Activation and Spreading Sequence for Spreading Activation Policy Selection Method in Transfer Reinforcement Learning

Hitoshi Kono[1], Ren Katayama[2], Yusaku Takakuwa[3], Wen Wen[4], Tsuyoshi Suzuki[5]
Department of Engineering, Tokyo Polytechnic University, Kanagawa, Japan[1]
Department of Electronics and Mechatronics, Tokyo Polytechnic University, Kanagawa, Japan[2]
Department of Information and Communication Engineering, Tokyo Denki University, Tokyo, Japan[3,5]
Department of Precision Engineering, The University of Tokyo, Tokyo, Japan[4]

***Abstract*—This paper proposes an automatic policy selection method using spreading activation theory based on psychological theory for transfer learning in reinforcement learning. Intelligent robot systems have recently been studied for practical applications such as home robot, communication robot, and warehouse robot. Learning algorithms are key to building useful robot systems important. For example, a robot can explore for optimal policy with trial and error using reinforcement learning. Moreover, transfer learning enables reuse of prior policy and is effective for environment adaptability. However, humans determine applicable methods in transfer learning. Policy selection method has been proposed for transfer learning in reinforcement learning using spreading activation model proposed in cognitive psychology. In this paper, novel activation function and spreading sequence is discussed for spreading policy selection method. Further computer simulations are used to examine the effectiveness of the proposed method for automatic policy selection in simplified shortest-path problem.**

***Keywords*—Reinforcement learning; transfer learning; spreading activation theory; policy selection**

I. Introduction

Intelligent robot systems are increasingly being developed to solve practical problems. For example, house cleaning, conveyance systems in warehouses, and agricultural systems [1]–[3]. Particularly, reinforcement learning framework is widely discussed in applications of machine learning such as deep Q-network [4], [5]. Reinforcement learning can be explored for optimal policy selection alternate to trial–and–error. This learning algorithm is used with other functions as framework in real world application. Traditional reinforcement learning techniques have long learning time; a disadvantage for implementation in robot systems. To address this problem, transfer learning is proposed for reinforcement learning [7], [8]. The concept of ç has appeared in psychology and education [9]. Transfer learning theory allows the application of a prior knowledge to another similar task. In reinforcement learning, a learning agent is used to draw and transfer policies from previous tasks (source task) to current tasks (target task) [7]. Advantages of transfer learning in reinforcement learning include [7], [8]:

- Fast adaptation to environments (Jump start).
- Learning faster than reinforcement learning (Time to threshold).
- Exploring more effective performances (Asymptotic performance).

The descriptions in parenthesis are the original representation of improvement in performance in transfer learning. In applications of transfer learning of reinforcement learning with pairs, source task and target task, the transfer schemes is called sequential transfer learning [10]. In sequential transfer learning, reusing policy is selected by humans as supervisors [11]–[13]. Parallel transfer learning [10] and multi-task and mutl-robot transfer [14] are proposed alternative approaches. They adopt multiple policy to improve performance of learning agents in target task. A typical disadvantage is if an agent includes an unprofitable policy. Fernández *et al.* and Takano *et al.* in [15]–[18] proposed some policy selection approach. Other related approaches are available in [19]–[21] Here, a learning agent selects a policy from stored multiple policies. Optimal policies are then selected for specific tasks. However, previous policy selection methods have high computational cost if an agent decides to reuse policy from sets of policies such as database.

For humans, many concepts are memorized using a schematic representation as network structure in brain. Spreading activation theory is regarded as realizing *cognitive economy* in cognitive process of humans [22]. Quillian in [23], [24] thoroughly discusses spreading activation theory. An example of semantic network structure for human memory is shown in Fig. 1. Each concept (*e.g.* red, orange, yellow, and green) is connected using paths with semantic distance. That is, connected concepts are related in semantic. In spreading activation theory, each concept has an activation value that can be activated or deactivated by external stimuli such as visual information. If activation value increases beyond the threshold value, the concept remain in human brain. This phenomena is called recall. Activated values spread to related concepts that are connected via paths of semantic distance. If a related concepts' activation value exceed threshold value, they get recalled.

Spreading activation model is an autonomous distributed

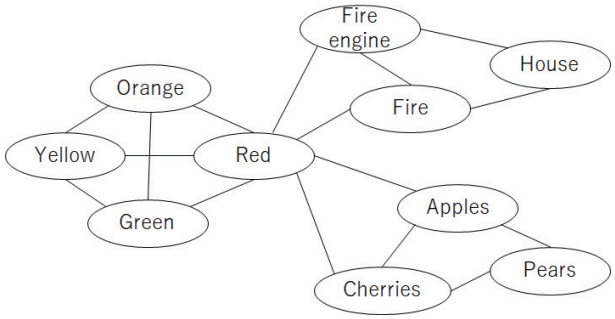

Fig. 1. A schematic representation of network structure including concepts. It is adapted from reference [22].

type. Many existing approaches have centralized decision system architecture. Thus, spreading activation model based on policy selection method is proposed in this study. Stored policies are connected to other policies in the network structure. The network paths denote the semantic distance of spreading activation model. All policies have values equal to the activation value. Concrete functions such as activate function, and spreading function are not formulated in psychological domain. Gaines *et al.* proposed SAN-RL learning algorithm that learns the structure of spreading activation network by reinforcement learning [25]. His study does not consider selecting and leveraging of policies. This study proposes some necessary functions for implementation in a policy selection method based on spreading activation model. From computer simulations, the proposed method for reinforcement learning enables selection of effective policies for task by a learning agent.

The rest of this paper is organized as follows. Section 2 gives an overview of learning algorithms such as reinforcement learning and transfer learning and spreading activation mode in cognitive psychology. Section 3 discusses spreading activation policy network and selection algorithm for transfer learning in reinforcement learning. Section 4 presents the computer simulation experiments and results. Section 5 presents concluding remarks.

