Looking for a suitable strategy for each problem

-Multiple tasks approach to navigation learning task-

Akitoshi Ogawa* **Takashi Omori***
*Graduate School of Engineering, Hokkaido University
Sapporo, 065-8628, Japan
akitoshi@complex.eng.hokudai.ac.jp
omori@complex.eng.hokudai.ac.jp

Abstract

We propose the functional parts combination (FPC) model, whereby a problem solving strategy is acquired depending on the tasks given. The model is based on the neuroscientific fact that each cerebral cortical area has a different role and is selectively activated depending on the task. FPC model is a meta learning model that consists of a set of functional parts and a sequence of control signals that specifies their combination. The functional parts are combined depending on the situation, to realize a processing circuit required for the situation. We use genetic algorithm for searching the control signals. We examine the model by evaluating the difference in acquired behavior of (1) two agents with different functional parts working on the same navigational task and (2) two agents with the same functional parts working on different tasks. We show that the agent using FPC model acquires learning strategies suitable for the given problems.

1. Introduction

The modeling of an intellectual development process corresponding to physical development is a good benchmark task for an intellectual model, in the sense of requiring adaptive behavior in a dynamical environment. Physical development enables the body to perform more tasks, and hence requires the control system to develop more and more task-solving strategies. In reality, it is impossible to prepare all the necessary strategies that will be required for new situations, tasks and conditions of physical development. And as physical and intellectual changes occur simultaneously, the intellectual system must acquire the strategy on-line. Until this problem is surmounted, we cannot argue that developing robots are intelligent.

For this problem, many methods of action search have been proposed (Sutton and Barto, 1998) (Goldberg, 1989) (Koza, 1992). Most of them present a strategy appropriate to a given task, and argue the generality of the strategy. However, as we can see from the history of AI, it is very difficult to realize an universal action learning method. It is therefore natural to think that each task has its own appropriate method of solution. To advance robtics research, what we need is a method that independently decides on a problem-solving strategy depending on the task. It is a meta learning model.

What then is the strategy or procedure for problem solving? This paper assumes the existence of internal information processing corresponding to each stage of a problem solving process. They are realized by determining a combination of processing sub-modules (neural networks) that produce an action appropriate to a given input depending on the situation. Here, we assume the processing to include not only the generation of physical action but also various internal processing, such as memorizing, recalling, calculation and search. Then, the problem solving is reduced to a search for the combination of sub-functions depending on the situation. We call this the *functional parts combination* model (FPC model) (Omori and Ogawa, 2001) (Ogawa and Omori, 2001).

As an example of FPC model in action, this paper shoes procedual the meta learning of procedure on a navigation learning task by a moving robot. The system acquires the learning procedure of navigation actions to reach a goal in a grid world by reinforcement learning. The choice of particular grid world is leargely arbitarary, because what we discuss is an acquisition of procedure depending on the task and situation. The conditions are these 1) that the system not know the procedure of reinforcement learning, and 2) that the system be given a set of fuctional parts that can realize the procedure. We adopted a variable length genetic algorithm as the search method for a variable length procedure. The environment and functional parts change with task, but our model would find procedures appropriate to the task when those changes occur.

2. FPC model

2.1 The role of selective activation in cortex

The cerebral cortex comprises many areas with different functions. Noninvasive research has confirmed the selective activation of the cerebral cortex depending on the task. Although the mechanism of activation is not known, it is accurate to say that the selective activation of brain functions is a part of the basic mechanism of intellectual information processing in the brain.

However, there is known to be a wide variety of connections between many areas of the cerebral cortex. For instance, areas in the visual system that process many sort of information from area V1 to ventral and dorsal pathways are hierarchically organized. Anatomical research shows many connections among these areas (Felleman and VanEssen, 1991) (VanEssen et al., 1992). An individual connection suggests a functional relation between two areas, but it is not clear by what principle these connections are used in each task.

