Multi-Controller Fusion in Multi-Layered Reinforcement Learning

Y. Takahashi M. Asada
Adaptive Machine Systems Graduate School of Engineering
Osaka University
Suita, Osaka 565-0871, Japan

Abstract

This paper proposes multi-controller fusion in multi-layered reinforcement learning based on which an autonomous robot learns from lower level behaviors to higher level ones through its life. In the previous work [1], we proposed a method enables the behavior learning system to acquire several knowledges/policies, to assign sub-tasks to learning modules by itself, to organize its own hierarchical structure, and to simplify the whole system by using only one kind of learning mechanism in all learning modules. However, it has a few drawbacks. The system cannot handle the change of the state variables. It is easily caught by a curse of dimension, if number of the state variables is large. In this paper, we propose an approach of decomposing the large state space at the bottom level into several subspaces and merge those subspaces at the higher level. This allows the system to reuse the policies learned before, to learn the policy against the new features, and therefore to avoid the curse of dimension. To show the validity of the proposed method, we apply it to a simple soccer situation in the context of RoboCup, and show the experimental results.

1 Introduction

There have been a lot of works on learning methods for behavior acquisition for robots based on the methods such as reinforcement learning, genetic algorithms, and so on (ex. [2], [3], and [4]). These methods have a couple of advantages such as higher capability of reactive and adaptive behaviors with little or no a priori knowledge. Therefore, they are embedded as components into autonomous robots expected to develop their behaviors through the interaction between environments and themselves in their lives. Almost of the previous works, however, concentrate on acquiring certain behaviors specified by the human designers who must define the desired goals, state and action spaces, and evaluation functions, in order to apply their methods to acquire the behaviors within reasonable learning time. Therefore, the robot needs learn from scratch if the task changes.

There are several works which reuse previously obtained policies (ex. [5], [6] and [7]). They concentrate on adjustment of certain behaviors to cope with new environments based on knowledge obtained in the previous environment. However, the problem is that these methods cannot handle the situations in which the state space or the state valuables are varying, therefore, the robots need to learn from scratch, again.

An autonomous robot is expected to develop its behaviors from the lower level to the higher one in its life time. In order to cope with various kind of situations, a hierarchical structure within leaning control system seems necessary, by which the control structure can be decomposed into smaller transportable chunks and previously learned knowledge can be applied to related tasks in a newly encountered situation.

Takahashi and Asada [1] proposed a method by which a hierarchical structure for behavior learning is self-organized. The modules in the lower networks are organized as experts to move into different categories of sensor output regions and learn lower level behaviors using motor commands. In the meantime, the modules in the higher networks are organized as experts which learn higher level behaviors using lower modules. This method enables the behavior learning system to acquire several knowledges/policies, to assign sub-tasks to learning modules by itself, to self-organize its own hierarchical structure, and to simplify the whole system by using only one kind of learning mechanism in all learning modules.

However, the proposed system has several drawbacks. First, the system cannot handle the change of the state variables because the system suppose that all tasks can be defined on the state space at the bottom level. This means that the system must discard the learned hierarchical structure and reconstruct new one when the robot obtains new features or variables which represent the situations of the environment. This prevents the robot from developing its behaviors. Next, if number of the state variables is large, it is easily caught by a curse of dimension. This means that the number of state at the bottom layer becomes huge and the system spends enormous computational resources.

Then, we propose an approach of decomposing the large state space at the bottom level into several subspaces and merge those subspaces at the higher level. This allows the system

- to acquire behaviors based on new state features by adding new learning modules layers based on them while it leaves the already learned hierarchical structure when it encounters new state features,
- to acquire behaviors based on state space constructed with large number of state valuables by

merging the modules at lower levels which acquire behaviors based on subspaces, and

- to save computational resources because the number of state could be small by decomposing the whole state space into small subspaces.

We apply the method to a simple soccer situation in the context of RoboCup, show the experimental results, and give discussions.

2 Basic Idea: Construction of a whole state space with decomposed small state spaces

The basic idea of multi-layered learning system is same as our previous work [1]. The robot prepares learning modules of one kind, makes a layer with these modules, and constructs a hierarchy with the layers. The hierarchy of the learning module's layers can be regarded as a role of task decomposition. The lower learning modules explore small areas in the given environment, and learn lower level, fundamental behaviors. They learn behaviors with narrower scopes and shorter time horizons, focusing on the more details. On the other hand, the upper learning modules explore large areas, and learn higher level, more abstracted behaviors based on the learning modules at the lower layers.

