Reward effect in Reinforcement Learning Systems

Lubna Zaghlul Bashir * and Zina Waleed*

Received: 5/10/2010
Accepted: 25/8/2011

Abstract

Learning Classifier Systems (LCS), are a machine learning technique which combines reinforcement learning, evolutionary computing and other heuristics to produce adaptive systems. The system **HRC** ***(Human – Rat - Cheese)*** focuses in creating artificial creature **(Rat)** using computer simulation, and learning it how to choose between two different basic behaviors, (approach / escape) combining them to perform complex behavior, which represents the final response in changing environment.

The HRC is built of two-classifier subsystems working together, each classifier system learns a simple behavior, and the system as a whole has as its learning goal the control of activities. Flat architecture was used. The flat organization allows distinguishing between two different learning activities: the learning of basic behavior and the learning of switch behavior. One classifier system learns basic behavior, (approach/escape), i.e., it is used to learn the simulated robot single step movement in every direction in the environment. Whereas the other classifier system learns to control the activities of basic classifier systems, i.e., it is used to learn to choose between basic behaviors using suppression as a composition mechanism to chose between two basic behaviors which represent complex behavior.

Simple experiments were executed for HRC: comparing and contrasting the effect of the reinforcement learning using reward & punishment with learning using reward only. Experiment results show that the run using reinforcement learning with reward only is unable to perform as well as the run with reinforcement learning with reward and punishment.

*Department of Building and Construction Engineering, University of Technology

1. Introduction

The ability to learn is an essential component of intelligent behavior in human society. Individual humans do not need to learn everything by discovering it from scratch for themselves. Instead, they learn from their peers and teachers by exchanging the knowledge and information that they have acquired. Learning from peers does not occur only in humans. It has also been found in some invertebrates, in many birds, aquatic mammals and of course primates. When reinforcement learning is used to solve problems that are composed of many different stages, learning is like a search process, in which the agent searches the world for states that maximize reward and thus minimize punishment. The time this search takes depends strongly upon the size and structure of the state space and upon *a priori* knowledge encoded in the learning agent's initial parameters. When *a priori* knowledge is not available, the search through the state space is unbiased and can be excessively long. In nature, co-operative mechanisms help to reduce the size of the search area, and hence the search time, by providing the learner(s) with auxiliary sources of experience. Furthermore, such mechanisms can allow agents to successfully learn problems that would otherwise be too difficult [7].

2. Reinforcement Learning

Reinforcement learning is a trial and error approach to learning, in which an agent explores its environment to learn how to achieve its task(s). The agent learns by adjusting the actions it takes in different situations, on the basis of positive or negative feedback that it receives whilst trying those actions. This feedback only specifies how good or bad the chosen actions were. It does not indicate which other actions would have been better or worse. The agent should converge to the policy (set of actions) that maximizes the received reward [7].

Reinforcement learning allows an agent to learn sequential decision tasks by trial and error interaction with the environment. The agent is able to perform one of a number of *actions* in order to interact with the environment. The environment is represented by a number of parameters which collectively define the current *state* and the task of the learning process is to determine the appropriate action to apply for any given state. The various techniques that comprise reinforcement learning are used to derive a matrix of state action pairs that determine behavior which is known as a *policy*. Reinforcement learning model is shown in Figure (1) [5].

Reinforcement learning differs from *supervised learning* which provides the learning process with information on the correct action to select for a particular state. Instead of information on the appropriate action to perform, the reinforcement learning process receives a *reward* value as it operates in the environment. This reward is matched with information on the current state and used to create a policy which provides progressively better action selection as the agent gains experience in the environment [5].

3. What is a Learning Classifier System (LCS)?

A Learning classifier system can be considered as a black box having detectors in input (input interface), effectors in output (output interface) and placed in a certain environment. The input interface obtains information about the environment and the messages sent by this interface can be thought of as description of the environment in different states. The bits contained by the messages picked up by the output interface control the effectors.

Messages are passing between input and output interfaces according to a set of "IF-THEN" rules and also called classifiers. These messages are binary strings and they are kept in a data structure called the "message list". The message list contains a finite number of messages having a fixed format and containing only 0 and 1. Classifier conditions contain the alphabet {0, 1, #} and are matched against messages in the message list. The symbol # is a wildcard and matches either a 0 or a 1 in the input. If the conditions of a classifier are met, the classifier is activated. This rule-based system learns simple rules to make decisions and to guide the system it controls in the environment. Figure (2) illustrates architecture of learning classifier system. [6, 9, 10].

Learning classifier systems contain two learning sub-systems:

● **Credit assignment sub-system**: This sub-system evaluates the usefulness of the existing rules in the population. It assigns each of them strength and selects the best one in case of a conflict resolution because more than one classifier can apply at a time. The design of this sub-system is based either on a Bucket Brigade [1] or Q-learning algorithms [12]. This work uses Bucket Brigade algorithm.

The Bucket Brigade algorithm (BBA) service economy contains two components: an *auction* and a *clearinghouse*. When classifiers are matched, they do not directly post their messages. Instead, having its tradition matched qualifies a classifier to participate in an activation auction in which it maintains a record of its net worth, called its strength. Each matched classifier makes a *Bid* proportional to its strength in this way; rules that are highly fit (have accumulated a large net worth) are given preference over other rules. The auction permits appropriate classifiers to be selected to post their messages. The selected classifier must clear its payment through the clearinghouse, paying its bid to other classifiers for matching message rendered. A matched and an activated classifier send its bid B to those classifiers responsible for sending the messages that matched the bidding classifier's condition. The bid payment is divided in some manner among the matching classifiers. This division of payoff among contributing classifiers helps ensure the formation of an appropriately sized sub population. Thus different types of rules can cover different types of behavioral requirements without undue inter species competition [1,12].

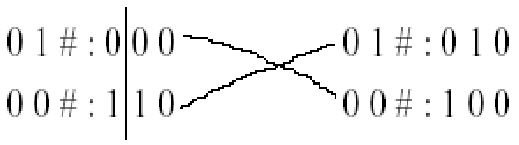
● **Rule discovery subsystem**: Rules constantly need to be generated and the less useful ones must be deleted in the purpose of learning. The rule discovery sub-system often consists of a Genetic Algorithm (GA) which can either use the calculated strength calculated by the credit assignment sub-system to evaluate the fitness of IF-THEN rules, or else use a new measure based on the accuracy of these rules [3, 9].

Most of the LCS belong to this strength based type and they use a measure called "strength" to evaluate the usefulness of existing rules. This strength parameter serves as a predictor of future pay off and also as the classifier's fitness for the Rule discovery sub-system and its genetic algorithm [9, 11].

