New Approach to Controller-Adaptor Based Intelligent Control Systems

Liang-Hsuan Chen* and Cheng-Hsiung Chiang**

Department of Industrial Management Science, National Cheng-Kung University, No.1 University Rd., Tainan Taiwan 701, R. O. C.

* E-Mail: lhchen@mail.ncku.edu.tw     ** E-Mail: r3886103@sparc1.cc.ncku.edu.tw

Abstract - A novel approach to intelligent control systems is proposed. It has two main modules: the fuzzy controller and the adaptor, which is to explore new actions to enhance control performance. The simulation results of robotic path planning showed the robot could reach the target point without collisions in various environments.

I. INTRODUCTION

Recently, the concept of human intelligence has been studied in the fields of cognitive science, soft computing, artificial life (A-life), computational intelligence, artificial intelligence, and intelligent control, and so on [4]. Intelligent control systems require the ability to sense the environment, and then to make decisions, and to generate control actions. Higher levels of intelligence may include the ability to recognize objects and events, to represent knowledge, and to reason about and plan for the future [1]. The three basic elements of intelligent control system are: perception, decision-making, and action.

The “intelligence” is consisted of multiple elements; therefore, it has to combine more than one technology to realize intelligent control system. Several intelligent computing technologies are becoming useful to construct the intelligent control systems such as fuzzy logic, neural networks, evolutionary algorithms, case-based reasoning, expert systems, reinforcement learning, affective computing, etc. For intelligent control research, some studies aimed at the conceptual model [1]-[2], [5], and some proposed the computing technology-based model [3]-[4], [6], [8]. Many intelligent control systems are applied for robotic systems that the typical problem is path planning of a mobile robot [3]-[4], [8]. One interesting application is the plant (such as tomato) producing system [6].

Albus [1]-[2] proposed the conceptual architecture of intelligent systems with four functional elements: sensory processing (SP), value judgment (VJ), world model (WM), and behavior generation (BG). The objective of SP is to transform information from sensor into meaningful representations. From the inputs of SP, WM constructs, updates, and maintains a knowledge database. VJ is then to compute the cost, risk, or benefit of actions and plans. The BG plays the role of planner and executor to develop schedules for possible actions or to produce output commands directly in the environment. This study offered an exhaustive introduction of intelligent model to imitate human’s intelligence but did not introduce how to realize the intelligent system by actual algorithms, such as artificial intelligence technologies.

Fukuda et al. [8] proposed the hierarchical intelligent control, which included a neural network and the knowledge-based approximation, for robotic manipulators in 1994. The model is constructed by symbols, which can be treated as objects containing their own characteristics, and attributes. This model is simple to design but we have to know the prior knowledge. Furthermore, different applications we have to design different symbolic knowledge and they did not offer a general manner to design the knowledge base.

Fukuda et al. [4] proposed the architecture of structure intelligence for a robotic path planning in 1999. For comparing with hierarchical intelligent control, this model performed the better designing efficiency and generality. Also, fuzzy controller is used instead of the neural controller and the knowledge is based on fuzzy rules but not symbols. They applied evolutionary algorithms to learn the membership functions and fuzzy rules in fuzzy controller. The sensory network is proposed to modify fuzzy rules in different environments. The model aimed at the robotic path planning. For another applications, we have to find other suitable manner to redesign the algorithms (e.g. sensory network).

However, for generating new behaviors, most intelligent control systems are based on the pre-designed mathematical formulation. It is lack of flexibility in the various environments. If the environments are unpredictable then the generated behaviors may not suitable. In addition, for adapting environments, most studies aim at the special applications but loss the generality. We proposed the new approach of intelligent control system named CAICS (Controller-Adaptor based Intelligent Control System). It is to generate new behaviors based on heuristic algorithm—multi-objective genetic algorithm (GA). Thus, the CAICS does not have to know the mathematical model for updating the system’s behavior and it can be generality used. Moreover, we proposed the three-stage adaptive mechanism to adapt various environments and it is suitable any complex environments.

