Systems and Computers in Japan, Vol. 35, No. 2, 2004
Translated from Denshi Joho Tsushin Gakkai Ronbunshi, Vol. J84-D-I, No. 12, December 2001, pp. 1635–1647

Delayed Reward-Based Genetic Algorithms for Partially Observable Markov Decision Problems

Yoshihide Yamashiro,[1,*] Atsushi Ueno,[1] and Hideaki Takeda[2]

[1]Graduate School of Information Science, Nara Institute of Science and Technology (NAIST), Ikoma, Nara, 630-0192 Japan

[2]National Institute of Informatics, Tokyo, 101-8430 Japan

SUMMARY

Reinforcement learning often involves assuming Markov characteristics. However, the agent cannot always observe the environment completely, and in such cases, different states are observed as the same state. In this research, the authors develop a Delayed Reward-based Genetic Algorithm for POMDP (DRGA) as a means to solve a partially observable Markov decision problem (POMDP) which has such perceptual aliasing problems. The DRGA breaks down the POMDP into several subtasks, and then solves the POMDP by breaking down the agent into several subagents. Each subagent acquires policies adapted to the environment based on the delayed rewards from the environment, and these policies are evolved using a genetic algorithm based on the delayed rewards. The agent adapts to the environment by combining effective policies that remain after natural selection. The authors apply this method to maze search problems in which perception is limited in order to demonstrate its validity. © 2004 Wiley Periodicals, Inc. Syst Comp Jpn, 35(2): 66–78, 2004; Published online in Wiley InterScience (www.interscience.wiley.com). DOI 10.1002/scj.10230

*Presently affiliated with Sanyo Electric Co.

Contract grant sponsor: Partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B)(2), 11480078.

Key words: reinforcement learning; genetic algorithm; partially observable Markov decision problem; perceptual aliasing problem.

1. Introduction

Many reinforcement learning methods target the Markov decision process (MDP). In many cases, however, the agent cannot perceive the complete information. This results in a problem in which several different states are perceived to be the same state. This is referred to as the perceptual aliasing problem [1]. In the environment of the partially observable Markov decision problem (POMDP) in which this kind of perceptual aliasing problem occurs, the agent cannot accurately know the current state, and so good learning results cannot be achieved using an MDP algorithm.

However, with respect to an agent undertaking a task, the agent does not need to know the environment completely, but only needs to have inside itself a model of the environments to complete the task. It is from this perspective that in the authors' research, an agent is broken down into several subagents, and control is passed from one subagent to another before the perceptual aliasing problem affects task completion. Subagents resulting from such an approach can treat this problem as MDPs and can resolve a POMDP. The subdivision of the task is performed by setting subgoals for each subagent. The purpose of each subagent is to complete its subgoal. The problem can be broken down

into two levels: the learning of subgoals in order to effectively subdivide the task, and the learning of MDP in each subagent.

In this kind of incomplete perceptual problem, abstraction can be performed effectively because perception is incomplete, and the same policy can be used in different situations. When different behaviors must be selected at different but same-looking points, completing the task becomes difficult. In contrast, when the same behavior can be selected at the same-looking points, knowledge can be effectively reused. Thus, in this paper, the authors propose a Delayed Reward-Based Genetic Algorithm for POMDP (DRGA) as a way to address the problem of the same small problem appearing repeatedly in various situations in the large-scale problem of incomplete perception. In DRGA, the problem is subdivided into MDPs by looking at the task overall, measures which may be useful in different situations survive, and a large-scale POMDP can be addressed by using these useful measures in situations where they are effective.

HQ-learning [2] has been proposed in prior research to solve this problem by breaking down POMDP into subtasks. However, in HQ-learning, each subagent is trained completely independently, and so the reuse of effective measures is not considered. When dealing with a large-scale problem of incomplete perception that can be broken down into identical small problems, completely separate learning is inefficient. In this paper, the authors demonstrate the effectiveness of DRGA through experiments which compare it with HQ-learning.

2. POMDP

2.1. The perceptual aliasing problem in grid space

In this research, the perceptual aliasing problem is addressed in the following grid space. The perceptual capacity of the agent is limited to knowing whether or not there is a wall in the adjacent grid in the grid environment. In other words, the perceptual input of the agent is represented with 4 bits. When this input is one in sequence starting from the first bit, this indicates that there is a wall in the grid in the north, east, south, and west directions [Fig. 1(a)]. In Fig. 1(b), the agent cannot know which position it is in because it is getting the same perceptual input from positions S1 and S7, and S2 and S6. In this problem, although the environment is Markovian, it appears to be POMDP because of the lack of the agent's perceptual capacity.

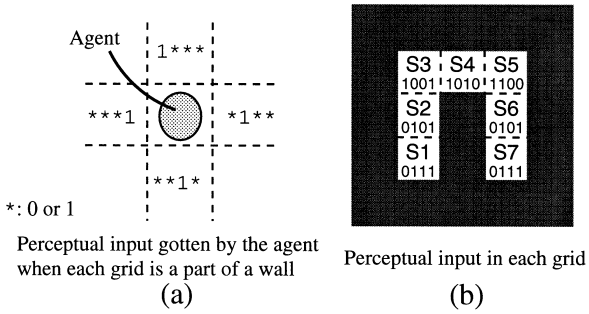


Fig. 1. Perceptual aliasing in grid space.

2.2. HQ-Learning

HQ-learning is an algorithm expanded hierarchically from Q-learning [3]. It is used to resolve perceptual aliasing problems deemed to be POMDP problems. In a partially observable problem, it breaks down a task into subtasks by setting up subgoals, and Q-learning is performed for each subtask. Figure 2 shows the agent architecture in HQ-learning.

An agent consists of several subagents, with control being passed in sequence from subagent 1 to subagent *m*, where *m* represents a predetermined number. Control is transferred when each subagent achieves its particular subgoal. A subgoal is defined as one perceptual state among the perceptual states.

The behaviors of each subagent were determined using a Max-Boltzmann rule based on a Q-table. A Q-table is a table of Q-values (behavior values). If the set of perceptual states is *O* and the set of behaviors is *A*, then there are

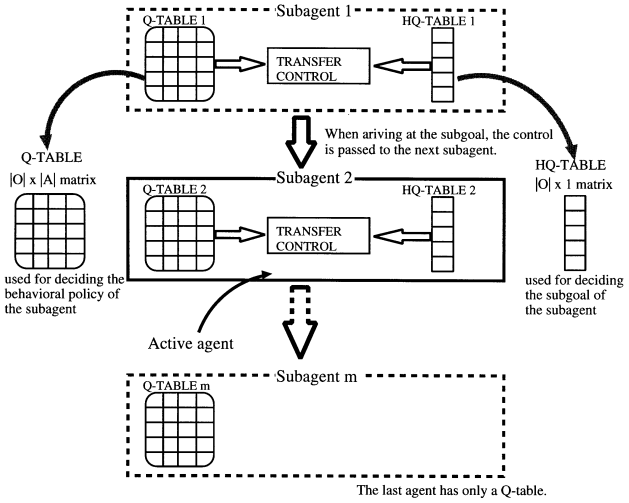


Fig. 2. Agent architecture in HQ-learning.

