2005 IEEE/RSJ International Conference on Intelligent Robots and Systems

Enhanced Learning Classifier System for Robot Navigation*

Petr Musílek, Sa Li, and Loren Wyard-Scott
Department of Electrical and Computer Engineering
University of Alberta
Edmonton, Alberta, Canada T6G 2V4
{*musilek, sali, wyard*}*@ece.ualberta.ca*

***Abstract*— This paper describes an enhanced learning classifier system used to evolve obstacle-avoidance rules used in mobile robot navigation. The robot learns these rules via feedback from the environment, available as sonar readings. Conventional classifiers, when used in this application, show evidence of shortcomings: becoming trapped in local minima, loss of (desirable) rules, and favouring of generalized rules. Enhancements to the classification system are described and tested using a simulated robot and environment. The enhancements prove to be worthwhile in that they overcome the limitations, and can generally handle more complex situations.**

***Index Terms*— Robot, Navigation, Learning Classifier Systems, Reinforcement Learning, Genetic Algorithms**

I. Introduction

This paper describes a method to automatically develop behaviours used to guide an autonomous mobile robot through unfamiliar environments. This navigation problem proves to be difficult to solve using a machine-learning approach and, as such, much of the work in the past few decades concentrates on the use of artificial intelligence (AI) techniques. This paper explores an alternate, sophisticated technique wherein robot intelligence is evolved rather than designed. The focus of the work is in the area of evolutionary robotics (ER) [1] wherein evolutionary computing methods are used to evolve populations of controllers for use in autonomous robots. In the evolutionary computing domain, a classifier system (CS) is a machine-learning system that evolves rules in order to perform effectively in a dynamic environment [2].

Autonomous mobile robots can be programmed to learn through experience. The relation between a robot and its environment can be described by condition-action rules in which the conditions represent the environmental messages and the actions modify the environment. Consequently, learning is the process of discovering an appropriate set of rules. Implementing such a learning algorithm is difficult on mobile robots since they usually operate unsupervised and therefore have need to evaluate their actions according to feedback from the environment. This feedback provides a measure of the extent to which a need is satisfied. It is realistic to assume that the robot has limited memory for storage of experiences and must therefore learn incrementally without direct storage of events in their original form [3]. Various techniques have been developed to execute rapid online modifications to a robot's behaviour. The main contributions in this area can be grouped into three paradigms: artificial neural networks, fuzzy systems, and evolutionary computing.

An artificial neural network (ANN) imitates the adaptive abilities of a biological neural system [4] and is promising for use as a mobile robot controller. An ANN designed to accomplish mobile robot navigation typically consists of numerous simple elements connected in parallel to map sensory inputs to robot actuator commands. Most such systems use multi-layer feedforward networks [5]. If the number of inputs is small, such a network can be trained effectively. However, for large input spaces, as in the case of mobile robot navigation, its performance decreases [6]. Multi-layer feedforward networks are suited for supervised learning scenarios, making on-line incremental learning impossible.

Fuzzy systems can alleviate many problems in real robotic systems in which knowledge of the environment is partial or approximate, sensing is noisy, and/or the dynamics of the environment can be only partially predicted. Fuzzy control systems [7], [8] bring many advantages to the design process including simplicity and the ability to deal with uncertainty. They express and implement human knowledge in the form of linguistic rules that can be applied to robot control. In [8], a simple fuzzy controller handles complicated situations, allowing the robot to avoid obstacles detected by sensors. Although fuzzy controllers are simple to design and faster to prototype than conventional control systems, they require more simulation and fine-tuning before they are usable. In addition, they do not have an inherent ability to adapt which further reduces their usefulness in systems exposed to a dynamic, changing environment, unless augmented by an appropriate learning mechanism.

Evolutionary Computing (EC) is the general term describing a family of computational techniques based on evolution in the natural world. ER applies EC methods to evolve populations of robot controllers. The fundamental goal of ER is to develop automatic methods of autonomous mobile robot controller synthesis that do not require hand design or in-depth human knowledge of the robot task. A large amount of experimental

*Support provided by the Canada Foundation for Innovation (CFI), Natural Sciences and Engineering Research Council (NSERC), and the Department of Electrical and Computer Engineering, University of Alberta, is greatly appreciated.

research in the ER area [9], [10] has lead to synthesis of controllers that produce desirable complex behaviours.

