Learning of Subgoals for Goal-Oriented Behavior Control of Mobile Robots

Sang Hyoung Lee[1], Sanghoon Lee[1], Il Hong Suh[1], and Wan Kyun Chung[2]

[1] The Department of Electronics Computer Engineering, Hanyang University, Seoul, Korea
zelog@incorl.hanyang.ac.kr, {shlee,ihsuh}@hanyang.ac.kr
[2] The Department of Mechanical Engineering, POSTECH, Pohang, Korea
wkchung@postech.edu

Abstract. Subgoal learning is investigated to effectively build a goal-oriented behavior control rule with which a mobile robot can achieve a task goal for any starting task configurations. For this, states of interest are firstly extracted from successful task episodes, where the averaged occurrence frequency of states is used as threshold value to identify states of interest. And, subgoals are learned by clustering similar features of state transition tuples. Here, features used in clustering are produced by using changes of the states in the state transition tuples. A goal-oriented behavior control rule is made in such a way that proper actions are sequentially and/or reactively generated from the subgoal according to the context of states. To show the validities of our proposed subgoal learning as well as a goal-oriented control rule of mobile robots, a Box-Pushing-Into-a-Goal(*BPIG*) task is simulated and experimented.

1 Introduction

A goal-oriented behavior control rule can be considered as a sequence of behaviors to complete a task. Such behavior control rules are usually provided by planners, reinforcement learning, or human programmers. Planning or reinforcement learning technologies have been known to be partially effective in recovering working conditions by building new behavior plans or policies to cope with exceptional cases. Most of them, however, require building of a new plan or policy to go from a current situation to a goal, which would be costly when exceptional cases occur frequently.

If a robot is able to autonomously learn subgoals, it will be possible to complete a task by changing the order of subgoals [1], or, by producing a sub-plan or a sub-policy only for coping with the current subgoals. This is more time- and cost-efficient than newly planning or learning the entire task, saving efforts for building a new plan, or reducing search-space. Here, subgoals refer to the states which robots necessarily or frequently go through in the course of task execution. By learning subgoals, the robot is able to explore more effectively and accelerate learning in other task in the same or similar environments where the same subgoals are useful [2]. Subgoals are also useful in producing the nominal sequence of behaviors for executing tasks with behavior control rules.

Stolle *et al*. [3] developed a control approach that used a library of trajectories to establish a global control law or policy. A library of trajectories is used to create a global

M. Köppen et al. (Eds.): ICONIP 2008, Part I, LNCS 5506, pp. 64–71, 2009.

policy by nearest-neighbor look up. However, since subgoals for a task have not been considered in their approach, a new library of trajectories will be required if a goal location is changed. Fukazawa *et al*. [4] proposed an algorithm that acquires intermediate goals between initial states and goal states for an agent executing multiple tasks. They focused on generating the priorities of subgoals. But, they have not discussed how to generate actions to reach subgoals. Sutton *et al*. [5], Mannor *et al*. [6], and Shen *et al*. [7] have proposed option-based reinforcement learning techniques for robot to learn behaviors more effectively for a task, where options imply trajectories between subgoals. However, they assumed that subgoals were available ahead of learning.

As we have seen above, there have been few works investigating subgoal learning. In this paper, we will propose an algorithm for subgoal learning to effectively build a behavior control rule, with which a mobile robot can achieve a task goal for any starting task configurations. In this case, a goal-oriented behavior control rule is built in such a way that proper actions are sequentially and/or reactively generated from subgoals according to the context of states.

The organization of this paper is as follows. In section II, details on subgoal learning will be described. In section III, a goal-oriented behavior control rule will be explained. In section IV, simulations and experimental results will be given to verify validities of the subgoal learning and the goal-oriented behavior control rule for a Box-Pushing-Into-a-Goal(*BPIG*) task. Finally in section V, concluding remarks will be provided.

2 Subgoal Learning

A goal-oriented behavior control rule of a task is defined as a sequence of *state-action* pairs which are reordered according to changes of states in such a way that a task is completed under several uncertainties due to nondeterministic environment, incomplete perception and/or probabilistic action effect.

