9

A Q-learning with Selective Generalization Capability and its Application to Layout Planning of Chemical Plants

Yoichi Hirashima
Osaka Institute of Technology
Japan

1. Introduction

Under environments that the criteria to achieve a certain objective is unknown, the reinforcement learning is known to be effective to collect, store and utilize information returned from the environments. Without a supervisor, the method can construct criteria for evaluation of actions to achieve the objective. However, since the information received by a learning agent is obtained through an interaction between the agent and the environment, the agent must move widely around the environment and keep vast data for constructing criteria when complex actions are required to achieve the objective. To conqure these drawbacks, function approximation methods that have generalization capability have had the attention as one of effective methods. The challenge of this chapter is focused on improving learning performances of the rainforcement learning by using a function approximation method, a modefied version of Cerebellar Model Articulation Controller (CMAC) (Albus, 1975a; Albus, 1975b), used in the reinforcement learning.
CMAC is a table look-up method that has generalization capabilities and is known as a function learning method without using precise mathematical models for nonlinear functions. Thus, CMAC is used to approximate evaluation functions in reinforcement learning in order to improve learning performance (Sutton & Barto, 1999; Watkins, 1989). In the CMAC, the numerical information is distributively stored at memory locations as weights. Each weight is associated with a basis function which outputs a non-zero value in a specified region of the input. The CMAC input is quantized by a lattice constructed by basis functions. In order to speed up learning and increase the information spread to adjacent basis functions, the CMAC updates a group of weights associated with basis functions that are close to a given point, and thus yields generalization capability. The concept of closeness stems from the assumption that similar inputs will require similar outputs for well-behaved systems. The structure of lattice determines how the CMAC input space is quantized and how the generalization works. However, the conventional CMAC has a fixed lattice and a fixed shape of region covered by the effects of generalization. Although the size of the region can be changed by adjusting the quantization intervals for lattice, the shape of the region is not adjustable. The required size and shape of the regions are not same for different cases, and thus, the CMAC has difficulties to obtain appropriate generalization for each case.

Source: Theory and Novel Applications of Machine Learning, Book edited by: Meng Joo Er and Yi Zhou,
ISBN 978-3-902613-55-4, pp. 376, February 2009, I-Tech, Vienna, Austria

To conquer the drawback, a design method of CMAC that has a selective generalization capability is introduced. In this method, several CMACs are selected by extended input that can adjust the shape and size of region covered by the effects of CMAC generalization. The extended input is generated by a function obtained by using a priori knowledge of the addressed problem. By the proposed method, appropriate generalization can be obtained for cases that the conventional CMAC is not efficacious. The proposed method is applied to a allocation problem of a plant based on the fictitious chemical production plant as a case study.
In chemical plants, layout of plant-elements should be determined considering at least the accessibility for maintenance and fire fighting, operability, and construction cost. From a pure safety perspective, tanks and reactors should be placed as far as possible from each other, in order to minimize the effect of a fire or explosion which a tank or reactor have on adjuscent equipment. But a larger distance between these elements of chemical plant, such as tanks and reactors, requires a larger pipe-length for connecting these elements and the efficiency of production and operation get worse. Hence in the layout of the chemical plant, there is a problem to minimize risk due to fires and explosions and maximize efficiency of production and operation simultaneously. Thus, the element allocation in a chemical plant is a multi-objective optimization problem. This problem was approached by using mathematical programming (Ceorgiadis et al, 1999; Vecchietti & Montagna, 1998). Or it may be considered to use Neural Network or Genetic Algorithm. But the problem has so many variables and freedoms to be chosen and also so many constraints among variables. Hence solving the problem by using those methods may take too much time and is not adequately efficient. Recently, a new reinforcement learning method has been proposed in order to solve allocation problem (Hirashima et al, 2002). The method is derived based on Q-learning (Watkins, 1989; Watkins, 1992) and hybrid-cordination that only a certain part of the plant-layout is recognized by the distance and rotational angle of each element measured from a `basis element' . In this method, rotated and/or shifted plants that have the same layout are identically recognized (Hirashima et al, 2005). The CMAC with selective generalization capability is integrated to this method, and effectiveness of the new CMAC is shown by computer simulations.
The remainder of this chapter is organized as follows: section 2 gives a detailed explanation of the CMAC that has selective generalization capabilities. Section 3 explains allocation problem of chemical plants. Section 4 presents a Q-learning method for plant allocation problem. Section 5 depicts and compares several results of computer simulations for a plant allocation problem. Finally, section 6 concludes the chapter.

2. CMAC

In this section, CMAC that has Selective Generalization capability (SG-CMAC) is explained. Assume the SG-CMAC has n inputs and 1 output. Define the input of SG-CMAC as $\mathbf{s} = (\mathbf{s}_I, s_z)$, $\mathbf{s}_I = \{s_i\}$ $(0 \le s_i < \Delta_i,\ i = 1, \cdots, n-1; 0 \le s_z < \Delta_z)$, where $s_z = f(\mathbf{s}_I)$. s_z is quantized with m areas and a corresponding CMAC module is assigned to each quantized area. CMAC modules used by the proposed method are same as the conventional CMAC. In the next subsection, the structure of CMAC modules is explained.

2.1 CMAC modules

Consider a basis function whose output is 1 when the input lies in its *support* and 0 otherwise. We define the size of the support as $\boldsymbol{\rho}$ for each axis. A set of k overlays is formed so that each of the module inputs is covered by k supports. The overlays are displaced by ρ relative to each other. Each overlay consists of the union of adjacent non-overlapping supports. Edges of supports generate knots, and the subset of knots constructs a square lattice of size ρ which quantizes the whole input space. The quantization interval ρ is determined by the size of the interval of neighboring knots, and it controls the degree of generalization.