As one of the EC techniques, learning classifier systems have received much attention in recent years. The original CS concept was proposed by Holland [11] to model natural evolutionary processes and provides flexible operation with mechanisms for structural adaptation. Classifier systems appear to offer a broadly useful framework for addressing problems in mobile robotics [3]. However, only a few research studies have applied classifier systems to the problem of mobile robot navigation [12].

This paper is organized as follows. Section 2 provides an introduction to conventional classifier systems. The application of learning classifier systems to robot navigation is described in Section 3, along with experimental results showing shortcomings of conventional CS. In Section 4, several extensions are proposed to overcome these limitations and their benefits are demonstrated using additional experiments. Section 5 concludes the paper and provides an outline of future work.

II. Conventional Classifier Systems

The learning classifier system is a machine-learning system with strong learning capabilities. The system assimilates a perpetual stream of information about the environment and creates a set of competing rules without significantly disturbing those already obtained. CS is normally applied to situations that possess one or more of the following challenges [13]:

- information about the environment is available as a stream of potentially noisy and/or irrelevant data;
- instantiation of action needs to be continuous, and perhaps real-time;
- goals are inexactly or implicitly defined; and
- payoff is sparse in that long sequences of actions occur between reinforcement.

In the face of these complexities, classifier systems receive information about the environment, perform internal processing and then provide an action which has a perceived effect on the environment. Subsequent feedback relating the effects the system has had on the environment is used to modify the classifier system's internal structure and parameters: the system effectively learns from experience.

The theory of learning classifier systems is fairly new. Two approaches have been developed: the Michigan approach [14] and the Pittsburgh approach. In the Michigan approach, an individual of an evolutionary algorithm population encodes a single rule, whereas in the Pittsburgh approach each individual represents an entire set of rules for the problem at hand [2]. This paper employs the Michigan Approach which has undergone more development. The approach has three major components: a rule and message subsystem, an apportionment of credit subsystem, and a rule discovery mechanism [15]. Fig. 1 shows a block diagram of a classifier system.

A. Rule and Message Subsystem

Detectors collect information from the environment and codify it into messages that the classifier systems can recognize.

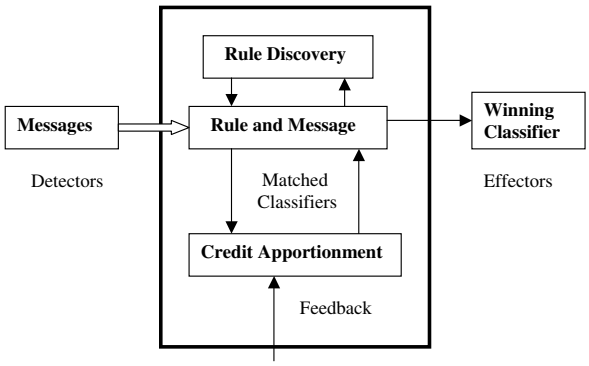

Fig. 1. Block diagram and data flow of a typical classifier system.

In the simplest version, all messages are of fixed length in a specific alphabet (typically k-bit binary strings [15]). Each classifier in the classifier population consists of a rule in the usual condition/action form. It has one or more conditions as the antecedent, drawn from the ternary alphabet (0, 1, #), an action part as the consequent composed of a binary string (0, 1), and an associated strength. The symbol '#' in the condition is a "don't care" and can assume a value of either '0' or '1'. The more "don't cares", the more general the rule. The number of non-'#' symbols in the antecedent, relative to its length, represents the specificity, f_i, of a classifier rule. The strength portion of the rule provides a measure of how well the rule has performed: the higher the strength the better the rule's performance and the more likely it is to be used when its condition matches an environmental message.

