is very effective in accelerating RL and that the proposed control mechanism successfully suppresses possible critical damages on the RL performance caused by errors and ambiguity in subgoal information.

KEY WORDS
Reinforcement learning, subgoals, acceleration, Q-learning

1 Introduction

Reinforcement learning (RL) [1] acquires a good action policy through interaction with the environment: a learning agent takes an action, which causes transition of environment state, and the environment gives a reward to the agent that indicates how good or bad the new state is and used to modify the policy. Since RL requires no other information e.g. an environment model than perceived state and rewards, it can be applied to a wide class of problems. The price for this advantage is its slow learning: it needs a large number of trials of actions to collect and organize pieces of plain information of states and rewards.

A solution to the problem is giving additional information to the learning agent to save its large number of trials. It is not, however, expected that very rich information is available for RL-suited problems. In this paper, information of subgoals is used. A subgoal here is a state or a subset of states that is believed to have to be visited in order to reach the goal state starting from the initial state. Human designers can give this kind of information in a large number of problems. In addition, knowledge of subgoals is often general information; we can find common subgoals in some different but similar problems.

Some researchers suggested to utilize additional information given by humans. Smart and Kaelbling proposed to use records of successful state-action pairs achieved by a human operator or a controller [2]. In the user-guided reinforcement learning of robots by Wang *et al*, a user teaches the robot a good action in real time [3]. Driessens and Džeroski used a policy given by the human designer [4]. In all of the above research, the additional 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. Finding good subgoal states and using them is easier and more efficient. Use of subgoals in RL has been proposed by researchers, and it is closely related to (hierarchical) decomposition of task and abstraction of actions. Singh used known subgoals to accelerate RL [5]. 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 [6]. HASSLE by Bakker and Schmidhuber learns subgoals for efficient RL [7]. A method was proposed by McGovern and Barto for discovering subgoals to create 'options' or temporally-extended actions [8], and an improved method by Kretchmar *et al* [9]. The subgoals in the above publications are used to partition the original problem into a set of distinct smaller problems, and thus the learning agent seeks one subgoal at a time.

In this paper, subgoals are used to accelerate RL. It is possible as proposed in [6, 7, 8, 9] to find subgoals by learning. But, in principle, every state can be a subgoal, and finding subgoals (especially multiple subgoals) by learning may take, unless some useful information on the subgoals is available, as long a time as the original RL itself. For efficient RL, subgoals are assumed here to be provided by humans. It is inevitable that errors and ambiguity are contained in subgoals provided by humans, and we cannot rely on them well enough to decompose the task or employ a hierarchical structure. Also it is risky for the learning agent to seek only one specific subgoal at any time because that subgoal can be wrong. Therefore, single-level flat problem structure is adopted in this paper, and a technique is proposed to appropriately control use of subgoals and suppress possible undesirable effects by errors and ambiguity.

In the next section, more descriptions will be given to the subgoals in RL, and their control technique will be proposed. In section 3, several examples are shown that

illustrate the validity of the proposed method, which is followed by conclusions in section 4.

2 Introduction and Control of Subgoals in Reinforcement Learning

2.1 Subgoals in Reinforcement Learning Problems

A subgoal here is a state or a subset of states that is believed to have to be visited in order to reach the goal state starting from the initial state. Here, by the word 'believed', it is implied that subgoals can be erroneous.

Imagine that you are in your office and you order an assistant robot to deliver a document to your colleague in an office upstairs. 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. But without this instruction, your robot will go round and round and take a long time to reach its destination. 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 reaches 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.

The structure of subgoals can be represented by a directed graph as shown in Figure 1 where B denotes the initial state and F the (final) goal state, and I_i, $i = 1, 2, 3$ stand for the subgoals. This figure indicates that, to reach the final goal F, the subgoal I_1 must be visited first, and then either I_2 or I_3 must be visited. Subgoals I_2 and I_3 are in parallel, and I_1 and I_2 (or I_3) are in series.

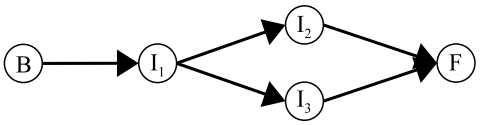

Figure 1. Subgoals and their structure.

Subgoals are '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. Here, we have to be careful because the subgoals may contain errors and ambiguity and because an already achieved subgoal becomes no longer useful. If a subgoal turns out to be wrong, we have to cease to use it in learning. If a subgoal is found to be useless or redundant, it is better to stop using 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 of them independently.

