Self Task Decomposition for Modular Learning System through Interpretation of Instruction by Coach

Yasutake Takahashi[1,2], Tomoki Nishi[1], and Minoru Asada[1,2]

[1] Dept. of Adaptive Machine Systems, Graduate School of Engineering,
[2] Handai Frontier Research Center,
Osaka University
Yamadagaoka 2-1, Suita, Osaka, 565-0871, Japan
{yasutake, nishi, asada}@er.ams.eng.osaka-u.ac.jp
http://www.er.ams.eng.osaka-u.ac.jp

Abstract. One of the most formidable issues of RL application to real robot tasks is how to find a suitable state space, and this has been much more serious since recent robots tends to have more sensors and the environment including other robots becomes more complicated. In order to cope with the issue, this paper presents a method of self task decomposition for modular learning system based on self-interpretation of instructions given by a coach. The proposed method is applied to a simple soccer situation in the context of RoboCup.

1 Introduction

Reinforcement learning (hereafter, RL) is an attractive method for robot behavior acquisition with little or no *a priori* knowledge and higher capability of reactive and adaptive behaviors [1, 2]. However, a simple and straightforward application of RL methods to real robot tasks is difficult because of the enormous time for exploration that scales exponentially with the size of the state/action space. The recent robots tend to have many kinds of sensors like normal and omni-vision systems, touch sensors, infrared range sensors, and so on. They can receive a variety of information from these sensors, especially vision sensors. This fact indicates that the difficulty of RL application to real robot tasks becomes more serious.

Fortunately, a long time-scale behavior might often be decomposed into a sequence of simple behaviors in general, and therefore, the search space can be divided into smaller ones. In the existing studies, however, task decomposition and behavior switching procedures are given by the designers (ex. [1, 3, 4]). Others adopt the heuristics or the assumption that are not realistic from the view point of real robot application (ex. [5–8]).

When we develop a real robot that learns various behaviors in its life, it seems reasonable that a human instructs or shows some example behaviors to

the robot to accelerate the learning before it starts to learn (ex. [9, 10]). This idea was applied to a monolithic learning module. To cope with more complicated tasks, this idea can be extended to a multi-module learning system. That is, the instruction will help a learner to find useful subtasks.

In this paper, we introduce an idea that the capability of a learning module defines the size of subtasks. We assume that each module can maintain a few number of state variables and this assumption is reasonable for real robot applications. Then, the system decomposes a long-term task into short-term subtasks with self-interpretation of coach instructions so that one learning module with limited computational resources can acquire a purposive behavior for one of these subtasks. We show experimental results with much more sensors such as normal and omni-vision systems and 8 directions infrared range sensors.

2 Basic Idea

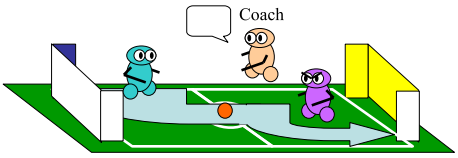

Fig. 1. A coach gives instructions to a learner

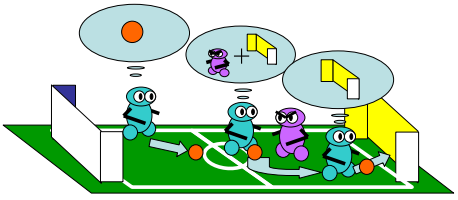

Fig. 2. The learner follows the instructions and finds basic behaviors by itself

There are a learner and a coach in a simple soccer situation (Fig. 1). The coach has *a priori* knowledge of a task to be played by the learner while s/he does not have any idea about the system of the learner. On the other hand, the learner just follows the instructions without any knowledge of the task. After some instructions given by a coach, the learner decomposes the whole task into a sequence of subtasks, acquires a behavior for each subtask, and coordinates these behaviors to accomplish the task by itself . In Fig. 1, the coach instructs an shooting a ball into a yellow goal with obstacle avoidance. Fig. 2 shows an example that the system decomposes this task into three subtasks and assigns them to three modules that maintain state spaces consist of ball variables, opponent and goal ones, and goal ones, respectively.