A situation may arise when the environmental message string may not find any matching classifier. The system should be robust enough to deal with such situations. This is dealt by unleashing the power of Genetic Algorithms. The tripartite process of reproduction, crossover and mutation is used to produce temporary classifiers. The fitness function for these classifiers is in accordance with the message sent.

- **Crossover**

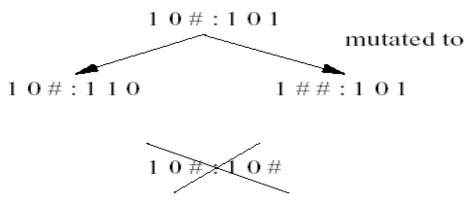
The GA crosses the classifiers in the normal way For example:



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

- **Mutation**

Mutation must now randomly select 1 of {0, 1, #} for the condition part of the classifier and 1 of {0, 1} for the action part. The action part is not allowed to have a wild card in it. For example:



Basic Operation of LCS

The cycle of an LCS, which is repeated sequentially is as follows:

1. The input interface generates messages which are posted into the message list
2. The messages are matched against the conditions of all classifiers and if the matching rules don't advocate the same action, conflict resolution occurs and an action is chosen.
3. The message list is emptied and the encoded actions are posted to the message list
4. Finally we activate the output interface from the message list and perform the actions they describe. We get rewards back from the environment and update the predictions of our classifiers [3, 9, 11]. Figure (3) illustrates LCS components.

4. Learning Classifier System (LCS) for Reinforcement Learning

Reinforcement learning consists of cycles in which a learning agent is presented with an input describing the current environmental state, responds with an action and receives some reward as an indication of the value of its action. The reward received is defined by the *reward function*, which maps state/action pairs to the real number line, and which is part of the problem definition. For simplicity we will consider only *single-step* tasks, meaning the agent's actions do not affect which states it visits in the future. The goal of the agent is to maximize the rewards it receives, and, in single-step tasks, it can do so in each state independently. In other words, it needs not consider sequences of actions in order to maximize reward.

When an LCS receives an input it forms the *match set* [M] of rules whose conditions match the environmental input. The LCS then selects an action from among those advocated by the rules in [M]. The subset of [M] which advocates the selected action is called the *action set* [A]. Occasionally the LCS will trigger a reproductive event, in which it calls upon the GA to modify the population of rules.

We will consider LCS in which, on each cycle, only the rules in [A] are updated based on the reward received – rules not in [A] are not updated [8].

4.1. The Standard Ternary LCS Language

A number of representations have been used with LCS, in particular a number of variations based on binary and ternary strings. Using what we'll call the *standard ternary LCS language* each rule has a single condition and a single action. Conditions are fixed length strings from $\{0,1,\#\}^l$, while rule actions and environmental inputs are fixed length

strings from $\{0,1\}^l$. In all problems considered here $l = 1$.
A rule's condition C matches an environmental input m if for each character m_i the character in the corresponding position C_i is identical or the wildcard (#). The wildcard is the means by which rules generalize over environmental states; the more #s a rule contains the more general it is. Since actions do not contain wildcards the system cannot generalize over them [8].

4.2. Strength-Based Fitness

In traditional strength-based systems, the fitness of a rule is called its *strength*. This value is used in both action selection and reproduction [8]. Strength is determined by following operations:

- **An Auction**

When condition parts of classifiers are matched by one or more messages they do not directly post their messages. Instead, having its tradition matched qualifies a classifier to participate in an activation auction in which it maintains a record of its net worth, called its strength. Each matched classifier makes a ***Bid*** proportional to its strength in this way; rules that are highly fit (have accumulated a large net worth) are given preference over other rules. The auction permits appropriate classifiers to be selected to post their 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 [2]. In other words a winner pays for posting a message on the message list [2, 12].

- **Clearinghouse**

The selected classifier must clear its payment through the clearinghouse, paying its bid to other classifiers for matching message rendered. A matched and activated classifier sends its bid to those classifiers responsible for sending the messages that matched the bidding classifier's condition. The bid payment is divided in some manner among the matching classifiers. This division of payoff among contributing classifiers helps ensure the formation of an appropriately sized sub population. Thus different types of rules can cover different types of behavioral requirements without undue inter species competition [10, 12]. 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 [12].

- **Taxation**

Each classifier is taxed to prevent freeloading, thereby biasing the population toward productive rules. There are two types of tax:

A. ***Life Tax*** removes redundant classifiers in the system and encourages the more frequent acting classifiers .It removes a tax at each time step from every classifier in the population, so fervors classifiers that can replace the lost strength. Unfortunately, it weakens classifiers in infrequent niches and chains starting classifiers**.** Life tax controls classifiers that ***never*** bid whereas, bid tax controls classifiers that bid too much.

B. ***Bid Tax*** is removed from classifiers in the match set that bid to control the auction. It is separate from the bid, which is only removed from ***unsuccessful*** active classifiers. The problem of over generals can be reduced as their strength is reduced to a greater extent than accurate generals (including those in default hierarchies). The bid tax sets a balance between specific and defaults classifiers' stable strength and so controls the likelihood of selection in rule discovery. This balance can be poor, as it is set prior to training [10].

5. Human – Rat- Cheese (HRC) System

The system focuses on creating an artificial creature (Rat) using computer simulation, and on learning it how to choose between two different basic behaviors, combining them to perform complex behavior, which represents the final response in a changing and unpredictable environment. Simulated Rat has an environment as in Figure (4):

- The environment will be a two dimentional grid of 32X32cells.This limitation of environmental size will be more realastic and will be fit to use in simulation program.
- In an environment we will use a piece of cheese which has a fixed position.
- the third creature (human) dose not appear in the environment but is only heard as a sound and is alwayes supposed to be far enough, so that the artificial Rat always has the time to run.
- There is a lair in our environment ,which is used by Rat for hidding.

Simulated Rat will have a whole view of its environment by using sensors in the main direction.

In this work we use two learning classifier systems interacting together to perform the complex behavior. The function of the two classifier systems will be as follows:

1. First classifier system (LCS-1) is used to teach simulated Rat how to move a single step in an eight direction to perform approaching or escaping behavior.
2. Second classifier system (LCS-2) is used to choose between the two basic behaviors to perform the last response.

The global behavior of the system Human – Rate – Cheese (HRC) is illustrated in Figure(5).

6. HRC System Structure.

The system is built of two different classifier systems. These two classifier systems will be learned independently and they interact together to produce an action as output to the environment. The structure of HRC system is shown in Figure (6).