Fuzzy controller is applied to realize the controller of CAICS, and adaptor is used to generate new behaviors to adapt different environments. CAICS is proposed to improve our previous study [3] to enhance the designing efficiency. CAICS is comprised of three functions: performance evaluator, controller, and adaptor. The three
functions correspond to perception, action, and decision-making described in [4]. Two elements: action explorer and rule constructor are built in adaptor to explore new actions and transform them to fuzzy rules. An application of robotic path planning is proposed to demonstrate the adaptive ability of CAICS.

II. THE ARCHITECTURE OF CAICS

Fig. 1 illustrates the block diagram of proposed intelligent control system. The important modules in CAICS are the controller and adaptor; therefore we called it the Controller-Adaptor based intelligent controller systems. According to the perceptual information from the environment, the performance evaluator evaluates the performance for controller. If the performance is satisfactory then keep on controlling the plant (controlled system). While the performance judged by performance evaluator is unsatisfactory, enable adaptor to explore new control actions.

There are two elements in adaptor: action explorer and rule constructor. We proposed the three-stage adaptive mechanism: return stage, exploration stage, and lead stage to implement action explorer. It is to explore the new actions for controller. Rule constructor then transforms these new actions to fuzzy rules and updates the corresponding rules in fuzzy controller. We apply multi-objective GA to realize action explorer. The three-stage adaptive mechanism corresponds human’s adaptive thinking: thinking back (return stage) to ensure when did the wrong decisions, thinking how to explore new decisions (exploration stage), and then thinking how much effort should be done (lead stage).

The return stage is to find the number of returning steps from current action (called failure point), which is caused poor performance, to past action (called return point). The lead stage is to determine the number of exploring actions, and exploration stage is to explore new better actions. After finding out new actions, rule constructor will transform them to fuzzy rules and replaces the corresponding rules in controller.

III. FUZZY CONTROLLER

The simple and intuitive Mamdani fuzzy inference system is applied to the controller. We have compared the control efficiency with fuzzy neural network-based controller [3] on the same application: robotic path planning through simulation. For the same fuzzy rules and numerical inputs, the simulation results of two controllers have similar performance. However, fuzzy neural network have to learn the parameters, therefore, it takes more learning time than fuzzy inference system. In general, a fuzzy if-then rule can be expressed as follows:

IF \( x_1 \) is \( A_1 \) … and \( x_n \) is \( A_n \)

THEN \( y_1 \) is \( B_1 \) … and \( y_o \) is \( B_o \)

where \( A_i \) and \( B_i \) are membership functions for the \( i \)th input and output, and \( n \) and \( o \) are the numbers of input and output. Here, we apply the Gaussian membership functions for input and output variables as follows.

\[
\mu_{Ai}(x) = \exp \left( -\frac{(x-c_{Ai})^2}{\sigma_{Ai}} \right) \\
\mu_{Bi}(y) = \exp \left( -\frac{(y-c_{Bi})^2}{\sigma_{Bi}} \right)
\]

(1)

(2)

where \( c_{Ai} / c_{Bi} \) and \( \sigma_{Ai} / \sigma_{Bi} \) are the central values and the widths of the input/output in the \( j \)th linguistic term of the \( i \)th input variable. Consequently, the fuzzy AND operation is to calculate the pair product \( P_k \) as follows.

\[
P_k = \prod_i \mu_{Ai}, \quad k = 1, 2, ..., n \quad \prod N_i
\]

(3)

\( N_i \) is the number of membership functions for input variable \( x_i \). For each input variable, pick up one membership function for calculating pair product value. Then, the fuzzy OR operation is to calculate the maximal value of \( P_i \) shown in Eq. (4)

\[
\gamma_i = \max \{ P_i \}, \quad i = 1, 2, ..., n; \quad j = 1, 2, ..., M_i
\]