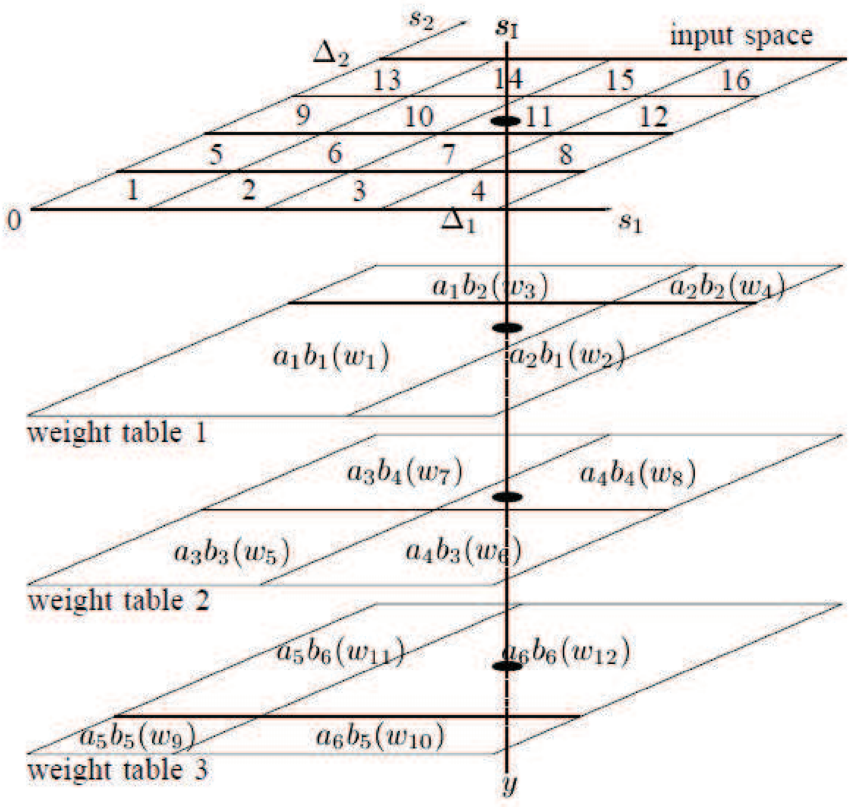

Fig. 1. Structure of a CMAC module

Fig.1 shows a CMAC module for the 2-dimensional input and 1-dimensional output consisting of 3 overlays and 12 basis functions. The lattice cells are numbered from 1 to 16. Assume the input to the CMAC module as $\mathbf{s}_I = \{s_1, s_2\}$ and the input space as $S = \{(s_1, s_2) \mid 0 \le s_1 \le \Delta_1,\ 0 \le s_2 \le \Delta_2\}$. Then s_1 and s_2 are quantized quantization interval ρ in the lattice. In the first overlay, s_1 is quantized into a_1 or a_2, and s_2 is quantized into b_1 or b_2, respectively. The pairs of $a_1b_1,\ a_2b_1,\ a_1b_2,\ a_2b_2$ express basis functions, and $a_2b_2(w_4)$ implies that the basis function a_2b_2 has the weight w_4.

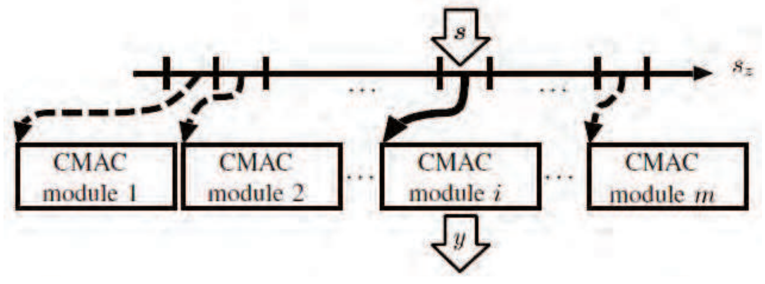

Fig. 2. Output of a SG-CMAC

2.2 Output of a SG-CMAC

The output of a module *u* is formed from a linear combination of basis functions. Given an input $\mathbf{s} = (\mathbf{s}_I, s_z)$ to the SG-CMAC, the input is quantized, the corresponding CMAC module is selected, a basis function is specified for each overlay in the selected module, and the weight value associated with the specified basis function is output from the overlay. The outputs of all the overlays are then summed up to yield the SG-CMAC output *y*, that is,

$$y = \sum_{j=1}^{k} w_j, \cdot \tag{1}$$

where w_j is the weight value associated with the basis function specified in the *j*th overlay.

In Fig.2, the input $(\mathbf{s}_I, s_z)$ is given to the SG-CMAC. Quantized value of s_z corresponds to the *i*th CMAC module. Then, $\mathbf{s}_I$ specifies a certain lattice cell in the CMAC module, for example the 11th cell as shown in Fig.1. In the figure, $\mathbf{s}_I$ specifies a_1b_1, a_4b_4 and a_6b_6. Then, the module outputs $w_1 + w_8 + w_{12}$ as the output of SG-CMAC.
Suppose that the desired signal for the input $\mathbf{s}_I$ is *d* and the learning rate is *g*. The SG-CMAC is then learned by adding the following correction factor $\boldsymbol{\delta}$ to all weights corresponding to $\mathbf{s}_I$ in the CMAC module specified by s_z:

$$\boldsymbol{e} = \boldsymbol{d} - \boldsymbol{y} \cdot \tag{2}$$

$$\boldsymbol{\delta} = \boldsymbol{g}\frac{\boldsymbol{e}}{\boldsymbol{k}} \cdot \tag{3}$$

When giving two similar inputs to the CMAC module, several basis functions are commonly specified. The existence of such common basis functions yields generalization capability. Since the weights corresponding to the input $\mathbf{s}_I$ are only in the CMAC module specified by s_z, the degree of generalization is adjustable by the quantization intervals for s_z. In addition, the shape of the region covered by the effects of generalization can be determined by the shape of function $s_z = f(s_I)$.