2.2 Control of Use of Subgoals

Let us assume that we have n subgoals $\mathrm{I}_1, \cdots \mathrm{I}_n$. The agent gives a positive virtual reward $r_{i,t}$ to itself when it is in subgoal I_i at time t. The real reward given by the environment at time t is denoted by r_t. To treat these real/virtual rewards separately, let us define action-value functions as follows:

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

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

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 [1]:

$$Q(s_t, a_t) \leftarrow (1-\alpha) Q(s_t, a_t) + \alpha\{r_{t+1} + \gamma \max_a Q(s_{t+1}, a)\}, \quad (3)$$

$$Q_i(s_t, a_t) \leftarrow (1-\alpha) Q_i(s_t, a_t) + \alpha\{r_{i,t+1} + \gamma \max_a Q_i(s_{t+1}, a)\}, \ i = 1, \cdots n, \quad (4)$$

where $\leftarrow$ means substitution and $\alpha > 0$ is learning rate.

We treat the RL problem in a flat non-hierarchical structure, and thus actions are selected based on the following composite action-value function:

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

where suffix t is dropped for simplicity. Coefficients $c_i, i = 1, \cdots n$ are introduced to control use of subgoals. They are initialized to 1.0, i.e. at the beginning, all the virtual rewards are considered as equally strongly as the real reward. Actual action selection is based on finding an action that maximizes $Q^A(s, a)$ plus an appropriate exploratory variation such as ϵ-greedy and softmax [1]. Therefore, learning of Q by equation (3) is an *off-policy* learning, and its convergence is assured just like the ordinary Q-learning under 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 avoiding visiting unnecessary state-action pairs by introduction of subgoals.

When a subgoal I_i is found to be either redundant or harmful, its corresponding coefficient c_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 $\mathrm{I}_j, j \in R_i$, where R_i is the set of suffixes of subgoals that are reachable from subgoal I_i in the directed graph. In other words, without help of subgoal I_i, the agent can find the optimal action that leads to the final goal or a subgoal closer to the final goal i.e. a more important subgoal. Let us define

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

Then the optimal action towards the subsequent subgoals and the final goal is given by

$$\tilde{a}_i^*(s) = \operatorname*{argmax}_a \tilde{Q}_i(s,a), \tag{7}$$

and the optimal action towards subgoal I_i by

$$a_i^*(s) = \operatorname*{argmax}_a Q_i(s,a). \tag{8}$$

Note that if $Q(s,a)$ and $Q_i(s,a)$ are zero or negative for any a it means that no positive real/virtual rewards have been received and that the above 'optimal actions' are meaningless. So, we need the following preconditions:

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

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 $\beta \in (0,1)$:

$$c_i \leftarrow \beta c_i. \tag{11}$$

Coefficient c_i is not set to zero at once because we only find that subgoal I_i is redundant in this particular state s but it may be useful in other states.

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 $\mathrm{I}_j, j \in R_i$, i.e. the action towards subgoal I_i contradicts with the action towards the final goal or a subgoal closer to the final goal. 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. We can say a subgoal I_i is harmful in state s if

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

with the preconditions (9). When a subgoal is judged to be harmful in state s, its associated coefficient c_i is reduced so that the subgoal does not do any harm in action selection. Again, the coefficient is not set to zero at once. To resolve the conflict (12), we reduce the value of $c_i Q_i(s,a)$ so that it, when used in action selection together with $\tilde{Q}_i(s,a)$, does not interfere the action selection by $\tilde{Q}_i(s,a)$ only. So, c_i is reduced so that the following holds for state s,

$$\operatorname*{argmax}_a \{c_i Q_i(s,a) + \tilde{Q}_i(s,a)\} = \operatorname*{argmax}_a \tilde{Q}_i(s,a). \tag{13}$$