(4)

where \( M_i \) is the number of membership functions for output variable \( y_i \). Eventually, the defuzzification can be computed as follows.
The three-stage action explorer can be treated as optimization problem as

$$\text{Max}./\text{Min.} \left\{ g_1(S_1, S_2, S_3), \ldots, g_k(S_1, S_2, S_3) \right\}$$

Subject to:

$$l_1 \leq S_1 \leq u_1$$
$$l_2 \leq S_2 \leq u_2$$
$$S_3 = [A_1, A_2, \ldots, A_{S_2}] \subseteq \mathcal{Z}$$
$$l_1, u_1, l_2, u_2 \in \mathbb{Z}$$

There are $k$ objective functions ($g_i, \ i = 1, 2, \ldots, k$) and three decision variables ($S_1, S_2$, and $S_3$). For the robotic path planning, the objective functions are to minimize the moving distance and avoid collisions. $S_1$ (return stage) and $S_2$ (lead stage) specify how many steps should we return and explore, and they belong to positive integer $\mathbb{Z}$. $S_3$ is a vector of new actions ($A_i, \ i = 1, 2, \ldots, S_2$) that is, the output of controller) belongs to feasible solution space $\mathcal{Z}$.

B. Genetic Algorithm for Action Explorer

There are many types of encoding way applied to specific problems, such as binary-encoded genes, discrete-encoded genes, real-encoded genes, gray-encoded genes, symbol-encoded genes, hybrid-encoded (combine numbers and symbols) genes, etc. In order to reduce the complexity of GA, the discrete-encoded type is suitable for our problem Eq. (7). Because the storage space of discrete-encoded type is smaller than binary-encoded type and the computing time is also less than real-encoded type. In fact, the discrete-type and symbol-type encoding ways are equivalents. Fig. 3 shows the encoding way of a chromosome.

The fitness function and the operators of discrete GA are introduced as follows.

1) Fitness function: $\text{fit}$

The fitness function $\text{fit}$ is weighted sum of objective functions formulated as follows.

$$\text{fit} = w_1 \cdot g(S_1, S_2, S_3) + \ldots + w_k \cdot g_k(S_1, S_2, S_3)$$

$$w_1 + w_2 + \ldots + w_k = 1$$

The three-stage action explorer is shown in Fig. 2. While the adaptor is enabled (the current control action is failure point), we first have to return to past (return point). Fig. 2 shows us two steps are returned (from point 6 to 4). Next, we explore seven new actions from point 1’ to 7’. The lead point (equal to 7) indicates how many actions should we explore. The new action-stream found in action explorer can be expressed as \{1 2 3 4 1’ 2’ 3’ 4’ 5’ 6’ 7’\}.

IV. ACTION EXPLORER

Action explorer is a decision maker to search some feasible decisions and determine the best one. It is the core of adaptor and it is enabled, as the performance of controller is poor. We use the simple rule to judge the performance in performance evaluator as follows.

$$J = \begin{cases} 
1, & \text{if performance is satisfied} \\
0, & \text{if performance is unsatisfied} 
\end{cases}$$

where $J$ denotes two values as 0 and 1 to specify the satisfactory and unsatisfactory situations. If $J = 0$, then the action explorer and rule constructor in adaptor will be activated. The procedure of three-stage adaptive mechanism and its mathematical model are introduced in the following. Furthermore, we engineered the GA to search the feasible solutions of the mathematical model.

A. Formulate the Three-Stage Action Explorer

The illustration of action explorer is shown in Fig. 2. While the adaptor is enabled (the current control action is failure point), we first have to return to past (return point). Fig. 2 shows us two steps are returned (from point 6 to 4). Next, we explore seven new actions from point 1’ to 7’. The lead point (equal to 7) indicates how many actions should we explore. The new action-stream found in action explorer can be expressed as \{1 2 3 4 1’ 2’ 3’ 4’ 5’ 6’ 7’\}.

