

# A method to embed human knowledge to reinforcement learning method\*

Tatsuo Unemi<sup>†</sup>

Dept. of Information Systems Science, Soka University, Japan

unemi@iss.soka.ac.jp

October 5, 2000

## Abstract

Reinforcement learning is a framework to learn from delayed reward and punishment for a model of both animal and robot learning. To make it more practical to design an intelligent machine, it would be better to be able to combine with human knowledge. This paper presents a method to introduce such a knowledge into a reinforcement learning system by embedding it as an intrinsic behavior, just like as animals acquired in the genetic information. Through two types of experimental simulation on mobile robot navigation, we show the effect of combination of episode-based reinforcement learning and intrinsic avoidance behavior. The result clarify that it is possible to improve the performance drastically by introducing relatively simple knowledge.

## 1 Introduction

Reinforcement learning (RL) is one of the key functions of animal learning in which the learner discovers better strategy to avoid harmful stimuli and to obtain pleasant sensation through its own experience. At the same time, it is also a suitable model for intelligent machines to adapt against unexpectable environment. A number of researchers have proposed different types of basic algorithms to realize this type of learning from delayed reward/punishment by machines, and a lot of orientations to extend this framework have also been proposed to apply them to more complicated practical problems.

One of the key issues for successful application of RL framework is to design an intrinsic behavior to try promising actions at the start point of learning, because RL methods are based on the iteration of *trial and error*. Learning does never progress if the trials always fail. The task domains in any researches and applications of RL using random actions have been designed or modified so that the learner has a chance of enough probability to try better action even if it acts randomly.

Natural animals never act randomly in their infant period but behave in some manner of useful trial to organize sensor-motor coordination necessary for survival. This paper focuses on a case study to certify the effect of a hand-coded intrinsic behavior through a computer

---

\*A part of this paper has appeared in the proceeding of the sixth international conference on intelligent autonomous systems (IAS-6) edited by E. Pagello *et al* held in Venice 25-27 July, 2000 as “Scaling up reinforcement learning with human knowledge as an intrinsic behavior.”

<sup>†</sup>A part of this work was done at AI Lab., IFI, University of Zurich, Switzerland during the author was staying as a visiting professor.

simulation of relatively simple task in which it is difficult to learn only by random actions. The basic idea was already proposed in our previous work[1], in which we examined combination of *instance-based* RL and Fuzzy rules. The new points of this paper are to use more powerful RL algorithm and weaker intrinsic strategy, and to try experiments on more difficult task for more precise comparison.

The following sections describe episode-based RL method that we employ as a basic algorithm to combine with an intrinsic behavior, our experiments on the computer simulation of path finding by a mobile robot with local sensors on two different types of mobility, and then its results and remarks.

## 2 Episode-based reinforcement learning method

We employ a modified version of *episode-based reinforcement learning method* (EBRL)[2] as the basic algorithm to which we add an intrinsic behavior, because this method makes it easy to measure how strongly the learner believes that the action leads better results. Additionally, it is capable to be applied to a non-Markovian domain, that is, context sensitive and/or with perceptual aliasing. This is not a necessary condition for this research but it is important to prepare somewhat complicated task to clarify the usefulness of the proposed method.

The main loop of performance is iteration of sensing, deciding and executing an action, acquiring reward, and then learning, in just same manner of ordinary RL methods.

### 2.1 Memory

EBRL can be seen as a derivative of instance-based learning (IBL)[3] in terms of learning by memorizing input/output pairs without any reformation. In IBL, the memory is a set of input/output pairs, but it is a set of their sequences in EBRL. An *episode* is a sequence of *memory elements* each of which contains one sense-action pair with two additional parameters, *reliability* and *utility*. The learner has a memory storage of fixed capacity to memorize episodes, and it forgets a memory element of the lowest reliability when there is no more room to store new experience. Reliability of each memory element decreases in each simulation step by multiplying a positive constant less than 1 named *time discount rate*. Reliability is modified also in the process of learning described later.