To generate such a meaningful sequence of *state*(current state, next state)-*action* pairs for the task completion (See Fig. 1), it is required to extract *state-action* pairs of interest for the successful task completion from task episodes which include various types of successful task episodes. Then, some of those *state-action* pairs of interest are abstracted as attentive *state-action* pairs (here is after, subgoals) and are nominally ordered as a behavior sequence for the task completion. At the time of execution of the

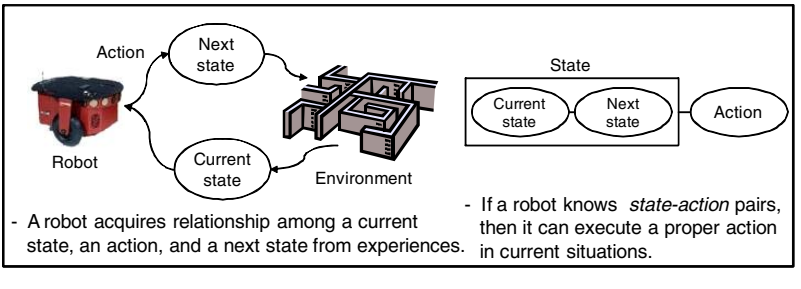

Fig. 1. A relationship of a *state* (current state, next state)-*action* pair

task, the nominal sequence could be reasonably reordered according to the context of the working environment [8].

2.1 Task Description Space(TDS) and Collection of Successful Episodes

TDS is a space for describing the task, in which a subgoal learner understands what task is given. Both a physical space (here is after, P_{space}) and a configuration space (here is after, C_{space}) [9] are composed of our proposed TDS. Tentative behavior sequences are found in a C_{space} and evaluated in a P_{space}. A state in the C_{space} of TDS may not be physically meaningful. Thus, a state needs to be checked whether the state is physically meaningful. This is done by using the P_{space}. Employing the C_{space} helps the subgoal learner to easily get many versatile successful task episodes from which a generic subgoal can be acquired. A state $\mathbf{S}$ of the C_{space} for a task can be defined as

$$\mathbf{S} = [d_{O_1O_2}, a_{O_1O_2}, ..., d_{O_{n-1}O_n}, a_{O_{n-1}O_n}] \tag{1}$$

where $d_{O_1O_2}$, $a_{O_1O_2}$, ..., $d_{O_{n-1}O_n}$, and $a_{O_{n-1}O_n}$ respectively, imply distance between the object O_1 and the object O_2 ($d_{O_1O_2}$), angle between the object O_1 and the object O_2 ($a_{O_1O_2}$), ..., distance between the object O_{n-1} and the object O_n ($d_{O_{n-1}O_n}$), angle between the object O_{n-1} and the object O_n ($a_{O_{n-1}O_n}$). A path in our TDS from an initial state ($\mathbf{S}_{init}$) to a goal state ($\mathbf{S}_{goal}$) can be found by applying the A^* algorithm in such a way that states to be tested are chosen in the C_{space}, and then those states are checked as physically possible states in the P_{space}.

2.2 Extraction of TDS States of Interest

All states of successful paths in TDS are TDS state vectors from which candidates of the TDS states of interest can be extracted. Here, we denote $\mathbb{S}_{total}$ to be the total set including TDS state vectors. In $\mathbb{S}_{total}$, there can be a lot of identical TDS state vectors. On the other hand, there are many TDS states with a single occurrence. The objective of our subgoal learning is to extract generic subgoals from the cases with multiple occurrences. For this, $frequency(\mathbf{S}_i, \mathbb{S})$ is defined as

$$frequency(\mathbf{S}_i, \mathbb{S}) \triangleq number\ of\ satisfying\ (\mathbf{S} = \mathbf{S}_i) \wedge (\mathbf{S} \in \mathbb{S}) \tag{2}$$

where $\mathbb{S}$ is the set including TDS state vectors, and $\mathbf{S}$ and $\mathbf{S}_i$ are a TDS state vector. And, now $\mathbb{S}_{duplicate}$ and $\mathbb{S}_{unique}$ are define as

$$\mathbb{S}_{duplicate} = \{\mathbf{S}_i \mid (\mathbf{S}_i \in \mathbb{S}_{total}) \wedge (frequency(\mathbf{S}_i, \mathbb{S}_{total}) \geq 2)\} \tag{3}$$

and

$$\mathbb{S}_{unique} = \{\mathbf{S}_i \mid (\mathbf{S}_i, \mathbf{S}_j \in \mathbb{S}_{duplicate}) \wedge (\mathbf{S}_i \neq \mathbf{S}_j)\}. \tag{4}$$

In Eqs. (3) and (4), $\mathbb{S}_{duplicate}$ and $\mathbb{S}_{unique}$ respectively, imply the set of TDS states having frequency greater than or equal to 2, and the set that does not allow the duplicated TDS state.

