Hierarchical Reinforcement Learning Introducing Genetic Algorithm for POMDPs Environments

Kohei Suzuki[1] and Shohei Kato[1,2]
[1]Dept. of Computer Science and Engineering Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan
[2]Frontier Research Institute for Information Science, Nagoya Institute of Technology, Gokiso-cho, Showa-ku, Nagoya 466-8555, Japan

Keywords: Reinforcement Learning, Genetic Algorithm, Perceptual Aliasing.

Abstract: Perceptual aliasing is one of the major problems in applying reinforcement learning to the real world. Perceptual aliasing occurs in the POMDPs environment, where agents cannot observe states correctly, which makes reinforcement learning unsuccessful. HQ-learning is cited as a solution to perceptual aliasing. HQ-learning solves perceptual aliasing by using subgoals and subagent. However, subagents learn independently and have to relearn each time when subgoals change. In addition, the number of subgoals is fixed, and the number of episodes in reinforcement learning increases unless the number of subgoals is appropriate. In this paper, we propose the reinforcement learning method that generates subgoals using genetic algorithm. We also report the effectiveness of our method by some experiments with partially observable mazes.

1 INTRODUCTION

There is a major problem in practical application of reinforcement learning in the robotics field. Many studies of reinforcement learning use Markov decision processes (MDPs), where agents observe states correctly. However, it is not always possible for agents to observe states accurately in real environments. For example, in case of autonomous mobile robots, perceptual aliasing (Whitehead and Ballard, 1991) arises from various factors such as sensor failure. Perceptual aliasing means that different states are recognized as the same state. Because of this, agents cannot learn the route correctly. This environment is called the partially observable Markov decision processes (POMDPs) (Whitehead and Ballard, 1990). A learning method under the POMDPs environment is necessary for practical use of reinforcement learning in the real world.

There are many types of solutions to perceptual aliasing: the methods using subgoals (Wiering and Schmidhuber, 1997; Nomura and Kato, 2015) (Suzuki and Kato, 2017), the methods using profit sharing (PS) (Miyazaki and Kobayashi, 2003; Arai and Sycara, 2001; Uemura et al., 2005), the methods using recurrent neural networks (Mnih et al., 2016; Sridharan et al., 2010; Sallab et al., 2017), the methods using bayesian (Ross et al., 2008; Poupart et al., 2006; Thomson and Young, 2010) and the methods using state transition history (Chrisman, 1992; McCallum, 1993; McCallum, 1995). In this paper, we focus on the methods using subgoals because these methods are able to adapt to complex tasks. HQ-learning (Wiering and Schmidhuber, 1997) and First Visit Profit Sharing (FVPS) (Arai and Sycara, 2001) are cited as solutions to perceptual aliasing. HQ-learning solves perceptual aliasing by using subgoals and subagents. The subgoals are determined by HQ-values, which are values of subgoals. However, subagents learn independently and have to relearn each time when subgoals change. In addition, the number of subgoals is fixed, and the number of episodes increases unless the number of subgoals is appropriate. Therefore, learning efficiency is poor. FVPS is a method that improves Profit Sharing (PS), and updates state-action pairs of each rule only once per one episode. However, it is difficult for FVPS to set the initial value of the state-action pairs which is suitable for the environment, moreover, FVPS accumulates value. Therefore, there is a high possibility of falling into local solutions.

In this paper, we propose the reinforcement learning method that generates subgoals using genetic algorithm (GA). We explain perceptual aliasing in Section 2, propose Subgoal Evolution-based Reinforcement Learning (SERL) in Section 3, and conduct experiments under POMDPs environment in Section 4. We propose Hybrid learning using PS and GA

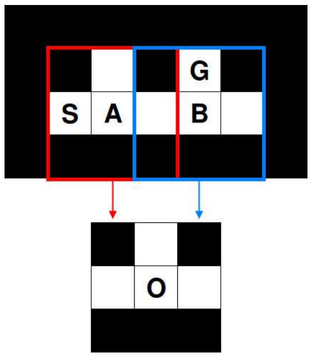


Figure 1: A POMDP environment.

(HPG) that improves SERL and report the effectiveness of our method by some experiments with partially observable mazes in Section 5 and Section 6.

2 PERCEPTUAL ALIASING

In this paper, we consider the grid world as shown in Figure 1. The agent can perceive only eight neighbor cells and performs self-localization by those cells. There are four types of action: “up” “ down” “left” and “right”. In Figure 1, the agent must go through the states “A” and “B” to reach the goal state “G” from the start state “S”. However, the agent is unable to distinguish between state “A” and “B” because these states have the same eight neighbor cells. This problem is called perceptual aliasing. This makes the agent select “up” in state “A” because it selects “up” in state “B” which is close to the goal. Therefore, the agent falls into a loop in state “A”. In this paper, we solve this problem.

3 SUBGOAL EVOLUTION-BASED REINFORCEMENT LEARNING (SERL)

SERL algorithm generates agents that solve perceptual aliasing using GA. Each agent has subagents and subgoals represented in the binary string. Subgoals divide a POMDPs environment into MDPs environments. Subagents perform reinforcement learning under the MDPs environments. In accordance with the result of the reinforcement learning, genetic operation is performed, and subgoals are generated. The overview of the proposed method is shown in Figure 2. The procedure of SERL is shown below.