### 2.2 Recall table

To catch up the current context in the learner's mind, the learner keeps a number of pointers for memory elements which seems similar to the current situation. The table containing these pointers is named *recall table*. Each element of the recall table is a pair of pointer and *weight* which indicates similarity between the memorized episode and the current situation. The learner replaces each element by the pointer of next memory element following along the forward link in the memorized episode, and modifies the weight by combination with the similarity between the sensory data in new element and current sensory input by:

$$\Delta W_i = \alpha \cdot (S(M_i, s) - W_i) \quad (1)$$

where  $W_i$  is the weight of  $i$ th element,  $S(M_i, s)$  is similarity between the current input  $s$  and the memorized data  $M_i$  pointed from  $i$ th element, and  $\alpha$  is a constance of  $(0, 1)$ . Additionally to renewed elements, the other memory elements similar to the current sensory input are also candidates of recall table elements. Assigning the value of similarity as the weight of newly

recalled memory element,  $N_r$  elements of the candidates with relatively larger weights remain in the recall table, where  $N_r$  is the capacity of the table.

Similarity  $S(\mathbf{x}, \mathbf{y})$  between vectors  $\mathbf{x}$  and  $\mathbf{y}$  of real numbers is calculated by the following equation.

$$S(\mathbf{x}, \mathbf{y}) = 1 - \sqrt{\frac{1}{2n} \sum_{i=1}^n \frac{(x_i - y_i)^2}{V_i}} \quad (2)$$

where  $n$  is the number of elements in a vector,  $x_i$  and  $y_i$  are  $i$ th element of  $\mathbf{x}$  and  $\mathbf{y}$  respectively, and  $V_i$  is the variance of  $i$ th element among all of memorized data. This equation is a simplified version of the original one in [2].

It should be modified into any other form if sensory inputs include different types of modality, such as distance sensors and touch sensors. Here we employ simple weighed sum of similarity values among different modalities, as the revised version of the definition.

$$S(\mathbf{x}, \mathbf{y}) = \frac{\sum_j w_j S'(x_j, y_j)}{\sum_j w_j} \quad (3)$$

$$S'(x_j, y_j) = 1 - \sqrt{\frac{1}{2n_j} \sum_{i=1}^{n_j} \frac{(x_j^i - y_j^i)^2}{V_j^i}} \quad (4)$$

where  $\mathbf{x}$  and  $\mathbf{y}$  are sets of vectors,  $n_j$  is the number of elements of vector  $x_j$ , and  $x_j^i$  is the  $i$ th element of vector  $x_j$ .

## 2.3 Policy

The agent should execute the action that seems to lead relatively high reward, by selecting one element from the recall table of which the following measure is maximum.

$$\min(W_i, U_i) \quad (5)$$

where  $U_i$  is the utility associated with the memory element pointed by the  $i$ th recall table element.

If the agent cannot find any element of enough value of the above measure, it takes random or intrinsic behavior. Of course, it should sometimes do a type of exploration behavior ignoring the above decision process. We call the probability of ignorance the *exploration rate* here.

## 2.4 Learning

The agent simply memorizes the experience as described above. To learn from delayed reward, the utility value associated with each memory element is modified by the following equation in every step.

$$\Delta U_j = r_t \cdot \gamma^{j-t} \quad (6)$$

where  $U_j$  is the utility value of the memory element memorized at time  $j$ ,  $r_t$  is the current reward,  $t$  is the current time, and  $\gamma$  is a constant of  $(0, 1)$  that is called *discount rate*. This operation is done by following the backward links in the episode when the agent acquires non-zero reward.

After doing an action expecting some amount of reward, the reliability of memory element that the agent referred for decision should be modified according to the difference between real and expected reward values. We implement this principle by the following equation.

$$\Delta R_i = \begin{cases} \rho \cdot r_t \cdot (1 - R_i) & r_t > 0 \\ \rho \cdot r_t \cdot R_i & r_t < 0 \end{cases} \quad (7)$$