2.2 Functional parts combination

From the facts above, we can interpret the selective activation of the cortical areas, or the connection between them, as a formation of neural circuit. When activated, the cortical neural circuit automatically produces output based on the input and its internal state. The circuit is the formation of an information processing function.

However, processes that satisfy the task requirement cannot be realized by a single circuit in a short time. Examples of such processes are those that require iteration, search and temporal inputs for processing. Selective activation of the circuits should be sequential depending on the task.

The following is a possible system model that realizes the system behavior. Fig. 1 shows the overall image of the model.

1. There are many neural networks that have different input and output connections. Each neural net is expected to learn to assume the role of its functional part. The network functions and topology are given pre-designed in our current model.

2. A parts selector system sends activation signals to each of the functional parts or to the connections. The signal works as an internal attention. The attention signals become a vector composed of a set of $\{0, 1\}$. We call the vector of the moment AV.

3. The attention vector sequence (AVS) is a sequence of AV's. AVS forms a sequence of processing circuits and realizes a procedure that fullfills the task requirement. An AV is switched to the next AV when the behavior of the selected circuit becomes stable.

4. The system holds a set of AVSs. One of them is selected and activated corresponding to the task. A task may be divided into sub-tasks to which a different AVS may correspond.

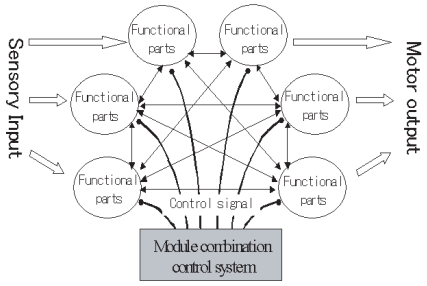

Figure 1: The structure of FPC model

2.3 Meta learning: search for functional parts combination

How is the combination of functional parts acquired in FPC model? It is natural to regard the use of reward driven search method as reinforcement learning. An experience based search in which learning is done by analogy is a strong candidate. For simplicity, we did not adopt it this time.

If searching for AVS, the processing circuit is aquired. For the purpose, we use the genetic algorithm that searches a combination of the variable number of fixed length gene groups. In this search task, the length of a gene representing an AV at any moment is fixed. To search for the variable length AVS that combines them, we use following algorithm, a combination of a mutation and an imbalanced cross-over.

1. Decide a fitness value of each individual gene based on a given definition of fitness for the task.

2. Select pairs of individuals by a roulette method in which the individuals are sorted by the fitness value and in which two neighboring individuals in the order are chosen. A Predetermined number of pairs are selected from the top of the fitness value.

3. A length of AVS is the length of AV multiplied by K. The value K is the length of the AVS. A border of AV in AVS is chosen as the cutting point of a crossover. As the length of AVS for the crossover pair is not same and the cutting points of both AVSs are arbitrary, the resulting AVS do not have the same length.

4. The mutation operation is applied to every AVS.

5. Repeat operations 1 to 4 until the fitness value becomes stable.

The unique characteristic of this method is the imbalanced crossover. The length of AVS corresponds to the number of operations for current input and states, and the algorithm searches for procedures of variable lengths.

This search is the meta learning that independently finds a learning procedure, because the AVS-controlled targets are neural networks that have learning ability. In the meta learning, the range of system adaptability is determined by the available functional parts and the meta learning method.

2.4 Multiplex tasks approach

In many conventional studies, a learning model is proposed and evaluated for a given task, and the generality of the learning model to other tasks is discussed. Such approach is inappropriate for our meta learning model study. We do not gear performance of the model to a given task, rather, we target the ability to acquire the learning procedure finding ability for various types of learning tasks. For the evaluation of the meta learning model, we have to analyze the realized learning model for each task and compare its processing structure and computational model with conventionally known learning models; then we can know the range of the tasks to which our meta learning model is applicable, and what are the limiting constraints of the system.