2.3 Numerical examples

Effects of CMAC generalization are explained by numerical examples. An input $(\mathbf{s}_I, s_z)$ is given to CMACs that each module has the 2-dimensional input and 1-dimensional output consisting of 3 overlays and 27 basis functions. Here, $\Delta_1 = \Delta_2 = 7$, $\boldsymbol{\rho} = 1$, $\boldsymbol{g} = 0.1$, $\mathbf{s}_I = (3.5, 3.5)$, $\boldsymbol{d} = 1.0$, and $\boldsymbol{m} = 20$. After giving $\mathbf{s}$ once, generalizations of following 3 CMACs are compared:
(A) Conventional CMAC.
(B) Proposed SG-CMAC, $s_z = s_2 - s_1 |_{S_z = 0}$
(C) Proposed SG-CMAC, $s_z = s_2 - 1 - (s_1 - 2)^2 |_{S_z = 0}$
After *k* and $\boldsymbol{\rho}$ are determined, the shape of the region covered by the effects of generalization for an input is fixed in the CMAC (A) as shown in Fig.3. In this case, 3 weights specified by $\mathbf{s}_I$ are updated, so that the effect spreads over colored area in the right figure of Fig. 3. While, in CMACs (B) (C), the region that the effects of generalization spread

is restricted in the CMAC module specified according to the value of s_z, as illustrated in Figs.4, 5. Thus, the shape and the quantization interval of s_z determins the degree of generalization in the SG-CMAC.

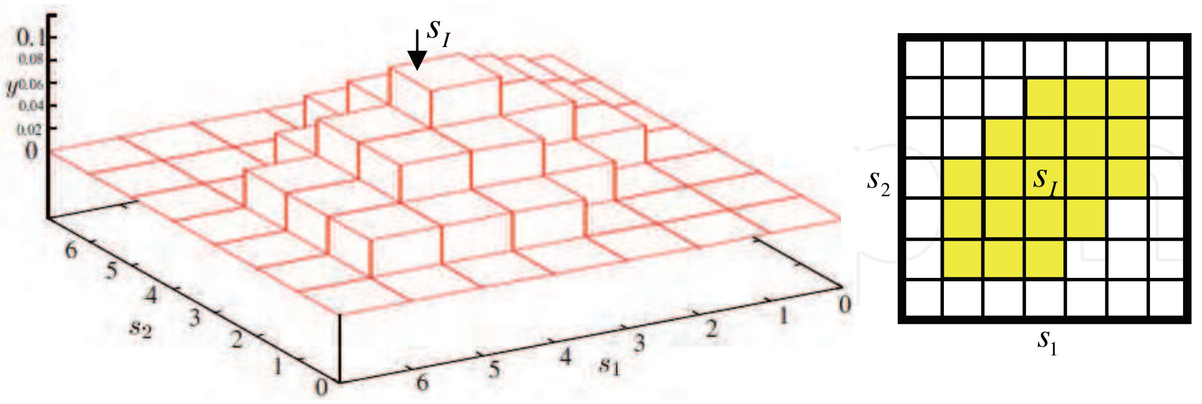

Fig. 3. Output of a CMAC (A)

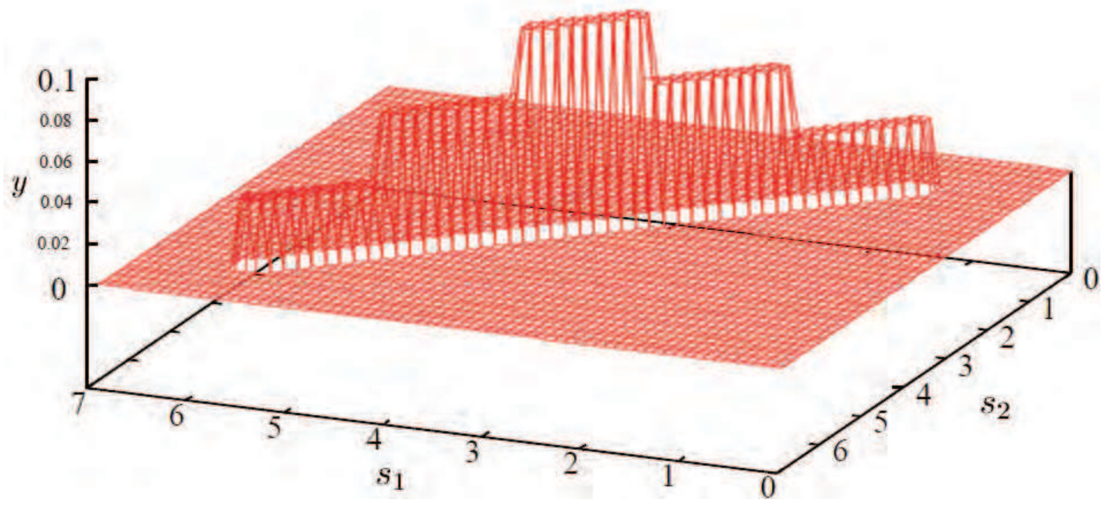

Fig. 4. Output of a SG-CMAC (B)

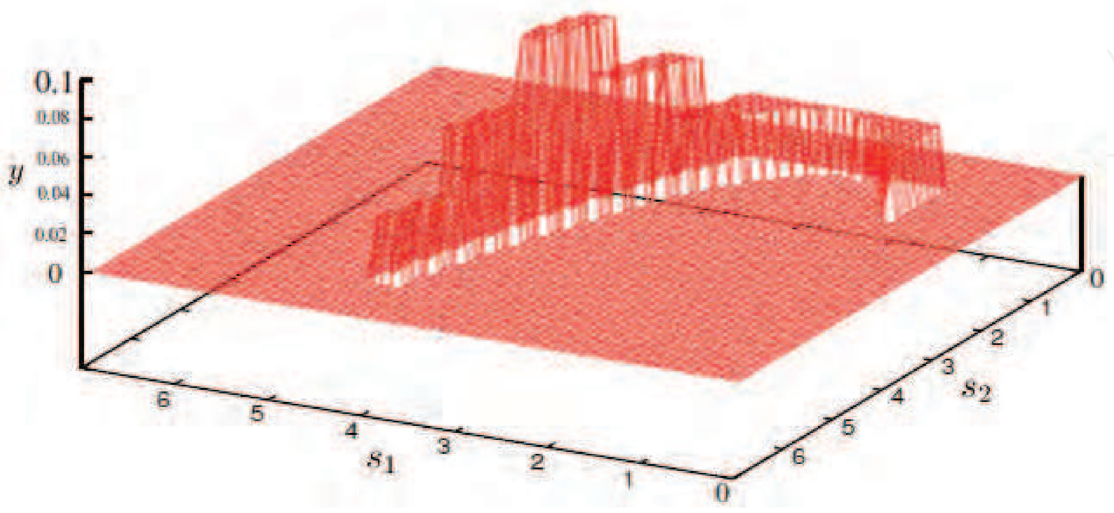

Fig. 5. Output of a SG-CMAC (C)