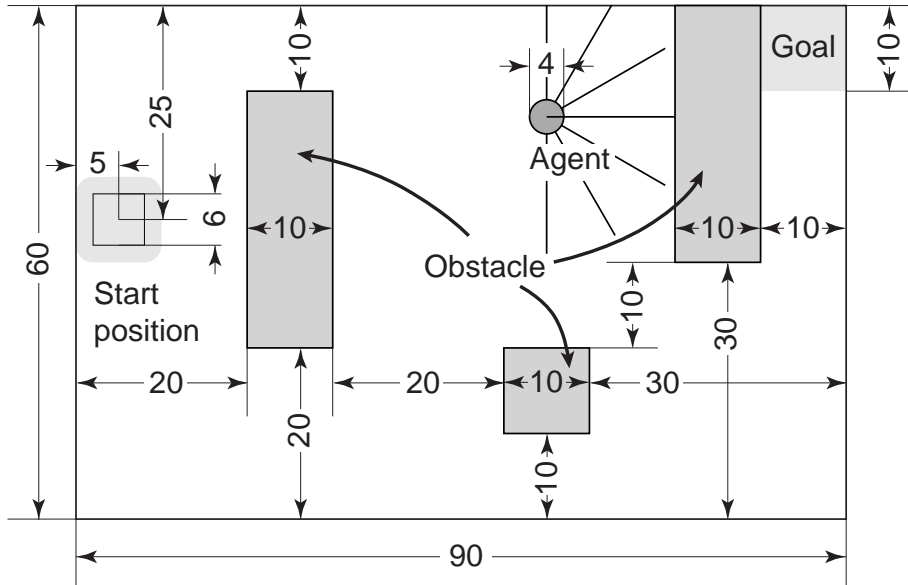


Figure 1: Field of example task.

where  $R_i$  is reliability of memory element and  $\rho$  is a constant of  $(0, 1)$  named *reliability modification rate*. This operation is also done by following the backward links starting from the memory element employed to decide the last action.

### 3 Experiments

We examined the effects of introducing an intrinsic behavior through two types of tasks of mobile robot navigation in a continuous field rather than a grid world, because it is not easy to be accomplished only by an intrinsic reactive behavior programmed by human hands. We call the robot *agent* here.

Both types of example tasks use a same environment but in the first one we assume a robot with two wheels independently controlled, and in the second one we assume a robot floating on a slippery floor accelerated by jet.

#### 3.1 Environment

The field where the agent moves around is a room surrounded by walls and includes some rectangular obstacles as shown in Figure 1, a continuous version of the testbed by Sutton[4]. The task of agent is to move from the start position to the goal position as fast as possible. It has neither a map of the room nor a detector of the global position, but only local distance sensors. The agent has to cope with a problem of perceptual aliasing, because the range of the sensors are not wide enough to distinguish the position in the room.

The start position is at 5 units from left edge and 25 units from upper edge of the field, and direction of the agent at the start point is right. We introduce  $\pm 3$  units and  $\pm 15^\circ$  as a fluctuation of start positioning to make it more realistic.

The goal area is a square of 10 units at the upper right corner of the field. The agent acquires a reward valued 1 when its center point reaches into this area.

## 3.2 Sensing

The agent has a number of distance detectors each of which measures the distance from the surface of agent's body to a wall and an obstacle. The range of sensing is 14 and the distance is encoded into an integer from 0 to 255, where it is 0 when nothing detected and 255 if it touches something. To make it more realistic, the encoded value includes a noise of pseudo Gaussian distribution<sup>1</sup> where the standard deviation is 5% of the maximum value, that is,  $255 \times 0.05 = 12.75$ .

**Two wheels robot** has seven sensors evenly arranged in  $180^\circ$  of its front, that is, the angle between each neighboring sensors is  $180^\circ / (7 - 1) = 30^\circ$ .

**Floating robot** has eight sensors around it, that is, the angle between each neighboring sensors is  $360^\circ / 8 = 45^\circ$ . In addition to these distance sensors, it can also detect the velocity of both horizontal and vertical directions. We use equation (3) to calculate the similarity using 0.7 and 0.3 for the weights of distance sensors and velocity respectively, because they have different modality. This proportion was set through some preliminary trials.

## 3.3 Action

Actions of two types are quite different as follows.

**Two wheels robot** moves forward by 2 units or turns by 1 radian maximum in each step. The demand of action is indexed by a real number of  $(-1, 1)$  where  $-1$  means turning left, 0 means going straight ahead, and 1 means turning right. Intermediate values indicate the proportion between turn and move. For example,  $-0.4$  means to turn left by 0.4 radian then to move straight by 1.2 units. If the agent cannot move any more because of collision with an object, the rest portion of move is transferred into the angle of turn. Similarly to the sensing, the action also includes some amount of fluctuation by maximum 5% discount from demanded value in both turning angle and moving length.

**Floating robot** moves constantly by its own velocity until it bumps against a wall or an obstacle. The demand of action is the amount of acceleration for both horizontal and vertical direction represented by a pair of real numbers of  $(-1, 1)$ . The initial velocity is zero. We set the maximum value of acceleration as 0.2 units per square step. For example, the action demand  $(-0.2, 0.3)$  means that the horizontal factor of velocity decreases by 0.2 and the vertical factor of velocity increases by 0.3 before the move in current step.