This paper attempts to solve some types of navigation problem using FPC model with the meta learning mechanism. The functional parts for each of the tasks may be minimal or redundant. When the number of parts for the task is minimal, we expect the meta learning to find the minimal learning procedure that can be realized with the parts. But if the parts are redundant for solving the task, we expect the meta learning to find an easy strategy at first, and then increasingly effective procedures through trial and error. In some cases, the acquired procedure may be trivial, that is, effective only for the given task situation. But when the task is more difficult, the trivial task may change to more general one.

The essence of the strategy is that the system is expected to become increasingly effective and robust to the change of task. We call this method for the meta learning evaluation the *multiplex task approach*. Fig. 2 shows structure of our approach. Multiple combinations of the task requirement and the functional parts find the problem solving procedures each of which can resolve its corresponding task based on a common meta learning algorithm. Through a evaluation and comparison of the implemented procedure, we can know characteristics of the architecture.

3. Computer simulation

The task is a navigation learning problem in a grid world. Reward is given to an agent when it reaches a goal. The agent must discover a valuable path, a shortest path or its equivalent, to the goal to maximize total reward.

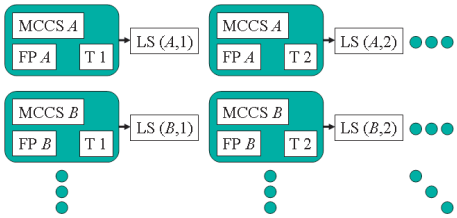

Figure 2: Functional parts (FP) are combined suitably by a Module combination control system ($MCCS$) in each Task (T). Learning strategy (LS) is acquired automatically, because a $MCCS$ decides the combination of FP depending on a T.

We performed computer simulations of the following two cases in order to evaluate our model:

Case 1: The task is the same and the functional parts are different.

Case 2: Functional parts are the same and the tasks are different.

3.1 Case 1: different functional parts on the same task

The grid world of the task is shown in Fig. 3. An agent receives the distance to the wall in four directions as a sensory input, and has internal states corresponding to each grid. When the agent reaches the goal, it receives reward and is returned to the starting point. When the agent collides with the wall, it receives no reward and returns to the starting point.

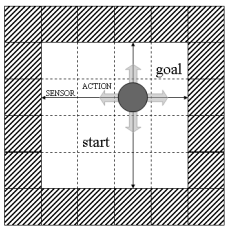

Figure 3: The environment for procedure acquisition in the case 1

We prepare two agents, one basic and one evolved, that have different functional parts. The basic agent shown in Fig. 4 consists of three functional parts (a state recognition part, an action selection part and a reinforcement learning part) and the attention system that determines the connection of those parts. The reinforcement learning part has a sufficient number of input recognition neurons corresponding to the place and action combination. They are used for the Q-table in this task. The evolved agent shown in Fig. 5 consists of the internal model of the environment and the working memory, in addition to the parts for the basic agent.

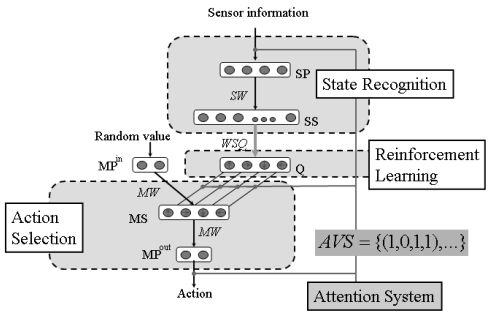

Figure 4: The structure of the basic agent in case 1

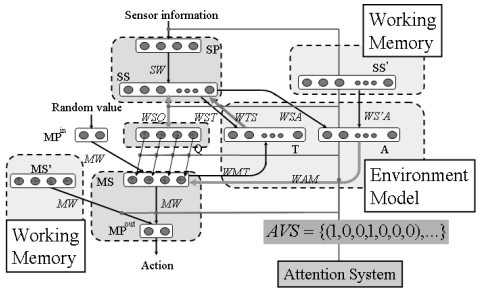

Figure 5: The structure of the evolved agent in case 1

3.1.1 The basic agent

The part for state recognition Distance sensor input $\boldsymbol{RS}$ is normalized and given to the SP-layer that works as an input buffer.

