Classifier Systems As Robot Architecture For Behavior Coordination

Bushra Abdulkareem Abdulazeez
Foundation Of technical Education
Institute Of Technical Medicine/Baghdad

ABSTRACT

Classifier systems (CS) are used as control architectures for robots in order to decide what to do at each time. We will explain in this research why these systems are good candidates for behavior coordination. We will introduce our system called RACE for collecting objects according to their colors , a race is held between two robots one of them is controlled by our system the other is controlled by another external system , the job of our robot is to collect balls before the other robot catch them.

أنظمة التصنيف كمعمارية للانسان الآلي في تنسيق السلوك

بشرى عبد الكريم عبد العزيز
هيئة التعليم التقني
المعهد الطبي التقني

ملخص البحث

تُستعمل أنظمة التصنيف كهندسة سيطرةٍ معمارية للإنسان الآلي لكي يقرر كل عمل في وقته . سَنُوضّح في هذا البحث أسباب ترشيح هذه الأنظمة في تنسيق سلوكِ الإنسان الآلي . سنقدم نظامِنا المسمى سباق RACE المصمم لجمع الأجسام بالاعتماد على لونِها , حيث يُعقد سباق RACE بين اثنين من الرجال الآليين أحدهم تَحْتَ سَيْطرَة نظامِنا بينما الآخر تَحْتَ سَيْطرَة نظام خارجي ، واجب انساننا الآلي أنْ يَجْمعَ الكراتَ حسب لونها قبْلَ أنْ يَمْسكُ بها الانسان الآلي الآخرَ.

استخدمنا في هذا النظام نظامَ تصنيف موزّعَ بهندسة معماريةِ هرمية المستوى Hierarchical Architecture .

1. Introduction

A classifier system (CS) is a machine learning system that learns syntactically simple string rules, called classifiers. These classifiers guide the system's performance in an arbitrary environment.

A classifier system derives its name from its ability to learn to classify messages from the environment into general sets and is similar to a control system in many respects. As a control system uses feedback to "control" or "adapt" its output for an environment, a classifier system uses feedback to "teach" or "adapt" its classifiers for an environment [1,2]. The classifier system has developed out of the merging of expert systems and genetic algorithms as originated by Holland [1975]. This synthesis has overcome the main drawback to expert systems; namely, the long task of discovering and inputting rules. Using a genetic algorithm, the CS learns the rules needed to perform in an environment [3].

Classifier system is an induction self-learning system based on a set of simple logical rules called classifiers. Each rule has the following structure: "if <condition> then <action>". Classifier system is being optimized by using learning rule called "bucket brigade" and evolutionary algorithms (genetic algorithms). During learning process rules priorities (strengths) are changed. In case of success current and previous activated rules are encouraged. Evolutionary methods are used for new rules searching.

2. Simple Classifier System

Classifier System is a kind of method of Genetic Algorithm-Based Machine Learning (GBML). CS consists in Performance Function based on Production System, Reinforcement Learning based on Credit Assignment and Rule Generation Function based on GA [4].

According to Holland , a classifier system consists of four principal components [5]:

- **List of classifiers** (population of classifiers).
- **List of messages** that plays the role of a "message board" for communications and short term memory.
- **Input interface** (detector) that represents the environment state.
- **Output interface** (effector) that ensures interaction with the environment or its change.

A Classifier System receives environmental information through sensors called ***detectors***. This incoming *message* is decoded into some standard message

-format (may be one or more messages). These messages are posted to a finite-length *message list* where the messages can then activate the classifiers from the ***classifier store***. If activated, a classifier may then be chosen to send a message to the message list for the next cycle. These messages may then invoke other classifiers or they may cause an action to be taken through the system's action triggers called ***effectors***. Messages are bit strings of fixed length k defined as <**message**> **:: = {0, 1}** . Each classifier consists of a condition and action part given as :

<classifier> :: = <condition> : <action> where <condition> ::= {0, 1, #}. Each condition specifies a subset of the set of all messages and is also of *length k*. The action is a message that can be sent to the message list. Whether the candidate classifier posts its message is determined by the outcome of an activation auction, which in turn depends on the evaluation of the classifier's value [6].