## 3.4 Intrinsic behavior

It is obviously better for the mobile robot to have ability to avoid any useless behavior for seeking the optimal path. One of the useful strategy is to avoid collision against walls and obstacles which is relatively easy to implement into a robot with distance sensors, that is, the rule that move the agent apart from the obstacles.

---

<sup>1</sup> A random number of pseudo Gaussian distribution is generated using Box-Müller transformation in our simulation program. The value is  $x = \sqrt{-2 \log u_1} \cdot \sin 2\pi u_2$  where  $u_1$  and  $u_2$  are independent random numbers of  $(0, 1]$  from uniform distribution.

Table 1: Settings of learning parameters

memory size	$(T)1024, (F)4096$
recall table size $N_r$	7
$\alpha$	0.5
exploration rate	0.05
discount rate $\gamma$	0.99
time discount rate	0.99
reliability modification rate $\rho$	0.5

$(T)$  : for *two wheels robot*.  $(F)$  : for *floating robot*.

**Two wheels robot** can turn as its action. It is effective to have rules that if there is something at the left side then turn right, and if at the right side then turn left. To make the avoiding behavior smoother, we employ the following equation to decide the portion of turn.

$$a = \frac{\sum_{i=1}^{\lfloor n/2 \rfloor} i \cdot (s_i - s_{n+1-i})}{\sum_{i=1}^{\lfloor n/2 \rfloor} i} \quad (8)$$

where  $a$  is the value of action demand described above, and  $s_i$  is the sensing value of  $i$ th element.  $s_1$  is of the left most sensor and  $s_n$  is of the right most sensor.

**Floating robot** can change its velocity for arbitrary orientation. It is effective to apply a negative acceleration against an obstacle, but it should not be too sensitive because it might stop at around the center of broad space. We employ the following equation to determine the horizontal acceleration.

$$a_h = \begin{cases} \frac{b_h + \theta}{1 - \theta} & \text{if } b_h \leq -\theta \\ \frac{b_h - \theta}{1 - \theta} & \text{if } b_h \geq \theta \\ \text{not changed,} & \text{otherwise} \end{cases} \quad (9)$$

$$b_h = 0.5(s_l - s_r) + 0.25(s_{ul} + s_{dl} - s_{ur} - s_{dr}) \quad (10)$$

where  $a_h$  is the value of horizontal acceleration,  $\theta$  is a threshold value determining the sensitivity, and  $s$  is the sensing value scaled into  $[0, 1]$ . The suffix letters  $l, r, u, d$  of  $s$  indicates the direction of the sensor; left, right, upper, and lower respectively. The value of vertical acceleration is determined by the symmetrical equation with the above one. Here we assign 0.25 as the value of  $\theta$ .

This kind of reactive strategy might prevent from trying useful behavior. We should introduce some probability of random action to avoid this drawback. Here we set the probability to be 20% in both types of experiments.

The demand value of random action is produced using pseudo Gaussian distribution of which mean value is 0 and standard deviation is 0.5.

### 3.5 Learning parameters

We use settings for learning parameters as shown in Table 1. These values were tuned by hand through several times of preliminary experiments.