3. Allocation problem of chemical plant

As an application example, SG-CMAC explained above is used to solve an allocation problem of chemical plant. The objective of the problem is to minimize the total distance of the links under the constraint that the distance of two elements cannot be set smaller than a certain value in order to avoid influence of explosions. Precise roles of the elements included in the original plant are omitted in the simplified problem and every element is called ``unit''. Now, the allocation space is assumed to be normalized into square field, and quantized by square cells that have the same quantization interval. Also, assuming that the number of units is k_c, the number of lattice cells is m_c, each unit is recognized by an unique name c_j $(j = 1, \cdots k_c)$ and the position where a unit is placed is discriminated by discrete position number $1, \cdots m_c$, then, the position of the unit c_j is described by d_j, $(j = 1, \cdots k_c, 1 \le d_j \le m_c)$. The state of the allocation space is determined by $x = \lfloor d_1, \cdots, d_{k_c} \rfloor$. Here, if c_j is not allocated, $d_j = 0$. Since units are allocated into a lattice cell, the maximum number of candidate positions where a units can be allocated is m_c. The unit to be allocated is defined as c_T $(T = 1, \cdots k_c)$ and a position u that c_T is to be allocated is selected from candidate positions $(1, \cdots m_c)$. c_T must be allocated into a position where the distance $l_{Tj}\,(1 \le j \le k_c, T \ne j)$ between c_T and every other unit is larger than certain distances $L_{Tj}\,(1 \le \boldsymbol{j} \le \boldsymbol{k}_c, \boldsymbol{T} \ne \boldsymbol{j})$. Then, the plant is described as $x' = f(x, u)$, where $f(\cdot)$ denotes that allocation of c_T is processed.

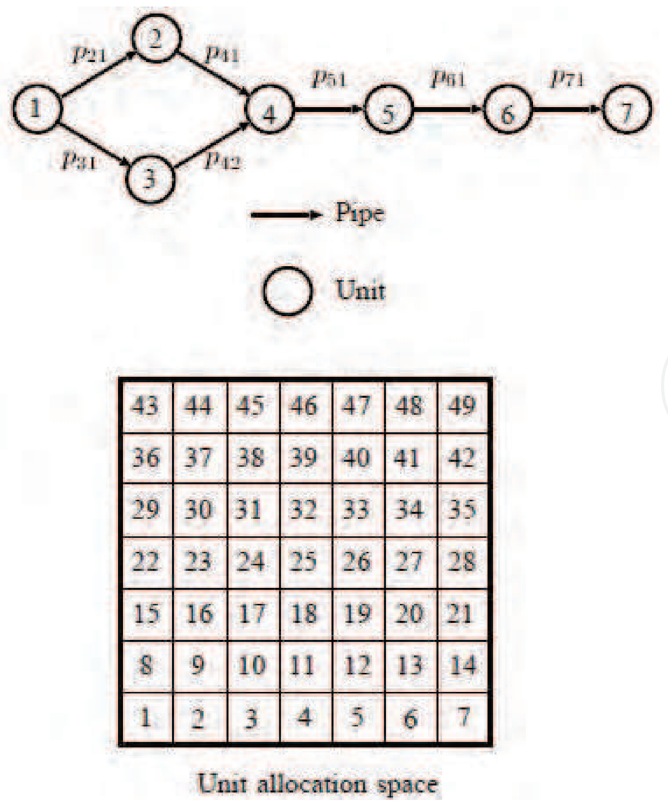

Fig. 6. Plant for allocation problem

Fig.6 shows an example of a plant, where m_c=49, k_c=7. In the figure, positions of units are discriminated by integer $1,\cdots,49$. $p_{ij}\,(i=1,\cdots,7;\ j=1,\cdots,n_T)$ denotes a length of *j*th intake pipe of *i*th unit, where n_T is the number of intake pipe to the *i*th unit. n_T is determined according to product process. In this example, the first unit is a mixer in which some raw materials are mixed before reaction in either of the two reactors (unit 2 or 3). These two reactors produce to intermediate products, which then react to produce the desired product in the next reactor (unit 4). After this reactor follows some purification steps. The first is a simple settler, which separates solids and liquids (unit 5). The desired product is assumed to be in the liquid phase, and is isolated in the crystallizers (unit 6 and 7). Two crystallization steps are needed to get the desired purity of the final product.
The objective of the proposed method is to find the plant layout that can reduce the total length of pipe with minimized risk.

4. A Q-Learning for plant allocation problem

In the conventional Q-learning algorithm, Q-table has to store evaluation-values for all the plant state. In unit allocation problems, the state of the allocation space is described by the positions of all the units *x* and c_T. Since units are allocated by predetermined order, c_T can be determined by units that have already allocated. A Q-value is thus stored for each pair *x*=$\lfloor d_1,\cdots,d_{k_c}\rfloor$and u_i $(i=1,\cdots m_c)$. In this case, the number of states and Q-value is $\prod_{i=0}^{k_c}(m_c-i)$ that increases by the exponential rate with increase of k_c. Moreover, in realistic situations, the number of lattice cells is often large, then required memory size to store information for all the state of the allocation space also becomes large (Baum, 1999).

4.1 Update rules

The proposed learning procedure consists of 3 update rules: (1) to update Q_1 for evaluation of the position of unit 1, (2) to update Q_2 for evaluation of the position of units 1 and 2, (3) to update Q_T for evaluation of the positions of unit *T* $(T=3,\cdots k_c)$. In the update rule (1), the input is the position of each unit u_i $(i=1,\cdots,m_c)$. $Q_1(u_i,x)$ is updated by eq. (4).

$$Q_{1_t}(u_i,x)=(1-\alpha)Q_{1_{t-1}}(u_i,x)+\alpha\gamma\max_{u\in U}Q_{2_{t-1}}(u,x)\,. \tag{4}$$