II. Previous Works

A. Probabilistic Policy Reuse

Fernández *et al.* proposed a policy selection method using probabilities in [15], [16]. In this method called PRQ-learning, the reusing policy is decided based on Boltzmann distribution selection method (Eqn. (1)) from policy library $L = \{\Pi_1, \Pi_2, \cdots, \Pi_n\}$,

$$P(\Pi_i) = \frac{e^{\tau W_j}}{\sum_{p=0}^{n} e^{\tau W_p}},\ \forall \Pi_j, 0 \leq j \leq n. \tag{1}$$

where τ is a temperature parameter. The expected average reinforcement per episode, W, as defined by

$$W = \frac{1}{K} \sum_{k=0}^{K} \sum_{h=0}^{H} \gamma^h r_{k,h}, \tag{2}$$

where γ, ($0 \leq \gamma \leq 1$), denotes reducing value for future rewards, and $r_{k,h}$ represents immediate reward obtained in step h of the episode k in a total episodes K. Reusing policy decided using the above method enables an agent to reuse policy with the following π–reuse strategy.

$$a = \begin{cases} \Pi_{past}(s) & w/\text{prob.}\psi \\ \epsilon - \text{greedy}(\Pi_{new}(s)) & w/\text{prob.}(1-\psi) \end{cases}. \tag{3}$$

Here past policy is reused with probability ψ and new policy is exploited with probability $1 - \psi$.

PRQ-learning has remarkable success in computer simulation with navigation task based on grid world and keeps away soccer task . It adjusts policy selection based on expected rewards, and adjusts balance between policy reuse and exploring new policies. PRQ-learning does not consider environmental information where the agent act. In robotics applications, observable information are rich because aside reward information, they contain obstacle information from agent, environmental shape and color . Takano *et al.* noted that if a policy is selected; unnecessary actions are increased to select policy [18].

B. Transfer Learning with Forbidden Rule Set

Takano *et al.* proposed a policy selection method different from probabilistic policy reuse. It's selection is based on forbidden rules that pair with policy in policy library [17], [18]. Before policy selection, the agent records forbidden action and state information to the database as forbidden rules set F as a source task. If the agent executes a forbidden action, state and action pair (s, a) are added to F. After building database, the agent selects a policy using a step-by-step selection action in target task.

This approach is applicable in maze exploration simulation to prevent negative transfer until new learning in target task. It considers small maze environment between source task and target task. This method depends on forbidden rule generated in source task, and cannot modify the selection rule. In broad and dynamic environments, the selection frequency or the selection probability is desired to change flexibly.

III. Learning Algorithms and Psychological Theory

A. Reinforcement Learning

Reinforcement learning (RL) is a method of machine learning [4].RL agent explores for optimal solution via trial–and–error and creates its own policies. Thus, RL does not need training data sets unlike supervised learning. Many types of RL algorithms have been developed in decades. In this study, Q-learning is adopted as the learning algorithm [6]. Q-learning is defined by

$$Q(s,a) \leftarrow Q(s,a) + \alpha\{r + \gamma \max_{a' \in A} Q(s',a') - Q(s,a)\}, \tag{4}$$

where $s, s' \in S$ are state of environments in state space S, $a \in A$ is the action of agent in action space A, α is learning rate ($0 < \alpha \leq 1$), γ is discount rate ($0 < \gamma \leq 1$), and r denotes reward. $Q(s,a)$ called Q-table contains all state of environment and each action value pair.

In this study, Boltzmann distribution is adopted as action selection function. The action selection probability adapted from Boltzmann distribution is given by

$$P(a|s) = \frac{\exp\{Q(s,a)/T\}}{\sum_{b \in A} \exp\{Q(s,b)/T\}}, \tag{5}$$

where T is a parameter that determines the randomness of the action selection.

B. Transfer Learning

Transfer learning is a method that reuses prior knowledge in human brain. This concept has been applied in machine learning domain for decades. Transfer method is proposed for reinforcement learning [7]. Transfer policy is given by

$$Q_c(s,a) \leftarrow Q_t(s,a) + Q_s(s,a), \tag{6}$$

were, $Q_c(s,a)$ is current policy including reusing policy. $Q_t(s,a)$ is a policy in target task and $Q_s(s,a)$ is reusing policy from source task. However, $Q_c(s,a)$ has high probability of over fitting if Eq. (6) is used in the target task. To reduce the value of action in $Q_c(s,a)$, transfer rate is proposed to prevent over fitting [26]. Transfer rate similar parameter ζ is proposed by Takano [18]. ζ is analogously used to adjust value of reusing policy, and Kono *et al.* leveraging ζ (denoted by τ in his paper) to prevent over fitting. Transfer method including transfer rate τ $(0 < \tau \leq 1)$ is

$$Q_c(s,a) \leftarrow Q_t(s,a) + \tau Q_s(s,a). \tag{7}$$

In this study, Eq. (7) is adopted for implementation of transfer learning.

In using transfer learning for heterogeneous agents, Taylor *et al.* proposed mapping function, $\chi(\cdot)$, called inter-task mapping (ITM) between source task agent and target task agent. ITM presents the correspondence of state-space and action-space between source task and target task. In this study, homogeneous agents are assumed in source task and target task. Therefore, ITM $\chi(\cdot)$ is not needed for transfer learning.

In the transfer learning, if the agent obtains good effect through reusing policy, it is called positive transfer. In contrast, if the agent encounters the bad situation, this situation is called negative transfer in general.

C. Spreading Activation Model

Spreading activation model is proposed in cognitive psychology as a recall of concept in human brain [22]. In Fig. 1, if concepts "red" is recalled, the activation value of proximate concepts (*e.g.*"orange", "fire" and "Cherries") increase based on activation value of "red". This is spreading of activation value. If activation value is beyond the threshold value, the proximate concepts are recalled simultaneously. Activated concept's activation value can spread to neighboring concepts during the spreading activation process.

TABLE I. VARIABLE DECLARATION IN PROPOSED METHOD

Character	Description
π_i	a policy
Π	Set of policies
$\mathbb{A}_i$	Activated value in π_i
A_a	Activation coefficient
Π^c	Set of candidate policies for selection
T_a	Threshold for activation of policy
T_r	Threshold for recall
$\Delta\mathbb{A}_i$	Decaying value for $\mathbb{A}_i$
w_{ij}	Weight in SAP–net between π_i and π_j
w_p	Adjustment value for w_{ij} with positive transfer
w_n	Adjustment value for w_{ij} with negative transfer

IV. PROPOSED METHOD

This section discuss the proposed method. Relevant parameters of the method is defined in Table I. The subsections in sequel explain necessary functions used in the proposed method. We commence with descriptions of preliminary functions. We assume that the agent has policies and policy network structure in initial state, and all policies have variable activated values.