1. Initial population generation
   Generate Y agents. Each agent has $(X+1)$ subagents, which have a subgoal and Q-table. Subgoals are generated randomly.
2. Reinforcement learning
   Subagents performs reinforcement learning using Q-learning(Watkins and Dayan, 1992) in each agent. Reinforcement learning is performed in order of subagents. The order shifts to the next subagent when the subagent has reach the subgoal.
3. Genetic operation
   Subgoals are inherited by crossover and are generated by mutation.
4. Repeat the procedure 2 and 3 for the number of G generations.

3.1 Initial Population Generation

Agents that have subagents and subgoals are generated in initial population generation. Subgoals are represented in the binary string as shown in Figure 3, where “1” and “0” means a wall and a road respectively, because the observation range of the agent is eight neighbor cells in this paper. In SERL, each agent randomly generates subgoals because the agents do not know what kinds of state exist in unknown environment.

3.2 Reinforcement Learning

In reinforcement learning, subagents perform reinforcement learning using Q-learning. SERL sets the maximum number of steps in an episode and finishes reinforcement learning if the number of steps in an episode reaches the maximum number. This is because there is a possibility that the subagent does not reach the subgoal which agents generated randomly.

3.3 Genetic Operation

3.3.1 Fitness Calculation

After reinforcement learning, agents perform one episode using greedy selection in order to remove the randomness of action selection. Fitness is calculated using the result of this one episode. The fitness function is shown below.

$$F1 = \begin{cases} Re + \dfrac{Max\_step - step}{sub \times a} & \text{(reach the goal)} \\ \dfrac{goal}{b} & \text{(not reach the goal)} \end{cases}, \quad (1)$$

where, Re is the goal reward value; Max_step is the maximum step number, $step$ is the number of

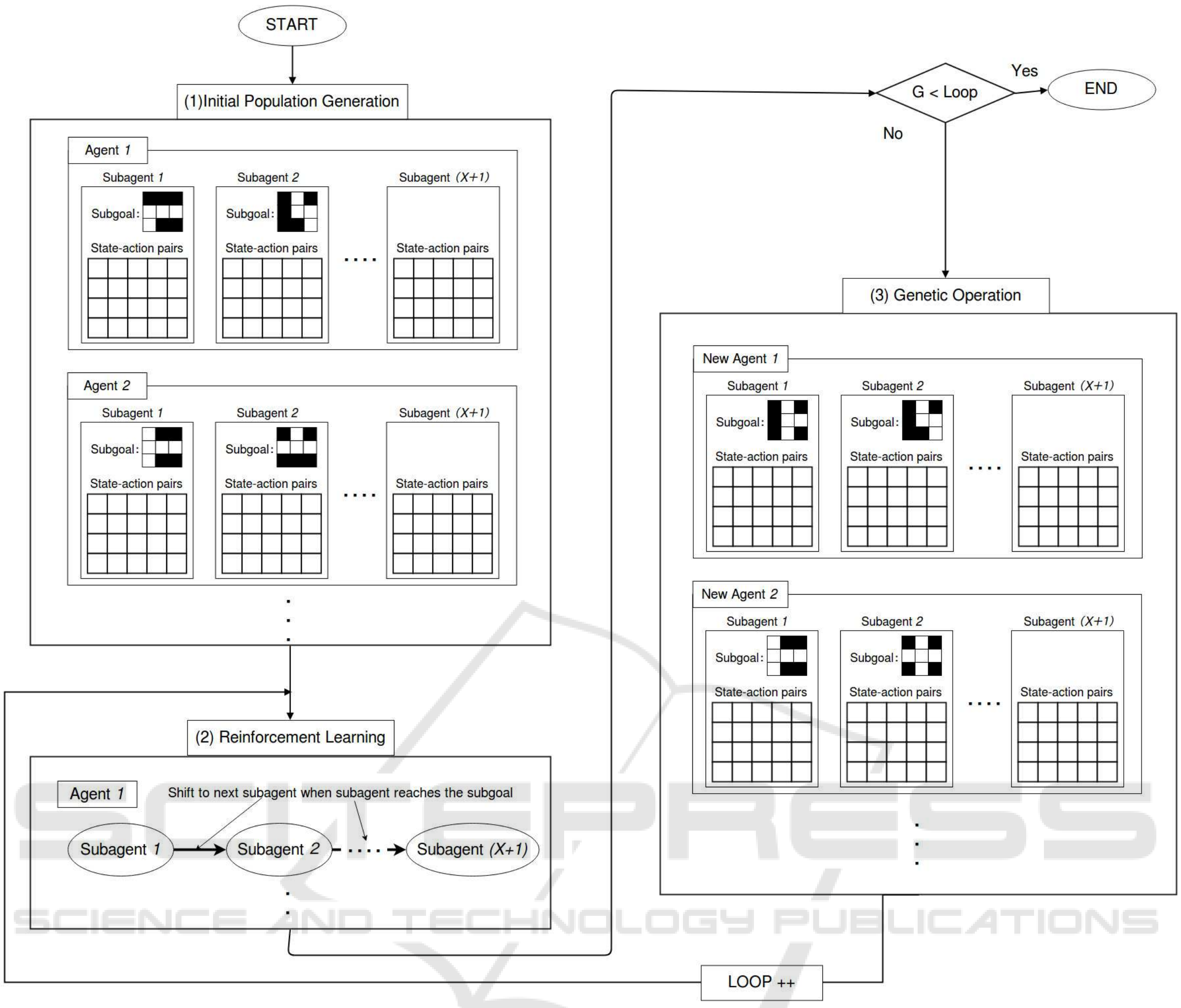


Figure 2: Overview of the proposed method.