6.1. LCS-1 for learning movement.

The first learning classifier system is used to learn artificial Rat to move a single step in each cycle towards lair in escaping behavior when there is a signal that the human will appear and to move one step towards the piece of cheese when there is a signal that the human is far. The movement capability is completely symmetric along the two axes.

6.1.1. Coding LCS-1 environmental message.

The way that the environmental message is coded differs from one problem to another depending on the environmental parameters involved and on the type and number of sensors used to detect them. A mapping procedure should be created to map the sensor output to the coding scheme used in the system, depending on the type of sensors used. The message will enter the system in an acceptable form. We could imagine that our simulated rat has four sensors that sense the relative position of the piece of cheese and lair in the environment and detects environmental message. The mapping process is not difficult. The detected message is information about relative position of cheese or lair from artificial rat. Relative position is represented by an eight direction coded as three bit from (0 to 7). The possible direction is in Figure (7).

Since messages will have the form and meaning as in Table (1)

6.1.2 Coding LCS-1 actions.

The desired action should be the same as the system environmental message. Therefore we have eight legal actions. Action has the form and meaning as in Table (2).

The performance system is the heart of CS. It is also called (rule and message system). LCS-1 consists of a message list and classifier store. There is only a single message in the message list that is used to match against a condition part of all classifier stores and there is no other message to be received in the current cycle until the system produces an action. The classifier store of LCS-1 contains a set of rules called classifiers, which represents the knowledge and controller of the system at execution time.

Each classifier within LCS-1 consists of a condition part of three bit representing the position of objects in the environment and action part of three bits representing action to be done in the environment. For example rule has the form: 000:000 that means when a relative position of an object to be sensed (cheese or lair) from our simulated rat is to the north then the action to be taken by simulated rat is moving to the north, and so on.

6.2. LCS-2 for learning switching.

LCS-2 is a learning system that is to be learned independently from LCS-1.LCS-2 is used to learn artificial rat to choose one of the two basic behaviors, approaching, that is, moving towards cheese and eat it or escaping, that is, moving towards lair when the human enters the environment(hearing human steps sound).

6.2.1 Coding LCS-2 environmental message.

LCS-2 received message from environment mapping which is represented by an eight state from (0 to 7) of three bits only. First bit represented presence or absence of a human in the environment. Second bit represented the distance of a piece of cheese from artificial rat (0 far and 1 close). The third represented the distance of lair from artificial rat (0 far and 1 close).When there is only one step for rat to reach cheese or lair then it is close, otherwise it is far.LCS-2 environmental message has the form and meaning as in Table(3).

6.2.2. Coding LCS-2 actions.

LCS-2 has an action consisting of one bit, therefore we have two legal actions.LCS-2 actions have the form and meaning as in Table (4).

The message list and classifier store are the main component in the performance system. Message list consists of a single message during each cycle and classifiers store consists of a set of rules. For example a rule has a form: 000:0 means that when human is absent and the cheese is far away from artificial rat and lair is far from rat then the rat action taken in the environment will be approaching cheese.

7. The Performance System for (HRC) System

As the performance system is the heart of the classifier system, the matching procedures are the heart of the performance system. Performance system of the HRC consists of a message list and classifier store. There is only single message in the message list that is used to match against condition part of all classifiers in the classifier store and there is no message to be received in the current cycle until the system produces an action. The two routines are responsible for matching classifiers to the environment message: match and match classifiers.

The function match performs a match between a single condition and a single message and returns a Boolean true value if match succeeds. The function match is illustrated in Figure (8).
The procedure matches all classifiers against the environment message and constructs the match list data structure. Figure (9) illustrates the procedure match classifiers.

8. Apportionment of Credit (AOC) for (HRC) System

The procedure (AOC), in the system (HRC) calls three routines: auction, clearing house and tax collector.

First procedure, the function auction holds a noisy auction to select a winning classifier from the set of matched classifiers. auction cycle through the matched classifiers , successively calculates each classifiers base (bid) and its effective bid ((Ebid) as illustrated in equations: (1),(2),(3),(4) the function auction keeps track of the classifier index with highest effective bid and returns this value upon relinquishing control to procedure (AOC).

Formally for HRC system, a classifier's bid is defined as a product of C_{bid} and a linear function of classifier's specificity, and classifier strength [2].

$$Bid_i = C_{bid} * f(SP) * S_i \quad \textbf{(1)}$$

$$f(SP) = bid_1 + bid_2 * SP \quad \textbf{(2)}$$

Where: C_{bid} is the bid coefficient, usually C_{bid} is a constant less than 1. Specificity (SP) = number of non # /Total Length of condition part. bid_1, bid_2 are input parameters. S Is strength, i is classifier index.

The effective bid (EB) for each matched classifier is the sum of its deterministic bid and a noise term:

$$EB_i = Bid_i + N(\sigma_{bid}) \quad \textbf{(3)}$$

where the noise N is a function of the specified bidding noise standard deviation σ_{bid} .

The winning classifiers place their messages on the message list and their strengths are reduced by the amount of their bids. Typically, if $S_C(t)$ denotes the strength of a winning classifier C , at time t and $B_C(t)$ denotes its bid at the same time t , then its strength at time $t+1$ is:

$$S_c(t+1) = S_c(t) - B_c(t) \quad \textbf{(4)}$$

There after procedure clearinghouse is invoked to reconcile payments. the current winners strength is simply decreased by the amount of its bid value for the HRC problem, bucket brigade algorithm flag is usually set to false because reward is available at every time step and because there is no relationship between successive signals.

For HRC system if C' sent the message matched by C above, then the strength of C' at time $t+1$ is:

$$S_{c'}(t+1) = S_{c'}(t) + B_c(t) \quad \textbf{(5)}$$

The last routine called by the AOC procedure is tax collector. To discourage nonproductive classifiers, two different types of tax are collected from classifiers:

an existence tax and bid tax. The existence tax is assessed and collected from all classifiers at a tax rate specified in the real value population variable life tax. The bid tax is assessed and collected from all classifiers that bid in the last auction; this tax rate is specified by the real variable bid tax. Many schemes are available; a tax was simply collected proportional to the classifier strength:

$$T_i = C_{tax} * S_i \qquad \textbf{(6)}$$

Where: T is tax, C_{tax} is coefficient of tax, S is strength, i classifier index.