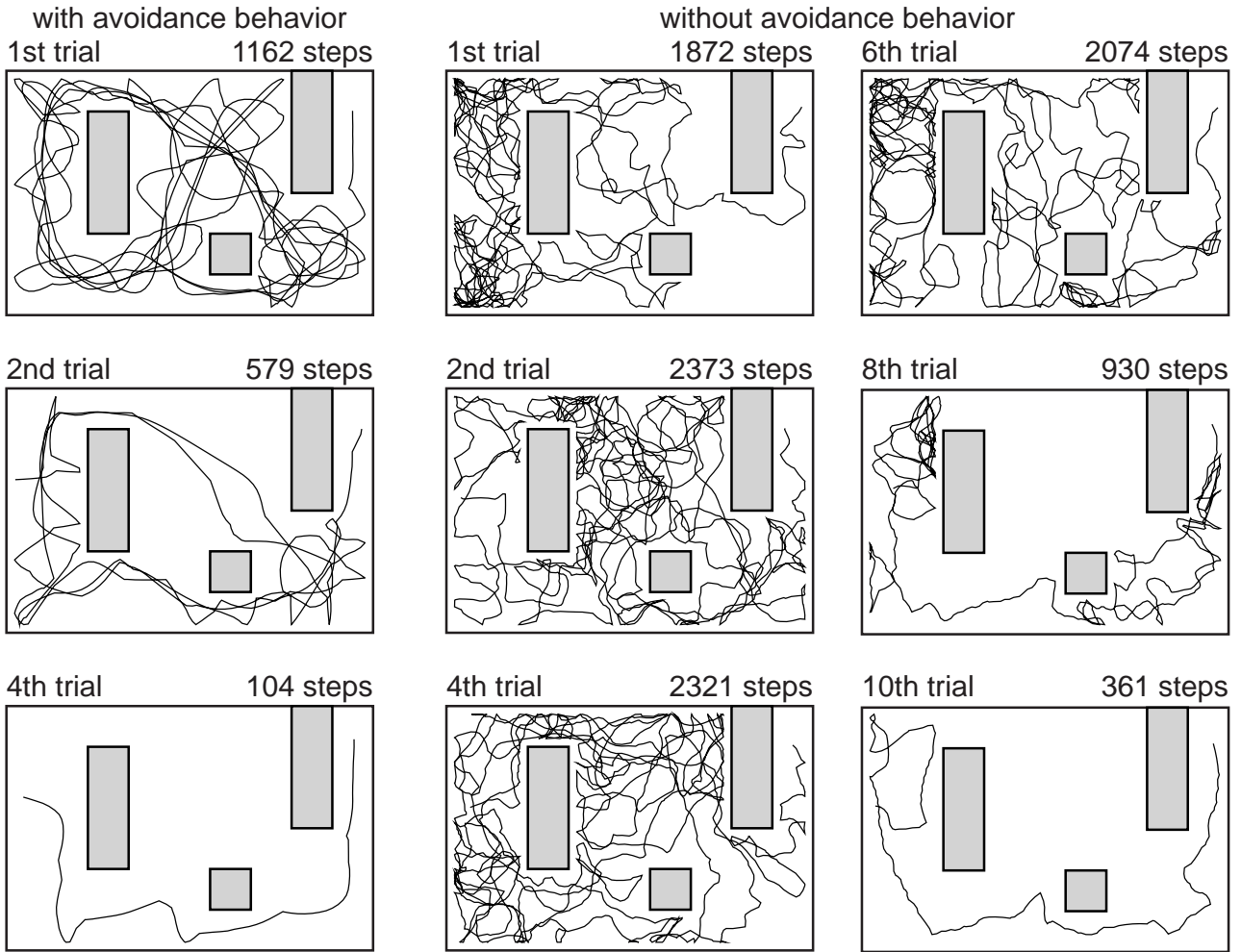


Figure 2: Typical examples of traces in successful cases of *two wheels robot*.

### 3.6 Results

To clarify the effects of intrinsic behavior, we tried 50 separated random number sequences for two cases, the case without avoidance behavior but only random walk, and the case with avoidance behavior.

**Two wheels robot** could achieve the goal without learning in 6,087 steps in average and the standard deviation was  $\pm 5,789$  over 49 cases<sup>2</sup> without avoidance behavior, and they were 956 in average and  $\pm 954$  in standard deviation over 50 cases with avoidance behavior. This means that the intrinsic avoidance behavior was quite effective to inhibit a type of useless actions.

The average number of steps per trial within 30,000 steps of learning can be referred as an evidence for learning performance. It was 3,844 ( $\pm 3,336$ ) without avoidance behavior, and was 283 ( $\pm 98$ ) with avoidance behavior. It could shrink the steps from the start position to the goal into 63% in the case without avoidance behavior, and into 30% in the case with avoidance behavior. This means that the learning was effective in both cases. It seems more effective with avoidance behavior if compared under same number of total execution steps, but this is obvious and unfair because learning works better with more times of trials.

<sup>2</sup> In one of 50 cases, it could not reach at the goal within 30,000 steps.

Figure 2 shows typical traces of trials in relatively successful cases. Figure 3 shows the cumulative reward of 50 cases. The number of cumulative reward at 30,000th step is less than 30 in 43 cases out of 50 cases without avoidance, though the number is 49 in the worst case with avoidance behavior.

**Floating robot** could achieve the goal without learning in 11,129 steps in average and the standard deviation was  $\pm 9,909$  over 50 cases without avoidance behavior, and they were 5,433 in average and  $\pm 3,977$  in standard deviation with avoidance behavior. This means that the intrinsic avoidance behavior was effective also in this task.

The average number of steps per trial within 100,000 steps of learning was 8,426 ( $\pm 2,630$ ) without avoidance behavior, and was 1,869 ( $\pm 699$ ) with avoidance behavior. It could shrink the steps from the start position to the goal into 76% in the case without avoidance behavior, and into 34% in the case with avoidance behavior. The learning was also effective in both cases. Figure 5 shows the cumulative reward of 50 cases.