The alphabet of condition coincides with the alphabet of messages and additionally has the symbol "#" (don't care) which means that the symbol may be either from alphabet A_{msg}. $A_{cond}=A_{msg}+\{\#\}$

For example, for alphabet $A_{cond}=\{0,1,\#\}$ the results of comparison can look like this:

Condition	Message	Matched
10##	1010	Yes
10##	1001	Yes
10##	1110	No

When the condition part of the classifier matches the input message, activation of the classifier occurs, i.e. the classifier puts one or more messages on the message list [7].

The input interface can be a device or subprogram that generates messages of length *L*. For example, if the classifier system is employed to control a robot, the input interface can be a set of detectors. They can transfer information on the objects that surround the robot [8].

The output interface is a device or subprogram that receives action messages and on their basis performs manipulations with the environment. For example, in the case of a robot, the output interface can be a set of devices (effectors) and can activate certain actions with their help (turn left or right, move forwards, etc.).

The main classifier system cycle is as follows:

1. ***Add messages obtained with the help of the input interface to the input message list.***
2. ***Compare all the messages on the message list with the condition parts of all classifiers and remember all the classifiers for which coincidence has been observed.***
3. ***Create new messages by activating the coinciding classifiers, i.e. perform some classifier actions.***
4. ***Pass the messages obtained through the output interface.***
5. ***Replace the contents of the message list with new messages.***
6. ***Go to Step 1***

The Bucket-Brigade Algorithm (**BBA**) is designed to solve the credit assignment problem for classifier systems and to determine the worth of each classifier. The credit assignment problem is that of deciding which of a set of early active classifiers should receive credit for setting the stage for later successful actions. To this end, a numerical quantity (called strength) is assigned to each classifier. This strength is adjusted continually by the algorithm to reflect the classifier's past usefulness. Each classifier whose condition part is satisfied by one or more messages makes a bid to post a message onto the message list. Only the highest bidders are allowed to become active and hence post messages. A classifier's bid depends on its strength and the specificity of its condition. The specificity measures the relevance of a classifier's condition to a particular message. Formally, a classifier's bid is defined as in equation (1) [9,10]:

$$Bid = a*strength*specificity \quad \text{……………………………………..(1)}$$

where **specificity** = number of non # /Total Length of condition part; and **a** = a constant less than 1. The winning classifiers place their messages on the message list and their strengths are reduced by the amount of their bids. Typically, if Sc(t) denotes the strength of a winning classifier, c, at time t and Bc(t) denotes its bid at the same time t, then its strength at time t+1 is as in equation (2):

$$Sc(t+1) = Sc(t) - Bc(t) \quad \text{……………………………………..(2)}$$

In other words a winner pays for posting a message on the list. Classifiers that sent messages matched by this winner have their strengths increased by the amount of its bid. For example, if c' sent the message matched by c above, then the strength of c' at time t+1 is :

$$Sc'(t+1) = Sc'(t) + Bc(t) \quad \text{……………………………………..(3)}$$

When system goals are attained, a pay-off (reward or punishment) is added to the strength of all the classifiers that are active at that time.

Since the Bucket Brigade algorithm only changes the classifier strength, the quality of the system heavily depends on the initially generated population. To avoid this shortcoming, genetic algorithms (discovery algorithms, evolutionary algorithms) can be employed.

A simple genetic algorithm is executed as follows:

Crossover :

The GA crosses the classifiers in the normal way. Example in Fig.1 describes crossover between two strings:

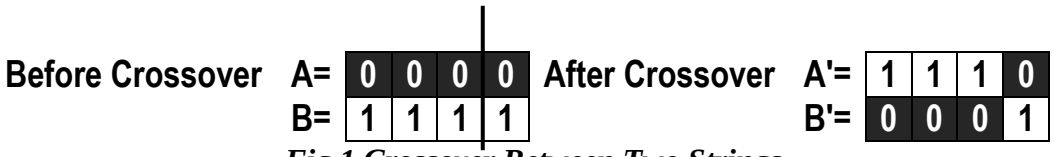

Fig.1 Crossover Between Two Strings

Classifiers to be used in crossover are selected by Roulette Wheel Selection.