Although the composite action-value function $Q^A(s,a)$ for the action selection includes the terms related subgoals closer to the *initial* state than subgoal I_i, we do not consider them in reducing c_i because preconditions (9) mean that subgoal I_i has been already achieved in the past trial. If any of those less important subgoals causes a contradiction with subgoal I_i, it is the coefficient associated with that subgoal that must be decreased to solve the contradiction. Now, considering equation (7), the above equation (13) holds when

$$c_i Q_i(s,a) + \tilde{Q}_i(s,a) \leq c_i Q_i(s, \tilde{a}_i^*(s)) + \tilde{Q}_i(s, \tilde{a}_i^*(s)) \tag{14}$$

is satisfied for all a. Then, by straightforward calculation, the maximum value of c_i that assure the above is derived as

$$\bar{c}_i = \min_{a \in \{a | Q_i(s,a) > Q_i(s, \tilde{a}_i^*(s))\}} \frac{\tilde{Q}_i(s, \tilde{a}_i^*(s) - \tilde{Q}_i(s,a)}{Q_i(s,a) - Q_i(s, \tilde{a}_i^*(s))}. \tag{15}$$

Note that there is a possibility that the original value of c_i is already smaller than $\bar{c}_i$, so, the coefficient is updated as

$$c_i \leftarrow \min\{c_i, \bar{c}_i\}. \tag{16}$$

Because update of c_i depends on c_j, $j \in R_i$ contained in $\tilde{Q}_i$, the update is done starting with the last subgoal namely the subgoal closest to the final goal down to the first subgoal that is the closest to the initial state.

The overall procedure is described in Figure 2. 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 useless or harmful. Here the action-values for other actions remain unchanged, and thus it suffices that the preconditions (9) are checked for s_t and a_t only.

```
Q(s,a) ← 0.0, Q_i(s,a) ← 0.0, ∀s, ∀a, i = 1, ··· , n.
c_i ← 1, i = 1.0, ··· , n.
Do until the terminate condition is satisfied {
    t ← 0, s_t ← B.
    Do until s_t = F 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, ··· n.
        Update action-value functions Q(s_t, a_t) and
        Q_i(s_t, a_t), i = 1, ··· n by equations (3) and (4).
        For I_i is the last subgoal down to the first subgoal do {
            If Q_i(s_t, a_t) > 0 and Q̃_i(s_t, a_t) > 0 then {
                If a_i^*(s) = ã_i^*(s) then
                    c_i ← β c_i.
                else
                    c_i ← min{c_i, c̄_i}.
            }
        }
        t ← t + 1.
    }
}
```

Figure 2. 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 in grid worlds. The grid worlds have several doors as shown in Figure 3. The agent must pick up a key to go through a door and 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 A opens door A, and key B is the key to door B. The subgoals can be represented by a directed graph shown in Figure 4. The agent receives a reward 1.0 at the goal F and also a virtual reward 1.0 at the subgoals. When it selects a move to a wall or 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 or 200 time steps have passed. Since the agent has to collect two keys to go through the two doors, it takes a large number of episodes to arrive at the final goal by random actions only.

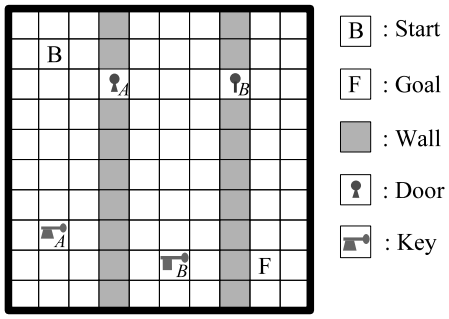

Figure 3. Grid world 1.

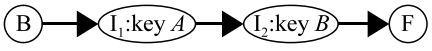

Figure 4. Subgoal structure of grid world 1.

Q-learning is performed with and without taking the subgoals into consideration. The parameters used are as follows: discount factor $\gamma = 0.9$, learning rate $\alpha = 0.1$ and β in (11) is 0.99. Softmax action selection is used with 'temperature parameter' being 0.05. The number of episodes required for the agent to reach the goal for the first time by greedy action based on the learnt Q^A and the number 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 random number sequences. The table indicates that consideration of the subgoals makes the learning more than 20 times faster, which verifies the validity of introducing subgoals to accelerate RL.

Table 1. Numbers of episodes required to achieve the goal (grid world 1).

	Numbers of episodes	
	first arrival at the goal	finding an optimal path
Without subgoals	9620.2	10397.2
With subgoals	462.8	465.2

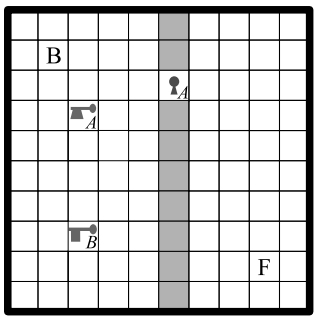

Figure 5. Grid world 2 with a correct key and a wrong key.