$|O| \times |A|$ combinations of perception and behavior. The Max-Boltzmann rule selects behaviors according to the Max-Boltzmann distribution: with probability P_{max} take the behavior with maximal Q-value; with probability $1 - P_{max}$ select a behavior according to the Boltzmann distribution represented as follows:

$$prob_o(a) = \frac{e^{Q(o,a)/T}}{\sum_{a' \in A} e^{Q(o,a')/T}} \quad (1)$$

Here, $prob_o(a)$ represents the probability of selecting the behavior a in the perceptual state o, and T represents a temperature parameter which adjusts the randomness when selecting behaviors.

Subgoal learning is performed using the HQ-table. The latter is a table of HQ-values (estimated value of each perceptual state as a subgoal), and is $|O| \times 1$ in size. A subgoal is determined using the Max-Random rule based on the HQ-table. In other words, the subgoal with maximal HQ-value is selected with probability P_{max}, and a random subgoal is selected with probability $1 - P_{max}$.

In HQ-learning, each state transition is recorded, and the Q-values and HQ-values are updated at once when agent completes the task. The Q-values are updated using offline Q(λ)-learning [4], and HQ-values are updated as follows:

$$R_i = \sum_{t=t_i}^{t_{i+1}-1} \gamma^{t-t_i} R(s_t, a_t)$$

$$HQ'_i(\hat{o}_i) \leftarrow R_i + \gamma^{t_{i+1}-t_i}[(1-\lambda) \max_{o' \in O} HQ_{i+1}(o') + \lambda HQ'_{i+1}(\hat{o}_{i+1})]$$

$$HQ_i(\hat{o}_i) \leftarrow (1-\alpha_{HQ}) HQ_i(\hat{o}_i) + \alpha_{HQ} HQ'_i(\hat{o}_i) \quad (2)$$

Here, $\gamma (0 \le \gamma \le 1)$ is the reward discount rate; it shows the trade-off between future reward and current reward. $R(s, a)$ represents the reward obtained in state s when taking behavior a. Therefore, R_i shows the sum of the discounted rewards obtained while subagent i takes behavior (from time t_i to $t_{i+1} - 1$). $HQ_i(o)$ represents the HQ-value of the perceptual state o in subagent i; $\hat{o}_i$ represents the subgoal selected at this point by subagent i, and $\alpha_{HQ} (0 \le \alpha_{HQ} \le 1)$ represents the learning rate for the HQ-table. $HQ'_i(o)$ represents the HQ-value aimed for when using eligibility traces,* and $\lambda (0 \le \lambda \le 1)$ is the decay-rate parameter which represents the extent to which eligibility traces are being used.

*Eligibility traces refer to calculations which take into consideration not only the evaluation values of the state immediately after itself on the state transition sequence, but also the evaluation values of all later states when computing the evaluation (Q-value or HQ-value) in a state. This has the effect of making allocation of rewards in reinforcement learning more efficient and addressing a partial non-Markov problem. Refer to Ref. 5 for details.

2.3. Problems with HQ-learning

In HQ-learning, the problem is broken down until it can be represented using MDPs, and then subagents are used in the predetermined sequence. In a method which determines subagents based only on internal variables in the agent, there is no index which reflects which policy is effective when switching subagents. As a result, the only measures available consist of trying existing policies at random, or learning completely independently. In HQ-learning, the latter measure is used, and reuse of effective policies is not taken into consideration at all. When dealing with a large-scale problem of incomplete perception which can be broken down into identical small problems as is addressed in this paper, completely independent learning is inefficient. As a result, when many subagents are needed to reach a goal, the agent may not be able to learn the task. The problem addressed in the original paper [2] on HQ-learning was solved with a minimum of three subagents for even the most difficult problem, so that it is necessary to check the applicability for larger-scale problems.

3. The Purpose and Method in the Authors' Research

In this research, the authors address problems in which identical small problems appear in various situations in a POMDP. Then, they attempt to develop a method which can address a large-scale POMDP by reusing the same policies in different situations by using perceptual aliasing while breaking down the problem into MDPs by taking the overall task into consideration.

It may seem unnatural for the method to address a perceptual aliasing problem to leave out perceptual aliasing. When different points appear to be the same, difficult problems occur when the respective different behaviors must be selected. In contrast, when the same behavior can be selected regardless of the case, knowledge can be reused effectively. Therefore, a fundamental characteristic of a learning system which reuses policies is that perceptual aliasing occurs at the changeover points of subagents. In the authors' research, the aim is to solve problems efficiently by eliminating perceptual aliasing which impedes task completion, and using it effectively when it does not impede task completion.

In order to realize this aim, the authors propose selecting subagents based on perceptual information for the changeover point of subagents. This is derived from a human policy selection method in which the policy of "walk along the wall" is selected based on perceptual information which says that "there is a wall in front." When dealing with a problem in which a large number of identical small problems appear in various situations, which situation

a particular policy is effective in depends on the environment. By selecting the subagent based on the perceptual information for the changeover point, subagent switching appropriate for the environment should result.

4. DRGA

In this section, the authors describe the overall picture and the algorithm of the DRGA architecture, and explain how it is used.

4.1. DRGA architecture

In DRGA, the agent performs reinforcement learning based on experiences through trial and error in the environment, and when a particular number of trials is completed, the agent evolves using a genetic algorithm (GA).

DRGA consists of several subagents. Each subagent has perceptual information (starting information) about the starting point* where it is selected and launched. The number of subagents is the same as the number of perceptual states, and each subagent has several policies (Fig. 3). The number of policies in each subagent is fixed to N_P. Here, a policy represents the combination of a fixed mapping ($|O| \times 1$) from each perceptual state to a behavior and a perceptual state corresponding to the subgoal (subgoal state).†

An agent selects one of the policies held by a subagent corresponding to its perceptual information at the changeover point of subagents, and then undertakes behavior based on that policy until it reaches the subgoal. When the subgoal is reached, the subagent whose starting information is equal to the perceptual information takes over.

The repertoire of policies held by a subagent evolves using a GA, and each subagent grades the policies using Q(λ)-learning. GA and Q(λ)-learning are performed based on delayed rewards, some portions of rewards distributed to the states on the way to a goal. Subgoal learning and subagent behavior learning are both achieved using Q(λ)-learning within the search range defined by the GA.

4.2. Evolving a policy: GA

4.2.1. Encoding and decoding

In DRGA, deterministic behavioral policies for the agent's perceptual information are created by the GA. These policies are taken to be one form of innate behavior.