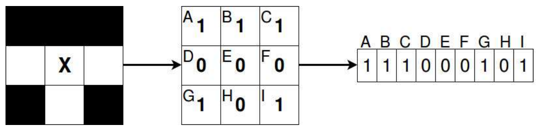


Figure 3: Subgoal expression of state X.

steps taken to obtain rewards, *sub* is the number of subagents, *goal* is number of times the agent reached the goal during reinforcement learning, and *a* and *b* are hyperparameter. The *b* is set to a value that $\frac{goal}{b}$ does not exceed the goal reward *Re*. The *goal* is used for fitness calculation when the agent does not reach the goal in one episode using the greedy selection. This is because, the agent is likely to have effective subgoals if the agent reaches the goal during reinforcement learning.

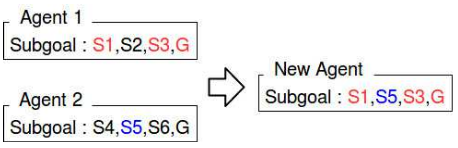
3.3.2 Crossover

SERL performs the following two kinds of crossover.

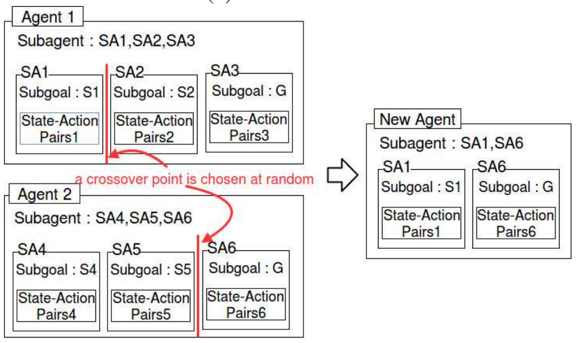
Crossover 1 performs uniform crossover of subgoals as shown in Figure 4(a). Subgoals are inherited and the state-action pairs are initialized. This is expected to avoid local solutions.

Crossover 2 performs single-point crossover of subagents as shown in Figure 4(b). Subgoals and the state-action pairs are inherited. Learning efficiency increases by inheriting the state-action pairs. In addition, the crossover points of two parents are not common points, but are random points in each agent, which makes the number of subgoals dynamically changes and is suitable for the environment.

In general, crossover generates two offspring from two parents. However, SERL just selects one of two offspring.



(a) Crossover 1



(b) Crossover 2

Figure 4: Crossover.

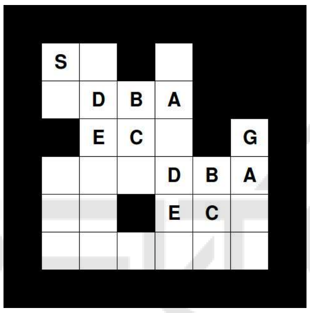


Figure 5: Maze of Yamamura(Yamamura et al., 1995).

3.3.3 Mutation

There is a possibility that effective subgoals are not generated in initial population generation. In this case, mutation is important so as to avoid local solutions. In mutation, SERL inverts a part of the binary string randomly, which expresses the subgoal. In addition, SERL changes a part of the binary string to "don't care". "Don't care" means either "0" or "1", that is, agent does not care whether corresponding cell is road or wall. This makes subgoals more diverse.

4 COMPARETIVE EXPERIMENTS UNDER THE POMDPS

We performed comparative experiments using maze of Yamamura (Yamamura et al., 1995) shown in Figure 5. Three methods are used: SERL, HQ-learning, and FVPS. As explained in Section 2, the observation range of agents is 8 neighbor cells, and the actions are of four types "up", "down", "left" and "right". Perceptual aliasing occurs in the states, "A", "B", "C", "D" and "E". The parameters for SERL is shown in Table 1. The number of episodes of each method is 6,000. The maximum number of steps in an episode is 150. The number of initial subgoals of SERL and the number of subgoals of HQ-learning are 2. The experiments were performed in 100 runs for each method, then the average value of results was calculated.

Table 1: Parameters.

Number of generations	10
Number of agents	20
Number of reinforcement learning episodes	30
Elite preservation (%)	20
Crossover 1 (%)	12.5
Crossover 2 (%)	87.5
Mutation rate (%)	5

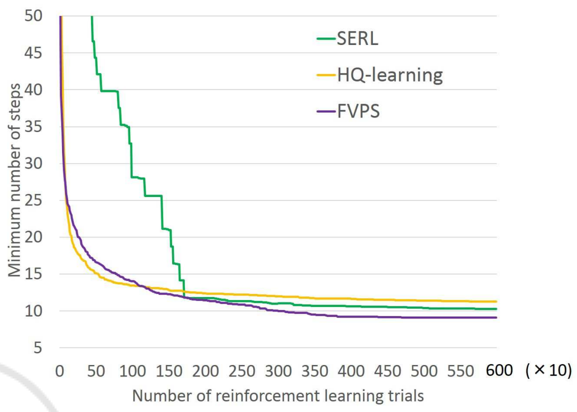


Figure 6: The result in maze of Yamamura.