$$SP_i = \frac{RS_i}{\sqrt{\sum_i RS_i^2}}$$

Each cell of the SS-layer represents the internal state corresponding to the place in the environment. Competition between the cells limits the number of firing cell at any moment to one.

$$ss_j = a_1 \sum_i SW_{ij} SP_i \quad (1)$$

$$SS_j = \begin{cases} 0 & : j \neq \arg\max_w ss_w \\ 1 & : j = \arg\max_w ss_w \end{cases}$$

The variable a_1 is the attention value over the connection SW_{ij} from the SP-layer to the SS-layer. It takes a value of zero or one to control use of the connection SW_{ij}.

The part for action selection Input to the MP-layer is a vector $(\cos\theta, \sin\theta)$ that represents a possible action of the agent. It is generated by a random action generator, where θ is a random value between 0 and 2π. This generator generates a random action when proper attention is set to the neural circuit.

Each cell of the MS-layer represents the discrete motor direction. Only one cell is allowed to fire at any moment.

$$ms_u = \eta \sum_r MW_{ru} MP_r^{in} + a_2 Q_u \quad (2)$$

η is a small positive number.

$$MS_u = \begin{cases} 0 & : u \neq \arg\max_{w'} ms_{w'} \\ 1 & : u = \arg\max_{w'} ms_{w'} \end{cases}$$

After the firing of an MS-layer cell, its corresponding input pattern MW_{ru} is recollected at MP^{out}-layer, and is outputted by a proper attention value.

$$MP_r^{out} = a_3 \sum_u MW_{ru} MS_u$$

The variable a_2 is the attention value over the connection from the Q-layer to the MS-layer. The a_3 works as an action trigger. Both variables take the value of zero or one, and control the use of the corresponding term in each equation.

The part for reinforcement learning After the place representing cell SS_j fires, the Q-layer receives the action-value (Q-value) of the place and works as a buffer for this value. Because there is a one-to-one correspondence between cells in the Q-layer and the MS-layer, each Q-layer cell activity represents the Q-value of each action k at the place j.

The connection WSQ_{jk} is equivalent to the Q-value table in Q-learning. Learning for WSQ_{jk} is performed by the following equation:

$$\Delta WSQ_{w_{t-1},w'} = \alpha(r + \gamma(\max_b WSQ_{w_t,b} - WSQ_{w_{t-1},w'}))$$

Here α is the learning rate, and γ is the discount rate. This is a typical Q-learning equation. Although acquisition of the learning rule itself is one of our final targets, we provide an a priori rule in this study.

The attention system The Attention System can control the topology of a neural network dynamically by changing the attention vector (AV). The use of AVS enables the neural network to be programmed through dynamic control of the network structure. Each AVS element AV is given to the network sequentially.

$$\boldsymbol{AVS} = (\boldsymbol{AV_0}, \boldsymbol{AV_1}, \boldsymbol{AV_2}, \ldots, \boldsymbol{AV_n})$$

3.1.2 The evolved agent

In addition to the functional parts of the basic agent, the evolved agent has additional memories that work as the internal model for the environmental map and the working memory. In relation to that agent, the parts, the state recognition and the action selection of the evolved agent is changed as follows.

The part for state recognition In addition to the input from the sensor, the SS-layer of the evolved agent receives input from the T-layer and the Q-layer. Equation (1) is changed as follows.

$$ss_j = a_1 \sum_i SW_{ij} SP_i + \\ +a_4 \sum_k WSQ_{jk} Q_k \sum_l WTS_{lj} T_l$$

The value of a_4 represents the attention over the input.