Mutation:

Mutation is the random alteration of a string position .Mutation plays a decidedly secondary role in the operation of GA , it is needed because , even though reproduction and crossover effectively search and recombine extant notions, occasionally they may become overzealous and loss some potentially useful genetic material (1's or 0's at particular locations). In a simple GA, mutation is the occasional (with small probability) random alteration of the value of a string position Fig.2.shows mutation.

Before Mutation

After Mutation

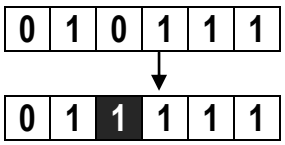

Fig. 2 Mutation in GA

The searching, not for the single best rule (classifier), but for a well adapted set of rules. The "crowding replacement" algorithm is used to choose the classifiers that should die to make room for new offspring. (Note that this implies combining the best of the parents and offspring.) Crowding replacement aims to replace a low performing classifier with a similar (potentially better) classifier). The crowding algorithm is [11]:

for 1 to crowing factor do
x := find worst of a random set
if this is more similar to offspring then past x then
set worst most similar to x
endif
endfor
replace worst most similar with offspring

Now the basic execution cycle of a learning classifiers system can be written as follows besides the description in Fig.3 :

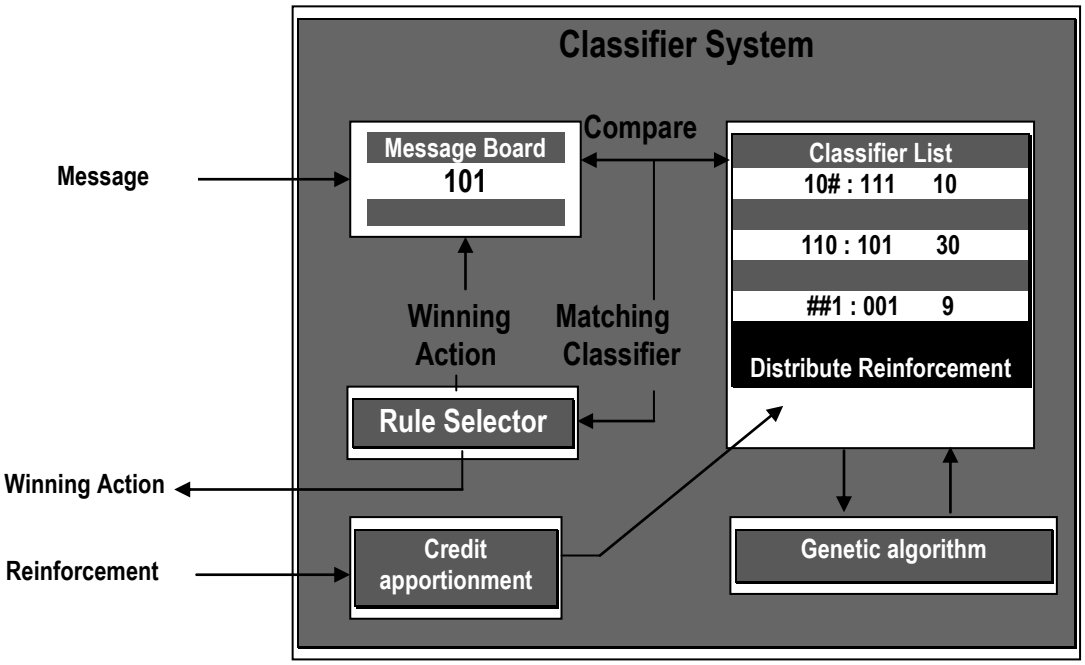

Fig. 3 Classifier System Interpretation

3. Behavior Coordination in Robot

Behavior Coordination Problem deals with "How can an agent select the most appropriate or the most relevant next action to take at a particular moment, when facing a particular situation" [12]:

A coordinator is needed in order to send only one command at a time to the motors. Fig.4 shows Structure of Behavior-based architecture.

