19

Multi-Legged Robot Control Using GA-Based Q-Learning Method With Neighboring Crossover

Tadahiko Murata[1,2] and Masatoshi Yamaguchi[3]
[1] *Policy Grid Computing Laboratory, Kansai University*
[2] *Department of Informatics, Kansai University*
[3] *Space Systems & Public Information Systems Division, NEC Aerospace Systems, Ltd.*
Japan

1. Introduction

Recently reinforcement learning has received much attention as a learning method (Sutton, 1988; Watkins & Dayan, 1992). It does not need a priori knowledge and has higher capability of reactive and adaptive behaviors. However, there are some significant problems in applying it to real problems. Some of them are deep cost of learning and large size of action-state space. The Q-learning (Watkins & Dayan, 1992), known as one of effective reinforcement learning, has difficulty in accomplishing learning tasks when the size of action-state space is large. Therefore the application of the usual Q-learning is restricted to simple tasks with the small action-state space. Due to the large action-state space, it is difficult to apply the Q-learning directly to real problems such as control problem for robots with many redundant degrees of freedom.

In order to cope with such difficulty of large action-state space, various structural and dividing algorithms of the action-state space were proposed (Holland, 1986; Svinin *et al.*, 2001; Yamada *et al.*, 2001). In the dividing algorithm, the state space is divided dynamically, however, the action space is fixed so that it is impossible to apply the algorithm to the task with large action space. In the classifier system, "don't care" attribute is introduced in order to create general rules. But, that causes the partially observable problem. Furthermore, an ensemble system of general and special rules should be prepared in advance.

Considering these points, Ito & Matsuno (2002) have proposed a GA-based Q-learning method called "Q-learning with Dynamic Structuring of Exploration Space Based on Genetic Algorithm (QDSEGA)." In their algorithm, a genetic algorithm is employed to reconstruct an action-state space which is learned by Q-learning. That is, the size of the action-state space is reduced by the genetic algorithm in order to apply Q-learning to the learning process of that space. They applied their algorithm to a control problem of multi-legged robot which has many redundant degrees of freedom and large action-state space. By applying their restriction method for action-state space, they successfully obtained the control rules for a multi-legged robot by their QDSEGA. However, the way to apply a genetic algorithm seems so straightfoward in their study. Therefore we propose a crossover and a modified fitness definition for QDSEGA (Murata & Yamaguchi, 2005; Murata & Aoki,

Source: Frontiers in Evolutionary Robotics, Book edited by: Hitoshi Iba, ISBN 978-3-902613-19-6, pp. 596, April 2008, I-Tech Education and Publishing, Vienna, Austria

2007). Through our computer simulations on a control problem of a multi-legged robot, we could make about 50% reduction of the number of generations to obtain a target state of the problem. In Murata & Aoki (2007), the same proposed crossover are used for controling multiple agents who convey several loads to a goal. The proposed crossover is effective to decrease the number of actions to attain the objective. In this chapter, we concentrate on showing a QDSEGA with our crossover for controling multi-legged robot.
In the previous study (Ito & Matsuno, 2002), the target of a multi-legged robot was fixed. That is the target does not move in their computer simulation. In this chapter, we try to move the target, and modify the learning algorithm of QDSEGA to follow the moving target. Simulation results show that our proposed method can control the multi-legged robot to follow the moving target more than the conventional method.

2. QDSEGA

In this section, we briefly explain the outline of QDSEGA (Ito & Matsuno, 2002). QDSEGA has two dynamics. One is a learning dynamics based on Q-learning and the other is a structural dynamics based on Genetic Algorithm. Fig. 1 shows the outline of QDSEGA. In QDSEGA, each action is represented by an individual of a genetic algorithm. According to actions defined by a set of individuals, action-state space called Q-table is created. Q-learning is applied to the created Q-table. Then the learned Q-table is evaluated through experiments. A fitness value for each action is assigned according to Q-table. After that, each individual (i.e., each action) is modified through genetic operations such as crossover and mutation. We show some details in these steps in the following subsections, and show our proposed method in the next section.