In DRGA, one policy is represented by a single chromosome, as shown in Fig. 4. In other words, the subgoal information and the mapping from each perceptual state to a behavior is described in the chromosome. For instance, if the number of behavior types for an agent is $|A|$ and the number of types of perception which can be perceived is $|O|$, then the number of types of chromosomes will be $A^{|O|} \times |O|$.*

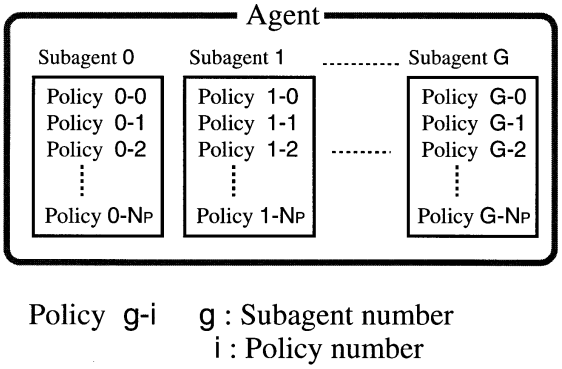


Fig. 3. Agent and subagent.

4.2.2. Evolutionary procedure

In general, in GAs, each individual behaves in the environment, and its fitness is measured. Better individuals are selected based on their fitness, and crossover or other GA operators are used on the surviving group members.

In DRGA, GA processing (selection and GA operators) is performed in the subagents. According to Goldberg [6], GAs are characterized by taking a group with high diversity (multiple points) to one with high fitness and low diversity (ultimately one point) in accordance with the building-block hypothesis. As a result, if all subagents undergo GA processing together, they will come to represent one agent, and so can only address a single subtask. Therefore, GA processing is not performed between different subagents.

First, an initial group (size N_P) of policies is created at random in each subagent at the start of learning. At this point, policies which clearly go into the wall are omitted so as not to belong to the initial group. Then, the agent becomes active in the environment, and fitness for each policy is found based on Q(λ)-learning which will be explained in Section 4.3. In DRGA, the Q-value [the evaluation based on the delayed reward in Eq. (5)] obtained during policy selection learning is used for the fitness of each policy.

GA processing is performed as shown below. It is the same for all subagents.

- Selection: Elite and roulette strategy

*Same as the "changeover point of subagents" in Section 3.

†As will be explained in the practical example in Section 4.5, the utility of a policy depends on its subgoal state.

*For the problem in Fig. 4, the GA search space is 6.9×10^{10} in size. Because the GA is a method appropriate for finding a quasi-optimal solution in a massive search space, the authors decided to use the GA for this kind of large-search-space problem.

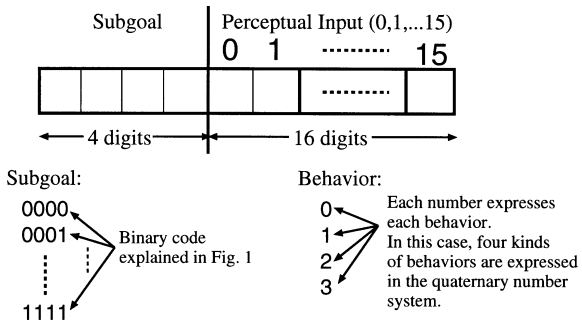


Fig. 4. Genotype in DRGA (case of Fig. 1).

- Crossover: Crossover 1–3 points
- Spontaneous mutation: Replacement of genes (random)

First, N_P elements are selected as parents for the next generation from the current group of policies. At this point, N_e elements are selected from the highest level of fitness downward using the elite strategy, and the remainder are selected using probability proportional to fitness. Two chromosomes are selected at random from the reproduced parent chromosome group, and are crossed over with probability C_p. Crossover is a process which selects at random crossover points (1–3 points) and then swaps the parent genetic code between the crossover points. Next, spontaneous mutation occurs with probability M_p in each gene in each chromosome. During spontaneous mutation, genes are exchanged at random, but mutations that make the agent clearly run into the wall are omitted. A new generation of policies is created by repeating this process $N_P/2$ times for the reproduced parent chromosome group.

4.3. Policy selection training: Q(λ)-learning

Each subagent performs *a posteriori* learning in order to select an appropriate policy from among the repertoire of policies obtained genetically. Q(λ)-learning is used for *a posteriori* learning.

4.3.1. Policy selection

In DRGA, reinforcement learning is used for policy learning in each subagent. In general, reinforcement learning is a method for behavior learning, but in DRGA is used for policy learning. The reinforcement learning algorithm used in DRGA is Q(λ)-learning. The Q-value in DRGA is an evaluation value for each policy, and is represented using a policy-table. A state in a Q-table equals a subagent in a policy-table. A behavior for each state in a Q-table is equivalent to a policy for each subagent in a policy-table. In DRGA, because mapping from perceptual states to behaviors is deterministic, whether or not the subgoal is achieved when a policy is selected in a particular subagent is determined by the environment.

When performing reinforcement learning, normally a behavior is selected while leaving a little search capacity, and the behavior for the maximum Q-value is not always selected. There are various methods available for selection policy: one method selects the behavior with the greatest evaluation value using the probability *prob* and then selects at random using probability 1 – *prob*, while another method determines the behavior using a Boltzmann distribution [8]. In DRGA, selection is performed using the Boltzmann distribution shown as follows:

$$prob_g(p) = \frac{e^{Q(g,p)/T}}{\sum_{p' \in P_g} e^{Q(g,p')/T}} \tag{3}$$

$prob_g(p)$ is the probability of selecting policy p in subagent g. P_g represents the set of policies in subagent g. $Q(g, p)$ is the evaluation value for policy p in subagent g, in other words the Q-value. T is the temperature parameter which adjusts the randomness in selecting a policy.

4.3.2. The update rule of the Q-values

Offline Q(λ)-learning is used to revise the Q-values. The update rules used by the system to revise the Q-value when policy p_i is performed by subagent g_i at time i are

$$Q'(g_i, p_i) \leftarrow R_i + \gamma[(1-\lambda) \max_{p' \in P_{g_{i+1}}} Q(g_{i+1}, p') + \lambda Q'(g_{i+1}, p_{i+1})] \tag{4}$$

$$Q(g_i, p_i) \leftarrow (1-\alpha)Q(g_i, p_i) + \alpha Q'(g_i, p_i) \tag{5}$$

Here, $i = 1, 2, \ldots$ is the discrete time which represents the transition of subagents. Although the actual time in which each agent operates is different, transfer of subagent control is represented by a count of 1. R_i represents the total reward obtained by subagent g_i. γ is the discount rate, and α is the learning rate, with both being $0 \le \gamma \le 1$, and $0 \le \alpha \le 1$. $Q'(g_i, p_i)$ is the Q-value to be reached when using eligibility traces (refer to the first footnote in Section 2), and $\lambda (0 \le \lambda \le 1)$ is the decay-rate parameter which represents the extent to which eligibility traces are used.