In the update rule (2), $Q_2(u_i,x)$ is updated when all the units are successfully allocated to the space by using the following rule:

$$Q_{2_t}(u_i,x)=(1-\alpha)Q_{2_{t-1}}(u_i,x)+\alpha\gamma R\,\cdot \tag{5}$$

In the update rule (3), the state is redefined by using the relative position of the *r*th unit pos_r $(r=2,\cdots k_c)$ measured from the position of unit 1 and the angle ϑ_r between the line that links units 1,2 and the line that link unit 1 and *r*th unit $(r=2,\cdots k_c,\vartheta_2=0)$. That is, $x_R=\{x_r\}$ $(r=3,\cdots k_c)$ is the state of the allocation space for the update rule (3), where $x_r = [pos_r, \vartheta_r]$. Also, u_i is redefined as the relative position on the basis of x_{R}. Then $Q_T(u_i, x_R)$ is updated by eq. (6).

$$\boldsymbol{Q}_{\boldsymbol{T}_{t+1}}(\boldsymbol{u}_i, \boldsymbol{x}_{\mathrm{R}}) = (1-\boldsymbol{\alpha})\boldsymbol{Q}_{\boldsymbol{T}_t}(\boldsymbol{u}_i, \boldsymbol{x}_{\mathrm{R}}) + \boldsymbol{\alpha}[\boldsymbol{R} + \boldsymbol{\gamma} \max_{\boldsymbol{u}\in\boldsymbol{U}} \boldsymbol{Q}_{\boldsymbol{T}_t}(\boldsymbol{u}, \boldsymbol{x}'_{\mathrm{R}}), (\boldsymbol{T} = 3, \cdots, \boldsymbol{k}-1) \cdot \quad (6)$$

In plant allocation problems, the objective is to reduce the total pipe length in the allocation space. Thus, in the proposed system, Q-values reflect the pipe length by adjusting the discount factor γ according to the pipe length. In the following, $L_{ij}\,(i = 1, \cdots, k_{\mathrm{c}};\, j = 1, \cdots, n_T)$ is defined as the minimum length of the *j*th intake pipe of *i*th unit, and *U* is defined as a set of positions that satisfy constraints $l_{Tj} > L_{Tj}$ in the allocation space. Then, γ is calculated by using the following equations :

$$\gamma = \begin{cases} 1 & \text{(for update rule (1))} \\ \dfrac{\sum_{i=1}^{k_{\mathrm{c}}} \sum_{j=1}^{n_T} L_{ij}}{\sum_{i=1}^{k_{\mathrm{c}}} \sum_{j=1}^{n_T} l_{ij}} & \text{(for update rule (2)).} \\ \dfrac{\sum_{j=1}^{n_T} L_{ij}}{\sum_{j=1}^{n_T} l_{ij}}, (1 \le i \le k_{\mathrm{c}}, i \ne 2) & \text{(for update rule (3))} \end{cases} \quad (7)$$

where n_T is the number of pipes that link c_T and allocated units, and $l_{ij}\,(j = 1, \cdots, n_T)$ is the length of the pipe that links c_T and the allocated unit. Here, *R* is given only when all the units has been allocated. Propagating Q-values by eqs.(4)-(6) as update rules, Q-values are discounted according to the pipe length. In other words, by selecting a position that has the larger Q-value, the length of pipe can be reduced. Each u_i is selected by the following probability (Sutton & Barto, 1999; Watkins, 1989) :

$$\boldsymbol{P}(\boldsymbol{u}_i, \boldsymbol{x}_j) = \frac{\exp(\boldsymbol{Q}_{t-1}(\boldsymbol{u}_i, \boldsymbol{x}_j)/\mathrm{C}}{\sum_{\mathrm{u}\in\mathrm{U}} \exp(\boldsymbol{Q}_{t-1}(\boldsymbol{u}, \boldsymbol{x}_j)/\mathrm{C}} \cdot \quad (8)$$

where C is a thermo constant. The learning algorithm is depicted in Fig.7.

4.2 Q-tables

In realistic problems the number of lattice cells is large, so that huge memory size is required in order to store Q-values for all the states. Therefore, in the proposed method, only Q-values corresponding states that have been searched are stored for unit *i* ($i = 3, \cdots k_{\mathrm{c}}$). Binary tree is constructed dynamically (Hirashima et al., 2005) during the course of the learning for storing Q-values.