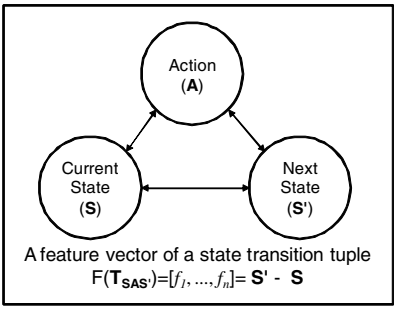

Fig. 2. A concept of a featrue vector in a state transition tuple

Let f_{th} be the averaged value of occurrence frequency which can be computed as

$$f_{th} = \frac{|\mathbb{S}_{duplicate}|}{|\mathbb{S}_{unique}|} \tag{5}$$

where $|\mathbb{S}_{duplicate}|$ and $|\mathbb{S}_{unique}|$ imply the cardinality of the set $\mathbb{S}_{duplicate}$ and $\mathbb{S}_{unique}$. Subgoals will be made for the TDS states having occurrence frequency greater than or equal to f_{th}. A set of such candidate TDS states, TDS states of interest ($\mathbb{S}_{interesting}$), can be obtained as

$$\mathbb{S}_{interesting} = \{\mathbf{S}_i \mid (\mathbf{S}_i \in \mathbb{S}_{duplicate}) \wedge (frequency(\mathbf{S}_i, \mathbb{S}_{duplicate}) \geq f_{th})\}. \tag{6}$$

It is remarked that states of interest can be obtained by means of occurrence frequency, because there should be same or similar situation for the robot to encounter during the execution of the tasks from various initial configurations to the same goal configuration. Those frequently encountered situations in a real environment, or states in our TDS could be possible subgoals to achieve the goal.

2.3 State Transition Tuple and Clustering

We produce state transition tuples by using TDS states of interest, which are the candidates of subgoals. Let $\mathbf{T}_{\mathrm{SAS}'}$ be the state transition tuple which is defined as

$$\mathbf{T}_{\mathrm{SAS}'} \triangleq (\mathbf{S}, \mathbf{A}, \mathbf{S}'). \tag{7}$$

where $\mathbf{S}'$ is a next state, $\mathbf{S}$ is a current state, and $\mathbf{A}$ is an action which forced $\mathbf{S}$ to become $\mathbf{S}'$. From state transition tuples $\mathbf{T}_{\mathrm{SAS}'}$, following information can be extracted (See Fig.2). (1) Is the state vector changed? (2) If so, what elements of the state vector are changed? (3) Is the value of change increased or decreased? The feature vector of $\mathbf{T}_{\mathrm{SAS}'}$ is now defined as

$$\mathbf{F}_{\mathbf{SAS}'} = [f_1, ..., f_n]. \tag{8}$$

In Eq. (8), $for\ all\ i = 1\ to\ n$, f_i is defined as

$$f_i = \begin{cases} 1, & if\ \mathrm{s}_i < \mathrm{s}'_i, \\ 0, & if\ \mathrm{s}_i = \mathrm{s}'_i, \\ -1, & if\ \mathrm{s}_i > \mathrm{s}'_i. \end{cases} \tag{9}$$

It is recalled that there may be a lot of different $\mathbf{T}_{\mathrm{SAS}'}$ for task episodes. And, different $\mathbf{T}_{\mathrm{SAS}'}$ can have the same $\mathbf{F}_{\mathbf{SAS}'}$ Thus, those state transition tuples are classified in such a way that $\mathbf{T}_{\mathrm{SAS}'}$ having the same $\mathbf{F}_{\mathbf{SAS}'}$ are included in the same cluster.

2.4 Generalization

It is necessary to find out the attentive (or representative) *TDS* state vector ($\mathbf{S}^{\mathbb{C}_i}$) for executing a task with each cluster. In this work, a concept learning approach is employed, where a general-to-specific ordering of hypothesis is applied to positive instances. The *more-general-than partial ordering algorithm* [10] is used to organize the search for an attentive *TDS* state vector $\mathbf{S}^{\mathbb{C}_i}$ with $\mathbf{T}_{\mathrm{SAS}'}$ in a cluster. An attentive *TDS* state vector was found by using the algorithm. The algorithm will output the attentive *TDS* state vector.

3 Goal-Oriented Behavior Control Rule

3.1 Ordering of Subgoals

Each attentive *TDS* state vector representing a cluster will be regarded as a subgoal. Using subgoals, a goal-oriented behavior control rule is made in such a way that proper actions are sequentially and reactively generated from the subgoals according to states which could be expected from previous robot behaviors, or unexpected ones. For this, subgoals need to be grouped according to the distance from a subgoal to the goal.