The part for action selection The MS-layer of the evolved agent also receives input from the A-layer. Equation (2) is modified to include the input from the A-layer that represents the place-place to action relation.

$$ms_u = \eta \sum_r MW_{ru} MP_r^{in} + a_2 Q_u \\ +a_5 \sum_v WAM_{vu} A_v$$

The MS'-layer is the working memory that represents the activity of the MS-layer at time $t-1$. The evolved agent outputs the action according to the activity of MS-layer at times t or $t-1$, or none. Equation (3) is changed as follows.

$$MP_r^{out} = a_3 \sum_u MW_{ru} MS_u + a_6 MS'_u$$

The part for environment learning The T-layer is a memory that represents the next state from the combination of the current internal state and the internal action. As the internal state corresponds to a place in the map, the T-layer represents the state transition relation of the environment. The number of T-layer cells is the product of the number of SS-layer cells and the number of MS-layer cells. WST_{jl} and WMT_{ul} are set by the following equations, where N is the number of SS-layer cells.

$$WST_{jl} = \begin{cases} 1 & : l = uN + j \\ 0 & : others \end{cases}$$

$$WMT_{ul} = \begin{cases} 0 & : l = um + 0 \ldots N-1 \\ -1 & : others \end{cases}$$

The value of a T-layer cell is calculated by the following equation.

$$T_l = \phi(\sum_j WST_{jl} SS_j + \sum_u WMT_{ul} MS_u)$$

$$\phi(x) = \begin{cases} 0 & : x \le 0 \\ x & : 0 < x < 1 \\ 1 & : x \ge 1 \end{cases}$$

The connection WTS_{lj} between the T-layer and the SS-layer represents state transition, (state, action) $\rightarrow$ state. The T-layer is used to predict the next state based on the next action. Learning of WTS_{lj} is performed by the following equation,

$$WTS_{lj} = WTS_{lj} + \alpha(T_l - WTS_{lj}) SS_j$$

where all the initial value of WTS_{lj} are 0.

The A-layer represents the action that bridges internal state at time t and $t-1$. The number of A-layer cells is equal to the square of the number of SS-layer cells. We assume the SS'-layer represents the activity of the SS-layer on time $t-1$. Connections WSA_{jv} and $WS'A_{j'v}$ are set by the following equations, where N is the number of SS-layer cells.

$$WSA_{jv} = \begin{cases} 1 & : v = j(N-1) + 1, \ldots, N-1 \\ 0 & : others \end{cases}$$

$$WS'A_{j'v} = \begin{cases} 1 & : j' + j(N-1) \quad (j' \neq j) \\ 0 & : others \end{cases}$$

The value of an A-layer cell is calculated by the following equation,

$$A_v = (\sum_j WSA_{jv} SS_j)(\sum_u WS'T_{j'v} SS'_{j'})$$

The connection WAM_{vu} lies between the A-layer and the MS-layer, and represents the directional relation between the places. Learning for WAM_{vu} is performed by the following equation,

$$WAM_{vu} = WAM_{vu} + \alpha(A_v - WAM_{vu}) MS_u$$

where all the initial values are set to zero.

3.1.3 Summary of the GA

The population size is set to 50, and the length of AVS is restricted to less than or equal to 10. All genes are initialized randomly.

One trial ends when an agent finds a valuable path. Fitness O is the number of necessary moves in each trial. When an agent reaches the goal three times consecutively in very few steps, we conclude that the agent has found the valuable path. It is unreliable to decide fitness from a single trial, because the action of the agent contains a random factor. Then, the following scaling is used:

$$f(O) = 1000 \sum_{i=1}^{10} O_i^{-1}$$

To reduce the influence of random factor, the sum of O^{-1} for ten trials per individual is calculated.

Probability of selection is decided in proportion to the scaled fitness. One-point cross-over is used, and a cross-over point is randomly chosen from the set of borders between the AV's in AVS. We used a higher cross-over rate 1.0 to accelerate the search. The mutation rate is 0.05.