1) The agent observes environmental information through sensors.
2) Extracted features form observed environmental information.
3) Corresponding policies are activated based on matches between extracted features and policies' label that are labeled using learned environmental information.
4) Activated value is spread to nearby policies from activated policy in this time.
5) Candidate policies are gathered using threshold value.
6) Selection of recall policies based on probabilistic function.
7) Transferring policy (This part is transfer learning).
8) Selecting action and learning (This part is reinforcement learning).
9) Evaluation of policy reuse effectiveness.
10) Adjusting of weights in policy network structure.
11) Back to process of 1).

From above, simplified system architecture of proposed method with reinforcement learning is shown in Fig. 2.

A. Spreading Activation Policy Network

To select the policy selection method in transfer learning through reinforcement learning method, spreading activation mode is adopted. Spreading Activation Policy Network (SAP–net) is a policy selection method of transfer learning in reinforcement learning. This was proposed by Takakuwa *et al.* in [27]. We propose effective functions based on Takakuwa *et al.*

In the initialization, obtained policy π_i is included in policy set Π as an undirected graph G is defined by $G = \{V, E\}$ where E denotes the set of edges between policies. SAP–net is defined as the adjacency matrix $\boldsymbol{A}$ given by G and the set of weights W, that is, $\boldsymbol{A} = (G, W)$.

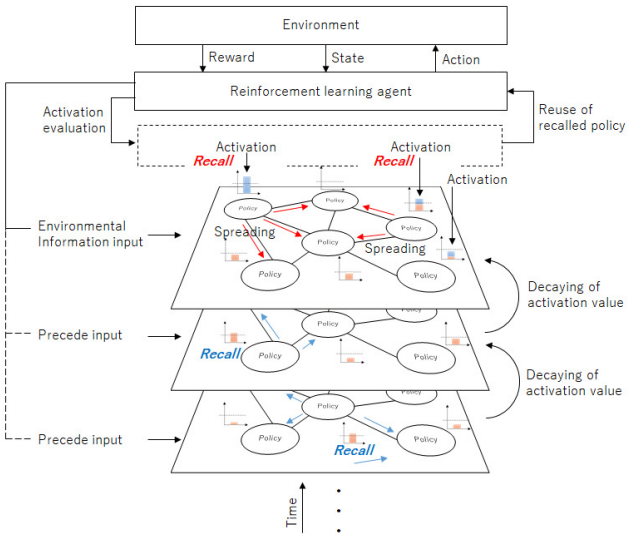

Fig. 2. Simplified system architecture of proposed method. This figure represents standard reinforcement learning concept overview with procedure of spreading activation for policy selection.

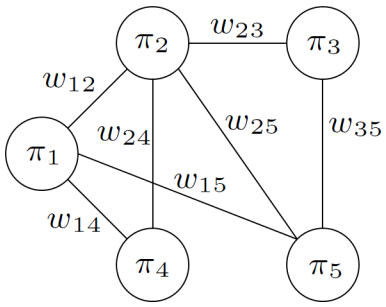

Fig. 3. Example of SAP–net. As default, all nodes are connected with full mesh. All path w_{ij} are assigned weight values and all π_k have activation value $\mathbb{A}_k$.

$$\boldsymbol{A} = \begin{array}{c} \\ \pi_1 \\ \pi_2 \\ \pi_3 \\ \pi_4 \\ \pi_5 \end{array} \begin{array}{c} \begin{array}{ccccc} \pi_1 & \pi_2 & \pi_3 & \pi_4 & \pi_5 \end{array} \\ \begin{pmatrix} 0 & w_{12} & 0 & w_{14} & w_{15} \\ w_{12} & 0 & w_{23} & w_{24} & w_{25} \\ 0 & w_{23} & 0 & 0 & w_{35} \\ w_{14} & w_{24} & 0 & 0 & 0 \\ w_{15} & w_{25} & w_{35} & 0 & 0 \end{pmatrix} \end{array} \tag{8}$$

is example adjacency matrix associated with Fig. (3) with $w_{ij} \in W$ and $w_{ij} \geq 1$. Note that large w_{ij} values indicate long paths. Initial state G is a full mesh network and behavior of graph depend on the weights w_{ij}. Extremely large w_{ij} indicate that path is disconnected asymptotically.

All policies have activated value $\mathbb{A}$. Therefore, node $v_i \in V$ of graph is configurated as $(\pi_i, \mathbb{A}_i)$. As the default $\mathbb{A}_i = 0$.

B. Activation Function

In match of a feature paired with corresponding policy, activated value $\mathbb{A}_i$ is updated by feature comparison function $C(\cdot, \cdot)$ between feature of observed environmental information

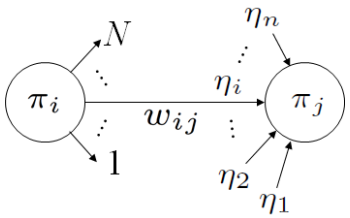

Fig. 4. Simplified illustration of spreading of activated value from π_i to π_j. In this situation, π_i is activated and outputs the activation value. π_j receives activated value η_i that is reduced by w_{ij}.

s and feature of policy that is environmental feature when the policy is obtained. Activation function is defined by

$$\mathbb{A}_i = \begin{cases} \mathbb{A}_i + A_a & (C(s, s_{\pi_i}) \geq T_a) \\ \mathbb{A}_i & (\text{Otherwise}) \end{cases}, \tag{9}$$

where A_a is activation coefficient value, $C(s, s_{\pi_i})$ is a function of extracted features similarity between s and s_{π_i}, and s_{π_i} is environmental feature when the policy π_i is obtained by the agent.