Let $L(\mathbf{S}^{\mathbb{C}_i})$ be the distance from $\mathbf{S}^{\mathbb{C}_i}$ (the attentive *TDS* state vector of the *i*th cluster) to the goal state. If current *TDS* state $\mathbf{S}$ of a tuple in a cluster $\mathbb{C}_j$ is the same as next *TDS* state $\mathbf{S}'$ of a tuple in other cluster $\mathbb{C}_i$, the distance of $\mathbb{C}_i$ to the goal is longer than $\mathbb{C}_j$ to the goal. $L(\mathbf{S}^{\mathbb{C}_i})$ is increased by one.

3.2 Action Selection Mechanism

N-best subgoals are selected by comparing probabilistic values of the current observed state $\mathbf{Z}$ and attentive *TDS* state vector $\mathbf{S}^{\mathbb{C}}$ of subgoals as

$$\mathbb{C}_N = \sum_{i=1}^{N} \arg \max_{\mathbb{C},\ \mathbb{C}_1 \notin \ldots \notin \mathbb{C}_{i-1}} P(\mathbf{S}^{\mathbb{C}_i} | \mathbb{C}, \mathbf{Z}), \tag{10}$$

where $\mathbb{C}$ implies a set of subgoals (clusters) and $\mathbf{T}$ implies state transition tuples. And, the best matched tuple is selected by comparing the probabilistic value of $\mathbf{Z}$ and all $\mathbf{S}$ in state transition tuple of *n*-best subgoals as

$$\mathbf{T}_{\mathbf{SAS}'} = \arg \max_{\mathbf{T}} P(\mathbf{S} | \mathbb{C}_N, \mathbf{Z}), \tag{11}$$

where $\mathbb{C}_N$ implies selected *n*-best subgoals. Then, the action of the best matched tuple is activated sequentially to avoid attraction to a passed subgoal, and to take successful transition to a state toward next subgoal as

$$\mathbf{A} = \Phi(\mathbf{T}_{\mathbf{SAS}'}) \tag{12}$$

where the function $\Phi()$ simply extracts an action in the selected tuple.

4 Simulations and Experiments

To show the validities of our proposed method of the subgoal learning for goal-oriented behavior control, a *BPIG* task is simulated and experimented. The *BPIG* task is having a robot push boxes to a goal across a room. To evaluate our subgoal learning for goal-oriented behavior control of the task, simulations are first performed, where simulations are executed in the same simulator used for P_{space}. But, staring configuration for each test is randomly selected so that it is totally different from those of episodes at the time of subgoal learning. A configuration space for the *BPIG* task, the C_{space} is a six-dimensional space. A state $\mathbf{S}$ of the C_{space} for the *BPIG* task can be defined as

$$\mathbf{S} = [d_{rb}, a_{rb}, d_{rg}, a_{rg}, d_{bg}, a_{bg}] \tag{13}$$

Successful episodes were collected by using the A* algorithm. The results summarized in Table 1 show the process of subgoal learning by using the successful episodes. In Figure 3, the results of the simulation and the experiment are shown, where the *BPIG* task is executed with designed subgoals and our goal-oriented behavior control rule in section 3.

Table 1. Simulation Results of the *BPIG* task

BPIG task			
# of successful episodes	1,024	# of *TDS* states of interest	8,593
# of dimensions of C_{space}	6	# of subgoals	12
# of *TDS* states	13,500	-	-

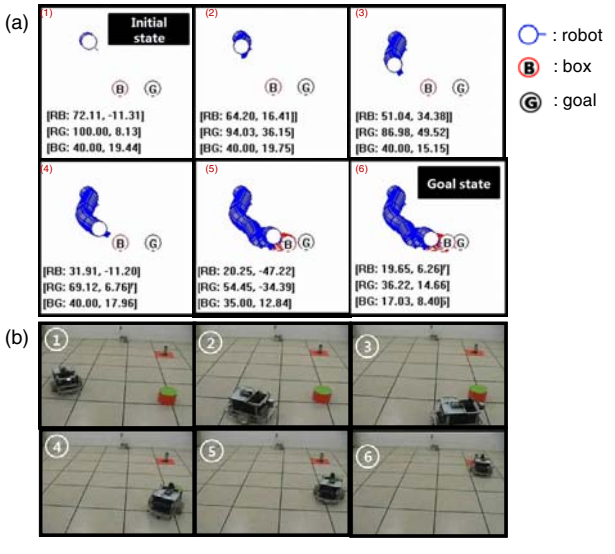

Fig. 3. Display of results for the *BPIG* task; (a) simulation results, (b) experimental results