In the previous work, the system dealt with a whole state space from lower layer to higher one. Here, we introduce an idea that the system constructs a whole state space with several decomposed state spaces. At the bottom level, there are several decomposed state spaces in which modules are assigned to acquire the low level behavior in the small state spaces. The modules at the higher level manage the lower modules assigned to different state spaces.

In this paper, we define the term "layer" as a group of modules sharing the same state space, and the term "level" as a class in the hierarchical structure. There might be several layers at one level (see Figure 1 and Figure 3 (a)).

2.1 Construction of State-Action Space of Upper Layer

2.1.1 Multiplicative Approach

The state space is constructed as direct product of module's activations of lower layers. This case occurs when the every layer deals with the different object from each other. For example, in the case of robot in the RoboCup field, one layer's modules could be the experts of ball handling and the other layer's modules the one of navigation on the field. The higher modules recognize the lower module's outputs of layers as different one (In Figure 1, two layers but it can be any.). We design the state-action space for the modules managing two layer's modules as shown in Figure 1. The system constructs $n \times m$-dimensional state vector and $n+m$-dimensional action vector, if one lower layer has n modules and the other has m modules.

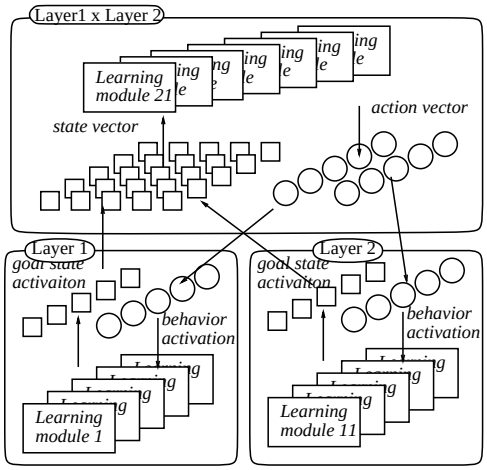


Figure 1: State-action space construction based on multiplicative approach

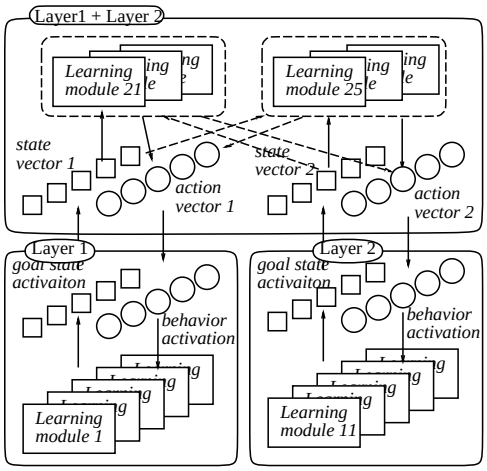
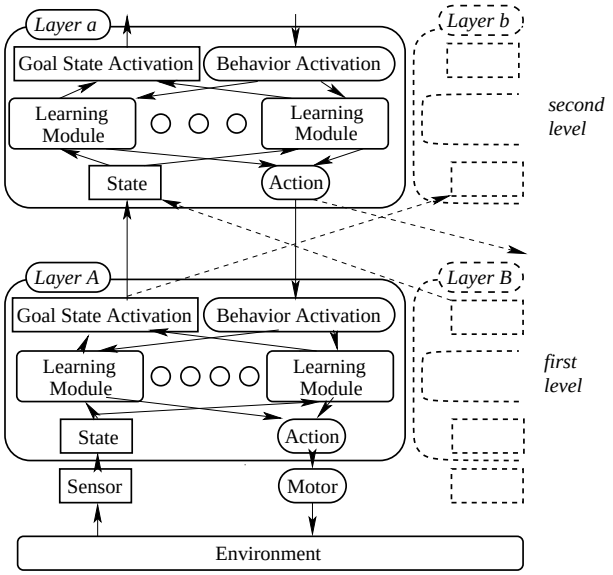