To implement a well-defined procedure the auction and payment scheme was detailed. Classifiers make ***bids*** (B_i) during the auction. Winning classifiers turn over their bids to the clearinghouse as ***payments*** (P_i). A classifier may also have ***receipts*** (R_i) from its previous message- sending activity or from environmental reward. In addition to bids and receipts, a classifier may be subject to one or more ***taxes*** (T_i) taken together, the equation governing the depletion or accretion of the classifier strength is as follows [10]:

$$S_i(t+1) = S_i(t) - P_i(t) - T_i(t) + R_i(t) \qquad \textbf{(7)}$$

The apportionment of credit equation recasts into a more useful form where all payments and taxes have been replaced by their strength equivalent [4].

$$S(t+1) = S(t) - C_{bid} * S(t) - C_{tax} * S(t) + R(t) \qquad \textbf{(8)}$$

9. Rule Discovery for (HRC) System

GA is a way of injecting new possibly better rules into the system (HRC) new rules are created by the rule discovery process, *Reproduction, crossover, and mutation.* These rules are then placed in the population and processed by the auction, payment and reinforcement learning to properly evaluate their role in the system.

- In the work (HRC) a quantity called the selection proportion were used, where that proportion of the population is replaced at a given genetic algorithm invocation.

The number of mate pairs to select =proportion select*n classifier*0.5
(9)

- The selection process for (HRC) is performed using ***roulette wheel*** selection where each classifier's strength value S is used as its fitness. The sum of the population fitness is calculated .The expected value of an individual is the individual's fitness divided by the average fitness of the population. Roulette wheel selection algorithm is illustrated in Figure (10).
- The crossover operator is implemented by taking two-parent string and generating two offspring string. In (HRC) problem crossover is implemented as follows: an integer position k along the string is selected uniformly at random between 1 and the string length less than one [1,l-1]. Two new strings are created by swapping characters between position k+1 and l inclusively. Figure (11) illustrates Procedure Crossover.
- When a mutation is called it changes the mutated {0,1,#} character to one of other two with an equal probability. The mutation algorithm is illustrated in Figure (12).

- **Selection of next Generation**

Now we are searching, not for the single best rule (classifier), but for a welladapted set of rules. Therefore we use the "crowding replacement" algorithm to choose the classifiers that should die to make room for new offspring. (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) [4,6]. The crowding algorithm is illustrated in Figure (13**).**

10. The Reinforcement Learning of HRC System.

The way that is used to provide the system with reinforcement learning is by allowing the system to control the reinforcement task by using reinforcement algorithm.

Reinforcement algorithm is preferable to be provided by a human trainer who observes the performance of the system and provides positive or negative reinforcement according to the 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. In HRC System we compare between 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 both LCS-1 and LCS-2.and using reward and punishment that is by rewarding (10) to reward good action and punishment (-10) for bad action for both LCS-1 and LCS-2.

Figure (14-a) describes function of reinforcement learning using reward only while Figure (14-b) describes the function of reinforcement learning using reward and punishment.

11. The HRC System Cycle.

The HRC system consists of two learning classifier systems. The mechanism of the two classifier systems is the same. First the system receives a message from environment through detectors into a single message list. This message matches against classifiers in the classifier store to perform match list. Second an action is held to choose a winning classifier. The action of wining classifier goes forward to the environment as output. At last reinforcement is provided to the system to reward the classifier of good action and punishment the classifier of bad action. Figure (15) illustrates HRC cycle.

Executing the HRC code, the system responds by presenting the initial report display in Figure (16-a), for LCS-1 using reinforcement learning with combination of reward and punishment. The classifier system run for **50** iterations, but the classifier system for LCS-1 execute using reinforcement learning with reward only shows in Figure(16-b) illustrates that the system run for 100 iterations.

Executing the HRC code, the system responds by presenting the initial report displayed in Figure(17-a) for LCS-2 using reinforcement learning with combination of reward and punishment, the system respond for **100** iterations, whereas the snapshot report displayed in Figure(17-b) illustrates LCS-2 responds for **750** iterations when using reinforcement learning with reward only,

The run of the program using reinforcement learning with reward only is unable to perform as well as the run using reinforcement learning with combination of reward and punishment, the time needed to match the signal is shorter when using reinforcement learning with reward and punishment and we are more likely to achieve good results quickly.

9. Conclusions

- Experimentally comparing between a reinforcement learning algorithm with a combination of reward and punishment produces better results than reward alone. Likewise, only rewarding the robot produces better results than only punishing it.
- Experiment shows the time needed to perform the task when using reinforcement learning with reward and punishment is less than the time needed when using reinforcement using only reward.
- Experimentally we obtained more accuracy of the signal when using reinforcement learning with reward and punishment.
- The system was able to learn rules for the given task using only a few training examples and starting with classifiers that were randomly generated.
- Reinforcement learning techniques used to implement robot controllers with interesting properties such as providing guidance to the system and shortening the number of cycles required to learn.

References:

1. Bacardit Jaume, Ester Bernad´o-Mansilla, and Martin V. Butz,(2009)" Learning Classifier Systems: Looking Back and Glimpsing Ahead". ASAP research group, School of Computer Science, Jubilee Campus, Nottingham,NG8 1BB and Multidisciplinary Centre for Integrative Biology, School of Biosciences, Sutton Bonington, LE12 5RD, University of Nottingham, UK.
2. Bhatia. Nikhil, Dixit. Prashant, Sood . Mohit.(1999)"GAMBLE: Genetic Algorithm Based Machine learning Expert" Indian Institute of Technology.
3. BROWNLEE, JASON,(2007)," Learning Classifier Systems" Technical Report 070514A,Complex Intelligent Systems Laboratory, Centre for Information Technology Research, Faculty of Information and Communication Technologies, Swinburne University of Technology ,Melbourne, Australia jbrownlee@ict.swin.edu.au.
4. Bull.Larry (2004), "Learning Classifier Systems: A Brief Introduction", Faculty of Computing, Engineering & Mathematical Sciences University of the West of England, Bristol BS16 1QY, U.K. Larry.
5. Crook. Stamati (2003), "Evolving expert systems for autonomous agent control using reinforcement learning", M.Sc Thesis. Evolutionary and Adaptive Systems. School of Cognitive and Computing Sciences, Sussex University.
6. Hunt. John (2002)" Learning Classifier Systems", JayDee Technology Ltd.Hartham Park, Corsham, Wiltshire , SN13 0RP, http://www.jaydeetechnology.co.uk
7. Kelly.Ian Darrell (1997)" The Development of Shared Experience Learning in a Group of Mobile Robots." ,Thesis submitted for the Degree of Doctor of Philosophy Department of Cybernetics, university of reading.
8. Kovacs Tim (2000)," Towards a Theory of Strong Over general

