Learning of Deterministic Exploration and Temporal Abstraction in Reinforcement Learning

Katsunari Shibata shibata@cc.oita-u.ac.jp
Dept. of Electrical and Electronic Engineering, Oita University, 700 Dannoharu, Oita 870-1192

Abstract: Temporal abstraction and exploration are both very important factors to determine the performance in reinforcement learning. The author has proposed to focus on the deterministic exploration behavior that is obtained through reinforcement learning. In this paper, a novel idea that deterministic exploration behavior can be considered as temporally abstract actions or macro actions was introduced. It was actually shown in some simulations that the deterministic exploration behavior obtained through the learning of a task accelerates the learning of another similar task without any definition of abstract actions. A recurrent neural network was used for the learning, but the knowledge obtained through the first learning was used effectively in the second learning without being destroyed completely even though it did not work in a more difficult task. Furthermore, when the agent was returned to the first task, the learning was still faster than the learning from scratch. An interesting phenomenon was observed in the simulation that context-based exploration behavior was acquired through the learning of a task that did not require such behavior.

Keywords: Temporal Abstraction, Deterministic Exploration, Reinforcement Learning, Recurrent Neural Network

1. INTRODUCTION

Reinforcement learning (RL) realizes autonomous and purposive learning, and that is a major reason why it is expected as a promising technique in robotics and other autonomous learning systems. The author also has a deep interest in explaining the emergence of various functions of real lives based on RL and in bridging the deep gulf between our intelligence and the modern artificial intelligence. Generally, abstraction in both space and time domains is expected to accelerate learning drastically. The author also expects at present that to investigate the emergent process of abstraction leads to understanding the origin of our intelligence.

As for the temporal abstraction, many researches have been done under the term of hierarchical RL or subgoal, and many of them have aimed at autonomous subgoal finding such as [1]. However, in the most cases, hierarchical structure is given beforehand and appropriate subgoals are found among some candidates through learning. Some techniques[2][3] have been proposed to find subgoals autonomously in the options framework[4] by looking for "bottleneck states" without giving a hierarchy beforehand. However, they need a special process to find the bottleneck states, and the transfer to another task is not mentioned. Furthermore, they seem difficult to be applied to continuous state space and also to be extended to higher-order abstraction.

The autonomous learning ability of RL is supported by trial and error based on reward and punishment. The trial and error is usually called "exploration" and it is clear that exploration also deeply influences on the performance of RL. Usually, it is realized by stochastic factors using random numbers in action selection such as ε-greedy or Boltzmann selection[5].

The author has noted the difference between what we are doing as exploration and the exploration that is usually implemented in RL. The former does not seem to be realized by just stochastic factors, but seems to be performed as deterministic action selections using the context and the knowledge obtained through the past experiences. As oppose to the random exploration, the deterministic exploration looks intelligent and need some knowledge.

For example, when we face a fork on a road as shown in Fig. 1, we don't choose a motion randomly at each actuator level, but choose one of the paths to go on. When we face a door that we don't know how to open and find a button just beside the door, we must try to push it to open the door. Therefore, the author has raised the necessity of such kind of exploration, and has shown by simulation of a simple learning task that an agent could learn a context-based deterministic exploration behavior by RL using a recurrent neural network[6][7].

In the simulation, no stochastic exploration on the abstract action level is done and also even what abstract actions are was not given beforehand. However, when we see the deterministic exploration obtained by RL, the agent seems to choose one of the abstract actions rather

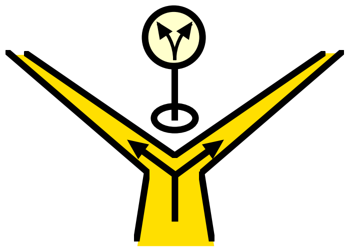
Fig. 1 The relation between deterministic exploration and temporal abstraction is explained in a simple fork problem. Before the fork, we don't choose a motion of each actuator randomly, but usually choose to go on one of the paths. In this case, it is supposed that we have already obtained abstract action through our past experiences.