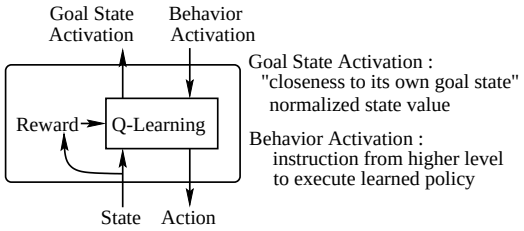
Figure 2: State-action space construction based on additive approach

2.1.2 Additive Approach

The state space is constructed in a additive manner (see Figure 2) when there is a correlation between activation patterns of lower layers. This approach seems to be an extension of the previous work [1]. The output of one layer is assumed to be complimentary to the other in this case; if the system acquires the output of one layer, it can predict the output of other one to some extent. The system does not need to recognize the outputs of two layers as different one, but as one output. We design the state-action space of modules which manages activations of two layers as shown in Figure 2. The system constructs $n+m$-dimensional state vector and $n+m$-dimensional action vector, if one lower layer has n modules and the other m modules. This means that the modules which has its goal state on the output of one lower layer, extrapolate the information from the other lower layer, if there is no active modules on the lower layer. This approach saves com-



(a) A whole system



(b) A behavior learning module

Figure 3: A hierarchical learning architecture

putational resources compared to the "multiplicative approach".

3 Hierarchical Learning System

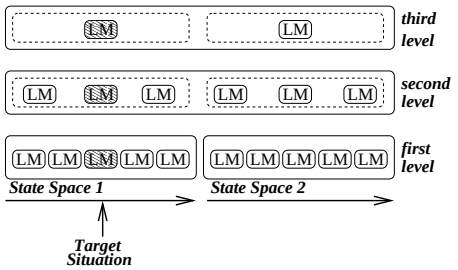
3.1 Architecture

The proposed architecture of the multi-layered reinforcement learning system is shown in Figure 3, in which (a) and (b) indicate a hierarchical architecture with two levels, and individual learning module embedded in the layers, respectively.

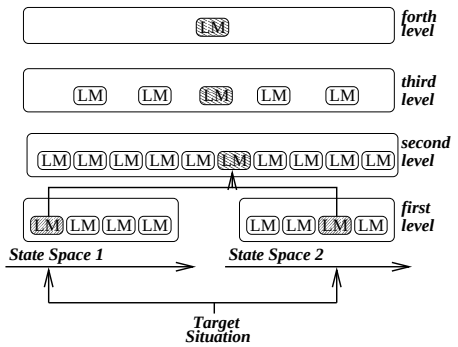
Each module has its own goal state in its state space, and learns the behavior to reach the goal, or maximize the sum of the discounted reward received over time, using continuous Q-learning [8]. The state and the action are constructed using sensory outputs and motor command, respectively at the bottom level (first level).

The input and output from/to the higher level are goal state activation and behavior activation, respectively, as shown in Figure 3(b). The goal state activation g is a normalized state value[1], and $g = 1$ when the situation is the goal state. When the module receives

[1]The state value function estimates the sum of the discounted reward received over time when the robot takes the optimal policy, and is obtained using Q learning.



(a) target situation on one state space



(b) target situation on two state spaces

Figure 4: Strategy in the multi-layered control structure. L.M. stands for learning module

the behavior activation b from the higher level modules, it calculates the optimal policy for its own goal, and sends action commands to the lower level. The action command is translated to actual motor command, then the robot takes the action in the world. The details of the method is described in [1].

3.2 Strategy in the Multi-Layered Learning System to Accomplish Task

The basic idea of the strategy in the multi-layered system is same as the previous work [1]. The target state is given to the multi-layered learning system in the state space at the bottom level. Figure 4(a) shows this situation. If the target situation is defined on one state space, the system executes the procedure [1]; first of all, the system searches a module which is nearest to the target situation. If the module can accomplish the given task, it applies its optimal policy. Else the system searches the module which is nearest to the target situation at the higher level. If the module can apply its policy, it sends behavior activation to lower modules, else the system does the same way at higher level.

If the target situation is defined on two state spaces at the bottom level, the system searches the lowest layer which has modules managing them. Figure 4(b) shows this case. The target situation is given in the two different state spaces. The system searches a module nearest to the target situation at the second level though there are two activated modules at the first level, because the given task is specified in the two state spaces and the layer at the second level manages them. If the module at the second level cannot handle the situation, the system searches a module at higher

level as the previous proposed method does.