C. Spreading Function

Each policy included in SAP–net can spread the activated value to neighboring policies. A simplified spreading is depicted in Fig. 4. If policy π_i has activated value $\mathbb{A}_i$, it spreads to neighbor π_j as propagating activated value η_i.

$$\eta_i = \frac{1}{N} \mathbb{A}_i e^{-w_{ij}}. \tag{10}$$

Here, N is number of output path of policy π_i. Therefore, policy π_i outputs same activated value calculated from Eq. (10). Policy π_j receives more than one activated value from neighboring policies as shown in Fig. 4. The total activated value received by π_j is given by

$$\mathbb{A}_j \leftarrow \mathbb{A}_j + \sum_{k=1}^{n} \eta_k. \tag{11}$$

For example, if two spreading target are exit in the activation scene like Fig. 5, each spread activation value is calculated by Eq. (10). Policies π_j and π_k spread values to each other after spreading activation of π_i. Policy π_j received spread value η_l from π_k, and π_k received spread value η_m from π_j. Spreading direction and spreading activation value are decided by following equation.

$$\eta_l = \begin{cases} {}^{\triangle}\eta & ({}^{\triangle}\eta > 0) \\ 0 & (\text{Otherwise}) \end{cases}, \tag{12}$$

$${}^{\triangle}\eta = \frac{1}{N-1} \mathbb{A}_k e^{-w_{jk}} - \frac{1}{N-1} \mathbb{A}_j e^{-w_{jk}}. \tag{13}$$

Here, ${}^{\triangle}\eta$ is deference of propagating activation value from π_j and π_k. In calculating propagating activation value, N is changed to $N-1$ because a node as policy is prohibited to use receiving path (*e.g.* η_j and η_k) for output path.

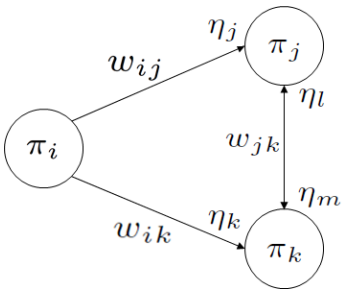

Fig. 5. Situation of multiple spreading of activation value. In this case π_j and π_k are affected to each other in same time. As a calculation result, activation value is propagated to either one like Eq. (13).

Note that, if SAP–net is contained with multiple nodes, and activation and spreading are emerged simultaneously, SAP–net will behave as the system, like a many-body problem. This SAP–net is set to sequential spreading of activation as a basic setting.

D. Threshold for Recall

Before selecting a reusing policy, candidate policy π_i is extracted with activation value $\mathbb{A}_i$. To create the set of candidate policies Π^c, π_i is filtered by following threshold function $\mathbb{T}(\cdot)$ using $\mathbb{A}_i$.

$$\mathbb{T}(\pi_i) = \begin{cases} \pi_i \in \Pi^c & (\mathbb{A}_i > T_r) \\ \pi_i \notin \Pi^c & (\mathbb{A}_i \leqslant T_r) \end{cases}, \tag{14}$$

where, Π^c is the set of candidate policies, and T_r is the threshold value. For the implementation, π_i is gives the optional value $\mathbb{A}_i^c$.

E. Policy Selection

We forbid the recall of multiple policy in same time. Therefore, a policy is selected by the following function. Reusing policies are selected from candidate policies Π^c. A reusing policy is decided by soft-max like function defined by

$$\mathbb{S}(\pi_i) = \frac{\exp \mathbb{A}_i}{\sum_{\pi_j \in \Pi^c} \exp \mathbb{A}_j}. \tag{15}$$

Reusing policy is decided in arbitrary timing. In further works, we will seek to reuse multiple policy in same time and their assimilation.

F. Decaying Process

Activated values decay in each time step. It synchronizes with time or actions. This mechanism is inspired by human oblivion phenomenon. Decay process is implemented for proposed method to re-flesh the state of SAP–net in long time learning.

In this study, decay value is calculated using

$$\Delta\mathbb{A}_i = \begin{cases} 0 & (\mathbb{A}_i = 0) \\ d & (\mathbb{A}_i > 0) \end{cases}, \tag{16}$$

where $\Delta\mathbb{A}_i$ is the decaying value for the activated value $\mathbb{A}_i$ in π_i, and d is the value of decay constant. This decaying value is used in the following equation to decrease present activated value $\mathbb{A}_i$

$$\mathbb{A}_i = \mathbb{A}_i - \Delta\mathbb{A}_i. \tag{17}$$

G. Activation Evaluation

Until an action by the agent, proposed method evaluates the effectiveness of selected policy reuse. If the agent observes positive transfer (PT) effect, weight in SAP–net is adjusted to small between previous used policy and current reuse policy. Thus, the network path's length becomes short in SAP–net. In case of negative transfer (NT), the weight is also adjusted to big value. Thus, the network path's length becomes long.

In this study, weights w_{ij} are adjusted in each action. Adjustment function is defined by.

$$w_{ij} = \begin{cases} w_{ij} - w_p & \text{if PT is emarged} \\ w_{ij} + w_n & \text{if NT is emarged} \end{cases}, \tag{18}$$

where w_{ij} is weight of connection between current reusing policy π_i and previous reused policyπ_j. Weight w_{ij} needs to be controlled using $w_{ij} < 1$ if positive transfer.

Activated policy's activated value is evaluated by action result. If negative transfer is emerged such as the agent collide obstacles based on action with selected policy, activated value gives penalty value. This function is defined by

$$\mathbb{A}_i = \begin{cases} \mathbb{A}_i - A_p & (\text{NT is emerged}) \\ \mathbb{A}_i & (\text{Otherwise}) \end{cases}. \tag{19}$$

V. Computer Simulation

A. Conditions

This experiment aims to confirm the emerging effect of transfer using the proposed method. To evaluate proposed method SAP–net, shortest path problem is adopted in this paper as the basic evaluation with single agent and learned multiple policies. Learning environment is set as Fig. 6 in computer simulation. In Fig. 6, if the agent achieved the goal, the agent can obtain reward from environment. Agent can observe self-localization and around environment such as wall or street with range defined by FOV (field of view) of agent. Reinforcement learning parameter is set as Table II. Positive reward is given to the agent if achieves goal position. Negative reward is given if the agent conflicts to the obstacle per every step, this situation means negative transfer in Eq. (18) and Eq. (19).