Updating is performed in the order $i = I, I-1, \ldots, 2, 1$. I is the time at which one of the stopping conditions (task completion or the lifespan limit of the agent for the task) is satisfied after the agent performs the task. One trial refers to the period up to when the agent satisfies one of the stopping conditions. At time $i = I$, $Q'(g_I, p_I) = R_I$.

Each policy persists until the subgoal is reached, the task is achieved, or the agent lifespan (*MaxStep*) is reached. When the subgoal is reached, a small reward is obtained,

and then the next subagent is given control based on perceptual information at that time. If the task is achieved, a large reward results. If the lifespan is reached, there is no reward. The agent life continues beyond the changeover points of subagents until it reaches its lifespan; sometimes an agent uses up its lifespan in one policy, and other times an agent uses up its lifespan with several policies.

4.4. DRGA outline

Figure 5 shows an outline of DRGA. An agent performs reinforcement learning based on experience obtained by trial and error in the environment. After a certain number of trials ($SIZE_T$) are performed, the agent itself evolves using the GA. At this point, an individual in the GA is equivalent to a policy in a subagent in DRGA, and the fitness, the evaluation standard for evolution, is represented using the Q-value for each policy. After evolution, the policy-table values are initialized, and the agent with new sets of policies repeats trial and error in the environment, and aims to complete the task.

Each policy receives a small reward when a subgoal is reached. As a result, the selection probability for a policy which is able to reach a subgoal rises in each subagent due to reinforcement learning. A policy which can reach a subgoal tends to move a certain distance within the environment. As a result, this policy is useful for expanding research range at the stage at which a goal has not even been reached once. In addition, because a large reward is obtained when a goal is reached once, policies which are useful for reaching a goal are strongly reinforced.

Since GA is performed based on the Q-values of policies, policies which are useful in different situations in the whole task increase progressively in each subagent. In the initial stage of learning, competition sometimes occurs between policies in a subagent. Using the local search capability of GA, the behavior and subgoal are slowly varied, pathways which do not cause competition are found in the environment and strengthened, and as a result competition progressively disappears. Then, the best policy in a subagent survives, and learning converges.

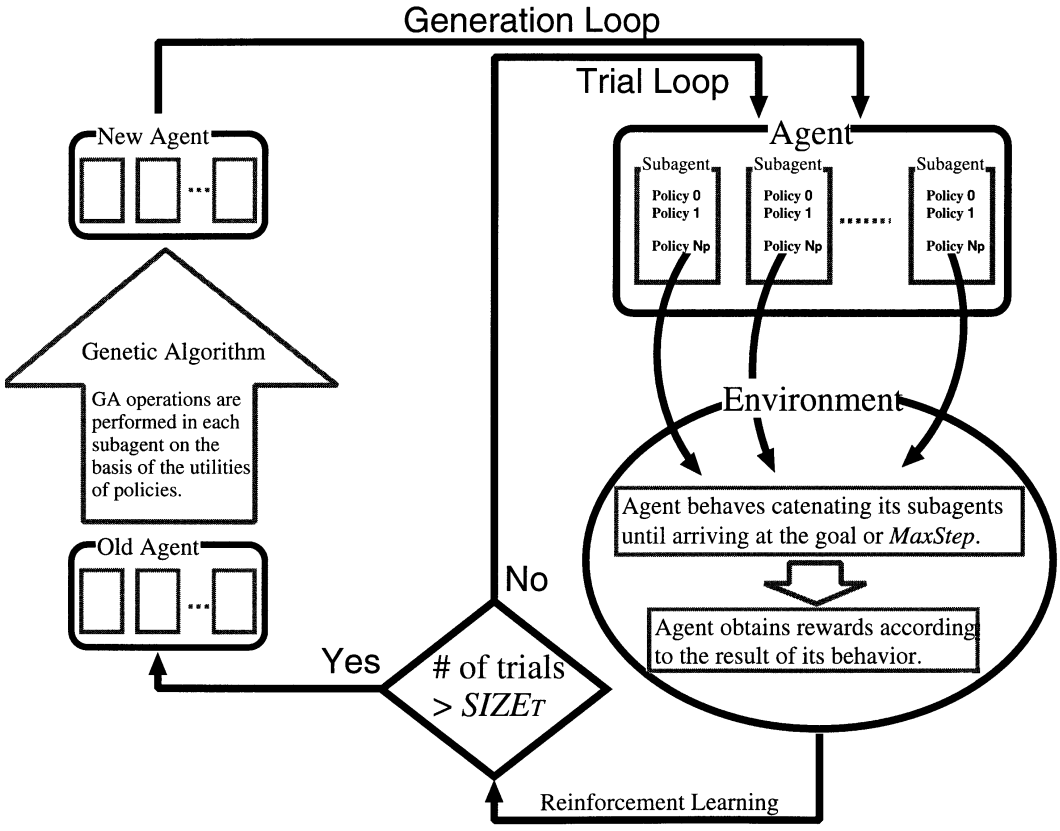


Fig. 5. Outline of DRGA.

4.5. A practical example of DRGA

The behavior of DRGA will be explained using the 10×10 maze shown in Fig. 6. The task is to go from the initial position S to the goal G, and the aim of DRGA is to find a route which can achieve this task. The agent can engage in four behaviors: "move one square to the north, move one square to the east, move one square to the south, move one square to the west." The only perceptual information the agent can get is whether or not there is a wall on the adjacent grid (refer to Fig. 1). There are only 16 types of perceptual information that can be had in this limited perceptual world, and so perceptual aliasing occurs. As a result of perceptual aliasing, this problem is not an MDP, and so cannot be solved using one subagent. For instance, if the shortest route, shown by the solid line in the figure, is taken, then for the perceptual state "0101" in which there are walls to the east and west, the first move is to the north (S $\rightarrow$ g4), the next move is to the south (g4 $\rightarrow$ g6), and the last move is to the north (g6 $\rightarrow$ G), otherwise G cannot be reached. In this problem, three subagents are required at a minimum.

When this problem is addressed using DRGA, first a subagent with the starting information "0011," which indicates that there are walls to the south and west, is launched. One policy is selected from among the group of policies in this subagent using Eq. (3). When learning starts, the selection is completely random because the Q-values for all policies are initialized to zero. If the behavior for the starting information in the selected policy is move north, then the subagent moves one square along the solid line (position g1). If the policy is move east, then the subagent moves one square along the broken line. Because policies which move into a wall are eliminated when initializing this policy group, there are no policies for moving south or for moving west in the group.