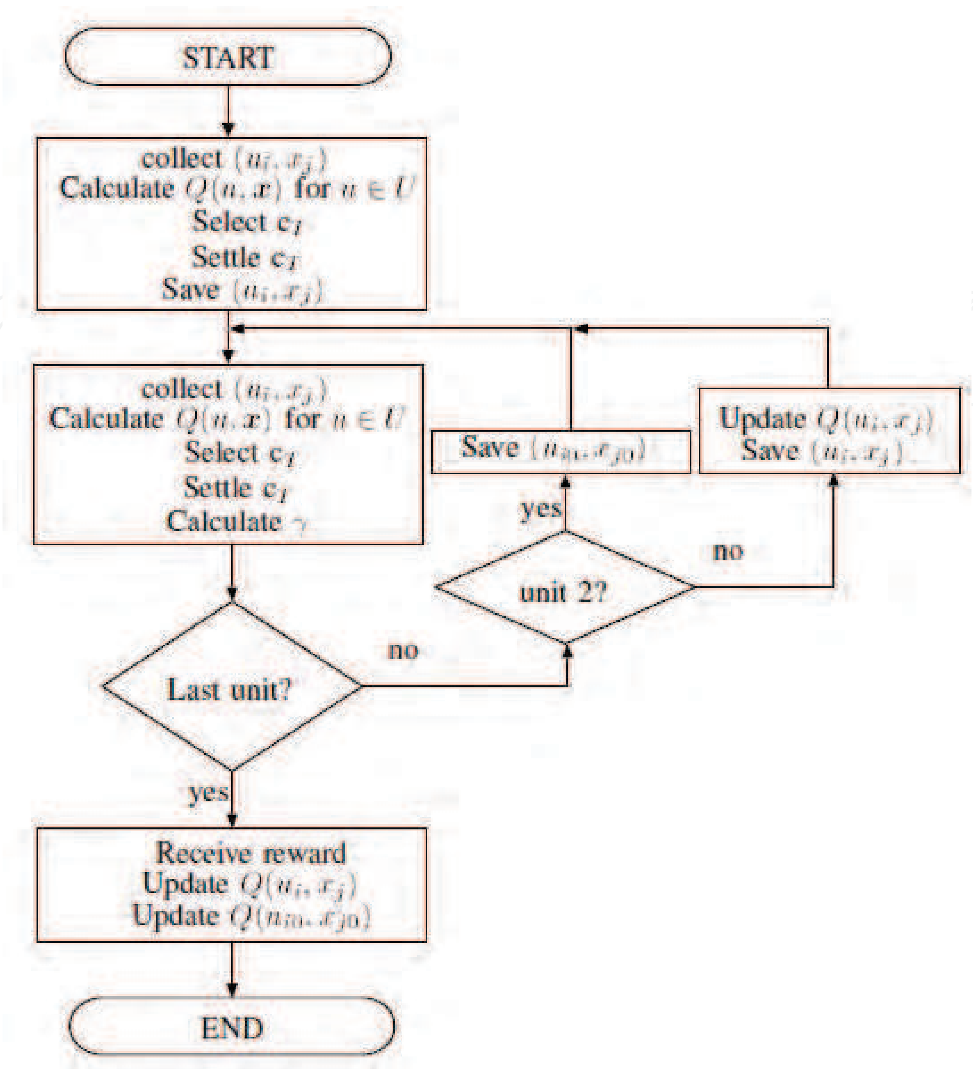

Fig. 7. Flowchart of the learning algorithm

Since similar layouts of the plant have similar evaluations, that is, difference of pipe lengths between such layouts is small, the learning performance can be improved by using appropriate generalization for evaluation of pipe length. By using CMAC as Q-tables for units 1,2, an evaluation for one input (for example, a position of unit 1) can be spread over adjacent inputs. However, the conventional CMAC has fixed shape of region covered by the generalization effects. Then, in Q-table for unit 2, the same evaluation is given to the layout that has longer pipe length and the one that has shorter pipe length when a similar layout is updated in the course of learning. Giving comparable evaluation to longer pipe length as compared to shorter pipe length is not appropriate, and thus, conventional CMAC can spoil the learning performance of the system.

In the proposed method, the CMAC that has selective generalization (SG-CMAC) is used as the Q-table for unit 2. Now, define positions of units 1,2, d_1, d_2 as $d_1 = (d_{1_x}, d_{1_y})$, $d_2 = (d_{2_x}, d_{2_y})$ by the *x*-*y* coordinate. The input of the SG-CMAC is $\mathbf{s} = (\mathbf{s}_I, s_z)$, where $\mathbf{s}_I = x$, $s_z = \sqrt{(d_{1_x} - d_{2_x})^2 + (d_{1_y} - d_{2_y})^2}$. s_z describes the distance between unit 1 and unit 2. Values of s_z are calculated based on quantized values of s_1, s_2. Corresponding CMAC module is assigned to each value of s_z, so that only the positions that have the same pipe length as the position specified by an input are affected by the generalization.

An example of generalization of SG-CMAC is illustrated in Fig.8. In the figure, $\Delta_1 = \Delta_2 = 7$, $\rho = 1$, $g = 0.1$, d=1.0. The position of unit 1 is $(d_{1_x}, d_{1_y}) = (2,3)$ that is blue colored position in the right figure of Fig. 8. The result is obtained after $\mathbf{s}_I$=(4.0,4.0) is given once. Only positions that the distance from unit 1 is same as the input are updated by the generalization of SG-CMAC. These positions are colored by yellow in the right figure of Fig. 8.

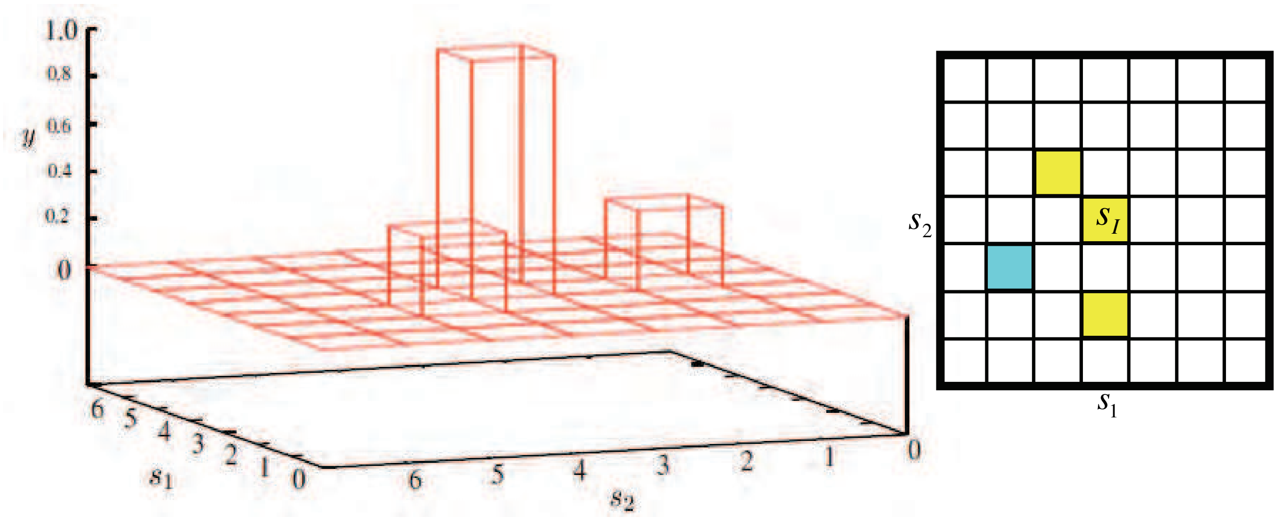

Fig. 8. Generalization of SG-CMAC

The outputs of all the weight tables are summed up to yield the output of the CMAC. For example, the output of the CMAC according to u_i and $\boldsymbol{x}$ is

$$\boldsymbol{q}_t(\boldsymbol{u}_i, \boldsymbol{x}_j) = \sum_{p=1}^{k} \boldsymbol{w}_{p,t}^{\text{ref}} \,. \tag{9}$$

where $w_{p,t}^{\text{ref}}$ $(p = 1, \cdots, k)$ are weights specified by the input u_i for the Q-table at time t. $q_t(u_i, x_j)$ is updated by the update law of the CMAC. In other words, defining the output error at time t as e_t, the desired signal at time t as $Q_t$and learning rate as g, weights are updated as follows:

$$\boldsymbol{e}_t(\boldsymbol{u}_i, \boldsymbol{x}) = \boldsymbol{Q}_t(\boldsymbol{u}_i, \boldsymbol{x}) - \boldsymbol{q}_t(\boldsymbol{u}_i, \boldsymbol{x}) \,. \tag{10}$$

$$\boldsymbol{w}_{p,t+1}^{\text{ref}} = \boldsymbol{w}_{p,t}^{\text{ref}} + \boldsymbol{g}\frac{\boldsymbol{e}_t(\boldsymbol{u}_i, \boldsymbol{x})}{\boldsymbol{k}} \quad (\boldsymbol{p} = 1, \cdots, \boldsymbol{k}) \,. \tag{11}$$