than one of the primitive actions. The author thought that this behavior must deeply relate to the temporal abstraction. Since a recurrent neural network is used, spatial abstraction[8] and the extension to higher-order abstraction can be also expected, the author thinks, and the application to continuous state space might be easy. In order to call such behaviors abstract actions, the agent, at least, has to explore efficiently in a similar task in which the abstract actions are still useful. However, since the exploration behavior is obtained in a recurrent neural network, it is concerned that the learned exploration behavior is destroyed in the learning of another task and does not accelerate the learning. In this research, it is examined whether the agent that learned deterministic exploration in a small room of grid world can learn the actions to the goal efficiently in a 4-room task in which the obtained exploration behavior can be utilized.

2. TEMPORAL ABSTRACTION ACQUIRED THROUGH REINFORCEMENT LEARNING

In [4], a 4-room task is introduced as an example, and it is explained that the learning is accelerated by introducing abstract actions named hallway options. However, hallway options, each of which is a combination of primitive actions to go to one of the two hallways in one room, are given beforehand. When we think of the fork case described above using Fig. 1 as a RL task, we deal with it as a binary decision task by defining two primitive actions, each of which is to go on one of the two paths. In order to build an intelligent robot or agent, autonomous acquisition of abstraction process is thirsted, but in this case, temporal abstraction based on our experience has been given by us unconsciously. The author thinks that to investigate the emergence of the abstraction process opens the way to higher level of abstraction that we are also doing usually.

In the fork task, it can be thought that we obtained the knowledge through our past experiences that to go on one of the paths is a better choice than to make random motions at each actuator level even though we don’t know which path is better to go on. In order to simplify this situation, the author set up a binary decision task on a grid-world room as shown in Fig. 2, where there are two landmarks of goal, but one of them is the real goal and the other is a fake. The agent does not know which is the real goal in advance. It was shown that the agent, who has a recurrent neural network, learned an action series to go to one of the landmarks at first, and then to go to the other one if the first landmark was not the real goal[6][7]. In the obtained behaviors, the distance to each landmark seems to be reflected to decide the landmark to go first.

In this case, we did not give and also hardly know definitely what are the options or abstract actions. However, as discussed above using the fork task, since the agent seems to know that it is a better choice to go to one of the landmarks to reach the goal, the author thinks the acquired behaviors can be interpreted as abstract actions. The reason why the abstraction is so important is that learning in a similar situation can be accelerated. The purpose of this research is to examine whether the agent after the learning of the binary decision task in a 1-room environment can learn faster in a 4-room task without giving what are the options or abstract actions definitely.

One thing to be noted to make the abstraction work effectively is that the choice of sensor is a critical factor. If the sensor provides some absolute information, for example, a camera equipped at the ceiling provides the world coordinate of a robot, the similarity for the robot between two situations is hardly expressed. Therefore, it is important to use a sensor that provides relative information such as a camera mounted on the robot. It is supposed that since the agent observes its absolute location in [2] and [3], it is not easy to transfer the knowledge to another task. Here, it is assumed that the agent in the simulations has the sensor as shown in Fig. 3 that has 9x9 visual field whose center is the location of the agent. This means that the visual field moves together with the robot.

3. LEARNING

The learning is very simple. Just a popular three-layer Elman-type recurrent neural network is employed. In an Elman-type recurrent network, the hidden outputs are fed back to the input layer at the next

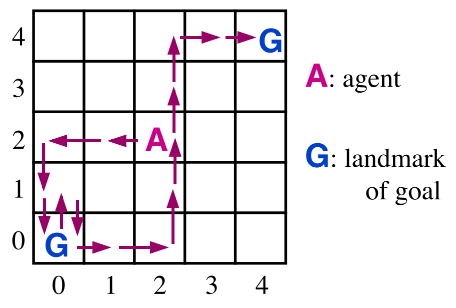

Fig. 2 1-room task and a sample exploration behavior acquired by RL using a recurrent neural network.