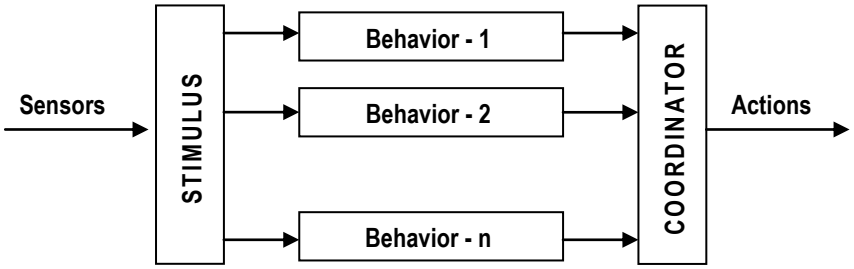

Fig.4 Structure of Behavior-Based Architecture

There are two primary coordination mechanisms to assemble behaviors competitive methods and cooperative methods [13,14]:

Competitive methods can often be viewed as a winner-takes-all strategy in which the single response for the winning behavior suppresses all the others and is directed to the robot's actuators for execution. Competitive methods result in the selection of the output of a single behavior, either by arbitration of action-selection. Arbitration requires that a coordination function serving as an arbiter selects a single behavioral response. The selection is usually based on some sort of predefined hierarchy. Action-selection methods select the output of a single behavior in a less autocratic manner. Here the behaviors actively compete with each other through the use of activation levels driven by both the robot's goals and incoming sensor information. The behavior with the highest activation level wins.

Cooperative methods offer an alternative to competitive methods by using behavioral fusion. This provides the ability to use the output of more than one behavior at a time. The central issue in combining the outputs of behaviors is finding a representation that allows fusion.

4. The RACE System

The system RACE is built to mimic the behavior of the natural racing persons in collecting objects , the COLLECTOR (our robot) should chase and collect balls which colored by red , blue, green and yellow color , in the same time there are another Robot (COMPETITOR) that competes with our robot for collecting balls.

The simulation environment of the system is a pool of a , 32 cells width and 32 cells length . Fig. 5 shows the objects that can be seen in this environment (Fig.5) , which are:

1- An artificial COLLECTOR Man that should learn how to approach balls and collect them . The COLLECTOR 's sensors are two eyes (cameras) each one limited with visual fields of 270º and one ear. The eye is able to detect

the presence of an object in it visual field . The ear is used to detect a signal to start chasing a specified ball for instance:

1 beep : for red ball;

2 beeps for blue ball;

3 beeps for green ball;

4 beeps for yellow ball.

the other job for ear is to detect warning signal (for instance continuous beeps) that appears when the COMPETITOR closed to the ball that our robot should catch it . Finally the COLLECTOR Man is provided with 4 wheel driver for walking .

2- The Second objects are 4 balls red , blue, green and yellow that change their position by the human trainer randomly, and the Man should catch them according to the order coming from the environment.

3- The COMPETITOR  that also programmed to compete with our robot COLLECTOR in order to catch the balls.