Classifiers". School of Computer Science ,University of Birmingham ,Birmingham B15 2TT United Kingdom ,Email: T.Kovacs@cs.bham.ac.uk

9. Kovacs Tim (2002)," advanced topics in machine learnin strength or accuracy Fitness Evaluation in Learning Classifier Systems" ,UNIVERSITY OF BRISTOL.
10. Robert Elliott Smith , Max Kun Jiang ,Jaume Bacardit , Michael Stout , Natalio Krasnogor ,Jonathan D. Hirst,(2010)," A learning classifier system with mutual-information-based fitness", UK Engineering and Physical Sciences Research Council(EPSRC).
11. Ryan J. Urbanowicz and Jason H. Moore,(2009)," Learning Classifier
12. Systems: A Complete Introduction, Review, and Roadmap", Department of Genetics, Dartmouth College, Hanover, NH 03755, USA.
13. Odetayo Michael O,(1990), " Machine Learning Using a Genetic-Based Approach". School of Mathematical and Information Sciences, Coventry University, Priory Street, Coventry CV1 5FB, UK.

Table (1): Form and meaning of LCS-1 environmental message.

The message	Its meaning
000	Relative position of cheese or lair from rat is to the north
001	Relative position of cheese or lair from rat is to the north east
010	Relative position of cheese or lair from rat is to the east
011	Relative position of cheese or lair from rat is to the south east
100	Relative position of cheese or lair from rat is to the south
101	Relative position of cheese or lair from rat is to the south west
110	Relative position of cheese or lair from rat is to the west
111	Relative position of cheese or lair from rat is to the north west

Table (2): Form and meaning of LCS-1 actions.

The action	Its meaning
000	Mean rat move to the north
001	Mean rat move to the north east
010	Mean rat move to the east
011	Mean rat move to the south east
100	Mean rat move to the south
101	Mean rat move to the south west
110	Mean rat move to the west
111	Mean rat move to the north west

Table (3): Form and meaning of LCS-2 environmental message.

The message	Its meaning
000	Absence human, far cheese, far lair
001	Absence human, far cheese, close lair
010	Absence human, close cheese, far lair
011	Absence human, close cheese, close lair
100	Presence human, far cheese, far lair
101	Presence human, far cheese, close lair
110	Presence human ,close cheese, far lair
111	Presence human, close cheese, close lair

Table (4): Form and meaning of LCS-2 actions.

The action	Its meaning
0	Approaching behavior is active(approaching cheese).
1	Escaping behavior is active(escaping to the lair).

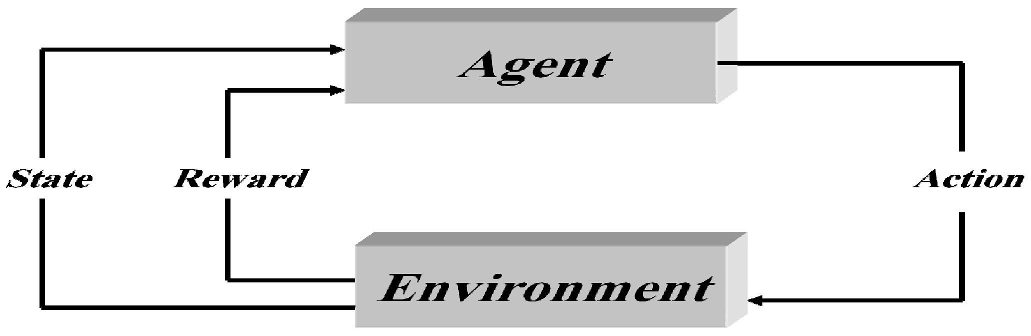


Figure (1) Reinforcement learning model.

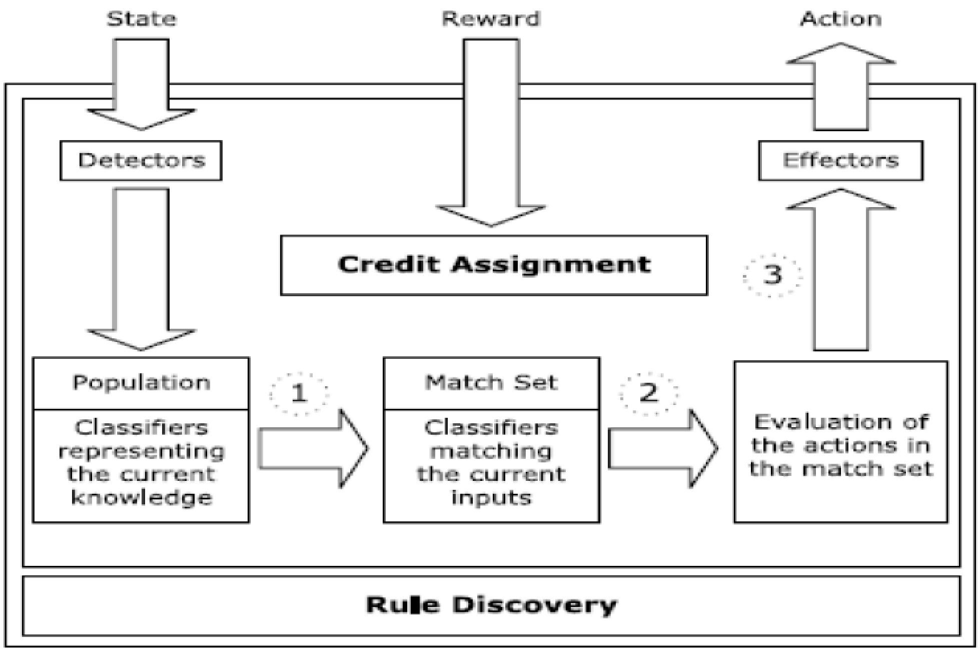

State
Reward
Action
Detectors
Effectors
Credit Assignment
3
Population
Classifiers representing the current knowledge
1
Match Set
Classifiers matching the current inputs
2
Evaluation of the actions in the match set
Rule Discovery