What is addressed in this paper is controlling subgoals in order to deal with possible errors and ambiguity. Now, let us consider another grid world shown in Figure 5 where key B does not open the door and only key A is the correct key. We apply the proposed method to this problem with each of subgoal structures shown in Figure 6 where subgoal structure (a) is the correct one. The average numbers of episodes before the first arrival at the goal and before finding an optimal path are shown in Table 2. With the correct subgoal given, the agent can reach the goal and find the optimal path faster than the case without a 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 works well. Finding the optimal path takes more episodes in the cases of 'wrong subgoal', 'correct and wrong subgoals in series' and 'correct and wrong subgoals in parallel'. This is because the coefficient associated with the wrong subgoal does not decay fast enough. The precondition for reducing the coefficient requires that the subgoal in question as well as one of its subsequent subgoals have been already visited. Naturally the subgoals closer to the *initial* state are more likely to be visited by random actions than those far from the *initial* state. This means that the coefficient associated with a subgoal closer to the *initial* state starts to decay earlier. Therefore, in 'correct and wrong subgoals in series' case, the coefficient on the cor-

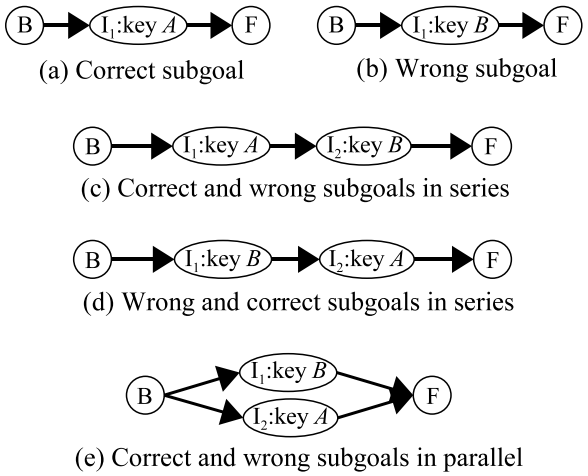

Figure 6. Possible subgoals for grid world 2.

Table 2. Numbers of episodes required to achieve the goal (grid world 2).

Subgoals used in learning	Numbers of episodes	
	first arrival at the goal	finding an optimal path
None	94.0	94.0
Correct	80.4	81.6
Wrong	102.6	210.6
Correct & wrong in series	116.4	271.0
Wrong & correct in series	90.6	118.2
Correct & wrong in parallel	113.2	272.8

rect subgoal decays faster and the coefficient on the wrong subgoal survives longer, which causes more delay in the learning. Similar situations occur in 'wrong subgoal' and 'correct and wrong subgoals in parallel' cases.

To confirm the effect of subgoal control, learning is performed with the coefficient control disabled, i.e. coefficients c_i are fixed to 1.0. In the case that the correct subgoal is given, the result is the same as that derived with the coefficient control. However, in other cases where a wrong subgoal is given, the optimal path is not found within 100000 episodes. Therefore, simply giving virtual rewards to subgoals does not work well when some wrong subgoals are included; appropriate control is needed.

In the results shown in Table 2, the learning is not accelerated much even when the correct subgoal is given, and the results with wrong subgoal are not too bad. Those results of course depend on the problem to be solved. Table 3 shows the results for a problem where the positions of key A and key B are exchanged in grid world 2. Also the results for grid world 3 depicted in Figure 7 are listed in Table 4. Here the correct and the wrong keys are located in the opposite directions from the start cell. By exchanging the two keys, the problem becomes more difficult than the original grid world 2 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 on the way from the start cell to the correct key, and therefore the wrong subgoal does not do much harm in this case. To the contrary, the wrong key in grid world 3 lies in the opposite direction to the correct key. So, this wrong subgoal has worse effect on the learning speed in Table 4.

In all of the above problems, the actual subgoal structure is that shown in Figure 4 (a). To investigate the per-

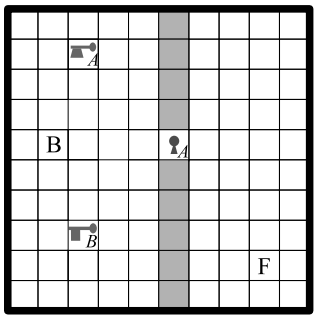

Figure 7. Grid world 3 with two keys in opposite directions.