It was also revealed that the optimal path could not be found in any of the above tasks and settings. EBRL potentially has ability to find the optimal path even if the problem includes perceptual aliasing, though not proven mathematically yet. However, the path that was found in our experimental simulation includes plenty of redundant behavior, which would be useful to find some landmark to change its action.

## 4 Conclusion

We examined an improvement of learning performance for a reinforcement learning by introducing a reaction for collision avoidance as an intrinsic behavior. Through our experiments by the computer simulation, it was certified that this method can be effective for somewhat difficult task, such as path finding by a mobile robot with local sensors.

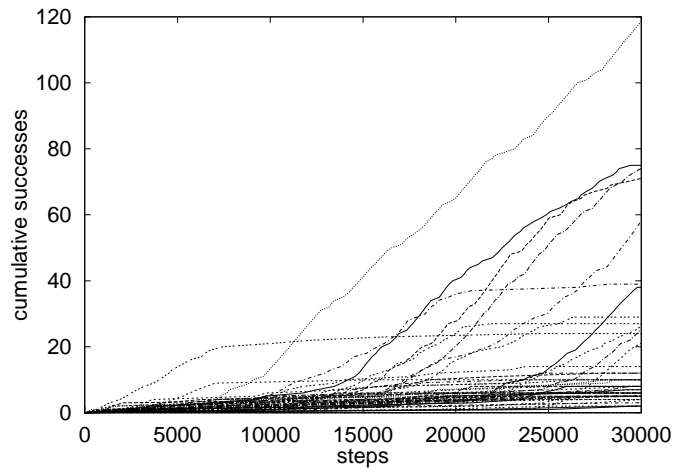
It was easy to introduce this type of strategy into EBRL, however we need to develop additional mechanism to measure the confidence of decision making for other types of RL algorithms such as look-up table, artificial neural network, classifier system, and so on. The variance of probabilities in the stochastic action selector might be useful to measure the confidence, if it uses this type of mechanism just like as roulette wheel action selection in Q-learning[5].

We applied the proposed method to two types of tasks, but both are still experimental example problems. Application to more practical task is also our future work. The proposed method could provide more effective profit for more complicated task on which human knows what types of actions are useless.

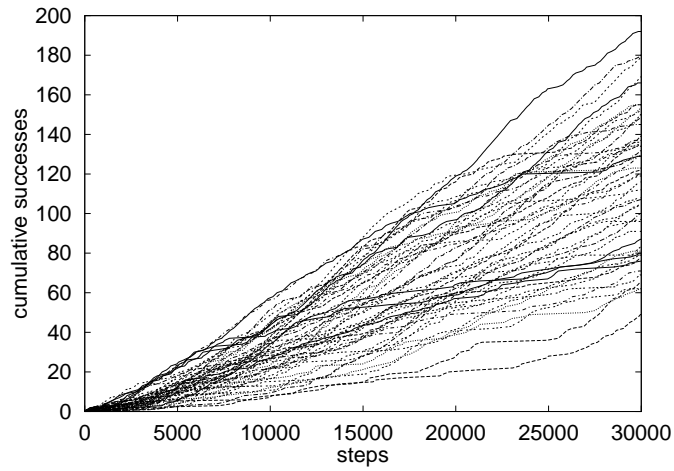
The optimization of the intrinsic strategy will be a target of an evolutionary approach, because natural animals also have acquired them through some billion years of evolutionary processes. In the combination method proposed here, the intrinsic knowledge is separated from the learning module, because we focused on a hand-coded knowledge from a standing point of engineering. The initial values of learning variables should also be set up appropriately as the intrinsic characteristics. These issues might be interesting from a view point of Artificial Life.