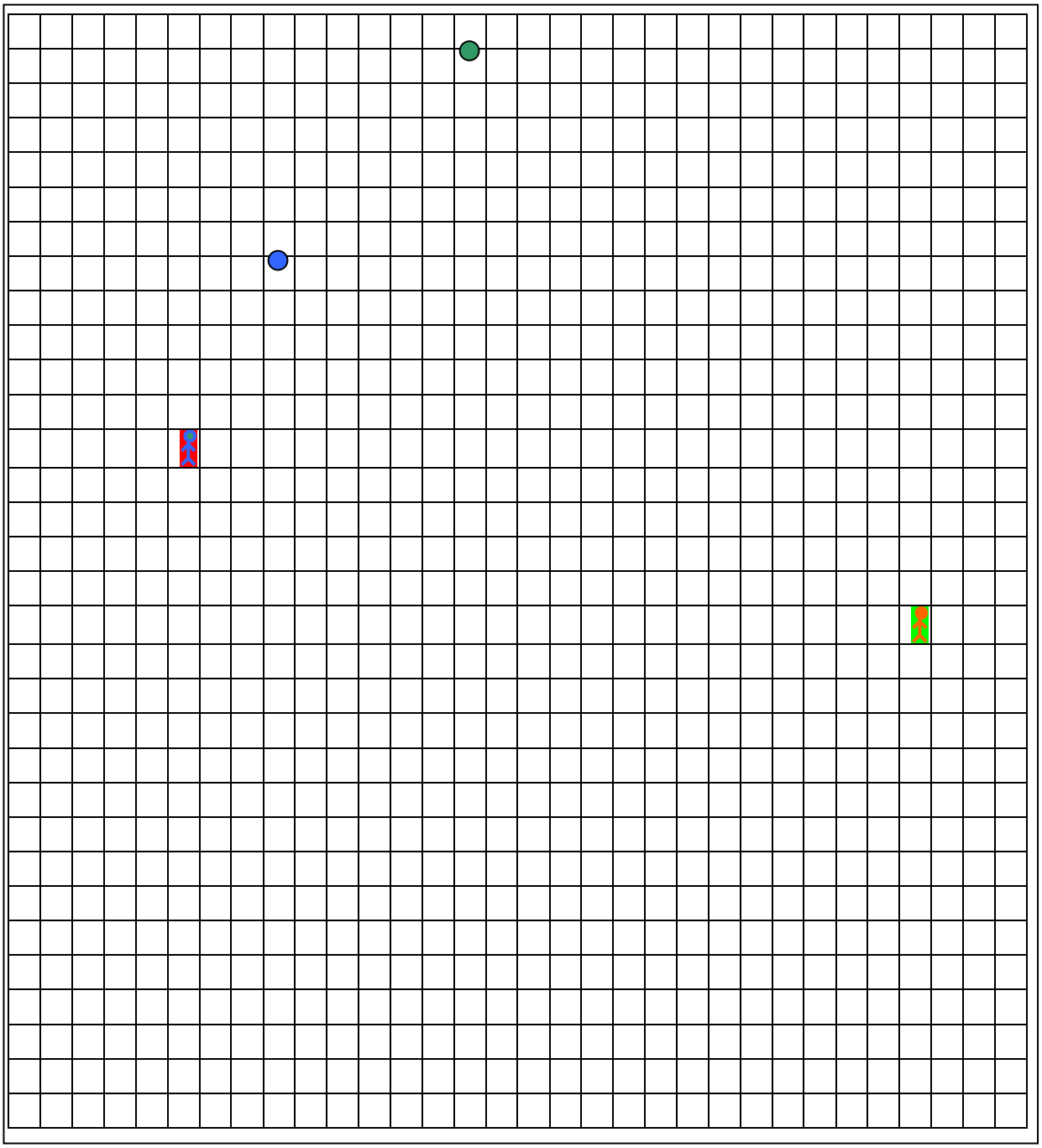

Fig.5 The Whole Simulation of the Environment of (COLLECTOR)

Fig.5 The Whole Simulation of the Environment of (RACE)

4.1 The Structure of (RACE)

The system RACE consists of three Learning classifier systems: Coordinator , 1-step and 2-steps organized in hierarchical architecture as shown in Fig 6.

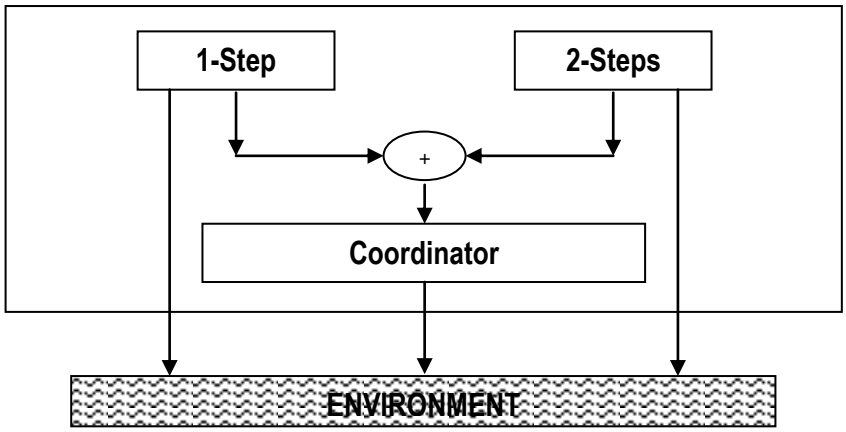

Fig. 6 The Architecture Of The System (COLLECTOR)

4.2 The Coordinator LCS

The LCS *Coordinator* is used to learn how to coordinate between two behaviors after its analysis of the environmental message which is received from the environment. The behaviors are move one-step toward specific ball and two-steps if it hears warning sound .