The experimental results are shown in Figure 6. The vertical axis of the graph shows the minimum number of steps to reach the goal, and the horizontal axis shows the number of reinforcement learning episodes. SERL learned slower than the other two methods. This is because subgoals were generated randomly at the initial population generation, which did not make many subgoals suitable for the maze. In addition, SERL did not converge to the optimal solution. SERL took time to find appropriate subgoals because a new subgoal, which the initial individuals do not have, is generated only by mutation. Therefore, SERL should generate effective subgoals at the initial population generation.

5 HYBRID LEARNING PROFIT SHARING AND GENETIC ALGORITHM (HPG)

We propose HPG algorithm that agents select subgoals among subgoal candidates at initial population generation. HPG judges aliased states as subgoal candidates using PS. The flow of HPG is the same as SERL. HPG improves initial population generation and reinforcement learning in SERL.

5.1 Initial Population Generation

The procedure of initial generation population is shown below.

1. An agent performs reinforcement learning using PS.
2. Subgoal candidates are determined based on state-action pairs in each state.
3. Subgoals of each agent are determined randomly among the subgoal candidates.

HPG makes subgoal candidates using PS, which is a kind of reinforcement learning.

PS (Miyazaki et al., 1994)(Grefenstette, 1988) is an offline learning method which collectively updates state-action pairs after earning rewards, and actions are often determined by roulette selection. The updating formula for state-action pairs $P(s_t, a_t)$ of action a_t in the state s_t is shown below.

$$P(s_t, a_t) \leftarrow P(s_t, a_t) + f(x), \tag{2}$$

where, $f(x)$ is a reinforcement function and x is the number of steps to earn rewards. The reinforcement function often uses a geometric decreasing function. However, HPG changes the way of updating state-action pairs to judge aliased states.

HPG equally updates the state-action pairs of rules selected in the same state. Because of that, each rule is updated only once in one episode, and the reinforcement function does not depend on the x. It is expressed by the following formula.

$$f(R, W) = \frac{R}{W}, \tag{3}$$

where, R is a reward and W is length of the episode. We propose two methods to generate initial population using this PS: HPG using action selection probability and HPG using entropy of action selection.

5.1.1 HPG Using Action Selection Probability (HPG-a)

HPG-a determines subgoal candidates using action selection probabilities. PS updates the rules equally in each state. The value of state-action pairs are the same when there are multiple essential rules to earn the reward in a certain state. The value of state-action pairs are equal if all actions have to be selected to earn the reward. Then, the action selection probabilities Pr are the value of the following formula.

$$Pr = \frac{1}{|Action|}, \tag{4}$$

There is a possibility that effective subgoals are not generated in initial population generation. In this case, mutation is important so as to avoid local solutions. In mutation, SERL inverts a part of the binary string randomly, which expresses the subgoal. In addition, SERL changes a part of the binary string to “don’t care”. “Don’t care” means either “0” or “1”, that is, agent does not care whether corresponding cell is road or wall. This makes subgoals more diverse. where, $Action$ is set of selectable actions in a state. The action selection probability of the necessary rules for earning rewards never falls below the value of (4) because the state-action pairs of the necessary rules are definitely updated every time when the agent earns a reward. In other words, the agent has to select different actions because there is a high possibility that perceptual aliasing occurs when the action selection probabilities of multiple rules are over the value of (4) in one state. Therefore, the states where action selection probabilities of multiple rules exceed the value of (4) are set as subgoal candidates.

5.1.2 HPG Using Entropy of Action Selection (HPG-e)

HPG-e determines the subgoal candidates using entropy of action selection and calculates the entropy $En(s)$ of action selection in each state s by the following formula.

$$En(s) = -\sum_{a \in Act(s)} P(a) \log_2 P(a), \tag{5}$$

where, Act is set of selectable actions, $P(a)$ is the action selection probability of action a. The value of (5) decreases when the action selection probability of one action increases. In contrast, the value of (5) increases when the action selection probability of all actions are equal. In other words, entropy of action selection expresses uncertainty of policy and increases under POMDPs.

When the agent has to select two action (a_1, a_2) on an aliased state s^*, these action selection probabilities approach $\frac{1}{2}$ and the others approach 0. Then, the entropy of action selection is the value of following formula.

$$\begin{aligned} En(s^*) &= -\sum_{a\in Act(s^*)} P(a)\log_2 P(a) \\ &= -\sum_{a\in\{a_1,a_2\}} P(a)\log_2 P(a) \\ &= -(\frac{1}{2}\log_2\frac{1}{2}+\frac{1}{2}\log_2\frac{1}{2}) \\ &= 1 \qquad (6) \end{aligned}$$

The entropy of action selection does not exceed (6) when only one action selection probability in a state is large. Whereas, the entropy exceeds (6) when the agent has to select at least two actions to reach the goal. It is highly likely to be aliased state when the state has the entropy exceeding (6). Therefore the state is set as a subgoal candidate in HPG-e. Then, each agent randomly determines subgoals among the subgoal candidates.