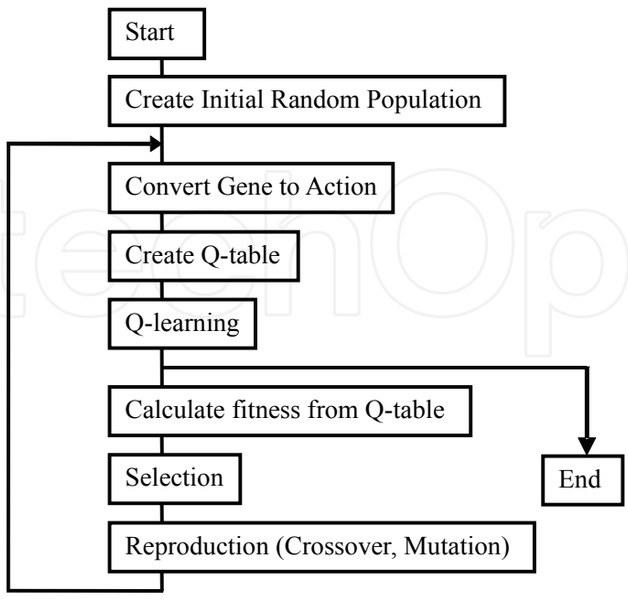


Figure 1. Outline of QDSEGA (Ito & Matsuno, 2002)

2.1 Action Encoding

Each individual expresses the selectable action on the learning dynamics. It means that a set of individuals is selected by genetic operations, and a learning dynamics is applied to the subset. After the evaluation of the subset of actions, a new subset is restructured by genetic operations.

2.2 Q-Table

An action-state space called Q-table is created from the set of individuals. When several individuals are the same code, that is, they are the same action, only one action is used in the action-state space in order to avoid the redundancy of actions. Fig. 2 shows this avoidance process.

2.3 Learning Dynamics

In QDSEGA, the conventional Q-learning (Watkins & Dayan, 1992) is employed as a learning dynamics. The dynamics of Q-learning are written as follows:

$$Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha\{r(s,a) + \gamma \max_{a'} Q(s',a')\}, \tag{1}$$

where $Q(s,a)$ is a Q-value of the state s and the action a, r is the reward, α is the learning rate, and γ is the discount rate.

2.4 Fitness

The fitness $fit(a_i)$ for each action is calculated by the following equation:

$$fit(a_i) = fit_Q(a_i) + k_f \cdot fit_u(a_i), \tag{2}$$

where $fit_Q(a_i)$ is a fitness value for action a_i calculated from Q-table, $fit_u(a_i)$ is a fitness value for action a_i calculated from the frequency of use, and k_f is a non-negative constant value to determine the ratio of $fit_Q(a_i)$ and $fit_u(a_i)$. We show the detail explanation of these factors in this subsection.

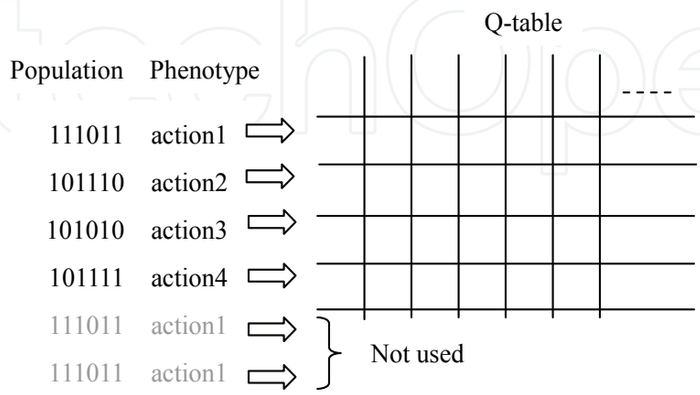


Figure 2. Q-table created from a set of individuals (Ito & Matsuno, 2002)