Figure (2) Simplified architecture of learning classifier system

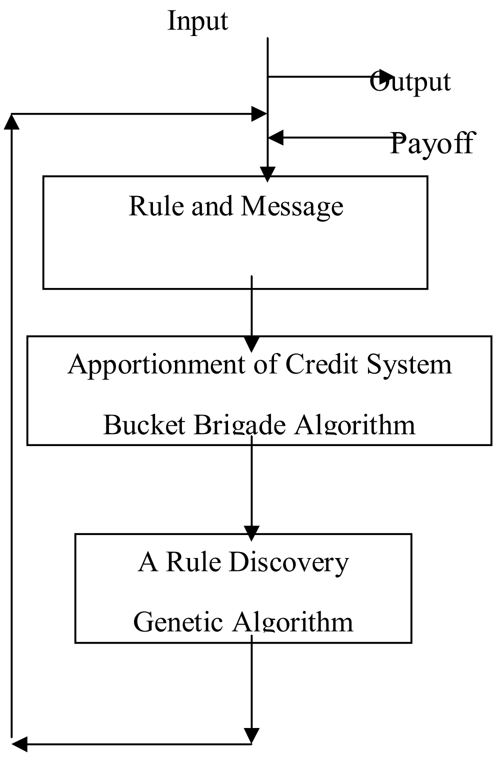
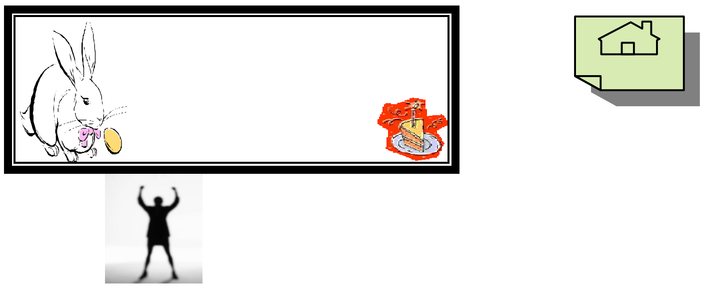


Figure (3) Learning Classifier System components.



Figure(4) HRC Environment

```
If there is a human
    Then escape to the lair
Else if cheese is near
    Then approach cheese
```

Figure (5) Global behavior of HRC System.

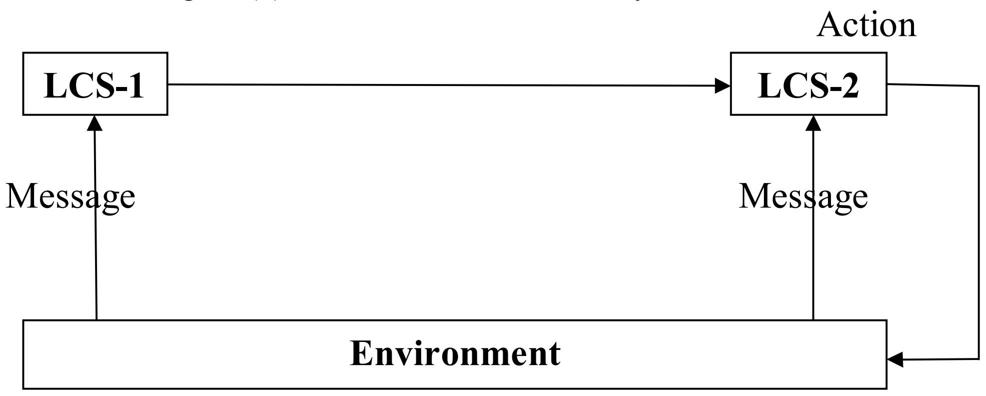


Figure (6) HRC System Structure.

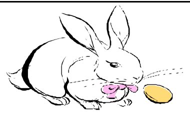
Northwest (111)	**North (000)**	**Northeast (001)**
West (110)		**East (010)**
Southwest (101)	**South (100)**	**Southeast (011)**

Figure (7) Rat Movement Direction.

```
Function match;
{match single condition to single message}
begin
      matchtemp:=true;
      while(matchtemp=true) and(nposition>0) do
      begin
           matchtemp:=(condition[nposition]=wildcard) or
            (condition[nposition]=message[nposition]);
           nposition:= nposition-1
       end;
  match:=matchtemp
end;
```

Figure (8) The Function Match

```
Procedure match classifiers;
{ match all classifiers against environmental message and create match list}
begin with population do with match list do
       begin
          nactive:=0;
          for j:= 1 to nclassifier do with classifier [j] do
           begin
     matchflage:=match(condition,environmentmessage,nposition);
              if matchflag then
                   begin
                      nactive := nactive+1;
                      conditionlist[nactive]:=j
                  end
            end;
end   end;
```

Figure (9) The Procedure Match Classifiers

```
Function roulette wheel;
 Begin
      P=0;
      J=0;
      Rand = Random * Sum of Strength;
      Repeat
              J=J+1;
              P = P + classifier [j]. Strength;
      Until ((P>Rand) OR (J = n classifier));
      Select j;
```

Figure (10) Roulette Wheel Selection Algorithm.

```
Procedure crossover;
Begin
        If flip (Pcrossover) then
           begin
           Sitecross: = rnd (1,nposition);
            ncrossover:=ncrossover+1;
           end
     else site cross: = nposition+1     {transfer but no cross}
     {transfer action part regardless of sitecross}
     Child1.action: = Parent1.action;
     Chiled2.action: = Parent2.action;
     {transfer and cross above cross site}
      j:= site cross;
     While (j<= n position ) do  begin
              Child2.condition [j]: = Parent1.condition [j];
              Child1.condition [j]: = Parent2.condition [j];
              j:=j+1
      end;
     j:=1;
     {Transfer only below cross-site}
     While (j< site cross) do begin
             Child1.condition [j]: = Parent1.condition [j];
             Child2.condition [j]: = Parent2.condition [j];
             j:=j+1
 end;
end;
```

```
Function Mutation;
Begin
        mutate: = flip (Pmutation);
         If mutate then
           begin
               nmutation: = nmutation + 1;
                mutation: = notalleleVal;
          End else
          mutation: = alleleVal;
End;
```

Figure (12) Function Mutation

```
for i= 1 to crowding factor do
        x:=find worst of a random set
        if this is not more similar to offspring then paste x then
        set worst more similar to x
        end if
end for
Replace worst most similar with offspring
```

Figure(13) Crowding algorithm.

```
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.
```

Figure (14-a) Reinforcement learning using Reward only

```
If (classifier output=desired output) then

Add (Reward=10) to classifier strength responsible for sending action.

Else

Add (Punishment =-10) to classifier strength responsible for sending action.
```