3.1.4 Result

The following is a typical AVS that the basic agent acquired in the first generation:

$$\boldsymbol{AVS} = (1, 1, 1)$$

By using this AVS, the basic agent recognizes the current state and acts according to the Q-value. It is a well-known greedy strategy.

The evolved agent acquired the following AVS in the 9th generation:

$$\boldsymbol{AVS} = ((0, 0, 0, 1, 0, 0), (1, 0, 1, 0, 1, 0))$$

An competition process embedded in the 1st attention vector finds the next place with the maximum total sum of Q-value within the range of single-step action. The agent actually moves to the place by the 2nd attention vector step. This is a prediction-based behavior that makes use of the environmental map. Fig. 6 shows change of the fitness value of the best adapted agent along generations. The fitness of the basic agent does not vary with the constraint of available parts. Fig. 7 shows the change of the best adapted AVS vector length along generations.

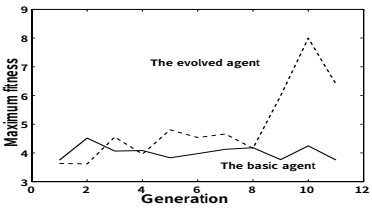

Figure 6: The fitness of the best adapted agent for each generation. The basic agent acquires the greedy strategy in the first generation. The evolved agent acquires the prediction-based behavior in the 9th generation.

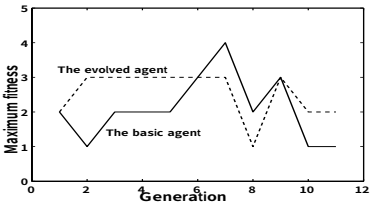

Figure 7: The length of the best adapted agent's AVS. Because each AVS (of the basic agent and the evolved agent) has redundant AV's, their AVS is long in the early generations.

3.2 Case 2: different tasks for a single set of functional parts

The environments for each of the tasks are shown in Fig. 8 and Fig. 10. The agent shown in Fig. 9 receives input indicating whether there is an obstacle at each grid neighboring the agent's position. The agent selects an action that prevents collision with an obstacle. The agent is a modified version of the evolved agent in case 1. We adopt not Q-learning but actor-critic model for reinforcement learning and the connection between the reinforcement learning part and the working memory of the state is added. The working memory of the action is removed.

The perceptual aliasing problem, in which the agent recognizes different places to be same because of the lack of sensor ability, arises in the environment II, but not in the environment I. Therefore, if we use a model free reinforcement learning, the agent cannot detect a valuable path in the environment II (Lin and Mitchell, 1992).

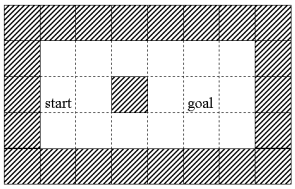

Figure 8: The environment I for a task in the case 2. The perceptual aliasing problem does not arise, because the sensory signal is enough to allow discrimination of each grid.

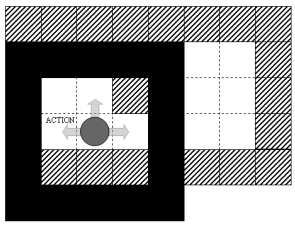

Figure 9: The range of the agent's sensor and action.

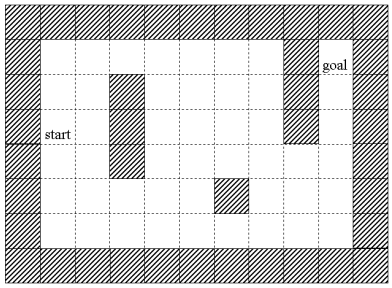

Figure 10: The environment II in the case 2. There are a few grids that the agent cannot discriminate because of its poor sensor ability.

3.2.1 The agent

All functional parts except the environment learning are modified as follows.