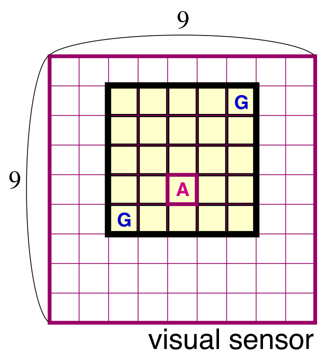

Fig. 3 The visual sensor used in the simulation. Its visual field moves together with the agent.

time step. Sensor signals of the agent are the external inputs of the network. The number of output neurons is the same as the number of the executable primitive actions, and each output is used as Q value for the corresponding action. The training signal $Q_{s,a_{t-1}}$ for the Q value of the previous action a_{t-1} is generated autonomously from the given reward r and the Q value for the present action a_t based on Sarsa algorithm[5] as

$$Q_{s,a_{t-1}} = r_t + \gamma Q_{a_t}(\mathbf{s}_t) \tag{1}$$

where γ is a discount factor. The training signal is given only to the output corresponding to the performed action at previous time step, and the other outputs are not trained. The network is trained by BPTT (Back Propagation Through Time)[9] after the outputs of the network from the input signals at the previous time step are computed again. The learning is episodic, and one trial means one episode. Before each trial, all the hidden outputs are reset to 0.0, and after reaching the goal, all the Q values are supposed to be 0.0. When the agent reaches the goal, it can get a reward $r = 0.8$, otherwise $r = 0.0$. Accordingly, just before reaching the goal, the training signal for the Q value for the performed action is identical to the reward value; otherwise the training signal is identical to the discounted Q value at present. The output function of each hidden and output unit is a symmetrical sigmoid function that ranges from -0.5 to 0.5. In order to adjust the value range to the range of the Q value, 0.4 is added to the actual output before substituting it in Eq. (1), while 0.4 is subtracted from the training signal derived by Eq. (1) before using it as the actual training signal in BPTT. Initial connection weights from the external inputs to the hidden layer are random small numbers from -0.5 to 0.5. The initial connection weights from the hidden layer to the output layer are all 0.0, and as for the feedback connections of the hidden layer, initial weights are 4.0 for the self-feedback and 0.0 for the other feedback. That enables the error signal to propagate to the past states effectively in BPTT without diverging.

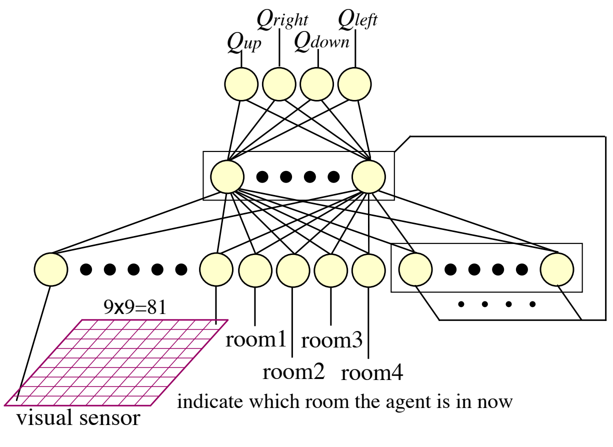

Fig. 4 An Elman network and its input and output signals in the simulation.

4. SIMULATION

4.1 Setups

The agent has a visual sensor that has 9x9=81 sensor cells. Each sensor cell has a receptive filed whose size is just the same as one square in the room, and catches whether a landmark exists on the square or not as a binary value as shown in Fig. 3. As mentioned, so as to perceive the relative information of the agent, the sensor moves together with the agent. It is assumed that the agent cannot see the outside of the room where the agent is, and so the output of every sensor cell whose receptive field covers outside of the room is 0.0. The agent can identify which room it is in now. If the agent is in the room 1, the four signals are 1, 0, 0, 0. In the 1-room environment, all the signals are 0. The number of the total external input signals is 81+4=85 as can be seen in Fig. 4.

The executable primitive actions are “going up”, “right”, “down”, and “left”. The output layer of the network has 4 units, each of which represents the Q value for the corresponding action. The state transition is deterministic. If the agent hits against a wall, it stays at the same square. ε-greedy is employed for the random exploration, and the value of ε is 0.1. The discount factor γ is 0.92. The neural network has three layers and the hidden layer has 30 hidden neurons.