Fig. 3 show a rough sketch of the idea of the task decomposition procedure. The top of the Fig. 3 shows a monolithic state space that consists of all state variables $(x_1, x_2, \cdots, x_n)$. The red lines indicate sequences of state value during the given instructions. As we assume beforehand, the system cannot have such a huge state space, then, decomposes the state space into subspaces that consist of a few state variables. The system regards that the ends of the instructions represent goal states of the given task. It checks all subspaces and selects one

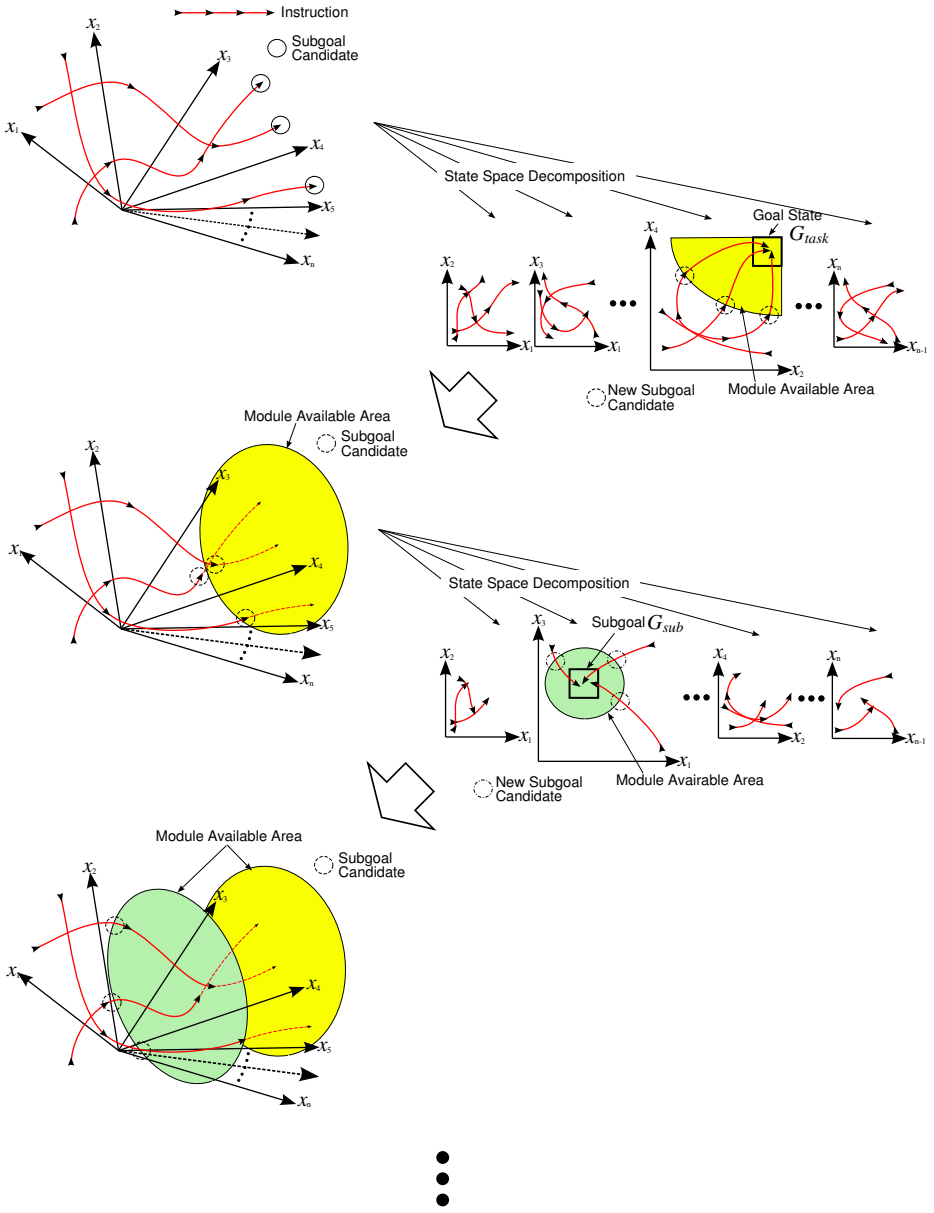

Fig. 3. Rough sketch of the idea of task decomposition procedure

in which the most ends of the instruction reach a certain area (G_{task} in Fig. 3). The system regards this area as the subgoal state of a subtask which is a part of the given long-term task. The steps of the procedure are as follows:

1. find module unavailable areas in the instructions and regard them as unknown subtask.
2. assign a new learning module.
   (a) list up subgoal candidates for the unknown subtasks on the whole state space.
   (b) decompose the state space into subspaces that consist of a few state variables.
   (c) check all subspaces and select one in which the subgoal candidates reach a certain area best (G_{sub} in Fig. 3).
   (d) generate another learning module with the selected subspace as a state space and the certain area as the goal state.
3. check the areas where the assigned modules are available.
4. exit if the generated modules cover all segments of instructed behaviors. Else goto 1.

3 Robot, Tasks, and assumption

Fig. 4. A real robot

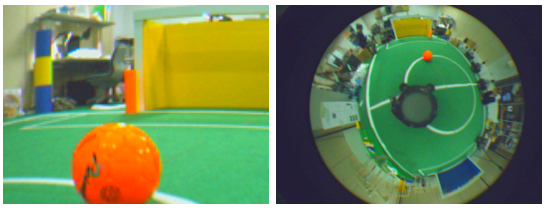

Fig. 5. Captured camera images

Fig. 4 shows a mobile robot we have designed and built. The robot has a normal camera in front of body, an omni-directional camera on the top, and infra red distance sensors around the body. Fig. 5 show the images of both cameras. A simple color image processing is applied to detect the ball, the goal, and an opponent in the image in real-time (every 33ms). The robot has also 8 directions infrared range sensors. The robot has totally 39 candidates of state variables. The details of the candidates are eliminated because of space limitations. The mobile platform is an omni-directional vehicle (any translation and rotation on the plane). The tasks for this robot are chasing a ball, navigating on the field, shooting a ball into the goal, and so on. We assume that the given task has

some absorbing goals, that is, the tasks are accomplished if the robot reaches to certain areas in state spaces which consist of a few state variables.

4 Availability Evaluation and New Learning Module Assignment

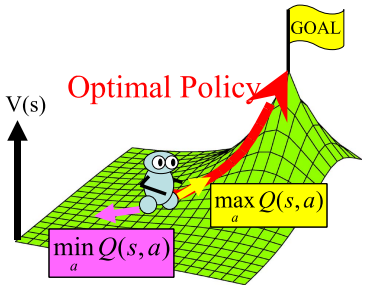

Fig. 6. Sketch of state value function and action value

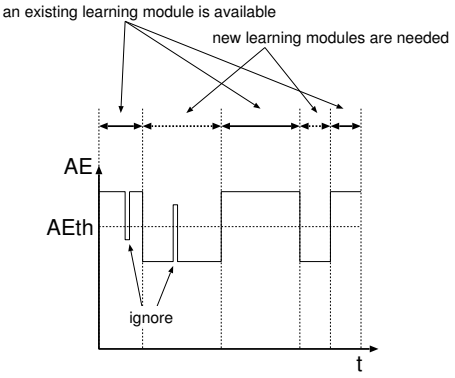

Fig. 7. Availability identification during the given sample behavior

The learner needs to check the availability of learned behaviors that help to accomplish the task by itself because the coach neither knows what kind of behavior the learner has already acquired nor shows perfect example behaviors from the learner's viewpoint. The learner should suppose a module as valid if it accomplishes the subtask even if the greedy policy seems different from the example behavior. Now, we introduce AE in order to evaluate how suitable the module's policy is to the subtask:

$$AE(s, a_e) = \frac{Q(s, a_e) - min_{a'}\, Q(s, a')}{max_{a'}\, Q(s, a') - min_{a'}\, Q(s, a')}\ , \tag{1}$$

where a_e indicates the action taken in the instructed example behavior. AE becomes larger if a_e leads to the goal state of the module while it becomes smaller if a_e leaves from the goal state (see Fig. 6). Then, we prepare a threshold AE_{th}, and the learner evaluates the module as valid for a period if $AE > AE_{th}$. If there are modules whose AE exceeds the threshold AE_{th}, the learner selects the module which keeps $AE > AE_{th}$ for longest period among the modules (see Fig. 7). In Fig. 3, "Module Available Area" indicates the one in which $AE > AE_{th}$.