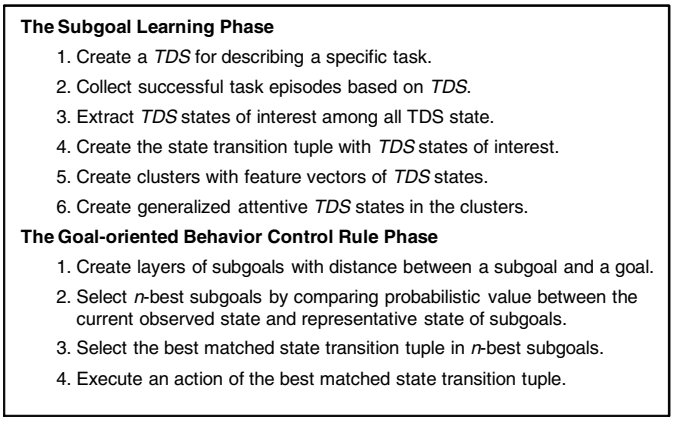

Fig. 4. Flow of Subgoal Learning for Goal-oriented Behavior Control Rule

To evaluate performances of our subgoal learning for goal-oriented behavior control, several experiments are carried out. Firstly, The *BPIG* task in an environment with sensing errors that have normal distribution $N(0,1)$ and $N(0,2)$. The task is observed for over 90% success rate in spite of sensing errors. Secondly, A robot executes a task in a dynamically changing environment by using subgoals for the *BPIG* task. The task was successfully completed even after boxes was added in the middle of task execution. This is an example showing that even in dynamically changing environments, a task can be completed by reordering subgoals that were already learned. Fig. 4 illustrates the entire flow of the previously-proposed subgoal learning for goal-oriented behavior control.

5 Conclusion and Further Works

We proposed a method for automatic subgoal learning to effectively build a goal-oriented behavior control rule with which a mobile robot can achieve a task goal for any starting task configurations. Our subgoals were made by extracting several attentive states from episodes, and by formulating a set of behaviors associated with such attentive states. Then, the states were organized to be an ordered collection of attentive states with their associated behavior sets, and were used to select a behavior from the set by means of the nearest neighbor look up. Thus, a goal-oriented behavior control rule could be robust since the subgoals have been designed by considering how to act for many possible states in episodes, which helps robots to cope with unexpected state transitions.

There are some points to be investigated in our future works. The subgoal learning for goal-oriented behavior control is to be applied to various kinds of service robot tasks. And, learning methods incorporating negative episodes and incremental learning technologies are to be investigated to deal with small size of episodes.

Acknowledgments. This work was supported by the IT R&D program of MKE/IITA. [2008-F-038-01, Development of Context Adaptive Cognition Technology].

References

1. Hue, C.W., Was, B.W., Chen, Y.: Subgoal ordering and Granularity Control for Incremental Planning. In: IEEE International Conference on Tools with AI (2005)
2. Goel, S., Huber, M.: Subgoal Discovery for Hierarchical Reinforcement Learning Using Learned Policies. In: International FLAIRS Conference, pp. 346–350 (2003)
3. Stolle, M., Atkeson, C.G.: Policies Based on Trajectory Libraries. In: IEEE International Conference on Robotics and Automation, Orlando, Florida, USA, pp. 3344–3349 (2006)
4. Fukazawa, Y., Trevai, C., Ota, J., Arai, T.: Acquisition of Intermediate Goals for an Agent Executing Multiple Tasks. IEEE Transactions on Robotics 22(5), 1034–1040 (2006)
5. Sutton, R., Precup, D., Singh, S.: Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning. In: AI, vol. 112, pp. 181–211 (1999)
6. Mannor, S., Menache, I., Hoze, A., Klein, U.: Dynamic Abstraction in Reinforcement Learning via Clustering. In: The 21st International Conference on Machine Learning, Banff, Canada, pp. 560–567 (2004)
7. Shen, J., Gu, G., Liu, H.: Automatic Option Generation in Hierarchical Reinforcement Learning via Immune Clustering. In: Systems and Control in Aerospace and Astronautics, ISSCAA, pp. 19–21 (2006)
8. Suh, I.H., Kim, M.J., Lee, S., Yi, B.J.: A novel dynamic priority-based action-selection-mechanism integrating a reinforcement learning. In: IEEE Conference on Robotics and Automation, pp. 2639–2646 (2004)
9. LaValle, S.M.: Planning Algorithms, ch. 4. Cambridge Press (2006)
10. Mitchel, T.M.: Machine Learning, ch. 2. McGraw-Hill, New York (1997)