At first, the agent learned deterministic exploration in a 5x5 small grid world where two landmarks of goal exist and one of them is the real goal. The task is called 1-room task here. Fig. 2 shows an example of the robot exploration behavior after the first learning. It can be seen that the robot went to one of the two landmarks at first, and then went to the other after it arrived at the first one and knew that it was not the real goal.

After that, the learning moves to the second stage. The agent is put on a random square in the 4-room world as shown in Fig. 5 that is similar to the 4-room task in [4]. In this world, there are four small rooms

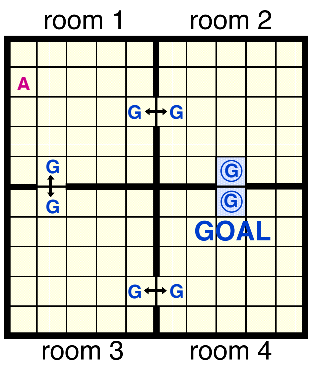

Fig. 5 4-room environment. There are four rooms and each of them has two doorways to the next room. When the agent arrives at the door (G), it is moved immediately to the next room. The configuration of the door changes at every some thousands of trials, but the real goal is always located between the room 2 and the room 3.

whose size is 5x5. The size of each room is identical to the 1-room task. The rooms are allocated as shown in Fig. 5, and are connected to the next room by a doorway. Each room has two doorways, and the doorway location is randomly chosen. A landmark is located just in front of each doorway, but the agent can observe the landmarks only in the room where the agent is now in even though the visual field reaches a landmark in another room. By using the visual sensor consisting of 9x9=81 visual cells, the agent can observe both landmarks wherever the agent is in the room. When the agent arrives at the square where a landmark exists, it is moved immediately to the square located at the opposite side of the doorway in the next room. When the agent arrives at the goal, it gets a reward and the trial finishes. The agent is put randomly on one of the squares in the world again and a new trial begins. The real goal is always located at the wall between the room 2 and room 3. In the 1-room task, the agent is required to keep the context information representing whether the agent already arrived at one of the landmark or not. In the 4-room task, the agent is not required to keep any context information, even though the size of the environment is larger than the 1-room task. This means that the agent can learn a solution using a regular feedforward neural network.

4.2 Results

Fig. 6 shows the learning curves when the door configuration is fixed. The vertical axis indicates the average number of steps to the goal. If the agent does not reach the goal within 2,000 steps, the trial is cut off. One plot is for the case of the agent after the learning of 1-room task and the other is the case when no learning had been done beforehand. For reference, the case that the action selection is always optimal except for the random action that appears with ε=10% probability and the case when the action selection is completely random are also plotted in the same graph. Each plot is the average over every 10 trials until the 100th trial, and the average over every 50 trials after the 100th trial. That is the reason why the value seems fluctuate more until 100th trials. Note that the vertical axis is magnified after 2,000th trial. The data is the average over 5 cases of initial weight set and over 30 door configurations for each initial weight set.

At the beginning of the learning, it can be seen that the agent that had not learned beforehand took around 100steps in average, but the agent after the prior learning took less than 30 steps. Even around the 5000th trial of learning, the performance is still different between them. The result indicates that the first learning accelerates the second learning.

In order to know how the learning is accelerated, the agent's trajectory at the first trial for each case is observed. Fig. 7 shows the first exploration behavior just after the agent finished the first learning and was put onto the 4-room world. The agent went for one of the landmarks at first without doing meaningless actions. After the agent was moved to the room 1, it moves toward the other landmark in the room. It can be said that the agent is able to search the goal so effectively utilizing the knowledge obtained through its past experiences even though some redundant actions were