The agent has 100 policies that are learned as a source task, for random start and goal positions. When the agent is in a target task environment (Fig. 6), selecting policies include helpful and useless policies for transfer. Helpful policies refer to policies that can be contributed to solve shortest path in target task environment. In contrast, useless policies refer to those policies that are not related to solve the shortest path; these policies have the probability for emerging as negative transfers to use useless policies. Examples of helpful and

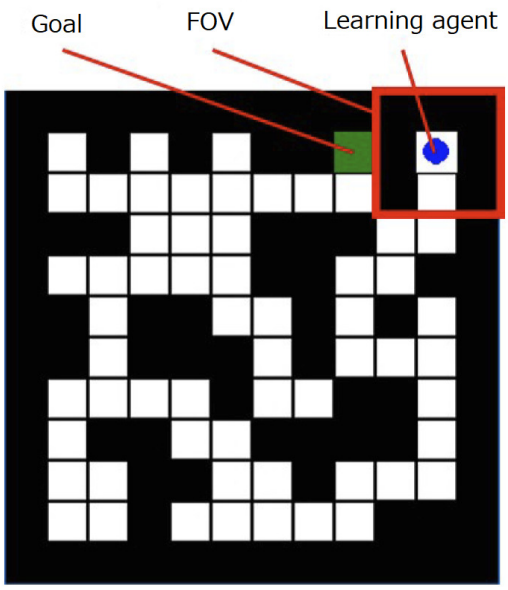

Fig. 6. Experimental environment in computer simulation. An environment includes an agent and a goal. The agent can observe self-localization and around environmental situation such as wall or street.

TABLE II. SET VALUES OF PARAMETERS FOR REINFORCEMENT LEARNING AND PROPOSED METHOD

Parameter	Source task	Target task
Learning rate α	0.9	0.9
Discount rate γ	0.1	0.1
Positive reward r^+	1.0	1.0
Negative reward r^-	0.0	-1.0
Boltzmann parameter T	0.01	0.01
Number of episodes	400	200
Number of trial	–	10

useless policies are shown in Fig. 7 and Fig. 8, respectively. The environmental shape of helpful policies are the same as the target task environment. In contrast, the environment of useless policies are different compared with target task environment. In this experiment, some shape of an environment is used in source tasks.

As default SAP–net configuration, all weights w_{ij} are five, all activation values $\mathbb{A}_i$ are zero. The network structure of the SAP–net is also set as a full mesh. The parameters of SAP–net are set as Table III. In this experiment, environmental information can be described with a discrete state, the feature comparison function $C(s, s_{\pi_i})$ helps determine whether the information is "matched" or not. Therefore, T_a needs not be adjusted in this experiment; the policy is activated only if there is a perfect matched between the current state s and the state s_{π_i}, which is observed when the policy π_i is learned.

B. Evaluation Factors

This experiment adopted the two types of basic evaluation factors. 1st is "learning" curve and area of learning curve, which is called "transfer ratio". The others are three main objectives in transfer such as "jumpstart improvement", "learning speed improvement", and "asymptotic improvement". Learning speed, asymptotic, and jumpstart improvement are proposed by Langlay in 2006 that are described in Lazaric's paper [8].

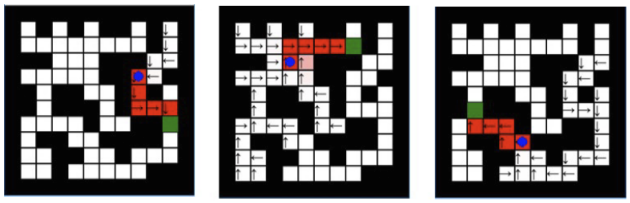

Fig. 7. Example of helpful policies for policy selection. Each learned path is included to the shortest path of the target task.

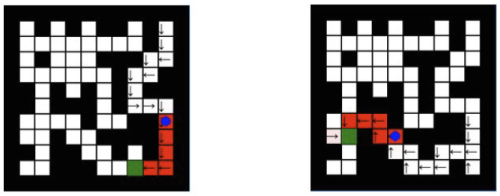

Fig. 8. Example of useless policies. The shape of the environment and path are slightly different.

TABLE III. PARAMETERS SETTING FOR SAP–NET FOR THE EXPERIMENTAL SETUP

Variable	value
Default activation value $\mathbb{A}_i$	0.0
Activation coefficient $\mathbb{A}_a$	1.0
Threshold for activation of policy T_a	Perfect match
Threshold for recall T_r	0.6
Decaying value $\Delta\mathbb{A}_i$	-0.001
Default weight between policies w_{ij}	5
Adjustment value of w_{ij} when PT	-0.5
Adjustment value of w_{ij} when NT	2.5

1) Learning curve: The learning curve is the most basic evaluation method among learning procedures. Reinforcement learning can also use the learning curve to evaluate the performance of the learning process (Fig. 9). The vertical axis of the learning curve denotes the performance of the learning agent, and the number of actions and problem solving time serve as its performance. The horizontal axis of the learning curve is time steps.

If the environment is defined as a Markov decision process (MDP), the learning curve is converged to optimum solution, which means that the fast problem-solving time, shortest path route, and so on. The converged learning curve aids understanding performance transition intuitively by humans. In a non-MDP environment such as a dynamic environment, learning curve is not converged; however, the shape of curve is emerged propensity for convergence with some fluctuations.

2) Jump start: Jump start is the difference between the value of the performance obtained with transfer and that without (Fig. 10). Normally, the learning curve of the initial state has some fluctuations; therefore, in this experiment, jump start value J_s is formulated as follow:

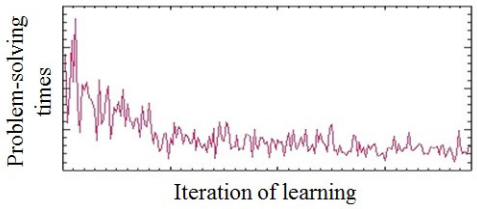

Fig. 9. Example of learning curve. This curve is not converged to optimal solution because this result is obtained by non-MDP environment experiment. However, an observer can read the increasing performance of the agent using this learning curve with every time step.

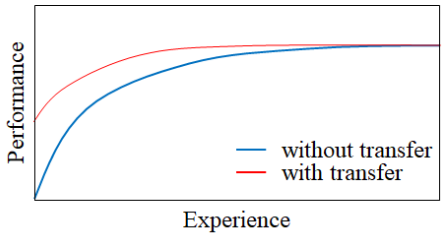

Fig. 10. Simplified example of the jump start that emerged in the learning curve. The blue curve explains the performance transition without transfer, which is the learning curve of reinforcement learning. The red curve denotes performance transition with transfer.