After the message from the environment is encoded, all classifier rules in the population are matched against the message. If the entries at all non-'#' positions match, the message satisfies the condition. Every rule in the population that matches the environmental message is sent to the apportionment of credit subsystem.

B. Apportionment of Credit Subsystem

The apportionment of credit subsystem is responsible for modification of rule strengths as the system learns [15]. All classifier rules that match the environmental message compete in an auction to determine the individual that will have an effect on the environment. The winning rule's action could be beneficial or detrimental to the environment. Feedback is used by the apportionment of credit subsystem to appropriately reinforce or punish the winning rule by increasing or decreasing its strength.

A classifier rule's bid is a function of its strength and specificity. The bid of rule i at iteration t, $B_i(t)$, is calculated as:

$$B_i(t) = \rho S_i(t) f_i \tag{1}$$

where ρ is a parameter that acts as an overall risk factor influencing what proportion of a rule's strength will be bid (and possibly lost) in a single step, $0 < \rho \leq 1$; $S_i(t)$ is the

current rule strength; and f_i is the specificity, $0 \leq f_i \leq 1$. Eq. (1) indicates that highly specialized rules with values of f_i close to 1 are preferred over general rules.

After all bids for rules in the matching pool are calculated, the winner of the auction is determined by selecting the individual with the highest bid. As in a real auction, the winner pays for the privilege to perform its action by decreasing its strength by the amount of its bid [15]. Other classifier rules that participated in the auction but did not win save their bids and their strength is not affected.

The strength of the winning classifier rule, $C_i(t)$, at the end of iteration t is:

$$S_i(t+1) = S_i(t) + R_i(t) - B_i(t) \tag{2}$$

where $R_i(t)$ is the reward from the environment at iteration t. Because the action caused by the winning rule could be beneficial or detrimental, the reward $R_i(t)$ will correspondingly hold a positive or negative value.

Two types of taxes, Tax_{life} and Tax_{bid}, are used to prevent the classifier population from being cluttered with individuals of little or no utility having artificially high strengths. For individuals that don't match a message in the current iteration, a fixed rate tax Tax_{life} is levied. Suppose a rule is inactive for n consecutive iterations. The strength is modified according to:

$$S_i(t+n) = S_i(t) * (1 - \text{Tax}_{\text{life}})^n \tag{3}$$

For rules that do not win the auction but are in the current matching pool, a fixed rate tax Tax_{bid} is applied to penalize their generality. Therefore, after n iterations of losing, a classifier rule has the strength:

$$S_i(t+n) = S_i(t) - \sum_{j=0}^{n} \text{Tax}_{\text{bid}} * B_i(t+j) \tag{4}$$

The general strength update equation can be obtained by combining Eqs. (2)-(4):

$$S_i(t+1) = (1 - \text{Tax}_{\text{life}})S_i(t) + R_i(t) - \text{Tax}_{\text{bid}} * B_i(t) \tag{5}$$

For the winning rule, Tax_{life} is set to 0 and Tax_{bid} to 1. For non-winning rules in the matching pool, Tax_{life} is set to 0 and Tax_{bid} to a fixed rate less than unity. Conversely, for non-matching rules, Tax_{bid} is set to 0 and Tax_{life} is set to a fixed rate less than unity. Reward $R_i(t)$ is received by only the winner and is equal to zero for the remaining rules, while bid $B_i(t)$ is determined by Eq. (1) for all those in the matching pool.

C. Classifier Discovery Subsystem

After a number of classifier system iterations, a genetic algorithm is applied to produce the next generation of rules. Based upon genetic operators, the search process is directed along trajectories influenced by the (above-average) strength of the current classifier population. The genetic algorithm normally selects the rules with greater strength and promotes reproduction among them. The offspring generated are launched into the next generation to compete with their parents and other individuals. If the new classifier rules are more effective in gaining payoff, they will survive and replace the weakest individuals in the population.

TABLE I
ENCODING OF ROBOT TURNING ANGLE.