## References

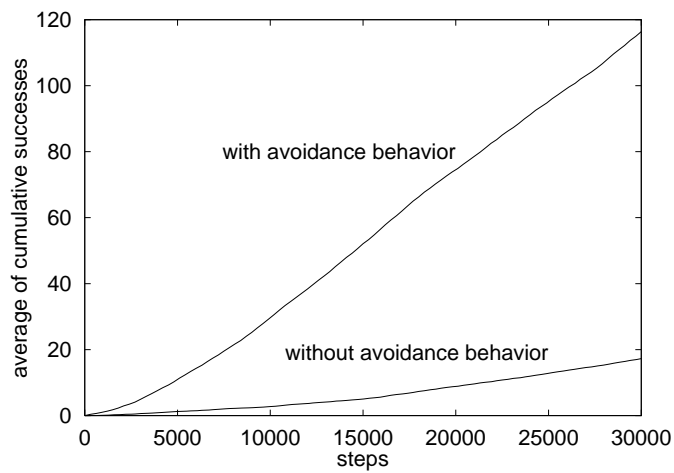
- [1] S. Ono, Y. Inagaki, H. Aisu, H. Sugie, and T. Unemi. Fast and Feasible Reinforcement Learning Algorithm. *Proceedings of the International Joint Conference of the Fourth International Conference on Fuzzy Systems and the Second International Fuzzy Engineering Symposium*, 1713–1718, 1995.



without avoidance behavior.



with avoidance behavior



Comparison in average.

Figure 3: Learning performance over 50 separated random sequences of *two wheels robot*.

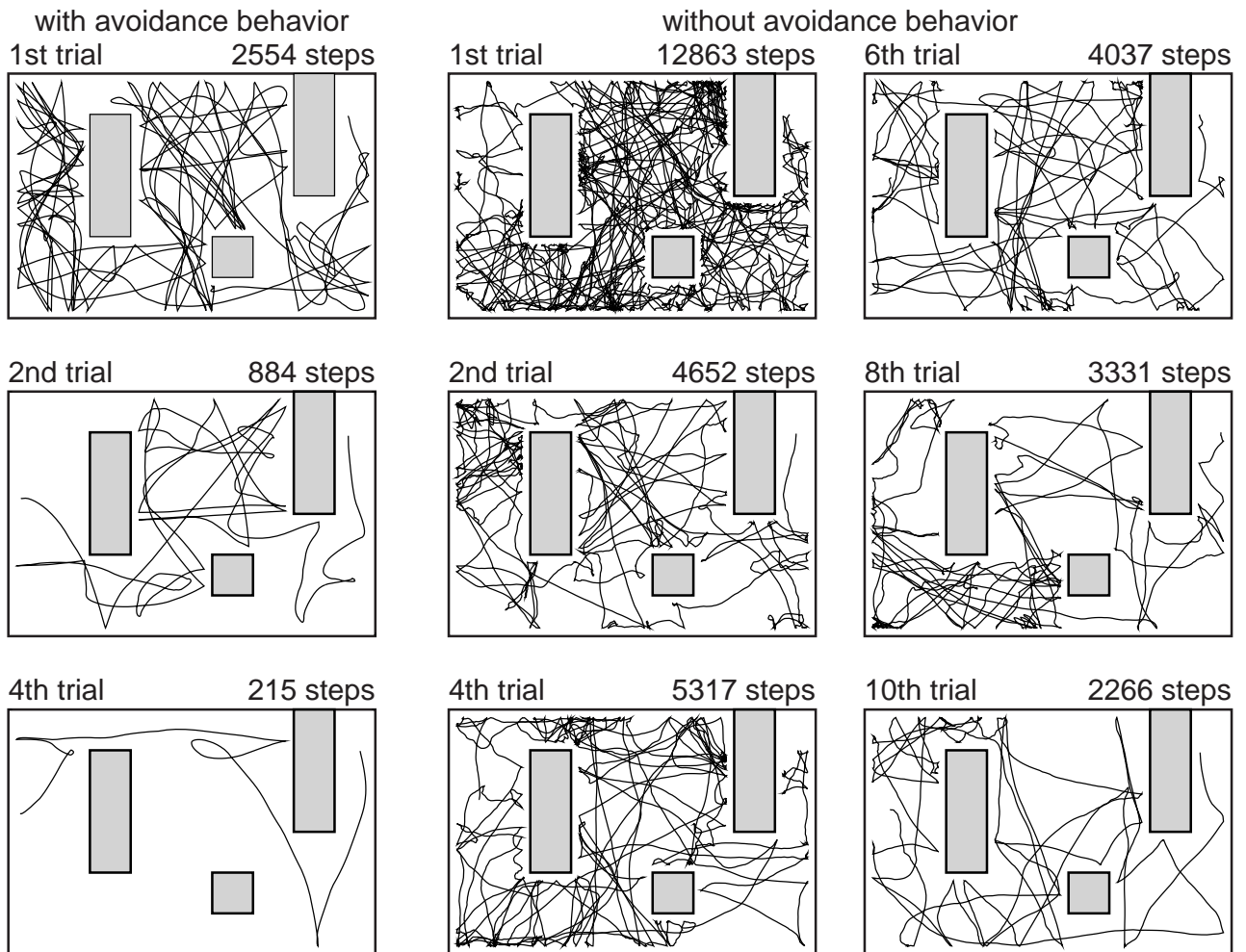
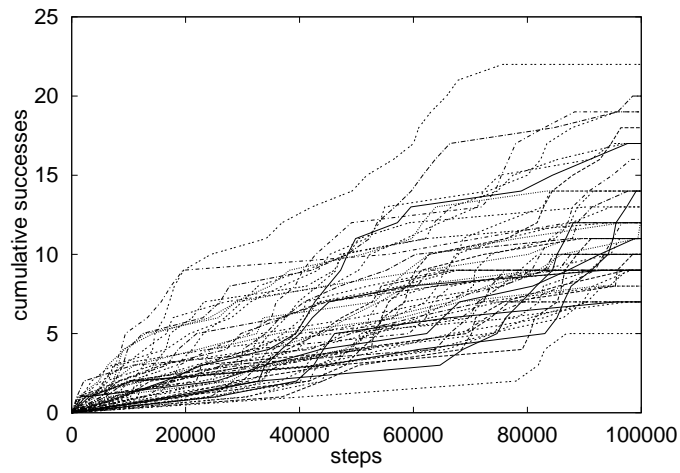
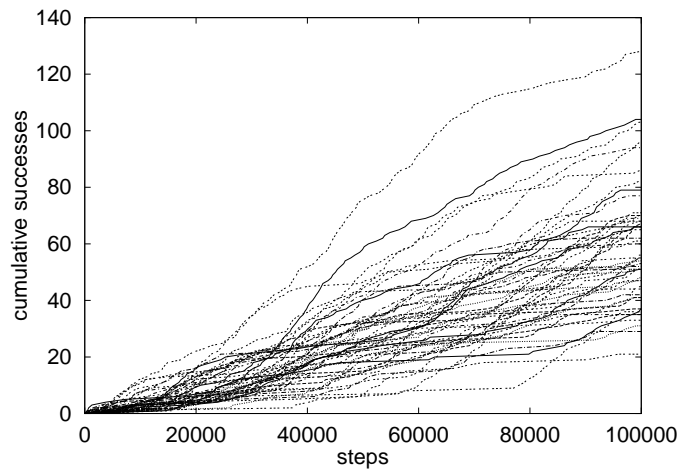