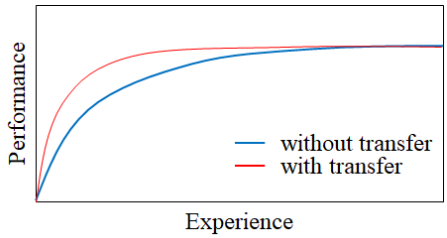

Fig. 11. Simplified example of learning speed improvement that emerged in the learning curve. The blue curve explains the performance transition without transfer, which is the learning curve of reinforcement learning. The red curve denotes performance transition with transfer.

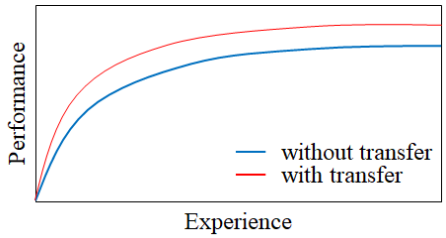

Fig. 12. Simplified example of asymptotic improvement which emerged in the learning curve. The blue curve explains the performance transition without transfer, which is the learning curve of reinforcement learning. The red curve denotes performance transition with transfer.

$$J_s = \frac{1}{n}\left(\sum_{i=1}^{n} L_t(i) - \sum_{i=1}^{n} L_{wt}(i)\right) \tag{20}$$

Here, i is episode number of learning simulation. $L_t(t)$ is learning curve in transfer, and $L_{wt}(t)$ is learning curve without transfer, which means that pure reinforcement learning. Eq. (20) is formulated by Kono *et al.* [26]. If jump start is calculated with an upward curve such as Fig.10, the value of the jump start is positive; in other words, the performance is increasing. In contrast, if the jump start has a downward curve such as learning curve (Fig. 9), the value represents a negative that means that learning time is decreasing.

3) Learning speed improvement: Learning speed improvement means a reduction of area of the learning curve even if the start performance is the same value. Fig. 11 shows that learning curve converges faster than the learning curve without transfer. In other words, this phenomenon is represented as the speed of convergence.

If the initial performance and converged values of the learning curve are the same with or without transfer, learning speed improvement can help calculate transfer ratio r as defined below:

$$r = \frac{\sum L_t(t) - \sum L_{wt}(t)}{\sum L_{wt}(t)}. \tag{21}$$

4) Asymptotic Improvement: Asymptotic improvement means that the final performance is improved by transfer. Fig. 12 shows the case of improvement; however, this emergence depends on the task and learned policy. This is because when depending on the task, there is an upper limit on performance, for example shortest path problem.

5) Transfer ratio: Taylor [7] proposed transfer ratio as a comparison method based on the area of the learning curves with or without transfer. If learning curve can be described as function $L(t)$, transfer ratio is defined by Eq. (21).

If the evaluation function is shaped as an upward curve (*e.g.* transition of rewards) as shown in Fig. 10, value r is positive. It represents the increasing performance or learning efficiency. In contrast, if the transfer ratio is evaluated as a downward curve, such as the learning curve, the value r becomes negative. It represents the decreasing of learning time or learning cost at the results .

C. Results and Discussions

First, criterial three learning curves are shown in Fig. 13. “Without transfer”, “Positive transfer” and “Negative transfer” are abbreviated as WT, PT and NT respectively. Proposed method as SAP–net is abbreviated as SAP. In the Fig. 13, represented learning curves are calculated averages from 10 trial, and standard deviation is represented with learning curve at 50 episode intervals. The red colored line is learning curve of WT equivalent to reinforcement learning. The blue colored line is learning curve of PT if the agent transferred policy from source task with helpful policy. This gives positive transfer. Green colored line is learning curve of NT. Here, the agent reused useless policy from the source task, and transferred policy inhibited learning in target task. This learning curve does not converge. It has huge fluctuation because the agent encountered dead lock with the wall in grid world, and it was difficult to re-learn the impeding behavior of dead lock until

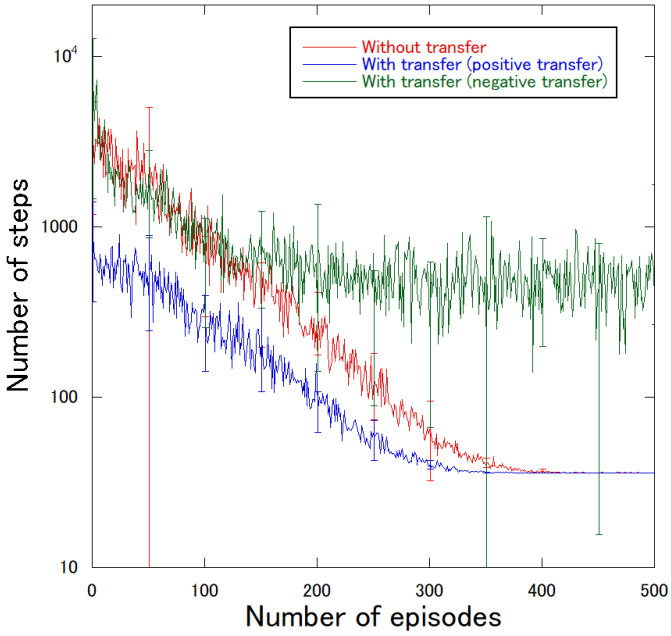

Fig. 13. Basic results of learning curve reinforcement learning, transfer learning with positive transfer and transfer learning with negative transfer.

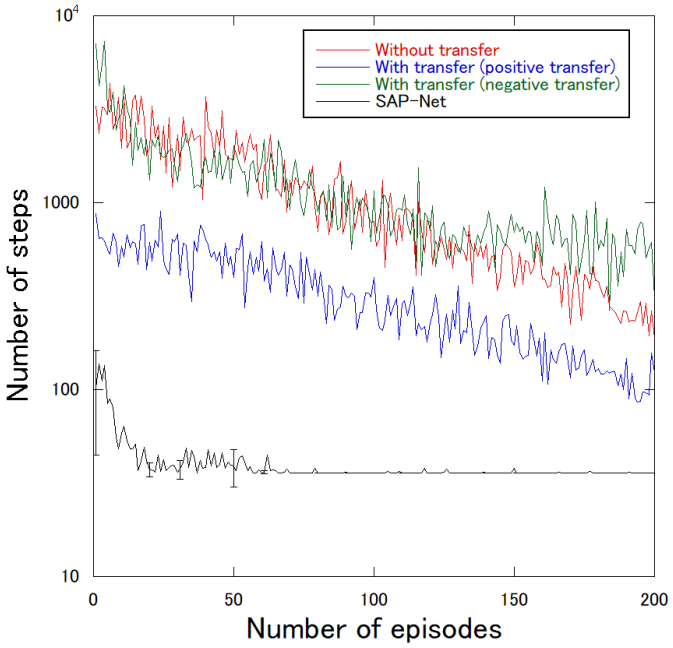

Fig. 14. Comparison among proposed method's learning curve and above basic results as Fig. 13. WT, PT and NT are the same as shown in Fig. 13.