The fitness of Q-table $fit_Q(a_i)$ is calculated from Q-values in the current Q-table. In order to calculate $fit_Q(a_i)$ for each action a_i the following normalization is taken place in advance as for the Q-values in the current Q-table.
First, calculate the maximum and minimum value of each state as follows:

$$V_{\max}(s) = \max_{a'}(Q(s,a')) , \tag{3}$$

$$V_{\min}(s) = \min_{a'}(Q(s,a')) . \tag{4}$$

Then $Q'(s,a)$ of the normalized Q-table is given as follows:

If $Q(s,a) \geq 0$ then

$$Q'(s,a) = \frac{1-p}{V_{\max}(s)} Q(s,a) + p , \tag{5}$$

else $Q(s,a) < 0$ then

$$Q'(s,a) = -\frac{p}{V_{\min}(s)} Q(s,a) + p , \tag{6}$$

where p is a constant value which means the ratio of reward to penalty. After this normalization process, we fix the action a_i and sort $Q'(s,a_i)$ according to their value from high to low for all states. We define the sorted $Q'(s,a_i)$ as $Q'_s(s,a_i)$, and $Q'_s(1,a_i)$ means the maximum value of $Q'(s,a_i)$, and $Q'_s(N_s,a_i)$ means the minimum value of $Q'(s,a_i)$, where N_s is the size of states. Using the normalized and sorted Q-value $Q'_s(s,a_i)$, the fitness of action a_i is calculated as follows:

$$fit_Q(a_i) = \sum_{j=1}^{Ns} (w_j \frac{\sum_{k=1}^{j} Q'_s(k,a_i)}{j}) , \tag{7}$$

where w_j is a weight which decides the ratio of special actions to general actions.
The fitness of frequency of use $fit_u(a_i)$ is introduced to save important actions. That fitness is defined as follows:

$$fit_u(a_i) = N_u(a_i) / \sum_{j=1}^{N_a} N_u(a_j) , \tag{8}$$

where N_a is the number of all actions of one generation and $N_u(a_i)$ is the number of times which a_i was used for in the Q-learning of this generation. Important actions are used frequently. Therefore the actions with high fitness value of $fit_u(a_i)$ are preserved by this fitness value.

2.5 Genetic Algorithms

The paper that proposed QDSEGA (Ito & Matsuno, 2002) says "the method of the selection and reproduction is not main subject so the conventional method is used (in this paper)." They employed a crossover that exchanges randomly selected bits between the parent

individuals according to the crossover probability P_c. They mutated each bit according to the mutation probability P_m. They did not replace parents individuals with offspring. Therefore the number of individuals is increased by the genetic operations. As for the elite preserving strategy, they preserve 30% individuals with the highest fitness value.

3. Proposed Method

We propose a crossover operation for the multi-legged robot control problem (MRC problem). As shown in Subsection 2.5, QDSEGA did not try to propose any specific genetic operations for MRC problems. We propose a neighboring crossover for QDSEGA for MRC problems. In order to show our crossover which is tailored to MRC problems, we explain a coding method of a multi-legged robot and Q-table used in this chapter first.

3.1 Coding of Individuals

Fig. 3 shows the multi-legged robot and its encoding. We control a robot which has 12 legs. Each leg has four states as in Table 1. As shown in Fig. 3, each individual shows a set of legs' positions of 12 legs. Therefore each individual indicates one state of this robot of the 4^{12} positions. Since we generate only the prespecified number of individuals in genetic algorithms, the Q-table includes a subset of 4^{12} states with respect to legs' positions.

3.2 Q-Table for 12-Legged Robot

Q-table represents a subset of the action-state space to control a 12-legged robot. In QDSEGA, individuals are used for the both of actions and states. Fig. 4 shows an example of a Q-table created by *n* individuals. In this problem, each individual is used twice in the Q-table. "Current state" indicates the current position of 12 legs. "Action" indicates a next position of 12 legs after a certain action. For example, if S1 is the current position of 12 legs and A3 is selected as an action, the 12-legged robot moves its 12 legs to become the next position as A3. If the action A3 causes the robot to lose balance, the trial terminates. In our simulation, we assume that the 12-legged robot can change its legs' position smoothly.