Let us view what happens when the policy move north continues. The next perceptual information obtained is "0101," and the behavior for this perceptual information in the current policy includes three possibilities: move north, move south, and stop at this square identifying it as a subgoal. If the subagent moves south, this returns it to S. Because the policy behavior is deterministic, this policy repeats the behavior "north south north south . . . " until *MaxStep*, and then has its lifespan end without any reward. On the other hand, if the policy is move north, the subagent moves one square to the north, and then with the same perceptual information, moves one square to the north again and again, finally reaching position g2. The perceptual information obtained here is "1001," and the behavior includes three possibilities: move south, move east, and stop at this square identifying it as a subgoal. If the policy is move south, the subagent moves one square to the south, and there its behavior must be to move north. As a result, this policy repeats the behavior "south north south north . . . " until *MaxStep*, and then has its lifespan end without any reward. If the policy is move east, the subagent follows the solid line again.

When viewed in this fashion, among the policies that move north from S, there are five types that will reach a subgoal, as shown in Table 1. Any behavior is acceptable for the 10 types of perceptual information not shown in the table. Policies p1–p4 make the agent move along the solid line and arrive at subgoals g1–g4, respectively. Policy p5 makes the agent pass through g1–g4 and arrive at g5. Then they obtain a small reward, and control is passed to the next subagent. In this manner, the policies which can reach the subgoal are reinforced progressively by small rewards.

Also in the subagents which take over after the first subgoal or move to the east from S along the broken line, the policies which can reach their subgoals are reinforced progressively. When G is reached using a combination of

Table 1. Policies that can reach a subgoal

Policy	Wall to the south and west 0011	Wall to the east and west 0101	Wall to the north and west 1001	Wall to the south and north 1010	Wall to the north 1000	Wall to the north and east 1100
p1	Move north	sg	*	*	*	*
p2	Move north	Move north	sg	*	*	*
p3	Move north	Move north	Move east	sg	*	*
p4	Move north	Move north	Move east	Move east	sg	*
p5	Move north	Move north	Move east	Move east	Move east	sg

sg: subgoal
*: any behavior

these policies, a large reward results, and all policies along the path are reinforced by the distribution of the reward based on Q(λ)-learning. The percentage of effective policies increases progressively in each subagent as a result of natural selection due to the GA based on the Q-value.

Here, let us consider using Eq. (6) for the large reward which results when the goal is reached, and Eq. (7) for the small reward which results when a subgoal is reached:

$$R_i = (MaxStep - step)^2/10.0 \quad (6)$$

$$R_i = (observenumber) \quad (7)$$

step represents the number of steps from the initial position until the goal is reached, and *observenumber* represents the number of types of perceptual information that a subagent perceives until reaching a subgoal. *observenumber* is not incremented while the same perceptual state continues to be observed. It is incremented when the subagent observes a new perceptual state.

For the five policies in Table 1, the reward when a subgoal is reached rises as this falls, and so the reward is *n* + 1 for policy pn. On the other hand, if it is assumed that the shortest path is taken from each subgoal, the reward when the goal is reached is 518.4 in the case of using p1–p4, and 384.4 in the case of using p5. Given this, p4 which is on the shortest path to the goal and moves the longest distance is reinforced the most.

Note, however, that the shortest path makes reaching the goal difficult because of perceptual aliasing at the subgoal. This will be described in detail in the next section.

4.6. DRGA benefits and problems

The fundamental benefits and problems of DRGA will be explained using the 10 × 10 maze task in the previous section. Under DRGA, a large problem is broken down into a number of similar small problems by eliminating perceptual aliasing which represents an impediment to achieving the task, and taking full advantage of perceptual aliasing which does not represent an impediment. As a result, an optimal solution is not necessarily converged on.

In the 10 × 10 maze task, the solid-line path which has subgoals at points g4 and g6 in Fig. 6 represents the optimal solution (path with the greatest reward). S and g6 have the same perceptual information "0011" and launched the same subagent. However, because at S the subagent must move north and at g6 the subagent must move east, competition between policies occurs in the subagent, and so frequently it cannot reach its goal effectively. In contrast, for the broken-line path which has subgoals at points g5 and g6, from S to g5 and g6 to G the same policy can be used, and so competition between policies does not occur. In this instance, only two policies have to be learned in order to reach the goal, and so learning is efficient. In practice, when learning is performed for this task using DRGA, this path is selected with a high probability (74.8%) after learning has converged.

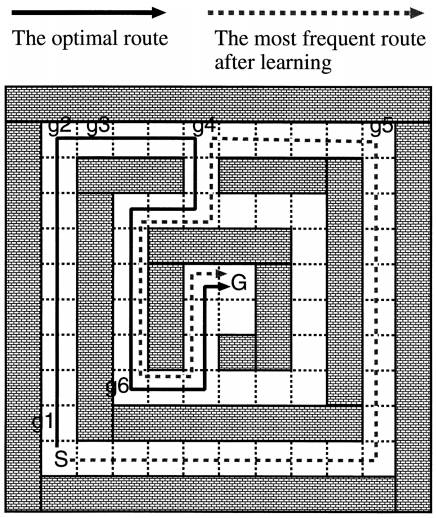


Fig. 6. 10 × 10 maze.

As this example shows, DRGA has a feature which reduces the number of policies necessary to reach a goal by reusing effective policies. As a result, the search space can be limited and learning can be performed more efficiently by finding effective policies which can be reused even for tasks which require switching between several subagents before reaching the goal.

On the other hand, the problems in DRGA include the optimal solution not necessarily being converged on, because the perceptual aliasing which represents a barrier to task completion is eliminated. When perceptual aliasing represents a barrier, competition among policies occurs in the subagent. In such cases, as learning progresses, the behavior and subgoals are slowly varied by the local search capability of GA, and a path which does not cause competition in the environment is found and strengthened. As a result, competition is progressively reduced. Eventually, the best policy remains in each subagent, and learning converges. In this fashion, even if perceptual aliasing does represent a barrier to task completion, DRGA still has a function which avoids the perceptual aliasing problem by finding a quasi-optimal solution.

In addition, sometimes perceptual aliasing which represents a barrier cannot be eliminated. This includes cases in which there is no path to the goal that avoids competition. For instance, the maze in Fig. 6 represents such a case if position g5 is blocked by walls. This kind of problem is an extremely difficult search problem. Although methods to select the subagent based on not only informa-

tion about the starting position, but also the change of the information before and after the position can be conceived, such flexible policy switching represents a future topic.

When compared with HQ-learning, the purpose of subgoal learning in DRGA is to effectively link good policies as well as to break down the problem into MDPs, which is the sole purpose of subgoal learning in HQ-learning. As a result, the subgoal learning in DRGA is a little more difficult than that in HQ-learning. On the other hand, because policies between subgoals can be reused in DRGA, the policy learning is extremely efficient relative to HQ-learning in which that is performed independently. In addition, DRGA also has the advantage of not needing to predefine the number of subagents necessary to reach a goal.