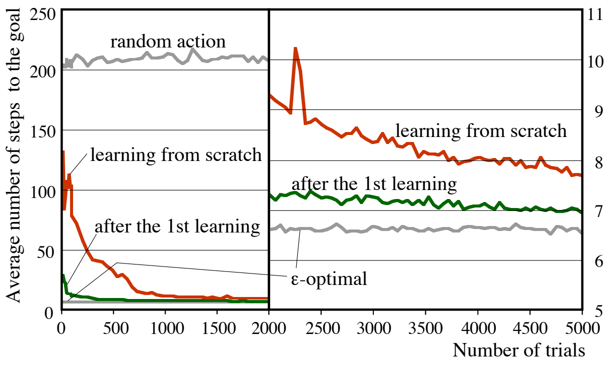

Fig. 6 Comparison of the learning curve with respect to whether the learning of the 1-room task had been done beforehand or not.

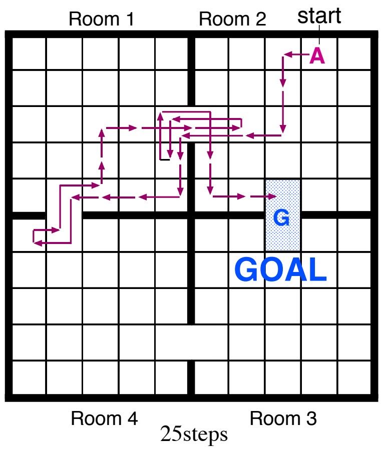

Fig. 7 The exploration behavior just after the agent who has already learned in 1-room environment was put onto the 4-room world.

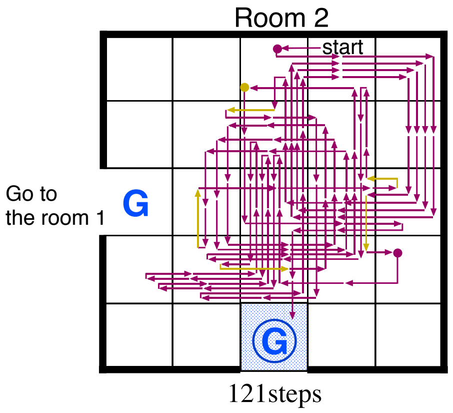

Fig. 8 The exploration behavior just after the agent who has not learned before was put onto the 4-room world. The agent was learning while moving. Light-colored arrows indicate non-greedy actions.

still taken such as the repetition of the move between the room1 and room 2. The result may be considered as a phenomenon within the range only to be expected, but it is important to show that even though the environment is similar to the previous one, obtained knowledge can be effectively utilized in another task without teaching definitely what knowledge is useful.

On the other hand, the exploration behavior of the agent without learning the first task is shown in Fig. 8. In this figure, the agent did not act randomly, but took the same path some times. The reason might be that at the first action in the first trial, all the Q values are 0.4, and as long as the agent did not arrive at the target, the Q value for the action is decreased. It is difficult to say how the Q value changes through learning because a recurrent neural network is used as a function approximator. However, it is supposed that chosen action changes periodically by the modification of the bias value in each output unit, and it is not easy to discriminate the different states for the network at the early stage of learning. Therefore, the agent took the same loop several times, such as square loop at the upper-right area. That can be thought as a kind of the effect of the "optimistic initial value"[1].

In the next simulation, the configuration of the doors changes at every 3,000 trials. The location of a door is chosen randomly at each wall between two adjacent rooms except for the configuration in which two landmarks are put on the same square. The goal location exists always between the room 2 and room 3. Fig. 9 or 10 shows the change of the learning curve according to the number of experiences of different door configuration for each of the cases of prior learning and non-prior learning respectively. Furthermore, each learning curve is average over 5 simulation runs with different initial connection weight set. Note that the scale of the vertical axis is different between Fig. 9 and Fig. 10.

It can be seen that in both cases, learning becomes faster according to the number of experienced configurations. It can be said that the agent obtained a strategy to respond appropriately to various configurations of doors and to go to the goal between the room 2 and 3 through the learning for many configurations. By comparing the two cases, it can be said that the previous learning accelerates such over-configuration learning. However, if the goal location changes as well as the configuration of doors, learning often failed. In this case, the agent sometimes has to move the opposite direction even though the configuration of doors and the agent location are both completely the same as one past state, and the knowledge obtained through the past learning might be broken. If we are in the same situation, we can learn the solution effectively. This can be a future issue.