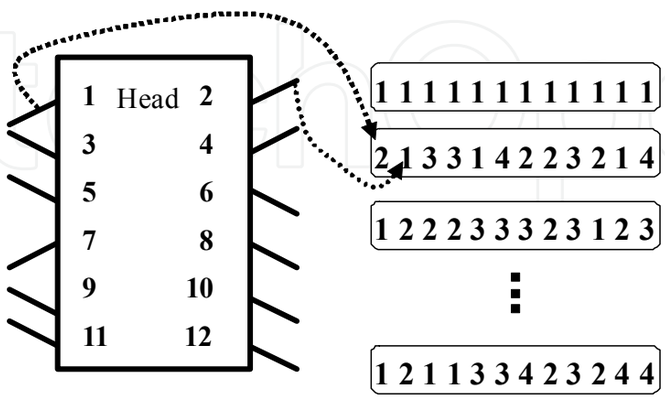


Figure 3. 12 Legs and representation in an individual

State Number	State
1	Forward and grounded
2	Backward and grounded
3	Forward and ungrounded
4	Backward and ungrounded

Table 1. Leg's position

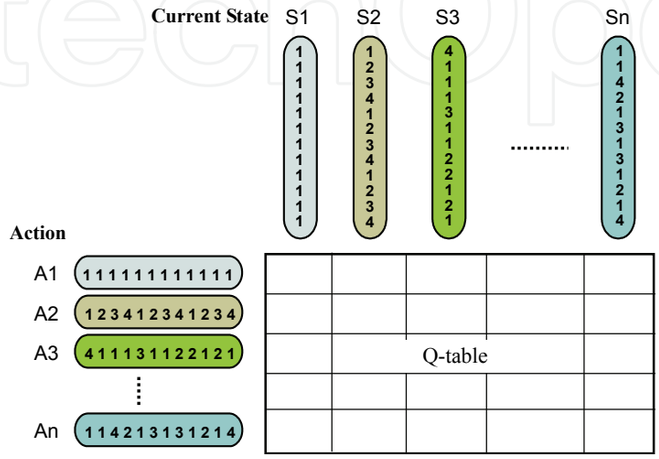


Figure 4. An example of Q-table created by n individuals

3.3 Neighboring Crossover

The crossover employed in QDSEGA (Ito & Matsuno, 2002) causes drastic change of legs' position because randomly selected bits are changed between two individuals. If the change of legs' position is so drastic, it may easily happen that the 12-legged robot loses its balance. In order to avoid losing the balance, we propose a crossover between similar parent individuals. We define the similarity by the number of the same genes in the same locus of a chromosome. The threshold for the similarity between two parents is denoted by k_{sim} in this chapter. Thus, the crossover is applied among similar individuals more than k_{sim}.

This kind of the restriction for the crossover has been proposed in the research area of distributed genetic algorithms. Researches on DGAs (Distributed Genetic Algorithm) can be categorized into two areas: coarse-grained genetic algorithms (Tanese, 1989; Belding, 1995) and fine-grained genetic algorithms (Mandelick & Spiessens, 1989; Muhlenbein *et al.*, 1991; Murata *et al.*, 2000). In the coarse-grained GAs, a population, that is ordinarily a single, is divided into several subpopulations. Each of these subpopulations is individually governed by genetic operations such as crossover and mutation, and subpopulations communicate each other periodically. Algorithms in this type are called the island model because each subpopulation can be regarded as an island. On the other hand, several individuals are locally governed by genetic operations in fine-grained GAs. In a fine-grained GA, each individual exists in a cell, and genetic operations are applied to an individual with individuals in neighboring cells. The DGAs are known to have an advantage to keep the variety of individuals during the execution of an algorithm, and avoid converging prematurely.

While we don't define any solution space such as cells or islands in our proposed crossover, our restriction may have the same effect of keeping variety in a population and attain the effective search for the sequence of states of 12 legs.