1000 episodes. From the learning curve without transfer and learning curve on positive transfer, jump start and learning speed were improvement. However, asymptotic improvement did not occur because of optimal policy as shortest path in the grid world is the same between conditions without transfer and positive transfer.

The result of the learning curve with proposed method is shown in Fig. 14 represented with 200 episodes. Black colored line is learning curve of proposed method as SAP–net. It is a calculated 10 trials average. Standard deviation is represented at 50 episode intervals. SAP–net's learning curve extremely decreases compared to conditions WT, PT and NT, and convergence speed is faster than other conditions. Comparison of jump start value is indicated in Table IV. From WT, result of condition PT and SAP are exhibited jump start. This phenomenon means that the agent can solve the problem with fast time compared to WT. Moreover, SAP has a lower jumpstart value than PT, indicating that performance is high in the early stages of learning. Condition NT does not give jump start value, thus the performance worsens. A comparison of transfer ratio is shown in Table V. Condition PT and SAP decrease the value of transfer ratio thus the overall learning time is improved. In this result, WT and NT have near value of transfer ratio r. From Fig. 13, NT has a higher transfer ratio than WT. From these results, learning speed of condition PT is improved compared to WT. SAP can shorten the total learning time compared to condition PT. Therefore, not only PT but also SAP exhibited learning speed improvement. This result indicate that policy selection by proposed method has effectiveness for transfer learning in reinforcement learning. Additionally, in this experimental condition, shortest path has minimum limits, therefore asymptotic improvement is not achieved from this results.

Each activation value of policies is shown in Fig. 15 if the

TABLE IV. COMPARISON IN JUMP START

Condition	Averaged steps (10 episodes)	Jump start
WT	3024.33	0.00
PT	623.07	−2401.26
NT	4253.29	1228.96
SAP	90.08	−2934.25

TABLE V. COMPARISON IN TRANSFER RATIO

Condition	Area under curve	Transfer ratio
WT	237463.40	0.00
PT	66689.40	−0.72
NT	248607.10	0.05
SAP	8033.40	−0.97

simulation is terminated. Activation value is averaged from 10 trials and standard deviation is calculated with 10 trials. Mainly seven policies are used and selected for solving shortest path problem. Initial activation value and weights of the path in SAP–net are same initialized value; therefore, activation value is adjusted by proposed method with learning in target task, and helpful policies' activation value become high value. Useless policies' activation value are not activated. Although some policies (*e.g.* policy number 36, 38 and 83) have low values, the activity value is updated a little. Transferred policies are generated in source task with random start position and goal position; therefore, a few policies selected were not effective. However, because the proposed method is learning algorithm, the process of selecting a policy through trial and error is important in transfer learning phase. Fig. 15 shows that final activation value of helpful policies are approximately 200; its higher value compared with initial activation value as 5. This is caused by insufficient tuning of decaying value

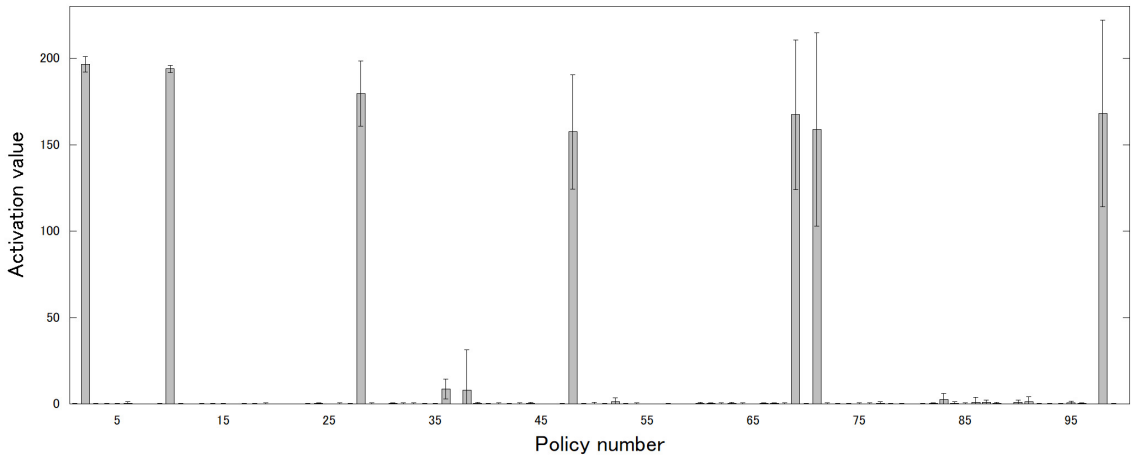

Fig. 15. Activation values when simulation terminated. Seven policies are founded useful from random generated source task's policy, and useless policies are not activated in SAP–net through this experiment.

$\Delta\mathbb{A}_j$. SAP–net is sensitive for parameter tuning, and many parameters are implemented. SAP–net effective from this experiment, however parameter setting needs to be discussed for generalization and applications.

VI. Conclusion

This paper proposed a novel policy selection method for transfer learning in reinforcement learning. Proposed method is inspired by the spreading activation theory discussed in cognitive science. SAP–net proposed contains functions such as networked stored policies, activation function, spreading function, decaying function, and recall function. Basic experiments were performed with shortest path problem using reinforcement learning agent that can select learned policies. From the experimental results, quantitative evaluation was performed, and results suggest that learning agent with SAP–net can solve the problem faster than WT and PT conditions. The agent with SAP–net selects policies adaptively from the environment.