From these results, the deterministic exploration obtained by the learning in the past experiences relates deeply to the temporal abstraction, and there is a possibility that this approach can be a solution for the discovery problem of temporal abstraction.

Finally, it is examined how much part of the knowledge obtained in the first learning is broken through the second learning, and also how the knowledge obtained in the 4-room task is useful in the 1-room task. In the 4-room task, although the environment is larger than the 1-room task, context information is not necessary to know the optimal path. Then the learning performance when the agent is returned to the 1-room environment is observed. Fig. 11 shows the learning curve for three cases. In one case, the agent has already learned the 1-room environment at first and then the 4-rooms environment. In another case, the agent has already learned only in the 4-room environment. In the other case, the agent learns in the 1-room environment without any previous learning.

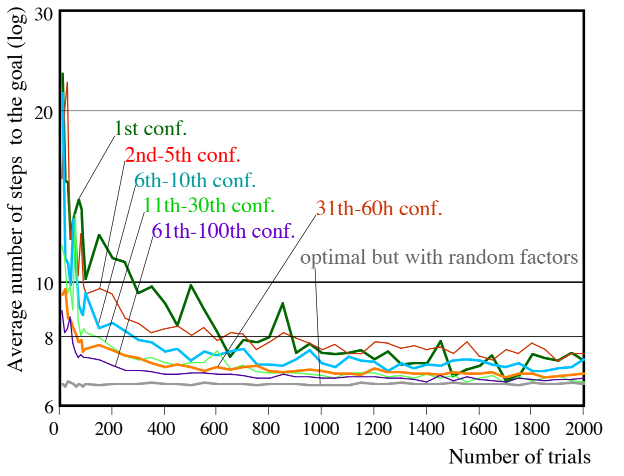

Fig. 9 Comparison of the learning curves with respect to the number of experienced door-configurations when the agent had already learned 1-room environment. As for the label of each line, "2nd – 5th conf.", for example, means that the average learning curve from the second configuration to the fifth configuration.

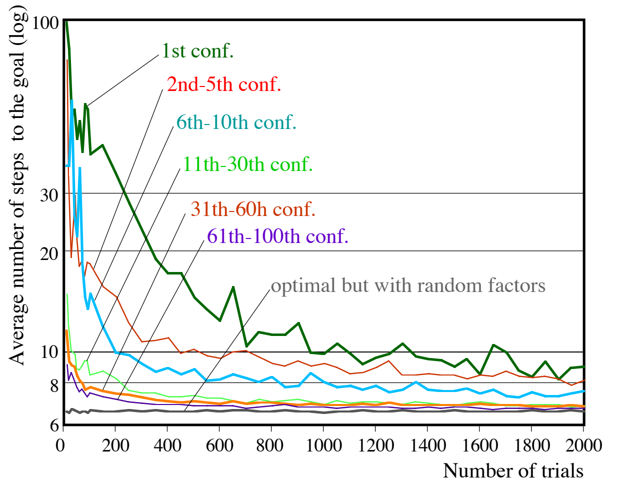

Fig. 10 Comparison of the learning curves with respect to the number of experienced door-configurations when the agent had not learned anything before.

At the early phase of the learning, the agent after the learning in the both environments can reach the real goal in the smallest number of steps in average, and even for the agent after only the learning in 4-room environment, the average steps to the real goal is smaller than the agent after no learning. Two possible reasons can be thought. One reason is that the agent learned to go to one of the landmarks and that helps the learning faster. The other reason is that the agent learned to go to the other landmark after it reaches one of the landmarks even though the 4-room task itself does not required such context-based behavior.

In order to know the reason, the agent behavior is just observed on a configuration of the landmarks that did not appear in the 4-room environment without learning in this environment. As shown in Fig. 12, it is surprising that the agent went to one of the landmarks and after reaching it, the agent went to the other landmark. The same behavior can be observed in another simulation run with a different initial weight set. Although more investigation is necessary, the author think that it is more likely that such behavior is obtained through the previous learning, because the task itself does not require context-based actions, but the context-based action realizes effective exploration.