3.4 Modified Reward for Moving Target

In the previous study, the target of the MRC problem was fixed. That is, the target stays at the same place. We consider a moving target as a variant problem of MRC problems. In order to let the multi-legged robot follow a moving target, we modify the reward $r(s,a)$ at time t in Equation (1) as follows:

If $-45[\text{deg}] \le L_z < -15[\text{deg}]$ and $d(t-1) - d(t) \ge 0$ then

$$r(s,a) = 50 \cdot (d(t-1) - d(t)), \tag{9}$$

If $-45[\text{deg}] \le L_z < -15[\text{deg}]$ and $d(t-1) - d(t) < 0$ then

$$r(s,a) = 200 \cdot (d(t-1) - d(t)), \tag{10}$$

If $15[\text{deg}] < L_z \le 45[\text{deg}]$ and $d(t-1) - d(t) \ge 0$ then

$$r(s,a) = 50 \cdot (d(t-1) - d(t)), \tag{11}$$

If $15[\text{deg}] < L_z \le 45[\text{deg}]$ and $d(t-1) - d(t) < 0$ then

$$r(s,a) = 200 \cdot (d(t-1) - d(t)), \tag{12}$$

If $-15[\text{deg}] \le L_z \le 15[\text{deg}]$ then

$$r(s,a) = 100 \cdot (d(t-1) - d(t)), \tag{13}$$

Else If($L_z < -45[\text{deg}]$ or $45[\text{deg}] < L_z$ or stuck) then

$$r(s,a) = -100. \tag{14}$$

In the above equations, L_z is the direction of the target in degrees (not in radian), and $d(t)$ is the distance from the head of the robot to the target at the time *t* (see Fig. 5 for the detail). L_z is zero [deg] if the target locates just ahead of the robot.

4. Multi-Legged Robot Control

4.1 Task

Fig. 5 shows the initial position of the 12-legged robot and the location of the target. At time zero, the head of the robot locates in (0,0) in the *x*-*y* plane. And the initial state of each leg of the robot is 1 in Table 1. We allow the robot to move 100 steps toward the target. If the robot can acquire the most efficient movement of 12 legs, it can move 0.4 for each step. Therefore the maximum gain toward the target is 40.

When we try to acquire the legs' movement for the moving target, the robot first learns the legs' movement for the fixed target over 50 generation. After that we put the target at $(100\sqrt{2}, -100\sqrt{2})$ in the *x*-*y* plane, and move it to $(100\sqrt{2}, 100\sqrt{2})$ by 1 [deg] in each step

on the circle whose center is (0,0) and the radius is 200. After 90 steps, the target reaches the point ($100\sqrt{2}$, $100\sqrt{2}$) and stay there until the end of 100 steps in each trial.

4.2 Simulation Model for Multi-Legged Robot

As Ito & Matsuno (2002) did, we also employed Minimal Simulation Model that was proposed by Svinin *et al.* (2001). This model is very simple so that the calculation cost becomes very low.

Fig. 6 shows a multi-legged robot. Each leg has two joints and has four motions. (Move forward and lift down, Move back and lift down, Move forward and lift up, Move back and lift up). The position of the robot can be calculated as follows:

$$f^r_{drv} = u(n^r_{12} - n^r_{21}), \tag{15}$$

$$f^r_{res} = v(n^r_{11} + n^r_{22}), \tag{16}$$

$$\text{if } \left|f^r_{drv}\right| \le f^r_{res} \text{ then } F^r = 0, \tag{17}$$

$$\text{else if } \left|f^r_{drv}\right| > f^r_{res} \text{ then } F^r = f^r_{drv} - f^r_{res}, \tag{18}$$

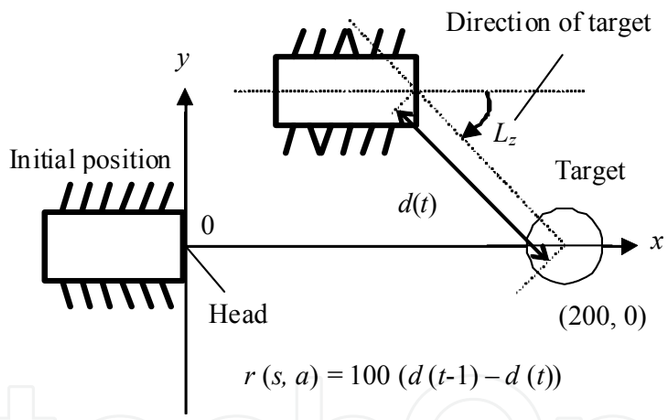


Figure 5. Initial position and the target (Ito & Matsuno, 2002)