If there is no module which has $AE > AE_{th}$ for a period, the learner creates a new module which will be assigned to the subtask (see procedure 2 in 2). To assign a new module to such a subtask, the learner identifies the state space and

the goal state. The system follow the two steps to select an appropriate state space and the goal state for the subtask:

- selection of one state variable that specifies the goal state, and
- construction of a state space including the selected state variable.

In order to find one state variable that specifies the goal state best, the system lines up the candidates for a goal region in term of state variables. On the other hand, in order to select another state variable, the system evaluates performance of Q value estimation.

The details of the procedure are eliminated because of space limitations.

5 Experiments

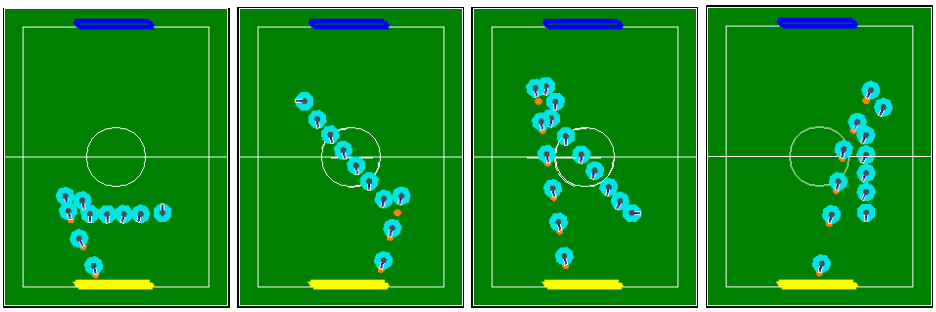

Fig. 8. Examples of Instructed behaviors

We have instructed the robot from a simple behavior (ball chasing) to a complicated one (shooting a ball with obstacle avoidance) in [11], however, there is a criticism that the step-by-step instruction implies task decomposition to the robot. Therefore we adopt only shooting behavior for the task decomposition and the coordination. Fig. 8 shows four examples of the behaviors instructed by the coach. The total number of instruction is 21 for this experiment.

According to the learning procedure, the system produced four modules for the instructed behaviors. The modules are $LM_1(A_{pb}, X_{pb})$, $LM_2(\theta_{og})$, $LM_3(Y_{ob}, X_{ob})$, and $LM_4(A_{ob}, \theta_{ob})$. For example $LM_1(A_{pb}, X_{pb})$ indicates that the modules has a state space that consists of the area of ball on the normal camera image ($A_o b$) and the x position of the ball on the normal camera image (X_{pb}). Fig. 9 shows sequences of the selected module, availability evaluations and goal state activations of modules through an instruction.

Fig. 11 shows the learned behaviors. The start positions of the behaviors are the same ones of the instructions for comparison. The trajectories of learned behaviors are different from the instructed behaviors. This fact indicates that the learner recognizes the subtasks based on its own modules, understands the objectives of the subtasks, and executes appropriate actions for them.

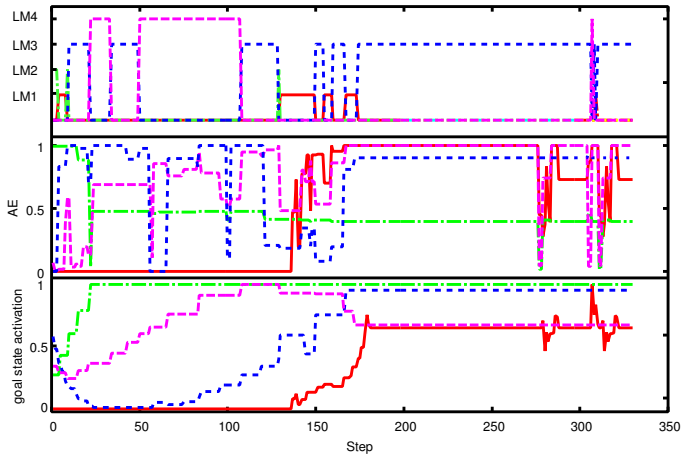

Fig. 9. Sequences of the selected module, availability evaluations and goal state activations of modules through an instruction

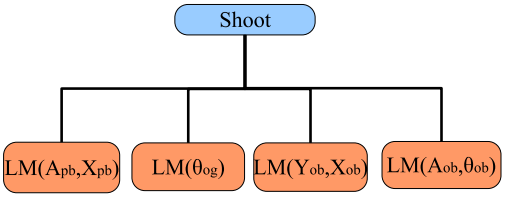

Fig. 10. Acquired hierarchy for the shooting behavior

6 Conclusion

We proposed a hierarchical multi-module learning system based on self-interpretation of instructions given by a coach. We applied the proposed method to our robot and showed results for a simple soccer situation in the context of RoboCup.