The environment message which Coordinator received it from the environment is always 10-bits as shown in Fig. 7.

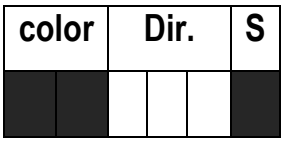

Fig. 7 The Structure of Environment Message of Coordinator_LCS

The meaning of bits is as follows:

Bit0,bit1(color of ball)

00: color of ball is red;

01: color of ball is blue;

10: color of ball is green;

11: color of ball is yellow.

Bit2,bit3,bit4 (The Ball's position)

The above bits are used to determine the relative position of the ball from COLLECTOR as shown in table-1 .Fig.8 shows the directions of COLLECTOR movement.

Table-1 : Shows the meaning of bit4,bit5 and bit6.

Input mes.	Its meaning
000	Relative position of ball from the COLLECTOR is to the Front
001	Relative position of ball from the COLLECTOR is to Front-Right
010	Relative position of ball from the COLLECTOR is to the Right
011	Relative position of ball from the COLLECTOR is to the Back-Right
100	Relative position of ball from the COLLECTOR is to the Back
101	Relative position of ball from the COLLECTOR is to the Back-Left
110	Relative position of ball from the COLLECTOR is to the Left
111	Relative position of ball from the COLLECTOR is to the Front-Left

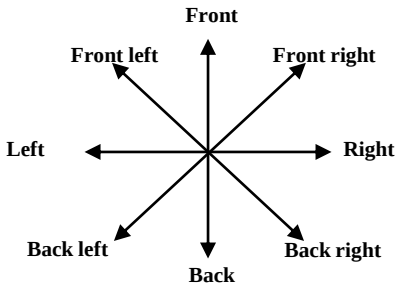

Fig.8 Shows the Directions of COLLECTOR Movements

Bit5 (Warning sound)

the above bit are used to warning the COLLECTOR by a sound if the distance of the second robot COMPETITOR becomes very closed to the ball.

The action of Coordinator is always 1 bit

If action = 1 then the message sent to **1-step_LCS** .

If action = 0 then the message sent to **2-steps_LCS** .

4.3 The 1-step_LCS

The 1-step_LCS is used to learn the COLLECTOR how to move one step toward cage in case of shark existence.

The input message that 1-step-LCS received it from the Coordinator_LCS is always 5-bits as shown in Fig.9.

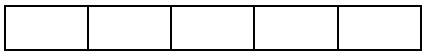

Fig.9 The Structure of Input Message of 1-step_LCS

The bits above are shown as follows:

Bit0,bit1(color of ball)

00: color of ball is red;

01: color of ball is blue;

10: color of ball is green;

11: color of ball is yellow.

Bit2,bit3,bit4 (The Ball's position)

The above bits are used to determine the relative position of the ball from COLLECTOR as shown in table-1 .

The action of 1-step-LCS is always 3-bits denoting to the movement of COLLECTOR toward ball.

Example:

Input Message						*Action*		
0	0	0	0	1	:	0	0	1

The above example means that if the color of wanted ball is red and relative position of it from the COLLECTOR to the front-right , then COLLECTOR should move one step toward ball according to that direction.

4.4 The 2-steps- LCS

The 2-steps_LCS is used to learn the COLLECTOR how to move two steps toward ball and catch it up .

The input message that 2-step-LCS received it from the Coordinator_LCS is always 5-bits as shown in Fig.10.

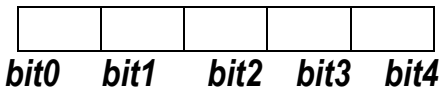

Fig.10 The Structure of Input Message of 2-steps_LCS

The bits above shows the Relative position of the ball from the COLLECTOR as shown in table 1.