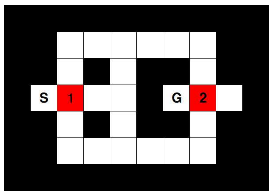
5.2 Reinforcement Learning

In SERL, subagents perform reinforcement learning using Q-learning (Watkins and Dayan, 1992). However, In HPG, subagents perform PS as described in Section 5.1. Uemura et al. (Uemura et al., 2005) showed that it is not inferior to random selection if the agent can select all the necessary rules for rewards with the same probability in the state, where the agent has to select multiple rules to reach the goal. HPG satisfies this condition and is able to solve perceptual aliasing because HPG gives rewards equally to rules in the same state. Therefore, it is possible to reach the goal even when subgoals do not divide the POMDPs environment well.

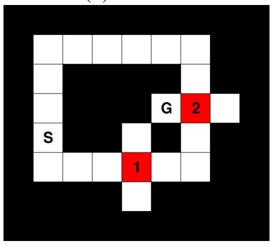
HPG involves hybrid learning using PS and GA. By combining the PS and GA, HPG compensates for their disadvantages: local optima and learning efficiency.

5.3 Peformance of Judgement of Aliased States

We compared the two methods (HPG-a and HPG-e) using the maze shown in Figure 7 and consider their respective performances of judgement of aliased states. Experimental setup is the same as in Section 4. Perceptual aliasing occurs in red cells in each maze. The maze in Figure 7(a) is an environment that the different actions must be selected in each red cell. In state “2”, the agent has to select “left”. However, In state “1”, it has to select an action except “left”. The maze in Figure 7(b) is an environment where the agent is able to go the route which the agent avoids perceptual aliasing. The number of PS episodes is set to



(a) Maze 1



(b) Maze 2

Figure 7: Mazes.

Table 2: Result of Maze 1.

(a) The number of times red cells is determined to be subgoal candidates (the sum of 100 runs)

Method \ Reward	50	100	300
HPG-a	100	100	97
HPG-e	100	100	100

(b) The sum of subgoal candidates (the average of 100 runs)

Method \ Reward	50	100	300
HPG-a	1.16	1.06	1.06
HPG-e	6.80	5.42	4.55

1000, and the parameters for PS are unified in both methods. The all initial values of the state-action pairs are set to 10 and the goal reward was set to any of 50, 100 and 300.

The results of Maze 1 are shown in Table 2. Table 2(a) shows the number of times red cells are judged to be aliased state in 100 runs. HPG-a and HPG-e were able to judge red cells as aliased state almost correctly. However, HPG-a failed to judge aliased state when reward was 300. In state “2”, “left” is selected and, in state “1”, any action of three actions (“up” “down” and “right”) is selected to reach the goal. The selection probability of “left” in red cells increased, whereas the selection probabilities of the other three actions were dispersed and did not exceed (4). Therefore, HPG-a could not judge aliased states correctly. Table 2(b) shows the average of the total number of subgoal candidates in 100 runs. In HPG-a, the number of subgoal candidates was close to 1 which is an ideal value. However, the number of subgoal candidates was large in HPG-e. This is because the difference in state-action pairs was small while reinforce-

Table 3: Result of Maze 2.

(a) The number of times red cells is determined to be subgoal candidates (the sum of 100 runs)

Method \ Reward	50	100	300
HPG-a	65	72	76
HPG-e	100	100	100

(b) The sum of subgoal candidates (the average of 100 runs)

Method \ Reward	50	100	300
HPG-a	1.31	1.19	1.18
HPG-e	8.67	6.77	5.31

ment learning was in progress, which made the entropy of action selection probabilities large. As value of reward increased, the reinforcement learning was completed early and the number of subgoal candidates was small.

The results of Maze 2 are shown in Table 3. HPG-a had poor precision for judgement of aliased state compared to result of Maze 1. There are two routes to the goal in Maze 2. The state-action pairs of ”left” became high in the red cell if the agent did not go to the route which through the state ”1”. In that case, there were no more rules that exceed (4). As the reward increased, the difference in state-action pairs between the two routes became larger and the precision increased. Whereas, although HPG-e had many subgoal candidates, red cells could be accurately judged as aliased states. This is because HPG-e judged the state where action selections were dispersed by using entropy. In such a state, there is a high possibility that there are multiple effective rules. Therefore, HPG could correctly judge aliased states.

HPG-a is able to correctly determine the aliased states as long as PS does not fall into a local solution at initial population generation. Although HPG-e has a loose determination criterion for aliased state and increases the number of subgoal candidates, HPG-e does not miss the aliased states. Therefore, HPG-a is more suitable for simple mazes and HPG-e is more suitable for calculating optimal solution.