5. Simulation Experiments

5.1. Task (a)—key and door task

5.1.1. Task setup

The first experiment involves a 26 × 23 maze, as shown in Fig. 7. This task is the most complex problem given in the original paper [2] on HQ-learning. In this experiment, the authors show that a problem complex enough to be solved using HQ-learning can also be solved using DRGA.

This task involves starting from initial position S, then (1) finding the key at position K, (2) putting the key into the door (gray area) to open it, and (3) reaching G.

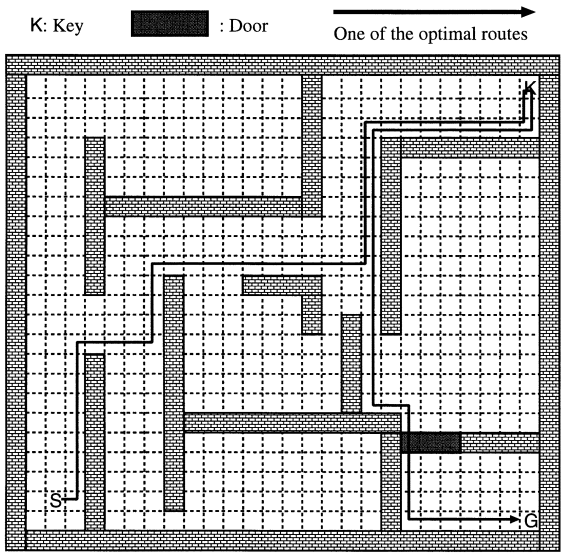


Fig. 7. Key and door task.

There are 11 types of perceptual information an agent can obtain in this maze. In this maze, the door cannot be opened unless the key is found. Thus, there are a total of 960 states when a distinction is made for whether or not the key is obtained. The shortest path to G is 83 steps, and the necessary minimum number of subgoals is two. Therefore, there is little possibility of reusing policies,* and as a result the effectiveness of reusability is not evaluated in this experiment.

According to Ref. 2, the goal was not reached even once during 20,000 trials when using a random walk with a limit of 1000 steps. In addition, the shortest path during 3000 trials using a random walk with a limit of 10,000 steps was 1174 steps.

5.1.2. Reward and parameter settings

The reward explained in Section 4.5 is used: the reward in Eq. (6) is given when the goal is reached, and the reward in Eq. (7) is given when a subgoal is reached.

The parameters for GA are set as follows: $N_P = 30$, $N_e = 4$, $C_p = 0.9$, and $M_p = 0.15$. The number of crossover points is 2. For Q(λ)-learning, $\gamma = 0.7$, $\alpha = 0.15$, and $\lambda = 0.4$, with the initial Q-value for each policy set to 0.0. The temperature parameter used for behavior selection, T, is set to 3.0. The *MaxStep* is set to 250. The $SIZE_T$ for the number of trials used to revise generations is set to 1000 as an initial value, and augmented by 100 for each generation, up to 7500. This represents the 65th generation. It remains at 7500 in all later generations. The number of generations until the end of the experiment is 100.

The authors perform HQ-learning on the same task for comparison. HQ-learning parameters are set in accordance with Ref. 2. The reward when reaching the goal is 500, and a reward of –0.1 is given for all behaviors that do not reach the goal. The update rule for Q(λ)-learning, the same as Eqs. (4) and (5) for DRGA, is used to update the Q-value, and Eq. (2) is used to update the HQ-value. $\gamma = 1.0$, $\lambda = 0.9$, $\alpha = 0.05$, $\alpha_{HQ} = 0.1$, $T = 0.2$, and *MaxStep* = 1000. P_{max} is set to 0.4 for the first trial, and linearly incremented until it reaches 0.8 at the last trial. γ, λ, and P_{max} are shared in Q(λ)-learning and HQ-learning. Here, *MaxStep* (agent lifespan) is significantly different from DRGA. DRGA behavior is deterministic; only during policy selection does it have a little randomness. As a result, there is no need to make the lifespan *MaxStep* large in DRGA.

*Policies whose starting information and subgoal information are equivalent cannot start and so are not useful. As a result, the policies before and after the subgoal must be different. Therefore, when there are only two subgoals, there is no possibility of reusing policies other than the cases in which the first policy is reused as the third.

5.1.3. Results—Task (a)

Figure 8 shows histograms of the number of steps to reach the goal. The graph of HQ-learning represents the results of an experiment performed using an 8-subagent system. It shows the distribution of the number of steps to the goal for sets of 600 trials during 60,000 trials using HQ-learning. These two graphs show the averages of 100 experiments. Table 2 lists the ratios of the top three which had the three greatest distributions in the histograms and the ratio of failure when the experiment ended (DRGA, 100th generation; HQ-learning, the final 600 trials). Based on Fig. 8 and Table 2, DRGA and HQ-learning can both be said to have solved the POMDP in this task.

5.1.4. Discussion—Task (a)

Both DRGA and HQ-learning reach the goal in the majority of cases (DRGA did not reach the goal at the end of learning in 31.1% of cases, and HQ-learning in 8.00% of cases). As a result, they can be said to have solved the POMDP for the difficulty of this task. The final task failure rate was rather high for DRGA. One reason for this is that the policy-table was cleared when the generation was updated, then policies were selected at random to start a new generation, and so the failure rate for the trials was high. However, there was not a single case in which the combination of subagents that could achieve the task was not created in the 100 experiments. Thus, DRGA can be said to be able to solve the perceptual aliasing problem in this task.

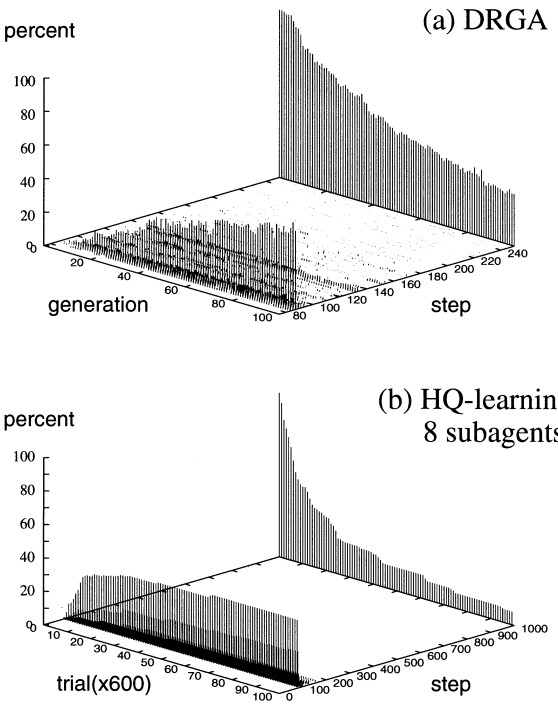


Fig. 8. Histograms of the number of steps to the goal.