The action of 2-steps-LCS is always 3-bits denoting to the movement of COLLECTOR toward ball.

Example:

					Input Message				***Action***
0	0	0	0	1	:	0	0	1	

The above example means that if the color of wanted ball is red and relative position of it from the COLLECTOR to the front-right , then COLLECTOR should move one step toward ball according to that direction.

4.5 Rule and Message System of (COLLECTOR)

The rule and message system is the main component in LCS , it contains two main parts : *classifier sore* and *message list* . The classifier store contains a set of rules called *classifiers* each classifier consists of two parts: *condition* and *action* , the part that should be matched with the environment message is the condition part.

The message list is used to collect messages coming from the environment . In each cycle only one message is sent from message list to the classifier store that means no other message to be received in the current cycle until the system produces an action.

As the rule and message system is the heart of the LCS , the matching procedure is the heart of the rule and message system. There are two routines responsible for matching classifiers to environment message :

1- *match* which performs a match between a single condition and a single message returns a Boolean true value if the match succeeds . The match performed position(*bit*) by position(*bit*) .

2- *matchclassifiers* that matches all classifiers against the environment message and construct a *messagelist* data structure .

The classifier store of rule and message system in *Coordinator_LCS* consists of (74) classifiers . The condition part of each classifier consists of (10) bits while the action part is (1) bit. All classifiers have the same strength value at the beginning.

The classifiers in our system are stored in a file to be read in the main program after running that program. The message List for *Coordinator_LCS* is an array to store the messages coming from the environment .

The classifier store of rule and message system in *Escape_LCS* consists of (80) classifiers . The condition part of each classifier consists of (3) bits while

the action part is (3) bits. All classifiers have the same strength value at the beginning. As in the message list of the *Coordinator_LCS* , we used an array to store the messages

The classifier store of rule and message system in *Approach_LCS* consists of (80) classifiers . The condition part of each classifier consists of (3) bits while the action part is (3) bits. All classifiers have the same strength value at the beginning. As in the message list of the *Coordinator_LCS* , we used an array to store the messages coming from the environment .

4.6 The Reinforcement Learning of (COLLECTOR)

Reinforcement algorithm is preferable to providing it by a human trainer who observes the performance of the system and provides positive or negative reinforcement according to action to be taken.

The reinforcement is provided in many ways by using reward and punishment , that is by rewarding positive action and punishing negative action or by only using reward to provide it. Using a reward algorithm only to provide reinforcement and the amount of reward is (10) to reward good action and (0) for bad action for the all three LCSs which are used in the COLLECTOR system. The description of the above statements is as follows:

If *classifier output = desired output* **then**

add (*reward =10*) to classifier strength responsible for sending action

Else

add (*reward = 0*) to classifier strength responsible for sending action

4.7 The Apportionment of Credit

The apportionment of credit of our system consists of two main component *auction* and *clearinghouse*. Classifier will be in match list when its condition matches the message coming from the environment. With auction algorithm we find the winner classifier which is one of the match list classifier[4].

Since classifier makes bids (B_i) during the auction, then winning classifier .turns over its bids to the clearinghouse payments (P_i). A classifier bids in proportion to its strength will be:

$$B_i = Cbid * (bid_1 * bid_2 * specify) * S_i \quad \text{……..(4)}$$

where Cbid is the bid coefficient, choosing Cbid =0.1 . Si is the strength with index i for each classifier in match list and (bid_1*bid_2*specifity) is a linear function of the specificity. Specificity represents the number of non #'s bit in condition part of classifier. The more specific rules are the best ones.

To choose an auction's winner we will calculate an effective bid (*EB*) for each matched classifier as the sum of its deterministic bid and a noise term as in equation (5)[4]:

$$Eb_i = Cbid * (ebid_1 * ebid_2 * specifity) * S_i + N(\sigma bid) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots(5)$$

where noise **N** is a function of the specified bidding noise standard deviation (σbid), also separate coefficients are used for the effective bid specificity function ($ebid_1$ and $ebid_2$), thereby providing additional flexibility for investigating alternative bidding structures.