Then the desired signal $Q_t(u_i, x_j)$ is calculated by the Q-learning algorithm eqs.(4)-(6).

5. Computer simulations

Computer simulations are conducted for the same plant described in Fig.6. That is, k_c = 7, m_c = 49. Minimum pipe lengths between two units are set as

$$L_{Tj} = \begin{cases} 2.0\,(T = 2,3,5;\, j = 1) \\ 1.5\,(T = 6,7;\, j = 1) \\ 2.5\,(T = 4;\, j = 1,2) \end{cases}$$

Then, learning performance of the following 3 methods are compared:
(A) proposed method that Q-tables for Q_2 are SG-CMAC,
(B) a method that conventional CMACs are used for Q-tables to store Q_1, Q_2,
(C) a method that conventional table look-up method without generalization is used to construct Q-tables for Q_2.

In CMAC modules in method (A) and CMACs in method (B) for storing Q_2, $\rho = 3, k = 3$. In CMACs in methods (A), (B) and (C) for Q_q, $\rho = 8, k = 6$. Parameters used in the proposed method are set as $\alpha = 0.8, g = 0.6, C = 0.1$. Reward R=1.0 is given only when all the units has been allocated. A trial starts from a initial state and ends when all the units are allocated. Fig.8 shows examples of plant-layout obtained by method (A). For each unit c_T $(T = 1, \cdots k_c)$, lengths of intake pipes l_{Tj} are

$$l_{Tj} = \begin{cases} \sqrt{5}\,(T = 2,3,5;\, j = 1) \\ 2.0\,(T = 6,7;\, j = 1) \\ 2\sqrt{2}\,(T = 4;\, j = 1,2) \end{cases}$$

They are best values that satisfy constrains $L_{Tj} < l_{Tj}$, so that the total pipe length of the optimal layout is $3\sqrt{5} + 4\sqrt{2} + 4 \approx 16.37$.

Fig. 8 shows simulation results. The proposed method could find several optimal solutions, and 4 results that have shortest pipe length are shown in the figure. In the figure, results (I),(II),(III) and (IV) have different layouts, and thus the state x for each solution is different to each other. While, the solution in result (I) is identical to solution (II) with horizontal shift. Also, the solution in result (II) is identical to solution (IV) with rotation, horizontal shift and vertical shift. Therefore, results (I) has the same x_R as result (II) after unit 2 is allocated, and thus, the same Q-value is referred for the layout in the course of learning and input-selecting phase. In the same way, results (III) and (IV) are obtained by using the same Q-values. Once, a good layout is obtained, then, the corresponding Q-value is used for several layouts that are rotated/shifted from the original layout. Moreover, the information of a good layout is spread over adjacent positions of second unit by the generalization capabilities of SG-CMAC. Therefore, the learning performance of the proposed method can be improved as compared to conventional methods that have fixed generalization capabilities. A simulation requires about 5 minutes as runtime and 500KBytes memory on a personal computer that has Pentium4 3GHz CPU.

Fig.9 depicts simulation results where the vertical axis shows the shortest pipe length that is found in the past trials and horizontal axis shows the number of trials. Each result is averaged over 10 independent simulations. In the figure, the proposed method (A) finds optimal solutions that have 16.37 as total pipe length in all simulations, whereas methods (B),(C) cannot. The learning performance of the conventional method (C) is better as compared to method (B) especially in early stages of learning. In method (B), by generalization of the conventional CMAC, inappropriate evaluations are spread over the region adjacent to inputs, so that the learning performance has been spoiled.

6. Conclusions

A design method for CMAC that has selective generalization (SG-CMAC) has been proposed. Also, a Q-learning system using SG-CMAC is proposed, and the proposed system is applied to allocation problem of chemical plant. In the computer simulations, the proposed method could obtain optimal solutions with feasible computational cost, and the learning performance was improved as compared to conventional methods.