4 Experiments

4.1 Setting

Figure 5: A mobile robot, a ball and a goal

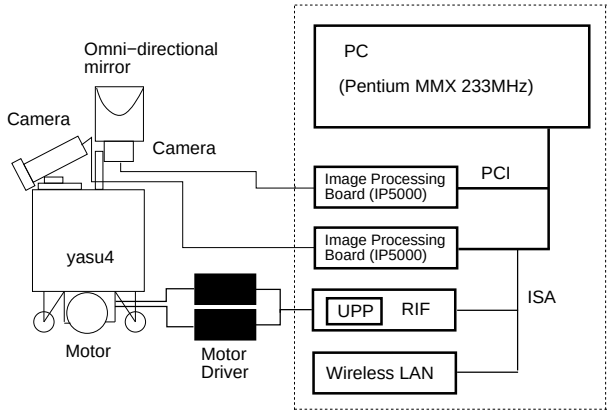


Figure 6: An overview of the robot system

To evaluate the proposed method, we apply it to a task of shooting a ball into a goal. Figure 5 shows a picture of the mobile robot we designed and built, the ball, and the goal. Figure 6 shows an overview of the robot system. It has two TV cameras; One with wide-angle lens of which visual angles are 35 degrees and 30 degrees in horizontal and vertical directions, respectively. This camera is tilted down 23.5 degrees to capture the ball image as large as possible. The other with omni-directional mirror is mounted on the robot. The driving mechanism is PWS (Power Wheeled System), and the action space is constructed in terms of two torque values to be sent to two motors that drive two wheels. These camera and kinematic parameters of the system are unknown to the robot, and it tries to estimate the mapping from sensory information to appropriate motor commands by the method. The environment consists of a ball, a goal, and the mobile robot. The target situation is given by reading the

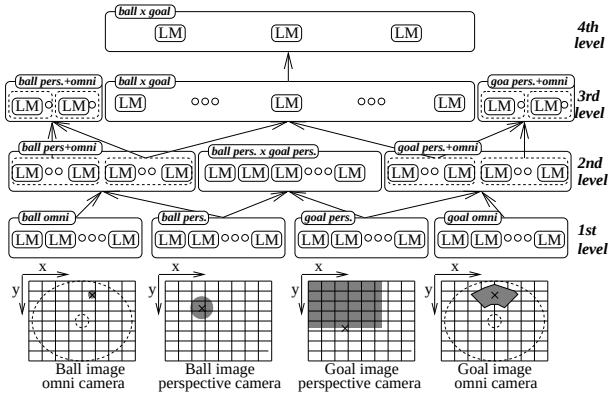


Figure 7: A hierarchy architecture of learning modules

sensor information when the robot pushes the ball into the goal; the robot captures the ball and goal at center bottom of the perspective camera image.

The state spaces at the bottom layers are constructed in terms of the centroids of a ball and a goal images of the two cameras and the perspective image and omni-directional one are tessellated into 11 by 21 grids and 15 by 15 grids, respectively. The action space is constructed in terms of two torque values and is tessellated into 5 by 5 grids. The representative state and action at the upper layer is constructed by the learning modules automatically assigned at the lower layer.

We construct the hierarchical structure as shown in Figure 7. At the lowest level, there are four learning layers, and each of them deals with its own logical sensory space (ball positions on the perspective camera image and omni one, and goal position on both images). At the second level, there are three learning layers in which one adopts multiplicative approach and the others adopt additive approach. The multiplicative approach of the "*ball pers.*×*goal pers.*" layer deals with lower modules of "*ball pers.*" and "*goal pers.*" layers. The arrows in the figure indicate the flows from the goal state activations to the state vectors. The arrows from the action vectors to behavior activations are eliminated. At the third level, the system has three learning layer in which one adopts multiplicative approach and the others adopt additive approach, again. At the levels higher than third layer, the learning layer is constructed as the previous one [1].

4.2 Experiment Result

The experiment is constructed with two stages, one is the learning stage and other the task execution one using the learned results. First of all, the robot moved at random in the environment for about two hours.