The function auction keeps track of the classifier index with highest effective bid and returns this value to procedure AOC. The winner pays its bids to the classifier responsible for sending a matching message.

Together, auction, clearinghouse, and taxcollector distribute and collect payments and taxes to help assure that good rules receive high strength and bad rules receive relatively low strength. Thereafter strength may be used as a fitness measure to facilitate a genetic search for new, possibly better rules.

4.8 Rule Discovery of (COLLECTOR)

The COLLECTOR is needed away of injecting new possibly better rules into the system. By using the genetic algorithm new rules are created by *reproduction*, *crossover* and *mutation*. These rules are then placed in the population and processed by the auction, payment and reinforcement mechanism.

The GA's are applied with a much lower frequency than the perform and apportionment of credit systems.

The COLLECTOR calls genetic algorithm by defining a quantity called genetic algorithm period T_{ga}. In this period the system reaches to the steady state.

1- the robot detects some environmental features.
2- encode the environmental information into a string message.
3- the message is compared to the condition part of each classifier in the classifier list.
4- one classifier among those whose condition part matches to the current message.
5- the corresponding action command is either directly sent to the effectors , or deposited on the message board.
6- in the latter case, the corresponding action message may be matched to the condition part of other classifiers, and the process returns to step (3).

in the former case, the behavior corresponding to the activated effectors is displayed in the environment .

7- each time an action command is executed, the strength value of the corresponding classifier is incremented or decremented relative to the resulting positive or zero outcome.

8- several classifiers are selected at the same time, the classifier with the highest strength value has the greatest probability of being activated.

9- new classifiers may be created by a genetic algorithm, according to so-called *mutations* and *crossovers* operators acting on classifiers with high strength values. Other classifiers may be removed from the list if they are associated with low fitness values.

5. Conclusion

Behavior Coordination Problem is the formulation of effective mechanisms for coordination of the behaviors' activities into strategies for rational and coherent behavior.

learning classifier systems have given rise to the field of evolutionary robotics especially in coordination of behavior field , robots are no longer restricted to simple repetitive task but are now able to explore new and uncharted horizons.

References

1- Guillot A. And J.A. Meyer. 2000. "From Sab94 To Sab2000 : What's New, Animat ?". In *From Animals To Animats 6*, J. A. Meyer, A. Berthoz, D. Floreano, H. Roitblat, And S. W. Wilson (Eds.), 3-12.

2- D. E. Goldberg . "Genetic Algorithms In Search Optimization, And Machine Learning". Addison – Wesley Publishing Company, Inc ,1989.

3- Robert A. Richards, 1995. "Zeroth-Order Shape Optimization Utilizing A Learning Classifier System", 1995.

4- D. Katagami And S. Yamada , "Interactive Classifier System For Real Robot Learning", Tokyo Institute Of Technology , Japan,2000.

5- Holland J. H. "Adaptation In Natural And Artificial Systems". Ann Arbor: The University Of Michigan Press, 1975.

6- Nikhil Bhatia, Prashant Dixit, Mohit S.od ,Gamble: Genetic Algorithm Based Machine Learning Expert, Indian Institute Of Technology, Roorkee ,India,2000.

7- A. Vasilyev . "Classifier Systems Learning In Dynamic Environment" ,1999.

8- Riolo R. L. Cfs-C: A Package Of Domain Independent Subroutines For Implementing Classifier Systems In Arbitrary, User-Defined Environments , Logic Of Computers Group, Division Of Computer Science And Engineering, 1988.

9- Paolo Pirjanian ."Behavior Coordination Mechanisms - State-Of-The-Art" , Usc Robotics Research Laboratory , University Of Southern California ,Los Angels, October 7, 1999 .

10- Michael O. Odetayo , "Machine Learning Using A Genetic-Based Approach". School Of Mathematical And Information Sciences, Coventry University, Uk ,1991.

11- Michael O. Odetayo , "Machine Learning Using A Genetic-Based Approach". School Of Mathematical And Information Sciences,Coventry University, Uk ,1991.

12- Pattie Maes. "How To Do The Right Thing". Technical Report Ne 43 - 836, Ai-Laboratory, Massachusetts Institute Of Technology, Cambridge, USA, 1989.