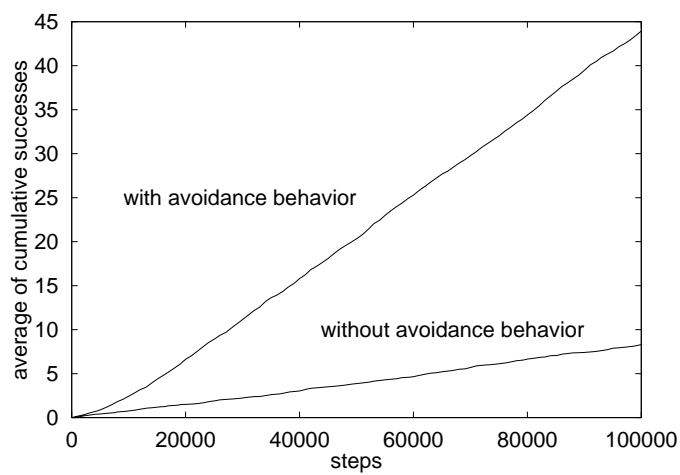
Figure 4: Typical examples of traces in successful cases of *floating robot*.



without avoidance behavior.



with avoidance behavior



Comparison in average.

Figure 5: Learning performance over 50 separated random sequences of *floating robot*.

- [2] T. Unemi and H. Saitoh. Episode-based Reinforcement Learning – an Instance-Based Approach for Perceptual Aliasing. *Proceedings of the 1999 IEEE International Conference on Systems, Man and Cybernetics*, V:435–440, 1999.
- [3] D. W. Aha, D. Kibler, and M. K. Albert. Instance-Based Learning Algorithm. *Machine Learning*, 6:37–66, 1991.
- [4] R. S. Sutton. Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming. *Proceedings of the Seventh International Conference on Machine Learning*, 216–224, 1990.
- [5] C. J. C. H. Watkins and P Dayan. Q-Learning. *Machine Learning*, 8:279–292, 1992.