As the future works, it is necessary to discuss the parameter settings. SAP–net is sensitive for activation and decaying and is related to trade-off. If a high activation value is set, the decaying value cannot cancel the activation value hence continues to rise. Further, decaying is strongly affected by the SAP–net and activation value may not rise. The proposed method is constructed with implementation as sequential, and calculation cost increases with number of policies. Order of proposed method is approximately $O(n^2)$, calculation is sequential processing. Parallelization method is also an important issue in implementation phase. Moreover, behavior SAP–net is connected to N-body problem, more theoretical consideration is required for system behavior.

Acknowledgment

This study was supported by JSPS KAKENHI Grant number JP18K18133 and JP19K12173, and this study was funded in part by the HAYAO NAKAYAMA Foundation for Science & Technology and Culture. We would like to thank Editage (www.editage.com) for English language editing.

References

[1] B. Ramalingam, A. K. Lakshmanan, M. Ilyas, A. V. Le, and M. R. Elara (2018). "Cascaded Machine-Learning Technique for Debris Classification in Floor-Cleaning Robot Application." *Applied Sciences* 8(12): 1–19.

[2] R. D. Andrea (2012). "Guest Editorial: A Revolution in the Warehouse: A Retrospective on Kiva Systems and the Grand Challenges Ahead." *IEEE Transactions on Automation Science and Engineering* 9(4): 638–639.

[3] T. Fukatsu, G. Endo, and K. Kobayashi (2016). "Field Experiments with a Mobile Robotic Field Server for Smart Agriculture." *Proceedings of the WCCA-AFITA2016*: pp.1–4.

[4] R. S. Sutton and A. G. Barto (1998). "Reinforcement learning: An introduction." MIT press.

[5] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis (2015). "Human-level control through deep reinforcement learning." *Nature* 518: 529–533.

[6] C. J. C. H. Watkins and P. Dayan (1992). "Q-Learning." *Machine Learning* 8: 279-292.

[7] M. E. Taylor and P. Stone (2009). "Transfer learning for reinforcement learning domains: A survey." *Journal of Machine Learning Research* 10(Jul): 1633–1685.

[8] A. Lazaric (2012). "Transfer in Reinforcement Learning: A Framework and a Survey. Reinforcement Learning. Adaptation, Learning, and Optimization." Berlin, Heidelberg, Springer. 12: 143–173.

[9] J. M. Royer (1978). "Theories of Learning Transfer." *Technical report No. 79, University of Illinois at Urbana–Champaign*: 1–54.

[10] A. Taylor, I Dusparic, E. Dalván–López, C. Siobhán, and C. Vinny. (2014). "Accelerating learning in multi-objective systems through transfer learning." *2014 International Joint Conference on Neural Networks*: pp.2298–2305.

[11] M. E. Taylor (2009). "Transfer in Reinforcement Learning Domains." Studies in Computational Intelligence 216: Springer.

[12] B. Lakshmanan and R. Balaraman (2010). "Transfer learning across heterogeneous robots with action sequence mapping." *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*: pp.3251–3256.

[13] M. Koga, V. F. Silva, F. G. Cozman, and A. H. R. Costa (2013). "Speeding-up reinforcement learning through abstraction and transfer learning." *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*: pp.119–126.

[14] C. Devin, A. Gupta, T¿ Darrell, P. Abbeel, and S. Levine (2017). "Learning modular neural network policies for multi-task and multi–

robot transfer." *2017 IEEE International Conference on Robotics and Automation*: pp. 2169–2176.

[15] F. Fernández, J. Garcia, and M. Veloso (2010). "Probabilistic policy reuse for inter-task transfer learning." *Robotics and Autonomous Systems* 58(7): pp. 866-871.

[16] F.Fernández, and M. Veloso (2013). "Learning domain structure through probabilistic policy reuse in reinforcement learning." *Progress in Artificial Intelligence* 2(1): 13-27.

[17] T. Takano, H. Takase, H. Kawanaka and S. Tsuruoka (2010). "Effective Reuse Method for Transfer Learning in Actor-critic." *Proceedings of the SCIS & ISIS 2010*: 137-141.

[18] T. Takano, H. Takase, H. Kawanaka, H. Kita, T. Hayashi, and S. Tsuruoka (2011). "Transfer Learning based on Forbidden Rule Set in Actor-Critic Method." *International Journal of Innovative Computing, Information and Control* 7(5(B)): pp. 2907–2917.

[19] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel (2017). "A unified game-theoretic approach to multiagent reinforcement learning." *Proceedings of the Neural Information Processing Systems* pp. 4190–4203.

[20] S. Doroudi, P. S. Thomas, and E. Brunskill (2017). "Importance Sampling for Fair Policy Selection." *Proceedings of the Thirty-Third Conference on Uncertainty in Artificial Intelligence*.

[21] A. Mittel and P. S. Munukutla (2019). "Visual Transfer Between Atari Games Using Competitive Reinforcement Learning." *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

[22] A. M. Collins and E. F. Loftus (1975). "A spreading-activation theory of semantic processing." *Psychological Review* 82(6): 407-428.

[23] M. R. Quillian (1967). "Word concepts: a theory and simulation of some basic semantic capabilities." *Behavioral Science* 12: pp.410–430.

[24] M. R. Quillian (1969). "The teachable language comprehender: a simulation program and theory of language." *Computational linguistics* 12(8): pp.459–476.

[25] D. M. Gaines, J. White, M. Wilkes, K. Kusumalnukool, S. Thongchai and K. Kawamura (2001). "SAN-RL: combining spreading activation networks and reinforcement learning to learn configurable behaviors." *Proceedings of the Intelligent Systems and Advanced Manufacturing*.

[26] Hitoshi Kono, Akiya Kamimura, Kohji Tomita, Yuta Murata and Tsuyoshi Suzuki (2014) "Transfer Learning Method Using Ontology for Heterogeneous Multi-agent Reinforcement Learning." *International Journal of Advanced Computer Science and Application* 5(10): pp.156–164.

[27] Y. Takakuwa, H. Kono, W. Wen, and T. Suzuki (2019). "A Study on Policy Reuse Method Using Psychologically Inspired Model in Transfer Learning." *Proceedings of the 24th Robotics Symposia*: pp.86–89 (in Japanese)