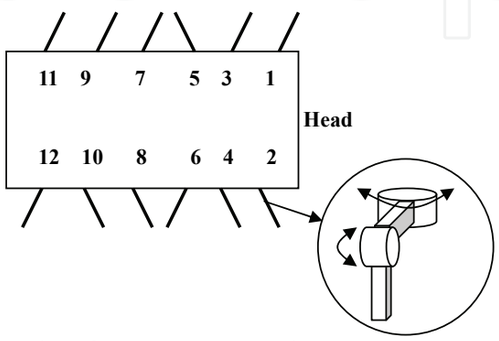


Figure 6. Multi-legged robot (Ito & Matsuno, 2002)

$$\text{else } f_{drv}^r < -f_{res}^r \text{ then } F^r = f_{drv}^r + f_{res}^r , \tag{19}$$

$$F = F^l + F^r , \tag{20}$$

$$M = F^r - F^l , \tag{21}$$

$$\Delta u = c_u F , \tag{22}$$

$$\Delta\theta = c_\theta M , \tag{23}$$

$$x(t+1) = x(t) + \Delta u \cos\theta(t) , \tag{24}$$

$$y(t+1) = y(t) + \Delta u \sin\theta(t) , \tag{25}$$

$$\theta(t+1) = \theta(t) + \Delta\theta , \tag{26}$$

where $f_{drv}{}^r$ and $f_{res}{}^r$ indicate driving force and resistance force, and $n_{ij}{}^r$ is the number of legs on the right side changing their configuration from the state i at the moment t to the state j at the moment $t + 1$. Similarly, we define n_{ij}^l for the left side. Let F and M be the force and the moment of the robot that act to the environment, respectively. And x, y, θ mean the position and orientation of the robot. Details are written in Svirin *et al.* (2001).

5. Simulation

5.1 Parameter Settings

We specified the parameter settings in the Q-learning, the genetic algorithm, and the multi-legged robot simulation model as follows:

[Q-learning]

The number of trials for each Q-table: 3 000 trials,

The number of steps in each trial: 100 steps,

Learning rate in (1): $\alpha = 0.7$,

Discounting rate in (1): $\gamma = 0.7$,

Temperature of Bolzmann distribution: $T = 150 \times e^{(-0.002 \times trial)+1}$.

[Genetic Algorithm]

The number of individuals: 50,

The number of generations: 200,

Crossover probability: $P_c = 0.5$,

Similarity for the crossover: $k_{\text{sim}} = 4$,

Mutation probability: $P_m = 0.2$,

Weight value in (2): $k_f = 200$,

Ratio of reward to penalty in (5), (6): $p = 0.01$,

Weight values in (7): $w_1 = 0.6$, $w_i = 0 \quad (i = 2,..., N_s - 1)$, $w_{N_s} = 0.4$.

[Multi-Legged Robot Simulator]

$u = 2.0$, $v = 1.0$, $c_\theta = 0.05$, $c_u = 0.05$.

5.2 Effect of Neighboring Crossover

We applied the conventional method (Ito & Matsuno, 2002) and the proposed method with the neighboring crossover in Subsection 3.3 100 times. Fig. 7 shows the average "Gain" over the generation. We defined "Gain" as the gaining distance of the robot toward the target. Thus, the maximum gain was 40 in this problem. From Fig. 7, we can see that the proposed crossover clearly enhanced the performance of QDSEGA.