Table 2. Ratios of the number of steps to the goal—Key and door task

—	DRGA	HQ-learning
No. 1	47.0% (87 step)	40.0% (87 step)
No. 2	3.63% (89 step)	21.0% (89 step)
No. 3	2.95% (85 step)	8.00% (91 step)
Task failure	31.1% (250 step)	8.00% (1000 step)

In this task, the goal was not reached even once during 20,000 trials for random walk of up to 1000 steps, while effective measures were obtained by repeating trials with 250 steps for DRGA and 1000 steps for HQ-learning. This is thought to be because the randomness of behavior was restricted and the directionality of behavior was augmented.

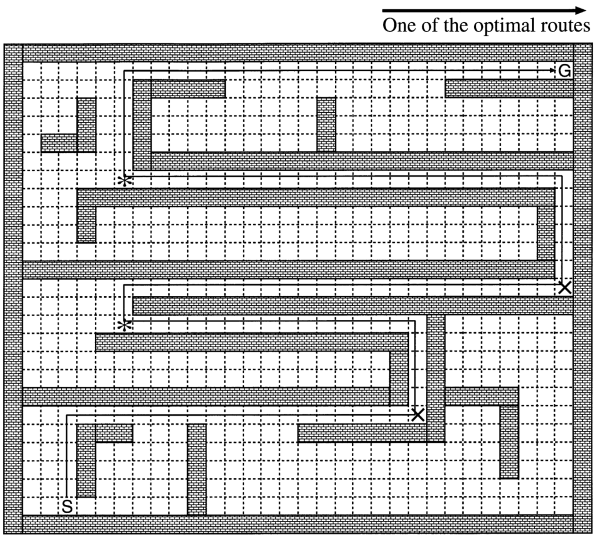
In a task like this which requires a lot of steps to reach its goal, the search range does not expand much while repeatedly searching the same place at the beginning of learning. Even if the goal does happen to be reached, the number of steps to the goal is too high, and rewards are not distributed well. By giving directionality to the agent's behavior, the search range can be expanded easily, and search efficiency can be improved.

DRGA is a method with very high directionality of behavior because behavior for each perceptual state is fixed, and so it is effective when solving tasks like this with long paths to the goal. HQ-learning also augments the directionality of its behavior by updating the Q-values at once when finishing a task, as described in Section 2.2. In other words, in such offline updating, the Q-values do not change during a trial. As a result, there is a high probability that the behavior with the highest Q-value will be selected repeatedly in the same perceptual state. In this task, the directionality is further augmented by giving a small negative reward for all behavior. In other words, due to the negative reward, a difference appears in the Q-values even when the goal is not reached at all, and so the probability that a behavior with a high Q-value will be repeated is raised.

5.2. Task (b)—Original maze (30×25 maze)

5.2.1. Task setup

As a second experiment, the authors created an original maze (Fig. 9) with a greater level of difficulty than the earlier Key and Door task. Using this task, the authors



If subgoals are located at ''*''s and ''×''s, the agent can reuse the policies.

Fig. 9. 30×25 maze.

demonstrate the validity of DRGA: a large-scale POMDP can be solved by reusing effective policies.

This maze is 30×25 in size, larger than the 26×23 maze in Task (a). Because there is no key or door, this task has less states than Task (a). In contrast to the 960 states in Task (a), this task has a total of 551 states. However, in DRGA and HQ-learning, the difficulty is primarily dependent on the number of subagents necessary to reach the goal and the length of the path (refer to Section 5.2.3). There are five places in this task in which the perceptual aliasing problem impedes task completion. In other words, subagents will have to be switched at a minimum of four times. The shortest route consists of 131 steps. Therefore, this task has a much higher difficulty than Task (a).

5.2.2. Results—Task (b)

This task was trained using DRGA with reward and parameters being the same as in Task (a). Figure 10 shows a histogram of the path length. This figure represents the averages of 100 experiments, as was also the case for Fig. 8 in Task (a). Table 3 lists the ratios of the top three which had the three greatest distributions in the histogram and the ratio of failure when the experiment ended (100th generation). Based on Fig. 10 and Table 3, it can be said that DRGA solved the POMDP effectively in this task. Only 18.8% of the trials in the 100th generation could not complete the task. This value represents good results when the fact that the policy-table was cleared in updating the generation is taken into consideration.

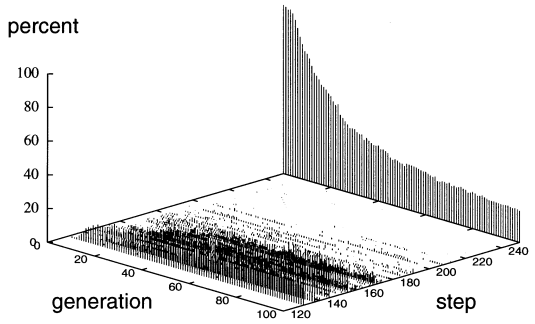


Fig. 10. Histogram of the number of steps to the goal.

5.2.3. Discussion—Task (b)

The learning space for the method to solve a POMDP by breaking it down into MDPs like DRGA is $(|A|^{|O|} \times |O|)^M$ in size. Here, $|A|$ represents the number of types of behaviors, $|O|$ the number of types of perceptual information, and M the number of subagents necessary to reach the goal. $|A|$ and $|O|$ are constants, and so what is important here is M. In addition, the difficulty of the task is significantly related to the length (l_{min}) of the shortest path to the goal. There is a tendency for the number of subagents necessary to rise as the length of the path to the goal increases. Furthermore, (1) it is extremely difficult to reach the goal at the first time without experiences; (2) the path between subgoals lengthens, the number of types of perceptual information to which each subagent needs to learn appropriate behaviors to reach its subgoal increases, and so behavior learning becomes more difficult. These factors increase the difficulty of the task.

Task (b) has a very high level of difficulty given its large values for M (= 5) and l_{min} (= 131) compared to Task (a) (M = 3, l_{min} = 83). However, in DRGA, the failure rate for this task after learning was 18.8%, low compared even to Task (a). This is because policies were reused by selecting subagents based on starting information. In DRGA, by placing subgoals at the "X"s and "*"s in Fig. 9, only the

Table 3. Ratios of the number of steps to the goal—DRGA

—	DRGA
No. 1	24.0% (145 step)
No. 2	18.2% (131 step)
No. 3	9.73% (155 step)
Task failure	18.8% (250 step)

two policies, a policy which started at X and a policy which started at * (or S), are necessary to reach the goal. Therefore, a search space of $(4^{16} \times 16)^5$ can be solved by being effectively limited to $(4^{16} \times 16)^2$. The search space for Task (a) is $(4^{16} \times 16)^3$, and this difference is linked to the inversion phenomenon seen in the task failure rates. Although the maze can be said to be a bit selfish, no *a priori* knowledge is given with respect to the maze structure, and so the $(4^{16} \times 16)^2$ search space can be found by effectively arranging the subgoals using the results of trial and error in the environment. DRGA has an ability of simplifying this kind of large-scale problem using the adaptability to determine the subgoals based on the environment.