After the learning stage, we let our robot do a couple of tasks. One is shooting a ball into the goal using this multi-layer learning structure. The target situation is given by reading the sensor information when the robot pushes the ball into the goal; the robot captures the ball and goal at center bottom in the perspec-

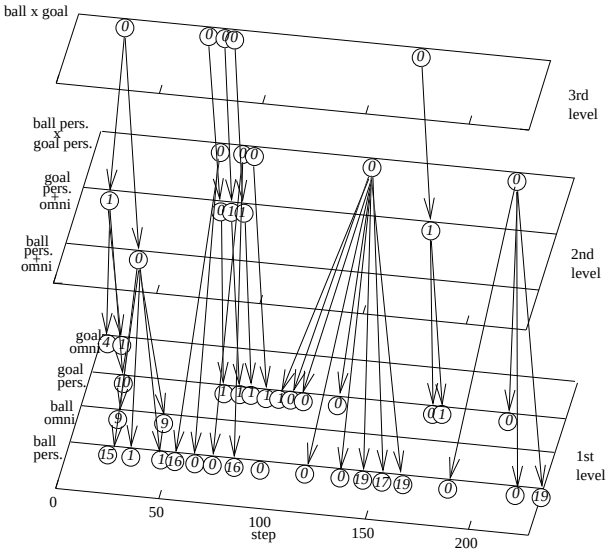


Figure 8: A sequence of the behavior activation of learning modules and the commands to the lower layer modules

tive camera image. As an initial position, the robot is located far from the goal, faced opposite direction to it . The ball was located between the robot and the goal.

Figure 9 shows the time development of the goal state and behavior activations of learning modules at first, second, and third levels while the robot shoots a ball into the goal. The arrows on the top of each series indicate the behavior activations, and the others indicate the goal state activation. Figure 8 shows the sequence of the behavior activation of learning modules and the commands to the lower layer modules. The down arrows indicate that the upper learning modules fire the behavior activations of the lower learning modules.

When the robot located at the initial position, the module 8 in the "ball omni" layer and the module 12 in the "goal omni" layer at the first level has high goal state activations. "ball pers." layer and "goal pers." layer at first level have no activated modules because the robot does not capture the ball or the goal with the perspective camera. When the robot located at the target position, the module 0 at all layers are near to their own goal states. The robot turns its body until 20 steps in order to capture the ball and goal with the perspective camera, and it dribbles the ball, then finally shoots it into the goal.

Figure 8 shows the rough sketch of the activated modules transition and the commands to the lower layer on the multi-layer learning system. First of all, the system tried to activate the module 0 of "*ball pers.*×*goal pers.*" layer at the second level, however, the module could not manage the current situation, because the robot doesn't capture the ball and goal with the perspective camera. Then the system tried to activate the module 0 of "*ball*×*goal*" layer at the third level, and this module activates the module 1 of "*goal pers.+omni*" layer and the module 0 of "*ball pers.+omni*" layer at second level, sequentially. These modules at second level activate adequate modules at first level. When the module 0 of "*ball pers.*×*goal pers.*" layer at second level is able to handle the situation, the module takes over all control of the robot. Sometimes the module 0 of "*ball*×*goal*" layer at the third level is activated when the module "0" of "*ball pers.*×*goal pers.*" layer at second level cannot handle the situation because the robot bumped the ball and the situation changed drastically. Finally, the module 0 of "*ball pers.*×*goal pers.*" layer at the second level leads the robot to the target situation in which the robot is capturing the ball and the goal at center bottom of the perspective camera image.

5 Discussions

This paper proposed a mechanism which constructs several configurations of learning modules at higher layers using several groups of modules at lower layers. We applied the method to a simple soccer situation in the context of RoboCup, show the experimental results.

One missing point in the current method is that it does not have the mechanism that constructs the learning layer by itself. We will extend the method to add new learning layers without human designer's intension.

We need take the action space hierarchy into consideration, while the current method deals with state space hierarchy. We expect we can apply similar approach to this problem.

Acknowledgments

This research was supported by the Japan Science and Technology Corporation, in Research for the the Core Research for the Evolutional Science and Technology Program (CREST) titled Robot Brain Project in the research area "Creating a brain".

References

[1] Y. Takahashi and M. Asada, "Vision-guided behavior acquisition of a mobile robot by multi-layered reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2000, vol. 1, pp. 395–402.