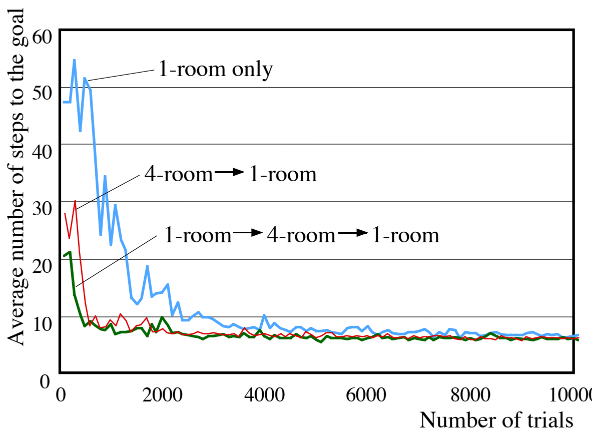

Fig. 11 Learning curve when the agent is returned to the 1-room environment after the learning in the 4-room env. and the comparison to the references.

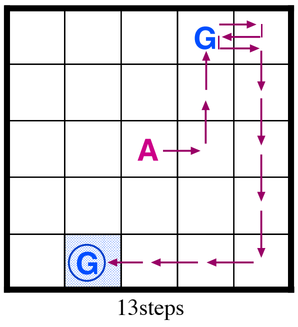

Fig. 12 An example of the agent behavior after only the learning in 4-room environment where context-based actions are not required. The agent did not learn, and just the behavior is observed.

5. CONCLUSION

A novel idea that deterministic exploration behavior obtained through learning can be considered as temporal abstract actions was introduced. It was actually shown in some simulations that the deterministic exploration behavior obtained through the learning of a task accelerates the learning of another similar task. A recurrent neural network was used for the learning, but the knowledge obtained in the first task was not destroyed completely but was used effectively in the second task even though the agent often failed a more difficult task. When the agent was returned to the first task, the learning was faster than the learning from scratch. Furthermore, context-based exploration behavior was acquired through the learning of a task that did not require such behavior. The detailed investigation should be done in the future.

ACKNOWLEDGEMENT

This research was supported by JSPS Grant in Aid in Scientific Research (B) #14350227and #15300064, and also MEXT Overseas Advanced Education and Research Application Assistance Program. I would like to say thank to Prof. R. S. Sutton at Univ. of Alberta, Canada for useful comments.

REFERENCES

[1] Dietterich, T. G., "Hierarchical reinforcement learning with the MAXQ value function decomposition", Proc. of ICML'98, 1998.

[2] McGovern, A., and Barto, A.G, "Automatic Discovery of Subgoals in Reinforcement Learning Using Diverse Density". *Proc. of the ICML'01*, pp. 361-368, 2001.

[3] Stolle, M., "Automated discovery of options in reinforcement learning", *Master's thesis, McGill University*, 2004.

[4] Sutton, R.S., Precup, D., Singh, S., "Between MDPs and semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning", *Artificial Intelligence,* Vol. 112, pp.181-211, 1999

[5] Sutton, R.S., Barto, A.G., "Reinforcement Learning: An Introduction", MIT Press, Cambridge, MA, 1998

[6] Shibata, K., "Acquisition of Deterministic Exploration Behavior by Reinforcement Learning", *Proc. of the 11th AROB* (CD-ROM), 2006

[7] Shibata, K., "Learning of Exploration Behavior by Reinforcement Learning", *Proc. of SICE SSI 2005*, pp. 11-16, 2005 (in Japanese)

[8] Shibata, K., "Spatial Abstraction and Knowledge Transfer in Reinforcement Learning Using a Multi-Layer Neural Network", *Proc. of ICDL5* (CD-ROM), 2006

[9] Rumelhart, D.E, Hinton, G.E., and Williams, R.J., "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing*, The MIT Press, pp. 318-362, 1986