Fig. 2 Illustration of three-stage action explore

The fitness function and the operators of discrete GA are introduced as follows.

1) Fitness function: $\text{fit}$

The fitness function $\text{fit}$ is weighted sum of objective functions formulated as follows.

$$\text{fit} = w_1 \cdot g(S_1, S_2, S_3) + \ldots + w_k \cdot g_k(S_1, S_2, S_3)$$

$$w_1 + w_2 + \ldots + w_k = 1$$

A gene (discrete value)

Fig. 3 Illustration of a chromosome

The fitness function and the operators of discrete GA are introduced as follows.

1) Fitness function: $\text{fit}$

The fitness function $\text{fit}$ is weighted sum of objective functions formulated as follows.

$$\text{fit} = w_1 \cdot g(S_1, S_2, S_3) + \ldots + w_k \cdot g_k(S_1, S_2, S_3)$$

$$w_1 + w_2 + \ldots + w_k = 1$$
The nonnegative weights $w_i, i = 1, 2, ..., k$, is decided by prior knowledge or the requirements of environment.

2) Genetic operation strategies

The conventional GA only performs three basic genetic operators: selection, crossover, and mutation. Nearcho [7] proposed another three operators: swap, insertion, and deletion. We presented a new operator: shift operator and combined other six operators (selection, crossover, mutation, swap, insertion, and deletion) for use in the proposed GA. The illustration of genetic operators is shown in Fig. 4. The shift operator is to shift each gene from right/ left to left/ right one or more positions.

V. RULE CONSTRUCTOR

Rule constructor is a simple fuzzy rule-generating algorithm proposed by Wang and Mendal [9]. We described the main steps of the algorithm as follows.

Step 1) Determine the input and output variables.
Step 2) Confirm the membership function for each input/output variable.
Step 3) Form the numerical data set. Here, the $k$th numerical data set can be formed as

\[(x'_1(k), x'_2(k), ..., x'_n(k)) \rightarrow (y'_1(k), y'_2(k), ..., y'_n(k))\]  

(10)

Step 4) Calculate the membership degree. For each input value, $x'_i(k)$, and corresponding output $y'_j(k)$, their membership degree is constructed for each variable by a row vector denoted by $\bar{x}_i(k)$ for inputs and $\bar{y}_j(k)$ for outputs as:

\[
\bar{x}_i(k) = \left[A^1_i(k), A^2_i(k), ..., A^N_i(k)\right] \quad (11)
\]

\[
\bar{y}_j(k) = \left[B^1_j(k), B^2_j(k), ..., B^M_j(k)\right] \quad (12)
\]

\[A^1_i(k), ..., A^N_i(k) \text{ and } B^1_j(k), ..., B^M_j(k) \] specify the membership degree for $x'_i(k)$ and $y'_j(k)$, respectively, according to Eq. (1)-(2).

Step 5) Assign importance degree to each data pair. The importance degree for each data pair as shown in Eq. (13) can be calculated as:

\[h(k) = \prod_{i=1}^{n} \text{max}\{\bar{x}_i(k)\} \quad (13)\]

In order to calculate the importance degree, first, pick up the maximum value for each vector $\bar{x}_i(k)$ and $\bar{y}_j(k)$, next, multiply them to get the importance degree.

Step 6) Construct fuzzy rules for each numerical data. Find the corresponding membership degree of $\text{max}\{\bar{x}_i(k)\}$ and $\text{max}\{\bar{y}_j(k)\}$ for each input/output variable. For instance, a fuzzy rule has this form:
Step 7) Delete the conflict fuzzy rules. As two rules have the same fuzzy sets in IF part but a different set in THEN part. The proper rule is selected according to highest importance degree.