5.4 Effectiveness of Initial Population Generation of HPG under the POMDPs

We report the effectiveness of HPG-a and HPG-e using the maze used in Section 4. Experimental setup is also the same as in Section 5.3. The number of episodes of PS at initial population generation is 100, the number of generations is 10, the number of agents

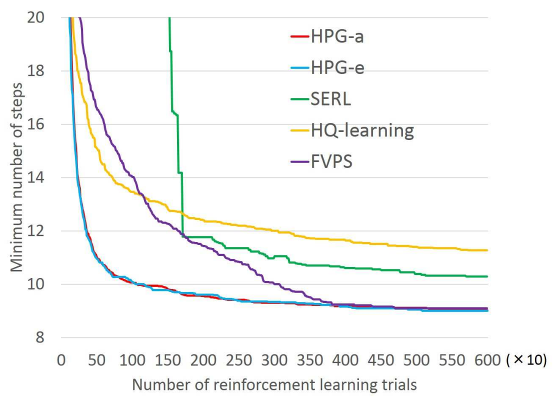


Figure 8: The result in maze of Yamamura.

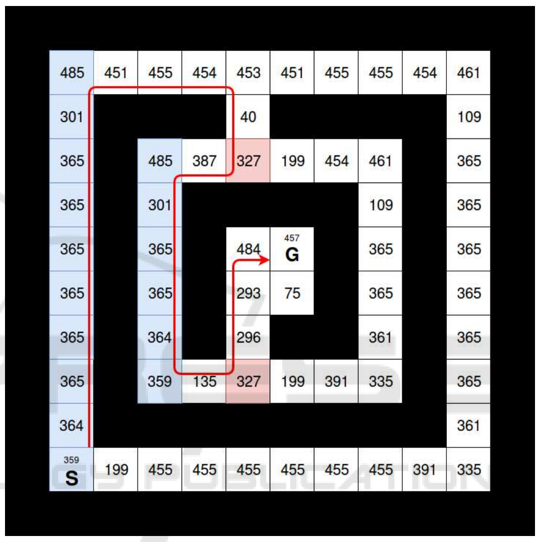


Figure 9: The maze of Wiering (Wiering and Schmidhuber, 1997).

is 20, the number of reinforcement learning episodes in each agent is 30, and the number of initial subgoals is 2.

The result is shown in Figure 8. HPG-a and HPG-e learned faster than FVPS and obtained good results. FVPS took time to learn because initial value of state-action pairs was small. HQ-learning did not find appropriate subgoals. This is because HQ-learning has 256 selectable subgoals, which is difficult to find appropriate subgoals. Whereas, HPG-a and HPG-e generated effective subgoals by PS at initial population generation, and the subgoals made the maze easier.

Table 4: Parameters.

(a) HPG-a and HPG-e

Number of PS episodes at initial population	15000
Number of generations	49
Number of agents	50
Number of reinforcement learning episodes	300
Elite preservation (%)	20
Crossover 1 (%)	12.5
Crossover 2 (%)	87.5
Mutation rate (%)	5
R (fitness calculation)	100
a (fitness calculation)	0.2
b (fitness calculation)	3

(b) SERL

Number of generations	50
Number of agents	50
Number of reinforcement learning episodes	300
Elite preservation (%)	30
Mutation rate (%)	5

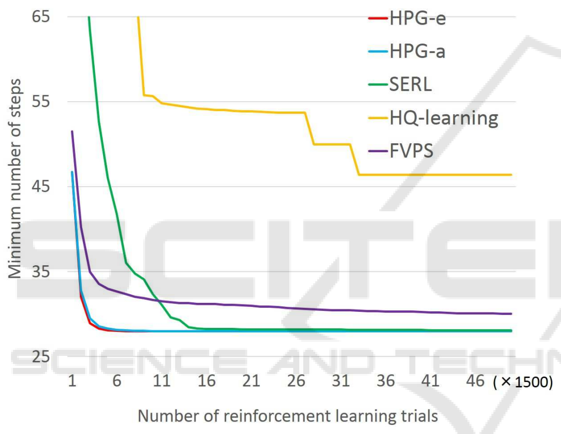


Figure 10: The result in a POMDP environment.

6 PERFORMANCE EXPERIMENTS FOR A POMDPS ENVIRONMENT

We performed experiments under the POMDPs environment using the maze of Wiering(Wiering and Schmidhuber, 1997) shown in Figure 9. The numbers in the cells represent observation information. These number is obtained by converting 9-digit binary number, which expresses subgoal as shown in Figure 3, into decimal number. The agent has to solve the perceptual aliasing to reach the goal in this environment. Perceptual aliasing occurs in the red cells and the blue cells on the shortest path indicated by the red arrow. We performed comparative experiments with five methods: HPG-a, HPG-e, SERL, HQ-learning and FVPS. The parameters for each method are shown in Table 4. The number of trials of each

Table 5: Minimum number of steps to reach the goal (the average of 100 runs).

	HPG-a	HPG-e	SERL	HQ-learning	FVPS
Minimum steps	28.00	28.00	28.10	46.04	30.06

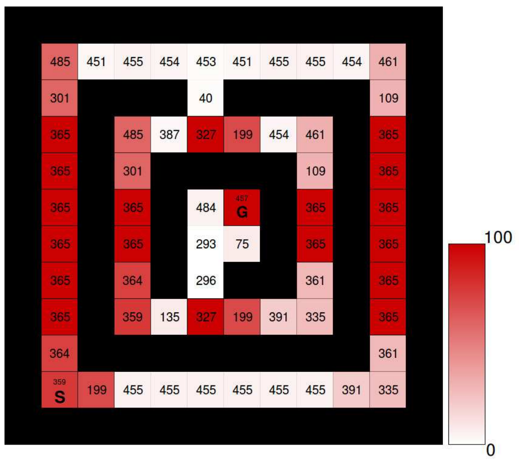


Figure 11: The distribution of subgoal candidates (HPG-a).

method is unified to 750,000 episodes. The shortest step in this maze is 28, the maximum number of steps in an episode is 150, and the number of initial subgoals is 3. The experiment was performed in 100 runs for each method and the average value of results was calculated.