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

Subgoals used in learning	Numbers of episodes	
	first arrival at the goal	finding an optimal path
None	350.2	386.2
Correct	105.6	106.0
Wrong	250.8	252.0
Correct & wrong in series	142.2	145.8
Wrong & correct in series	127.0	127.8
Correct & wrong in parallel	110.2	110.6

Table 4. Numbers of episodes required to achieve the goal (grid world 3).

Subgoals	Numbers of episodes	
	to arrive at the goal	to find an optimal path
None	155.4	236.0
Correct	80.2	81.6
Wrong	166.0	656.5
Correct & wrong in series	120.6	510.3
Wrong & correct in series	81.2	82.2
Correct & wrong in parallel	87.0	269.0

formance of the proposed method on problems with parallel subgoals, key B in grid world 2 is changed to a key A. So the environment has two correct keys, and the actual subgoal structure is just like Figure 4 (e) but with two correct keys. 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, it makes finding the optimal path slow. The subgoal structure 'near and far in parallel' gives the correct subgoal structure, but this gives the worst performance in finding the optimal path in the table. In this problem, both keys correspond to correct subgoals, but one (closer to the start cell) is more preferable than the other, and the less-preferable subgoal survives longer in this setting, which delays the learning.

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

Subgoals used in learning	Numbers of episodes	
	first arrival at the goal	finding an optimal path
None	128.2	128.2
Near	74.0	75.4
Far	119.2	272.6
Near & far in series	87.4	165.6
Far & near in series	90.6	118.2
Near & far in parallel	108.2	311.8

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 subgoals, especially when it occupies a position close to the initial state. However, even the wrong subgoals do not cause critically harmful effect such as impractically long delay and inability of finding the goal at all. This verifies the validity of the proposed control method of subgoals.

4 Conclusions

In order to make RL faster, use of subgoals is proposed. Since errors and ambiguity are inevitable in subgoal information provided by humans, a mechanism is proposed that controls use of subgoals. The method is applied to 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 do not cause critical damage on the learning performance.

Errors and ambiguity in subgoal information prevent us from using task decomposition or hierarchical structure. This puts certain limitations on the proposed learning method with respective to the memory and time efficiency. However, the memory increase caused by having a separate action-value function for each subgoal is not so significant because an action-value function is used until its associated subgoal is achieved and thus we do not need to prepare the action-value function over all state-action pairs. On the other hand, we cannot expect much gain in time efficiency when wrong subgoals are given in wrong places. The fact that decaying speed of a coefficient c_i depends on the position of its associated subgoal in the *initial_state-subgoals-goal_state* structure can be utilized when making use of ambiguous subgoal information effectively. Also a possible extension of the method is to combine it with a subgoal learning technique.

5 Acknowledgements

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

References

[1] R.S. Sutton and A.G. Barto, *Reinforcement Learning, An Introduction* (Cambridge, Massachusetts: A Bradford Book, The MIT Press, 1998).

[2] W.D. Smart and L.P. Kaelbling, Practical Reinforcement Learning in Continuous Spaces, *Proc. 17th Int. Conf. Machine Learning*, 2000, 903-910.

[3] Y. Wang, M. Huber, V.N. Papudesi and D.J. Cook, User-Guided Reinforcement Learning of Robot Assistive Tasks for an Intelligent Environment, *Proc. 2003 IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, 2003.

[4] K. Driessens and S. Džeroski, Integrating Guidance into Relational Reinforcement Learning, *Machine Learning*, 57, 2004, 271-304.

[5] S. Singh, The Efficient Learning of Multiple Task Sequences, in *Advances in Neural Information Processing Systems 4*, (Morgan Kauffman, 1992) 251-258.

[6] M. Wiering and J. Schmidhuber, HQ-Learning: Discovering Markovian Subgoals for Non-Markovian Reinforcement Learning, *Tech. Rep. IDSIA-95-96*, 1996.

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

[8] A. McGovern and A.G. Barto, Automatic Discovery of Subgoals in Reinforcement Learning using Diverse Density, *Proc. 18th Int. Conf. Machine Learning*, 2001, 361-368.

[9] R.M.Kretchmar, T. Feil and R. Bansal, Improved Automatic Discovery of Subgoals for Options in Hierarchical Reinforcement Learning, *J. Computer Science & Technology*, 3, 2003, 9-14.