VI. APPLICATION: ROBOTIC PATH PLANNING

A. Simulation Conditions and System Design

Path planning is an important task in navigation of autonomous mobile robots. We show the computer simulation results here. The robot is the cylinder structure and the goal is a circle zone. Given a start point of the robot, and it has to move forward to the goal with no collision. The size of the robot and goal are 1.5 and 3. The start and goal points are (10, 10) and (90, 90), respectively. Steering angle is restricted between 0 and 180. Moving distance of a robot is fixed as 2.25 units and rectangular obstacles based on the same start and goal points as first simulation (Fig. 6). The adaptor is enabled to explore a new path. The mobile robot first moves back to return point form failure point and moves forward to the lead point along the new explored path. The number of explored actions is 8 (5); Two new fuzzy rules are found as follows:

\[ R'(30): \text{IF} \quad d \quad \text{is} \quad \text{SC} \quad \text{and} \quad \phi \quad \text{is} \quad RS \quad \text{THEN} \quad \theta \quad \text{is} \quad FR \]

\[ R'(39): \text{IF} \quad d \quad \text{is} \quad F.A \quad \text{and} \quad \phi \quad \text{is} \quad RS \quad \text{THEN} \quad \theta \quad \text{is} \quad FR \]

The original THEN parts in \( R(30) \) and \( R(39) \) are all "\( \theta \) is RS". The changed surface is shown in Fig. 6 (b).

Third, the number of obstacles is increased to 20 and the simulation results are shown in Fig. 7. There are 3 collisions here; therefore, the adaptor is enabled three times. The numbers of explored actions are 22, 15, and 6 respectively (5). Seven new fuzzy rules are found as follows:

\[ R'(5): \text{IF} \quad d \quad \text{is} \quad GC \quad \text{and} \quad \phi \quad \text{is} \quad FR \quad \text{THEN} \quad \theta \quad \text{is} \quad RS \]

\[ R'(6): \text{IF} \quad d \quad \text{is} \quad GC \quad \text{and} \quad \phi \quad \text{is} \quad LT \quad \text{THEN} \quad \theta \quad \text{is} \quad RS \]

\[ R'(13): \text{IF} \quad d \quad \text{is} \quad VC \quad \text{and} \quad \phi \quad \text{is} \quad RT \quad \text{THEN} \quad \theta \quad \text{is} \quad RS \]

\[ R'(14): \text{IF} \quad d \quad \text{is} \quad VC \quad \text{and} \quad \phi \quad \text{is} \quad FR \quad \text{THEN} \quad \theta \quad \text{is} \quad LS \]

\[ R'(22): \text{IF} \quad d \quad \text{is} \quad MC \quad \text{and} \quad \phi \quad \text{is} \quad RT \quad \text{THEN} \quad \theta \quad \text{is} \quad FR \]

\[ R'(30): \text{IF} \quad d \quad \text{is} \quad SC \quad \text{and} \quad \phi \quad \text{is} \quad RS \quad \text{THEN} \quad \theta \quad \text{is} \quad RB \]

\[ R'(31): \text{IF} \quad d \quad \text{is} \quad SC \quad \text{and} \quad \phi \quad \text{is} \quad RM \quad \text{THEN} \quad \theta \quad \text{is} \quad FR \]

The original THEN part in above fuzzy rules are FR, LT, RT, FR, RT, RS, and RM, respectively. The control surface shown in Fig. 7 (b) is obviously changed from Fig. 5 (b). After the seven fuzzy rules are updated, the robot will move forward to the goal point based on new fuzzy rules in the future simulation.

VII. CONCLUSIONS

We proposed an intelligent control system with controller-adaptor architecture named CAICS. It can generate new behaviors for controller but not just modify the parameters in controller. The three-stage action explorer is proposed to explore new actions in adaptor. While the controller’s performance is poor it will be enabled and...
then rule constructor transforms explored actions to fuzzy rules to update corresponding rules in controller. The simulation results of robotic path planning demonstrated the CAICS is able to adapt various environments through changing its rule base.

Model the high-level knowledge (e.g. natural language) based system is a subject in the future. Good approaches of knowledge representation and reasoning can result more intelligent for a control system.

REFERENCES