Table 2 shows that the average number of generations in which the target gain was attained over 100 experiments. We specified the target gain as 38 in these experiments. We count a computational experiment as successful one when the robot gains over 38. We also show the standard deviation of the generations and the number of successful experiments. "Successful experiments" mean that the number of experiments that can attain the target gain. "Ito 2002" shows the conventional method in (Ito & Matsuno, 2002), and "Neighboring Crossover" is the method with the proposed neighboring crossover in Subsection 3.3. From Table 2, we can see that the number of generations was drastically reduced by introducing the neighboring crossover into QDSEGA. As shown in "# of successful experiments," the conventional method could not attain the maximum gain in all the 100 experiments while "Neighboring Crossover" attained the maximum gain in all the experiments. Through this result we can see the strong effect of the proposed neighboring crossover in QDSEGA.

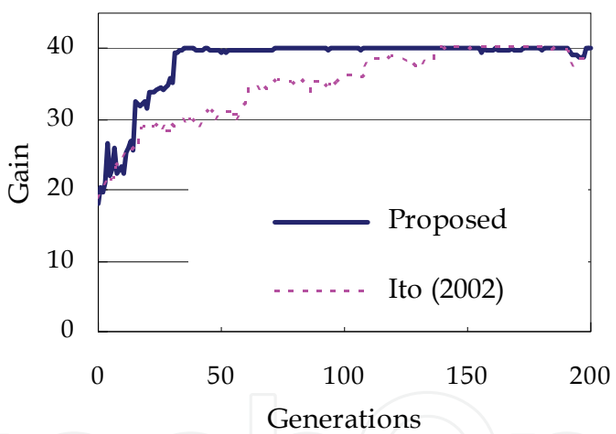


Figure 7. Average gain over the generation

	Ito (2002)	Neighboring Crossover
Average Generation	108.5	56.2
Standard Deviation	37.7	19.3
# of successful experiments	85/100	100/100

Table 2. Average generation of attaining the maximum gain over 100 experiments

Conventional	Without adjustment		
Target position	Left	Ahead	Right
Ratio (%)	21.7	50.7	27.5

Conventional	With adjustment		
Target position	Left	Ahead	Right
Ratio (%)	17.8	63.4	17.8

Table 3. Effect on the conventional method for moving target

Proposed	Without adjustment		
Target position	Left	Ahead	Right
Ratio (%)	21.0	50.6	28.2

Proposed	With adjustment		
Target position	Left	Ahead	Right
Ratio (%)	21.9	63.3	14.8

Table 4. Effect on the conventional method for moving target

5.3 Effect of Modified Reward for Moving Target

Tables 3 and 4 show the results on the MRC problem with the moving target. We applied our modified reward to QDSEGA (Ito & Matsuno, 2002) and QDSEGA with the proposed crossover. In these tables, "Target position" indicates that the ratio of target positions from the robot. That is, when the target locates ahead of the robot within -15 [deg] to 15 [deg], the number of "Ahead" is incremented. From this table, we can see that the ratio of "Ahead" was increased by the introduction of the adjustment of rewards in Q-learning in the both tables. Through the proposed modification of the reward we could improve the performance of the multi-legged robot to follow the moving target.

6. Conclusion

In this chapter, we proposed the neighboring crossover that improves the performance of QDSEGA for the multi-legged robot control problems. By introducing the neighboring crossover, we could reduce the number of generations to attain the target gain in the problem. We also show the fine effect of the modified reward for the problems with the moving target. When the target is moving in the problem, the robot needs to learn how to adjust its position toward the target. That is, it should learn how to adjust its direction toward the target, and how to convey its body toward the target. By adjusting the reward, we could improve the control of the multi-legged robot.

7. Acknowledgments

This work was partially supported by the MEXT, Japan under Collaboration with Local Communities Project for Private Universities starting 2005.

8. References

Belding, T.C. (1995). The distributed genetic algorithm revisited, *Proc. of 6th International Conference on Genetic Algorithms*, pp. 114-121.

Holland, J.H. (1986). Escaping brittleness: the possibilities of general purpose learning algorithms applied to parallel rulebased system. *Machine Learning II*, pp. 593-623.

Ito, K. & Matsuno, F. (2002). A study of reinforcement learning for the robot with many degrees of freedom –Acquisition of locamotion patterns for multi legged robot-, *Proc. of IEEE International Conference on Robotics and Automation*, pp. 392-397.