The state recognition part The input represents whether each of its 8 neighboring grid contains an obstacle.

$$SP_i = \begin{cases} 1 & : obstacle \\ 0 & : non-obstacle \end{cases}$$

In the SS-layer, the determination of the firing cell is based on the Euclidean distance inputs from the reinforcement learning part and the environment learning part.

$$\begin{aligned} ss_j &= a_1 \sum_i (SW_{ij} - SP_i)^2 \\ &\quad -a_2 WSV_j V \sum_l WTS_{lj} T_l \\ SS_j &= \begin{cases} 0 & : j \neq \arg\min_w ss_w \\ 1 & : j = \arg\min_w ss_w \end{cases} \end{aligned}$$

The working memory part The timing of copying from the SS-layer to the SS'-layer is controlled by the attention signal a_3.

$$\boldsymbol{SS' = SS} \quad \text{if } a_3 = 1.$$

The action selection part Equation (3) is modified. The random factor is removed, and the MS-layer always receives input from the Q-layer.

$$ms_u = Q_u + a_8 \sum_v WAM_{vu} A_v$$

Instead of removing the random factor, We decided the firing cell in the MS-layer with a probability in proportion to $\boldsymbol{ms_u}$.

As the working memory for action is removed, the agent's output changes.

$$MP_r^{out} = a_5 \sum_u MW_{ru} MS_u$$

The reinforcement learning part The activity of the Q-layer that sends output to the MS-layer is calculated as follows.

$$Q_u = a_4 \exp\left(\beta \left(\sum_j WSQ_{j,u} SS_j + a_6 \sum_{j'} WS'Q_{j',u}\right)\right)$$

β is a parameter that decides the affect of randomness.

When the agent move from one place to another, learning parameter δ is calculated.

$$\begin{aligned} \delta &= r + \gamma \left(WSV_{w_t} + a_7 WS'V_{w_t}\right) \\ &\quad - \left(WSV_{w_{t-1}} + a_7 WS'V_{w_{t-1}}\right) \end{aligned}$$

All the connections for reinforcement learning increase according to the following equation.

$$\left.\begin{array}{l} \Delta WSV_{w_{t-1}} \\ \Delta WS'V_{w_{t-1}} \\ \Delta WSQ_{w_{t-1}} \\ \Delta WS'Q_{w_{t-1}} \end{array}\right\} = \alpha\delta$$

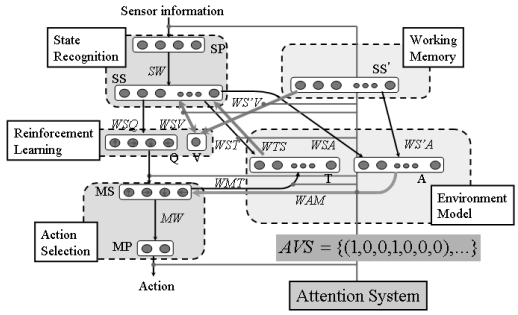

Figure 11: The structure of the agent in case 2.

3.2.2 *Summary of the GA*

The search algorithm for the AVS is largely the same as in case 1, except for following points. The population size is 10. Fitness O is the number of necessary trials. The following scaling function is used:

$$f(O) = \exp\left(-\xi \sum_{i=1}^{5} O_i^{-1}\right)$$

ξ decides the importance of fitness in the pair selection phase, and is set to 0.05. The cross-over rate is 0.9.

3.2.3 *Result*

The following is a typical AVS in the environment I at the 10th generation.

$$\boldsymbol{AVS_1} = ((0,1,1,0,1,0,1,1),(1,1,0,1,0,1,1,1))$$

The behavior that is generated by $\boldsymbol{AVS_1}$ is exactly the same the prediction-based behavior in case 1.

The following is a typical AVS in the environment II at the third generation.

$$\boldsymbol{AVS_2} = (1,0,0,1,1,1,1,1)$$

By using $\boldsymbol{AVS_2}$, the agent recognizes the current state and the state one step before, and both states are used for action learning. Meta learning finds a memory-based behavior like that in Lin and Mitchell reported (Lin and Mitchell, 1992).