References

1. J. H. Connell and S. Mahadevan, *ROBOT LEARNING*. Kluwer Academic Publishers, 1993.
2. M. Asada, S. Noda, S. Tawaratumida, and K. Hosoda, "Purposive behavior acquisition for a real robot by vision-based reinforcement learning," *Machine Learning*, vol. 23, pp. 279–303, 1996.
3. P. Stone and M. Veloso, "Layered approach to learning client behaviors in the robocup soccer server," *Applied Artificial Intelligence*, vol. 12, no. 2-3, 1998.
4. P. Stone and M. Veloso, "Team-partitioned, opaque-transition reinforcement learning," in *RoboCup-98: Robo Soccer World Cup II* (M. Asada and H. Kitano, eds.), pp. 261–272, Springer Verlag, Berlin, 1999.

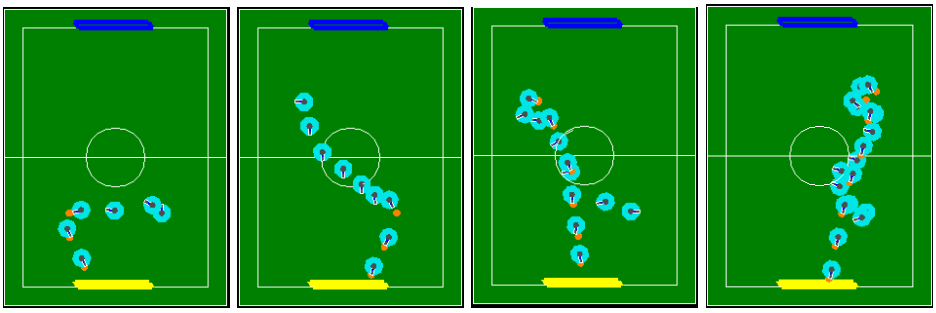

Fig. 11. Acquired behaviors for shooting task

5. B. L. Digney, "Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments," in *From animals to animats 4: Proceedings of The fourth conference on the Simulation of Adaptive Behavior: SAB 96* (P. Maes, M. J. Mataric, J.-A. Meyer, J. Pollack, and S. W. Wilson, eds.), pp. 363–372, The MIT Press, 1996.
6. B. L. Digney, "Learning hierarchical control structures for multiple tasks and changing environments," in *From animals to animats 5: Proceedings of The fifth conference on the Simulation of Adaptive Behavior: SAB 98* (R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson, eds.), pp. 321–330, The MIT Press, 1998.
7. B. Hengst, "Generating hierarchical structure in reinforcement learning from state variables," in *6th Pacific Rim International Conference on Artificial Intelligence (PRICAI 2000)* (R. Mizoguchi and J. K. Slaney, eds.), vol. 1886 of *Lecture Notes in Computer Science*, Springer, 2000.
8. B. Hengst, "Discovering hierarchy in reinforcement learning with HEXQ," in *Proceedings of the Nineteenth International Conference on Machine Learning (ICML02)*, pp. 243–250, 2002.
9. S. D. Whitehead, "Complexity and cooperation in q-learning," in *Proceedings Eighth International Workshop on Machine Learning (ML91)*, pp. 363–367, 1991.
10. M. Asada, S. Noda, S. Tawaratsumida, and K. Hosoda, "Vision-based reinforcement learning for purposive behavior acquisition," in *Proc. of IEEE Int. Conf. on Robotics and Automation*, pp. 146–153, 1995.
11. Y. Takahashi, K. Hikita, and M. Asada, "Incremental purposive behavior acquisition based on self-interpretation of instructions by coach," in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 686–693, Oct 2003.