Mandelick, B. & Spiessens, P. (1989). Fine-grained parallel genetic algorithms, *Proc. of 3rd International Conference on Genetic Algorithms*, pp. 428-433.

Muhlenbein, H. Schomisch, M. & Born, J. (1991). The parallel genetic algorithm as function optimizer, *Proc. of 4th Int'l Conf. on Genetic Algorithms*, pp. 271-278.

Murata, T. & Aoki, Y. (2007). Developing Control Table for Multiple Agents Using GA-Based Q-Learning With Neighboring Crossover, *Proc. of IEEE Congress on Evolutionary Computation 2007*, pp. 1462-1467.

Murata, T., Ishibuchi, H. & Gen, M. (2000). Cellular genetic local search for multi-objective optimization, *Proc. of Genetic and Evolutionary Computation Conference 2000*, pp. 307-314.

Murata, T. & Yamaguchi M. (2005). Neighboring Crossover to Improve GA-Based Q-Learning Method for Multi-Legged Robot Control, *Proc. of Genetic and Evolutionary Computation 2005*, pp. 145-146.

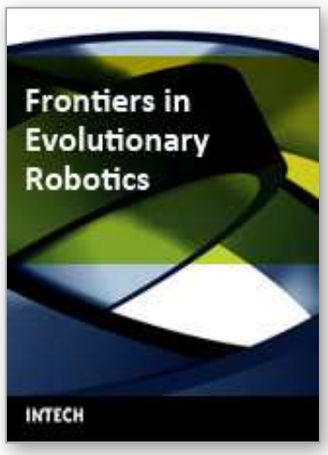
Sutton, R.S. (1988). *Reinforcement Learning: An Introduction*. The MIT Press.

Svinin, M., Ushio, S., Yamada, K. & Ueda, K. (2001). An evolutionary approach to decentralized reinforcement learning for walking robots, *Proc. of 6th Int. Symp. on Artificial life and Robotics*, pp. 176-179.

Tanese, R. (1989). Distributed genetic algorithms, *Proc. of 3rd Int'l Conf. on Genetic Algorithms*, pp. 434-439.

Watkins, C.J.C.H. & Dayan, P. (1992). Technical note Q-learning, *Machine Learning*, Vol. 8, pp. 279-292.

Yamada, K., Ohkura, K., Svinin, M. & Ueda, K. (2001). Adaptive segmentation of the state space based on bayesian discrimination in reinforcement learning, *Proc. of 6th International Symposium on Artificial life and Robotics*, pp. 168–171.



Frontiers in Evolutionary Robotics

Edited by Hitoshi Iba

ISBN 978-3-902613-19-6
Hard cover, 596 pages
Publisher I-Tech Education and Publishing
Published online 01, April, 2008
Published in print edition April, 2008

This book presented techniques and experimental results which have been pursued for the purpose of evolutionary robotics. Evolutionary robotics is a new method for the automatic creation of autonomous robots. When executing tasks by autonomous robots, we can make the robot learn what to do so as to complete the task from interactions with its environment, but not manually pre-program for all situations. Many researchers have been studying the techniques for evolutionary robotics by using Evolutionary Computation (EC), such as Genetic Algorithms (GA) or Genetic Programming (GP). Their goal is to clarify the applicability of the evolutionary approach to the real-robot learning, especially, in view of the adaptive robot behavior as well as the robustness to noisy and dynamic environments. For this purpose, authors in this book explain a variety of real robots in different fields. For instance, in a multi-robot system, several robots simultaneously work to achieve a common goal via interaction; their behaviors can only emerge as a result of evolution and interaction. How to learn such behaviors is a central issue of Distributed Artificial Intelligence (DAI), which has recently attracted much attention. This book addresses the issue in the context of a multi-robot system, in which multiple robots are evolved using EC to solve a cooperative task. Since directly using EC to generate a program of complex behaviors is often very difficult, a number of extensions to basic EC are proposed in this book so as to solve these control problems of the robot.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820

Fax: +385 (51) 686 166
www.intechopen.com

Fax: +86-21-62489821

IntechOpen