[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] Eiji Uchibe, Masateru Nakamura, and Minoru Asada, "Cooperative behavior acquisition in a multiple mobile robot environment by co-evolution," in *RoboCup-98: Robot Soccer World Cup II, Proc. of the second RoboCup Workshop*, Minoru Asada, Ed., 1998, pp. 237–250.

[4] Noriaki Mitsunaga and Minoru Asada, "Observation strategy for decision making based on information criterion," in *Proceedings of the 2000*

IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000, pp. 1038–1043.

[5] Sebastian Thrun, "A lifelong learning perspective for mobile robot control," in *In Proceedings of the IEEE/RSJ/GI International Conference on Intelligent Robots and Systems*, 1994, vol. 1, pp. 23–30.

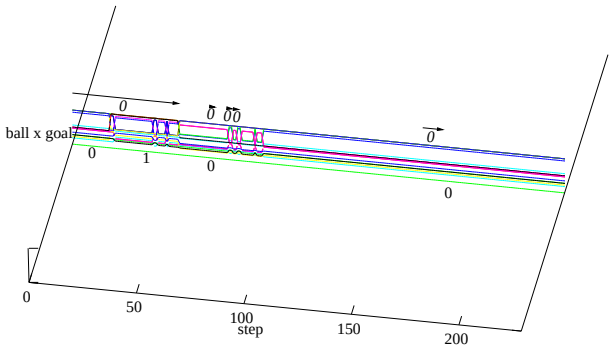
[6] Fumihide Tanaka and Masayuki Yamamura, "An approach to lifelong reinforcement learning through multiple environments," in *6th European Workshop on Learning Robots*, 1997, pp. 93–99.

[7] Takahashi Minato and Minoru Asada, "Environmental change adaptation for mobile robot navigation," in *Proc. of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1998, pp. 1859–1864.

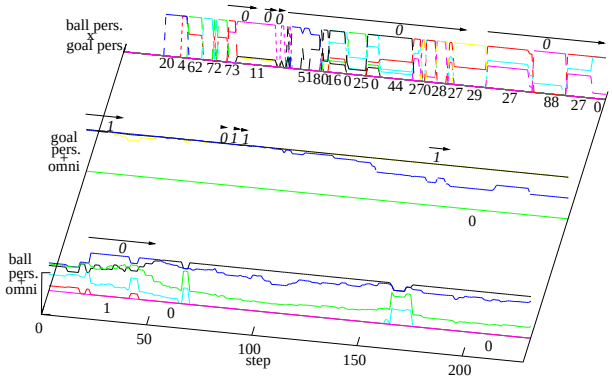
[8] Yasutake Takahashi, Masanori Takeda, and Minoru Asada, "Continuous valued q-learning for vision-guided behavior acquisition," in *Proceeding of the 1999 IEEE International Conference on Multisenso Fusion and Integration for Intelligent Systems*, 1999, pp. 255–260.

[9] J. Tani and S. Nolfi, "Self-organization of modules and their hierarchy in robot learning problems: A dynamical systems approach," Tech. Rep., Sony CSL Technical Report, SCSL-TR-97-008, 1997.

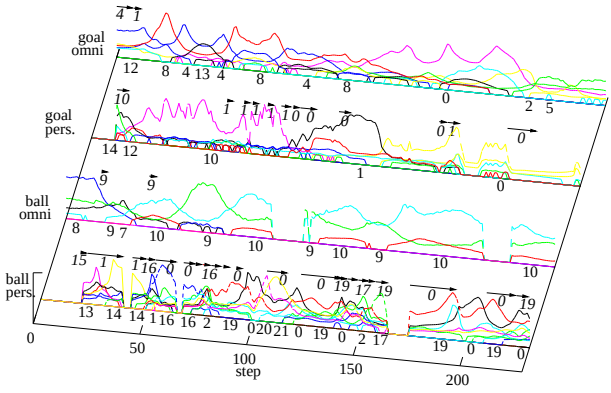
[10] C. J. C. H. Watins and P. Dayan, "Technical note: Q-learning," *Machine Learning*, vol. 8, pp. 279–292, 1992.



(a) third layer



(b) second layer



(b) first layer

Figure 9: A sequence of the goal state activation and behavior activation of learning modules