Figure (14-b) Reinforcement learning using Reward & Punishment

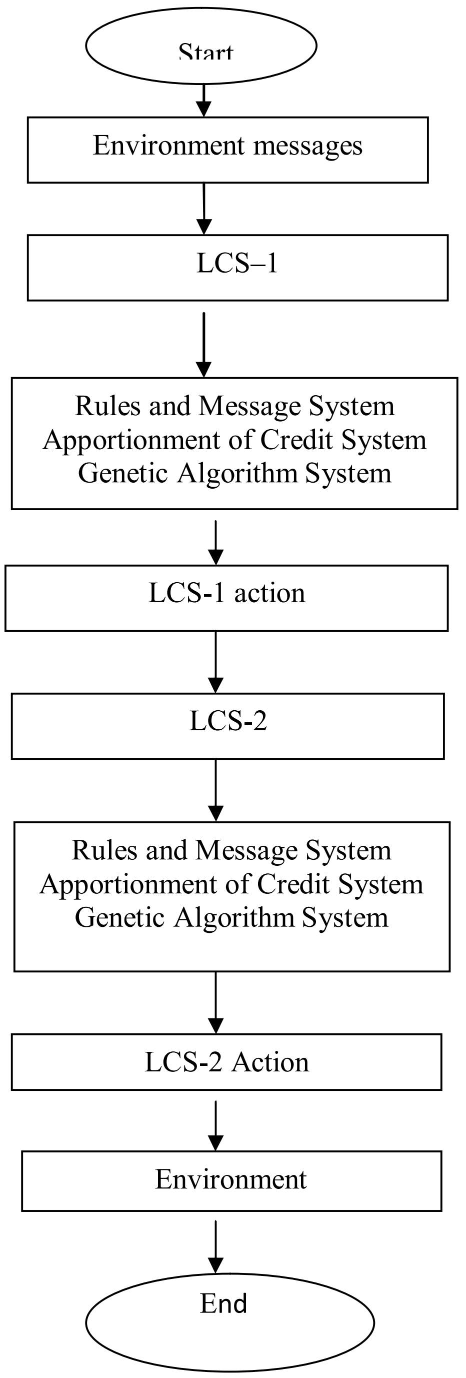


Figure (15) HRC System cycle.

```
--------------------------------------------
a learning classifier system( LCS-1 )
--------------------------------------------
population parameters
-------------------------
number of classifiers    =    20
number of positions      =     3
number of action         =     3
bid coefficient          =  0.1000
bid spread               =  0.0750
bidding tax              =  0.0100
existence tax            =  0.0000
generality probability   =  0.5000
bid specificity base     =  0.2500
bid specificity mult.    =  0.1250
edid specificity base    =  0.2500
ebid specificity mult.   =  0.1250

environmental parameters
--------------------------------
total number of signal  =     3

apportionment of credit parameters
-------------------------------------------
bucket brigade flag  =    false

reinforcement parameters
-------------------------------
 reinforcement reward   =   10.0

Timekeeper parameters
------------------------------
Initial iteration          =     0
Initial block              =     0
Report period              =     50
Console report period      =     50
Plot report period         =     50
Genetic algorithm period   =      1

Genetic Algorithm Parameters
-------------------------------------
Proportion to select/gen =  0.8000
Number to select         =     8
Mutation probability     =  0.0200
Crossover probability    =  0.8000
Crowding factor          =     3
Crowding subpopulation   =     3
```

```
snapshot report
---------------

[block: iteration]  -  [0:0]

current  Statius
----------------
signal     =    000
Decoded signal    =      0
desired output    =      0
classifier output  =      0

environmental message:      000

no.     strength       bid      ebid   M   classifier
------------------------------------------------------

   1    10.00      0.00     0.00       1##:[100]
   2    10.00      0.00     0.00       1##:[101]
   3    10.00      0.00     0.00       1##:[110]
   4    10.00      0.00     0.00       1##:[111]
   5    10.00      0.00     0.00       0##:[000]
   6    10.00      0.00     0.00       0##:[001]
   7    10.00      0.00     0.00       0##:[010]
   8    10.00      0.00     0.00       0##:[011]
   9    10.00      0.00     0.00       ##1:[001]
  10    10.00      0.00     0.00       ##1:[011]
  11    10.00      0.00     0.00       ##1:[101]
  12    10.00      0.00     0.00       ##1:[111]
  13    10.00      0.00     0.00       ##0:[000]
  14    10.00      0.00     0.00       ##0:[010]
  15    10.00      0.00     0.00       ##0:[100]
  16    10.00      0.00     0.00       ##0:[110]
  17    10.00      0.00     0.00       #1#:[010]
  18    10.00      0.00     0.00       #1#:[011]
  19    10.00      0.00     0.00       #1#:[110]
  20    10.00      0.00     0.00       #1#:[111]
new winner[1] : old winner[1]

Reinforcement report
-----------------------------
proportion correct (from start) =   0.0000
proportion correct (last fifty) =   0.0000
last winning classifier number  =       0
```

```
snapshot report
---------------

[block: iteration]  -  [0:50]

current  Statius
----------------
signal     =   101
Decoded signal    =      5
desired output    =      5
classifier output  =     5

environmental message:      101

no.    strength   bid     ebid    M     classifier
-----------------------------------------------------

   1   123.95     0.00    0.00          111:[101]
   2   120.33     7.60    7.52    x      101:[111]
   3   132.07     8.23    8.27    x      101:[101]
   4   119.11     6.02    6.03    x     #01:[010]
   5   122.45     7.73    7.63    x      101:[101]
   6   120.31     0.00    0.00          111:[010]
   7   121.45     0.00    0.00          111:[010]
   8   122.45     7.73    7.71    x      101:[101]
   9   122.81     7.75    7.78    x      101:[101]
  10    121.97     0.00    0.00          111:[101]
  11    125.48     7.92    7.88    x      101:[101]
  12    122.81     6.20    6.14    x     #01:[101]
  13    120.11     6.07    6.17    x     #01:[010]
  14    125.48     7.92    7.91    x      101:[101]
  15    121.61     0.00    0.00          111:[010]
  16    117.80     5.95    5.96    x     #01:[010]
  17    119.28     7.53    7.63    x      101:[101]
  18    120.48     0.00    0.00          111:[010]
  19    120.33     7.60    7.54    x      101:[101]
  20    122.17     0.00    0.00          111:[101]

new winner[3] : old winner[3]
```