Although problems (1) and (2) due to l_{min} growing in size cannot be avoided, with respect to (1), the goal can be effectively found by repeating the 250-step trial for this task owing to the high directionality of behavior in DRGA.

5.2.4. A comparison with HQ-learning

For the purpose of comparison, Task (b) was also trained using HQ-learning. The reward and parameters were the same as Task (a). However, the number of subagents was set to 16. In this task, a minimum of five subagents is required to reach the goal, and the number 16 is used to allow for a margin. In HQ-learning, as the number of subagents rises, the adaptability for the task increases. However, the convergence time for learning rises because each subagent has its own Q-table to be trained independently. In DRGA, there is no need to determine the number of switches of subagent beforehand, and so this kind of trade-off between adaptability and convergence time does not have to be taken into consideration.

Figure 11 shows a histogram of the path length. It shows the distribution of the number of steps required to reach the goal for sets of 600 trials during 120,000 trials performed under HQ-learning. Like Fig. 8 for Task (a), this figure also shows the averages of 100 experiments. In addition, Table 4 lists the ratios of the top three which had

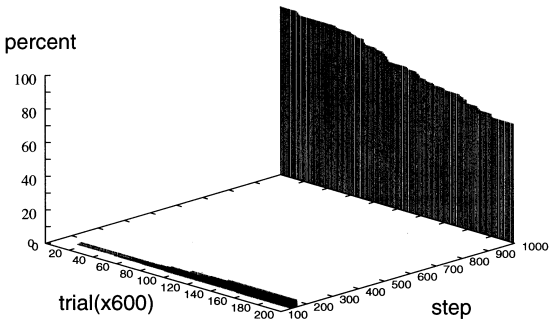


Fig. 11. Histogram of the number of steps to the goal.

Table 4. Ratios of the number of steps to the goal—HQ-learning

—	DRGA
No. 1	4.00% (161 step)
No. 2	4.00% (164 step)
No. 3	3.00% (153 step)
Task failure	71.0% (1000 step)

the three greatest distributions in the histogram and the ratio of failure in the final 600 trials.

Based on Fig. 11 and Table 4, it is difficult to say that HQ-learning can solve the POMDP in this task. The total number of steps for behavior over the 120,000 trials was about 106,610,000 steps. Even compared with the total of 97,010,000 steps over the 100 generations in DRGA, this is still a rather lengthy period of learning. In Fig. 11, the linear drop in the task failure rate is thought to be primarily a result of the linear drop in randomness of behavior. In reality, when this experiment was performed with the total of 60,000 trials, as was the case with Task (a), the task failure rate experienced a similar linear drop, ultimately reaching 67.0%. These results mean that no improvement was shown when the number of trials was doubled. As a result, even if the number of trials is further increased, major improvements in performance cannot be expected.

Because policies are not reused in HQ-learning, the search space for Task (b) remains at $(4^{16} \times 16)^5$, which is rather large compared to the space $[(4^{16} \times 16)^3]$ for Task (a). Whereas Task (a) was solved using HQ-learning, Task (b) was not. This result is consistent with the level of difficulty. In contrast, using the authors' proposed method of DRGA, a large-scale problem such as Task (b) can be solved by limiting the search space by effectively reusing policies.

6. Conclusion

In this paper, the authors proposed a DRGA for problems in which many identical small problems appear in various situations within the POMDPs. In DRGA, learning of the positions of subgoals and learning of behavior policies between subgoals are done at the same time by combining GA based on delayed reward and reinforcement learning. In DRGA, all of the problem is broken down into MDPs by looking at the task overall, and policies that are useful in various situations survive. By using these effective policies in various situations, a large-scale POMDP can be handled. Using simulation experiments, the authors showed that DRGA can solve a large-scale problem that cannot be

solved by previous methods which break down a POMDP into subtasks.

In DRGA, which subagent should be launched next is decided based on the perceptual information at the changeover point of subagents. As a result, there are some problems which cannot be solved owing to perceptual aliasing at the changeover points. When it is determined that perceptual aliasing at a changeover point cannot be avoided, a different index, such as state changes before or after, should be used to select the next subagent at the point while avoiding perceptual aliasing. Using such flexible policy switching represents a future topic. In addition, the authors plan to apply their method to real problems with the feature of the POMDP which can be broken down into MDPs.

Acknowledgment. This research was partially supported by the Ministry of Education, Science, Sports and Culture, Grant-in-Aid for Scientific Research (B)(2), 11480078.

REFERENCES

1. Whitehead SD, Ballard DH. Learning to perceive and act by trial and error. Machine Learning 1991;7:45–83.
2. Wiering M, Schmidhuber J. HQ-learning. Adaptive Behavior 1997;6:219–246.
3. Watkins CJCH, Dayan P. Technical note: Q-learning. Machine Learning 1992;8:279–292.
4. Lin LJ. Reinforcement learning for robots using neural networks. Ph.D. thesis. Carnegie Mellon University, 1993.
5. Sutton RS, Barto AG. Reinforcement learning: An introduction. MIT Press; 1998.
6. Goldberg DE. Genetic algorithms in search, optimization, and machine learning. Addison-Wesley; 1989.
7. Peng J, Williams R. Incremental multi-step Q-learning. Machine Learning 1996;22:283–290.
8. Whitehead SD, Ballard DH. Active perception and reinforcement learning. Proc 7th Int Conference on Machine Learning, p 162–169, 1990.

AUTHORS (from left to right)



Yoshihide Yamashiro received his M.E. degree in information science from Nara Institute of Science and Technology (NAIST) in 2000. He was a student in the Artificial Intelligence Laboratory at the Graduate School of Information Science at NAIST. He joined Sanyo Electric Co. in 2000.

Atsushi Ueno received his B.E., M.E., and Ph.D. degrees in aeronautics and astronautics from the University of Tokyo in 1991, 1993, and 1997. He is a research associate in the Artificial Intelligence Laboratory at the Graduate School of Information Science at Nara Institute of Science and Technology. His research interest is robot learning and autonomous systems.

Hideaki Takeda received his Ph.D. degree in precision machinery engineering from the University of Tokyo in 1991. He was an associate professor at Nara Institute of Science and Technology from 1995 to 2000. He has been an associate professor at the National Institute of Informatics since 2000. He has conducted research on a theory of intelligent computer-aided design systems, in particular experimental study and logical formalization of engineering design. He is also interested in multiagent architectures and ontologies for knowledge base systems.