4. Discussion

Case 1 It is natural that the basic agent acquired the greedy strategy, because it is the only strategy to find a valuable path by using given functional parts. The

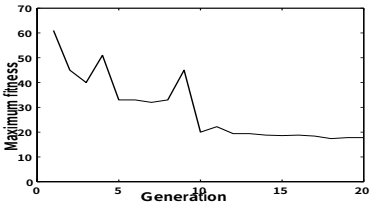

Figure 12: The maximum fitness in the map I. In 10th generation the agent acquired the prediction based behavior.

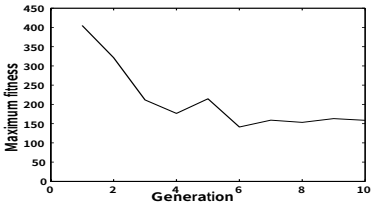

Figure 13: The maximum fitness in the map II. In 2nd generation the agent acquired the memory based behavior.

evolved agent has parts for environment learning and working memory that the basic agent does not have. Though two step prediction based learning was possible using these parts, the evolved agent acquired the one step prediction based behavior. Our pilot study confirmed that the fitness of the two step prediction based behavior is hardly better than the one step, It is difficult for the evolved agent to acquire the AVS of the two step prediction based behavior, because the search space is large and the fitness shape of the AVS in the search space is narrow. Consequently, the evolved agent acquires the 1 step prediction based behavior.

Case 2 Because the perceptual aliasing problem does not arise in the environment I, the state after the agent's action is unique by using the environment model. Consequently, the agent acquired the one step prediction based behavior. On the other hand, in the environment II, the agent acquired the memory based behavior using the current state and the state one step before. Though it is known that a model based reinforcement learning is more powerful (Atkeson and C.Santamaria, 1997), the agent could not find a valuable path in this case, because the state after the agent's action is not unique in environment II.

The important point in two cases is that each agent acquires behavior appropriate for each task, though each agent has no information about which strategy is superior for each task. By using our model, the agent acquires an appropriate strategy according to each task within the limits of its functional parts.

5. Conclusion

This paer proposed the FPC model that independently acquires a problem solving strategy depending on a task. Multiple navigation problems showed the following results.

1. Each agent acquired appropriate path search procedures, when tasks were the same and functional parts were different.

2. The agent acquired appropriate path search procedures, when tasks were different and functional part were the same.

The results show that, by adopting the FPC model, the agent can acquire effective procedures depending on the tasks and functional parts.

The next step is to apply the FPC model to non-navigation problems, to verify the effectiveness of the model, and to apply it to real navigation problems in ral robots.

References

Atkeson, C. G. and C.Santamaria, J. (1997). A comparison of direct and model-based reinforcement learning. *Proc. of International Conference on Robotics and Automation.*

Felleman, D. J. and VanEssen, D. C. (1991). Distributed hierarchical processing in the primate cerebral cortex. *Cereb Cortex*, 1(1):1–47.

Goldberg, D. E. (1989). *Genetic algorithm in search, optimization and machine learning.* Addison Wesley.

Koza, J. (1992). *Genetic Programming, On the Programming of Computers by means of Natural Selection.* MIT Press.

Lin, L. J. and Mitchell, T. M. (1992). Memory approaches to reinforcement learning in non-markovian domains. *Technical Report CMU-CS-92-138.*

Ogawa, A. and Omori, T. (2001). The acquisition of space search procedure depending on agent structure. *Proc. of ISHF2001.*

Omori, T. and Ogawa, A. (2001). Two hypothesis for realization of symbolic processing in brain. *Proc. of ICONIP2001.*

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction.* MIT press.

VanEssen, D. C., Anderson, C. H., and Felleman, D. J. (1992). Information processing in the primate visual system: an integrated systems perspective. *Science*, 255:419–423.