The experimental results are shown in Figure 10 and Table 5. HPG-a and HPG-e converged to the shortest step number. HPG-e obtained the optimal solution in the sixth generation and HPG-a did in the 14th generation. HPG-e completed learning in less than a half of the trials of HPG-a. Whereas, FVPS fell into a local solution and SERL could not find appropriate subgoals and did not obtain an optimal solution. HQ-learning had to relearn the policy each time when the subgoal changed and failed to obtain the optimal solution. It was confirmed that the effectiveness of subgoal generation by PS, and HPG-e is more suitable than HPG-a in complicated environments where there are multiple routes to the goal.

Figure 11 and Figure 12 show the distribution of subgoal candidates during 100 runs. The deeper the red, the greater the number of times the state is judged to be a subgoal candidate. Both methods accurately judged red and blue cells as aliased states in most of the runs. However, in HPG-a, “301” and “485” were judged as subgoal candidates as not many as in HPG-e. The agent has to solve perceptual aliasing occurring in these states to reach the goal in the shortest step. Whereas, HPG-e regarded these states as subgoal candidates in more than 90% of experiments. Because of this, HPG-e got the optimal solution earlier than HPG-a.

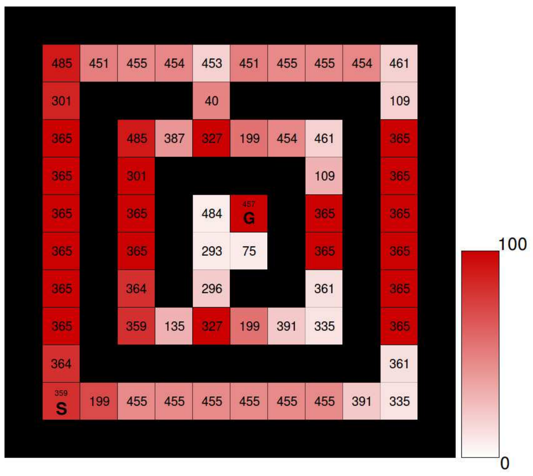


Figure 12: The distribution of subgoal candidates (HPG-e).

7 RELATED WORKS

7.1 HQ-learning

HQ-learning (Wiering and Schmidhuber, 1997) is a hierarchical extension of Q-learning and divides task into subtasks using subgoals. The agent has some subagents which have Q-table independently. The subagents perform Q(λ)-learning (Peng and Williams, 1994; Wiering and Schmidhuber, 1998) in order from first subagent and shift the next subagent when the subagent reaches the subgoal. Action are selected according to the Max-Bolzmann exploration (Kaelbling et al., 1996).

HQ-learning also learns appropriate subgoals. The subgoals are determined by HQ-values which are values of subgoals. HQ-value is updated when an agent accomplish the task. The updating formula for HQ-value of $subagent_i$ is shown below.

$$\begin{aligned} R_i &= \sum_{t=t_i}^{t_{i+1}-1} \gamma^{t-t_i} R(s_t, a_t), \\ HQ_i'(\hat{o}_i) &\leftarrow R_i + \gamma^{t_{i+1}-t_i}[(1-\lambda)\max_{o' \in O} HQ_{i+1}(o') \\ &\qquad + \lambda HQ_{i+1}'(\hat{o_{i+1}})], \\ HQ_i(\hat{o}_i) &\leftarrow (1-\alpha)HQ_i(\hat{o}_i) + \alpha HQ_i'(\hat{o}_i), \end{aligned} \quad (7)$$

where, γ ($0 \leq \gamma \leq 1$) is the discount-rate, R is the reward, $HQ_i(o)$ is HQ-value of state o, α is learning rate, and λ ($0 \leq \lambda \leq 1$) is a constant. Subgoals are selected using ε-greedy exploration (Sutton, 1996).

In HQ-learning, subagents have to relearn each time when subgoals change. In addition, the number of subgoals is fixed. Therefore, learning efficiency is poor. In HPG, State-action pairs are inherited by crossover and the crossover points are random points in each agent. Because of this, HPG obtained solutions more quickly than HQ-learning as shown in Figure 8 and Figure 10.

7.2 First Visit Profit Sharing

FVPS (Arai and Sycara, 2001) is an improved method of profit sharing. FVPS changes reinforcement function and updates state-action pairs of each rule equally. The reinforcement function $f(x)$ is shown below.

$$f_x = \begin{cases} \alpha_{o_x} & (First\, time\, to\, update\, rule\, \lambda) \\ 0 & (Otherwise), \end{cases} \quad (8)$$

where, α_{o_x} must be a constant value in state o_x.