Figure(16-a) LCS-1 using reinforcement learning with reward and punishment.

```
snapshot report
---------------
[block: iteration]  -  [0:100]


current  Statius
----------------
signal     =   101
Decoded signal     =      5
desired output     =      5
classifier output  =      5

environmental message:      101

no.    strength      bid     ebid   M   classifier
-----------------------------------------------------

   1   143.51     0.00     0.00        00#:[101]
   2   145.06     7.33     7.34   x    #01:[101]
   3   143.51     0.00     0.00        #11:[101]
   4   142.89     0.00     0.00        #11:[101]
   5   142.71     0.00     0.00        01#:[101]
   6   143.72     0.00     0.00        #11:[101]
   7   142.89     0.00     0.00        111:[001]
   8   142.83     0.00     0.00        11#:[101]
   9   144.07     9.03     9.04   x    101:[101]
  10    139.51     5.28     5.22   x    #0#:[010]
  11    142.95     0.00     0.00        111:[001]
  12    142.28     8.98     8.97   x    101:[101]
  13    143.45     0.00     0.00        111:[001]
  14    142.21     0.00     0.00        #1#:[010]
  15    142.45     0.00     0.00        111:[101]
  16    139.28     5.28     5.16   x    #0#:[010]
  17    142.89     0.00     0.00        111:[001]
  18    142.58     0.00     0.00        111:[001]
  19    144.55     0.00     0.00        #11:[101]
  20    142.57     0.00     0.00        11#:[101]

new winner[9] : old winner[2]
```

Figure (16-b) LCS-1 using reinforcement learning with reward only.

```
_____________________________________
a Learning classifier system( LCS-2)
------------------------------------
population parameters
---------------------
number of classifiers    =    20
number of positions      =     3
number of action         =     1
bid coefficient          =  0.1000
bid spread               =  0.0750
bidding tax              =  0.0100
existence tax            =  0.0200
generality probability   =  0.5000
bid specificity base     =  0.2500
bid specificity mult.    =  0.1250
edid specificity base    =  0.2500
ebid specificity mult.   =  0.1250

environmental parameters
------------------------
total number of signal   =     3

apportionment of credit parameters
----------------------------------
bucket brigade flag  =    false

reinforcement parameters
------------------------
 reinforcement reward    =   10.0

Timekeeper parameters
---------------------
Initial iteration          =     0
Initial block              =     0
Report period              =    50
Console report period      =    50
Plot report period         =    50
Genetic algorithm period   =    10

Genetic Algorithm Parameters
----------------------------
Proportion to select/gen =  0.8000
Number to select         =     8
Mutation probability     =  0.0200
Crossover probability    =  0.8000
Crowding factor          =     3
Crowding subpopulation   =     3
```

```
snapshot report
---------------

[block: iteration]  -  [0:0]

current  Statius
----------------
signal     =    000
Decoded signal     =       0
desired output    =      0
classifier output  =      0

environmental message:      000

no.     strength    bid     ebid    M   classifier
-----------------------------------------------------

   1   100.00     0.00     0.00       1##:[1]
   2   100.00     0.00     0.00       0##:[0]
   3   100.00     0.00     0.00       0#0:[0]
   4   100.00     0.00     0.00       #1#:[0]
   5   100.00     0.00     0.00       #0#:[0]
   6   100.00     0.00     0.00       #0#:[1]
   7   100.00     0.00     0.00       ##1:[1]
   8   100.00     0.00     0.00       ##0:[1]
   9   100.00     0.00     0.00       ##0:[0]
  10   100.00     0.00     0.00       ###:[1]
  11   100.00     0.00     0.00       ##1:[1]
  12   100.00     0.00     0.00       ###:[0]
  13   100.00     0.00     0.00       11#:[1]
  14   100.00     0.00     0.00       00#:[0]
  15   100.00     0.00     0.00       00#:[0]
  16   100.00     0.00     0.00       #11:[1]
  17   100.00     0.00     0.00       #00:[1]
  18   100.00     0.00     0.00       #00:[0]
  19   100.00     0.00     0.00       ##1:[1]
  20   100.00     0.00     0.00       ##0:[1]

new winner[1] : oldwinner[1]
```

```
snapshot report
---------------

[block: iteration]  -  [0:100]

current  Statius
----------------
signal     =   001
Decoded signal     =      1
desired output    =      1
classifier output  =      1

environmental message:      001

no.     strength    bid      ebid    M   classifier
------------------------------------------------------

   1    46.54      0.00     0.00         1#1:[1]
   2    43.16      0.00     0.00         1##:[1]
   3    41.07      0.00     0.00         011:[1]
   4    74.82      0.00     0.00         101:[1]
   5    70.58      3.64     3.53   x     0#1:[0]
   6    124.96     6.25     6.06   x     0#1:[1]
   7    45.53      2.93     2.85   x     001:[1]
   8    54.35      0.00     0.00         000:[1]
   9    71.44      0.00     0.00         100:[1]
  10     43.16     0.00     0.00         101:[1]
  11     50.45     0.00     0.00         1#0:[1]
  12     74.82     0.00     0.00         0#0:[1]
  13     48.48     0.00     0.00         110:[1]
  14     47.59     0.00     0.00         101:[1]
  15     64.47     3.32     3.34   x     0#1:[1]
  16     78.20     0.00     0.00         1#1:[1]
  17     40.82     0.00     0.00         100:[1]
  18     47.59     0.00     0.00         1#0:[1]
  19     43.76     0.00     0.00         011:[1]
  20     54.35     0.00     0.00         1#1:[1]

new winner[6] : old winner[6]
```

Figure (17-a) LCS-2 using reinforcement learning with reward and punishment.

```
snapshot report
---------------

[block: iteration]  -  [0:750]

current  Statius
----------------
signal    =   001
Decoded signal    =      1
desired output    =     1
classifier output  =     1

environmental message:      001

no.     strength    bid     ebid   M   classifier
-----------------------------------------------------

   1    70.77     4.56     4.48   x    001:[1]
   2    54.02     0.00     0.00         000:[1]
   3    35.66     0.00     0.00         000:[1]
   4   108.11     6.76     6.87   x    001:[1]
   5    56.01     0.00     0.00         101:[1]
   6    39.54     0.00     0.00         000:[1]
   7    38.32     2.47     2.43   x    001:[1]
   8    48.75     3.14     3.23   x    001:[1]
   9    37.20     0.00     0.00         100:[1]
  10     61.82     3.98     4.12   x    001:[1]
  11     43.93     2.83     2.73   x    001:[1]
  12     38.32     2.47     2.63   x    001:[1]
  13     43.93     2.83     2.84   x    001:[1]
  14     61.82     3.98     4.04   x    001:[1]
  15     70.77     4.56     4.50   x    001:[1]
  16     52.88     3.41     3.39   x    001:[1]
  17     34.07     2.20     2.16   x    001:[1]
  18     18.39     0.00     0.00         #00:[1]
  19     52.88     3.41     3.46   x    001:[1]
  20     38.32     2.47     2.43   x    001:[1]

new winner[4] : old winner[4]
```

Figure (17-b) LCS-2 using reinforcement learning with reward only.