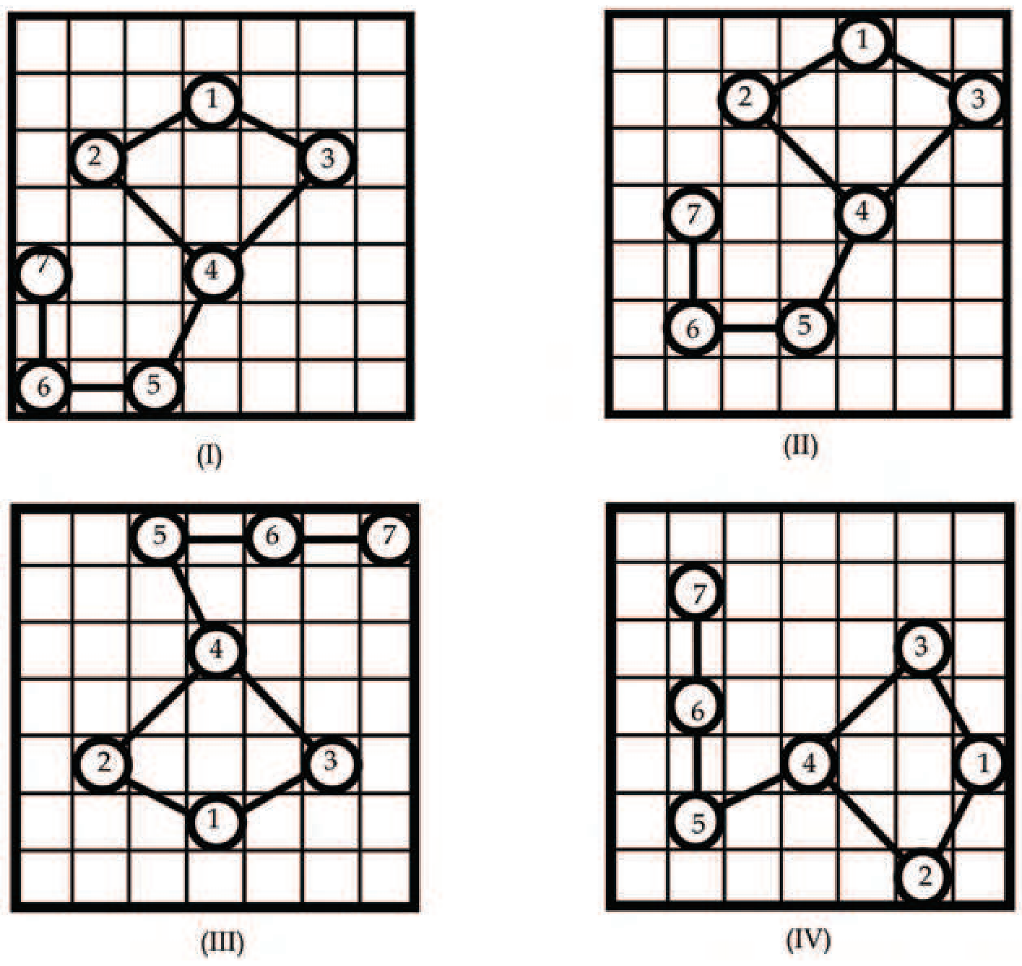

Fig. 8. Optimal lyaouts obtained by method (A)

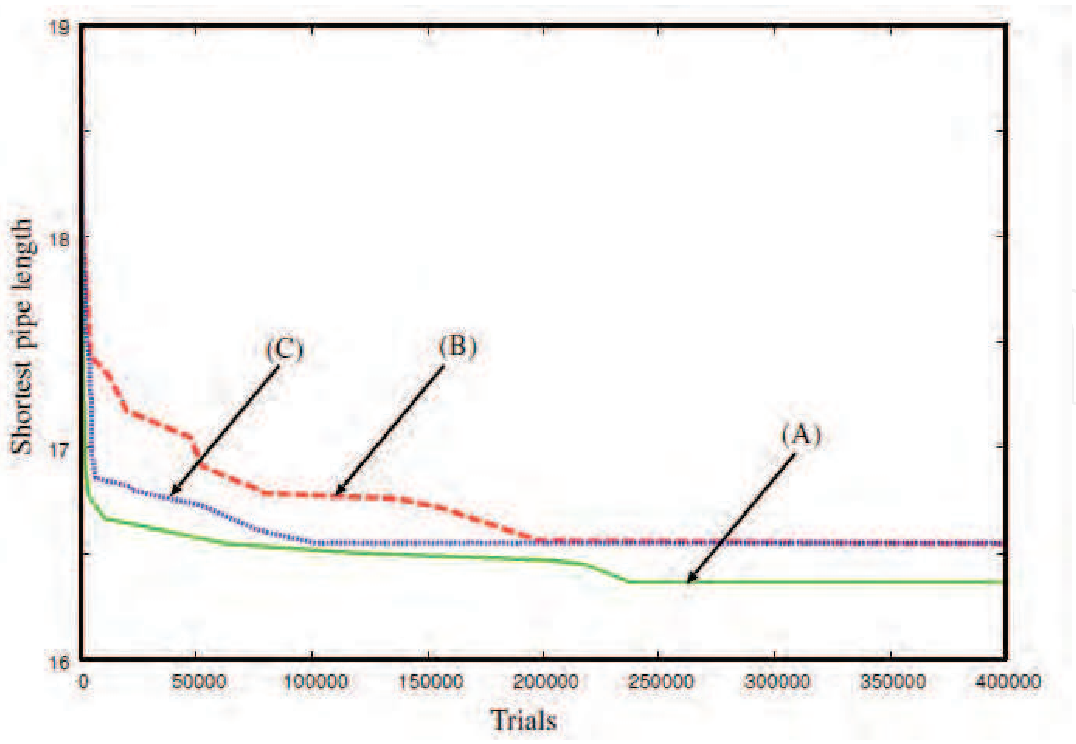

Fig. 9. Performance comparison

6. References

Albus, J. S. (1975a). A New Approach to Manipulator Control : The Cerebellar Model Articulation Controller (CMAC), Trans. ASME J. Dynam. Syst., Meas., Contr., Vol. 97, 220-227

Albus, J. S. (1975b). Data Storage in the Cerebellar Model Articulation Controller (CMAC), Trans. ASME J. Dynam. Syst., Meas., Contr., Vol. 97, 228-233

Baum, E. B. (1999). Toward a model of intelligence as an economy of agents, *Machine Learning*, Vol. 35, 155–185.

Ceorgiadis, M. C., Schilling, G., Rotstein, G. E. and Macchietto (1999). S. A General Mathematical Programming Approach for Process Plant Layout, Comput. Chem. Eng., Vol. 23, No. 7, pp.823-840.

Hirashima, Y., Deng, M., Inoue, A. (2005). An Intelligent Layout Planning Method for Chemical Plants Considering I/O Connections, Proceedings of SICE Annual Conference, pp.2021--2024.

Hirashima, Y., Inoue, A. and N. Jensen, A Method Placing Tanks and Reactors in a Chemical Plant to Minimize Risk Due to Fires and Explosions Using Reinforcement learning, Proc. International Symposium on Advanced Control of Industrial Processes, pp.381-386, 2002.

Sutton, R.S. and Barto (1999). A.G. Reinforcement Learning, MIT Press

Vecchietti, A. R. and Montagna, J. (1998). Alternatives in the Optimal Allocation of Intermediate Storage Tank in Multiproduct Batch Plants, Comput. Chem. Eng., Vol. 22,No. suppl.issue, pp.S801-4.

Watkins, C. J. C. (1989). Learning from Delayed Rewards, Ph.D. Thesis, Cambridge University, Cambridge, England.

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