In FVPS, it is difficult to set the initial value of the state-action pairs, which is suitable for the environment, moreover, FVPS accumulates the values of state-action pairs. Therefore, FVPS often falls into local solutions. In HPG, state-action pairs are initialized at crossover, which made HPG obtain optimal solution as shown in Figure 10 and Table 5.

8 CONCLUSION

In this paper, we proposed the reinforcement learning method that generates subgoals using GA under the POMDPs. SERL takes time to generate appropriate subgoals. Therefore, we proposed HPG that generates subgoal candidates using PS. It was confirmed that HPG-a using action selection probability is suitable for simple mazes and HPG-e using entropy of action selection is suitable for complicated mazes by some experiments. However, HPG has a problem which is appropriate subgoals are not found if PS cannot achieve the task at initial population generation. To solve the problem, we consider introducing swarm intelligence (Beni and Wang, 1993) to HPG. HPG introduces swarm reinforcement learning (Iima and Kuroe, 2008) and shares state-action pairs, and deals with complex problems.

In the future, we will improve the performance of judgment of aliased states. We will also conduct experiments in the real world using mobile robots.

REFERENCES

Arai, S. and Sycara, K. (2001). Credit assignment method for learning effective stochastic policies in uncertain domains. In *Proceedings of the 3rd Annual Conference on Genetic and Evolutionary Computation*, pages 815–822. Morgan Kaufmann Publishers Inc.

Beni, G. and Wang, J. (1993). Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712. Springer.

Chrisman, L. (1992). Reinforcement learning with perceptual aliasing: The perceptual distinctions approach. In *AAAI*, pages 183–188.

Grefenstette, J. J. (1988). Credit assignment in rule discovery systems based on genetic algorithms. *Machine Learning*, 3(2-3):225–245.

Iima, H. and Kuroe, Y. (2008). Swarm reinforcement learning algorithms based on sarsa method. In *SICE Annual Conference, 2008*, pages 2045–2049. IEEE.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. (1996). Reinforcement learning: A survey. *Journal of artificial intelligence research*, 4:237–285.

McCallum, R. A. (1993). Overcoming incomplete perception with utile distinction memory. In *Proceedings of the Tenth International Conference on Machine Learning*, pages 190–196.

McCallum, R. A. (1995). Instance-based utile distinctions for reinforcement learning with hidden state. In *ICML*, pages 387–395.

Miyazaki, K. and Kobayashi, S. (2003). An extention of profit sharing to partially observable markov decision processes : Proposition of ps-r* and its evaluation. *Journal of Japanese Society for Artificial Intelligence*, 18(5):286–296.

Miyazaki, K., Yamamura, M., and Kobayashi, S. (1994). A theory of profit sharing in reinforcement learning. *Journal of Japanese Society for Artificial Intelligence*, 9(4):580–587. (in Japanese).

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., Silver, D., and Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937.

Nomura, T. and Kato, S. (2015). Dynamic subgoal generation using evolutionary computation for reinforcement learning under pomdp. *International Symposium on Artificial Life and Robotics*, 2015(22):322–327.

Peng, J. and Williams, R. J. (1994). Incremental multi-step q-learning. In *Machine Learning Proceedings 1994*, pages 226–232. Elsevier.

Poupart, P., Vlassis, N., Hoey, J., and Regan, K. (2006). An analytic solution to discrete bayesian reinforcement learning. In *Proceedings of the 23rd international conference on Machine learning*, pages 697–704. ACM.

Ross, S., Chaib-draa, B., and Pineau, J. (2008). Bayesian reinforcement learning in continuous pomdps. In *International Conference on Robotics and Automation (ICRA)*.

Sallab, A. E., Abdou, M., Perot, E., and Yogamani, S. (2017). Deep reinforcement learning framework for autonomous driving. *Electronic Imaging*, 2017(19):70–76.

Sridharan, M., Wyatt, J., and Dearden, R. (2010). Planning to see: A hierarchical approach to planning visual actions on a robot using pomdps. *Artificial Intelligence*, 174(11):704–725.

Sutton, R. S. (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in neural information processing systems*, pages 1038–1044.

Suzuki, K. and Kato, S. (2017). Hybrid learning using profit sharing and genetic algorithm for partially observable markov decision processes. In *International Conference on Network-Based Information Systems*, pages 463–475. Springer.

Thomson, B. and Young, S. (2010). Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Uemura, W., Ueno, A., and Tatsumi, S. (2005). An episode-based profit sharing method for pomdps. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 88(6):761–774. (in Japanese).

Watkins, C. J. and Dayan, P. (1992). Q-learning. *Machine learning*, 8(3-4):279–292.

Whitehead, S. D. and Ballard, D. H. (1990). Active perception and reinforcement learning. *Neural Computation*, 2(4):409–419.

Whitehead, S. D. and Ballard, D. H. (1991). Learning to perceive and act by trial and error. *Machine Learning*, 7(1):45–83.

Wiering, M. and Schmidhuber, J. (1997). Hq-learning. *Adaptive Behavior*, 6(2):219–246.

Wiering, M. and Schmidhuber, J. (1998). Fast online q (λ). *Machine Learning*, 33(1):105–115.

Yamamura, M., Miyazaki, K., and Kobayashi, S. (1995). A survey on learning for agents. *The Japanese Society for Artificial Intelligence*, 10(5):683–689.