Turning angle	Codes	Probability
Left 15°	1100	0.0625
Left 10°	1000, 1110	0.125
Left 5°	0100, 1010, 1101	0.1875
0°	0000, 0110, 1001, 1111	0.25
Right 5°	0010, 0101,1011	0.1875
Right 10°	0001, 0111	0.125
Right 15°	0011	0.0625

III. Conventional Classifier System for Robot Navigation

In this section, a conventional classifier system used for robot navigation is described. The objective is to move the robot in its environment while avoiding obstacles. It is expected that this objective will be realized after a learning stage.

The robotic platform used in the experiment is a Pioneer 2DX simulated in the SRI robotic simulator. In the simulator, the robot is equipped with 16 sonars and controlled by the motor commands provided by the learning classifier system. Two types of simulated environments, the pilaster and polygon environments, are adopted to test the robot's performance. The pilaster environment is drawn in the same form as in [12] to allow qualitative comparison. No comparison of learning time is provided since the simulators and computing platforms differ. The major concern is whether the robot is learning or not, and the robot trajectories are sufficient for such assessment.

A. Representation: Encoding and Decoding

All learning systems need to represent real-world knowledge in a form that can be manipulated and classifier systems are no exception. In the case of robot navigation, 16 sonar signals are continuously transformed into binary values by a thresholding process.

To translate the action of a classifier rule into usable motor commands, a decoding mechanism is introduced. The robot's actions are determined by three effectors (represented by six bits): one to control the turning angle (in four bits), another to control the speed (as a single bit), and the third to control the forward/backward direction of travel (as a single bit). The turning angle is expressed in increments of 5° using a 4-bit binary code as shown in Table I. This symmetrical coding scheme has more entries for actions with smaller turns, increasing the probability that the robot will move ahead.

The total length of each rule is 22 bits plus an integer strength value. The rule population is initialized using a uniform random distribution by filling all positions (conditional parts with 0, 1, and #; action parts with 0 and 1). A population size of 400 is used, and the initial strengths of all rules are

set to a common value $S_i(0)$=100. An example of a complete classifier rule is shown here:

10#1000#11#0##10	1011	0	1	99
Conditional part to match the sonar inputs	Turn Angle (Right 5°)	Speed (Slow)	Direction (Forward)	Strength

B. Credit Assignment Algorithm

The credit assignment cycle starts with the matching process. The winning rule is selected from the pool of matches using the bids calculated by Eq. (1) with the risk factor ρ set to 0.5. The probability of selection of a rule is equal to its bid divided by the sum of bids of all rules in the matching pool.

To handle situations when no rule matches the input, the rule creation process [3] is used. The conditional part of a newly created classifier rule is a copy of the current input string, except that # symbols are inserted with a probability equal to the current percentage of all "don't cares" in the population. The action of the new rule is set randomly, and its strength is set to the mean strength of the current population. To make room for the newly created rule, one rule is deleted from the population according to the probability distribution over the inverse of the strengths of all rules.

The reinforcement component adjusts rule strengths according to the payoff received from the environment after performing an action. A reward is distributed to the winning rule, and its strength is reduced by the amount of its bid, B_i, as in Eq. (2). All other rules are updated by Eq. (5) with Tax_{bid} and Tax_{life} both set to 0.5.

C. Rule Discovery Algorithm

The genetic algorithm is invoked after a number of collisions has occurred [12]. In this experiment, the genetic algorithm is invoked after 10 collisions.

A classifier rule C_i in the population is selected using a probability distribution over the strengths of all rules in the population. With a probability of crossover P_c, classifier rule C_i is submitted to one-point crossover with each rule in the population. Each of the parents' strengths are reduced by one-third and the offspring's initial strength is set to the sum of the reductions. With a probability of mutation P_m, the mutation operation is applied to each individual. The strength of offspring resulting from mutation is not changed. When an offspring is added to the population, a rule is deleted using a probability distribution over the inverse of the strengths of all rules to keep the size of the population constant.

In summary, the genetic algorithm chooses higher-strength classifier rules to reproduce and generates new individuals by their recombination and mutation. In the experiments, P_c is set to 0.7 and P_m to 0.025. The value of P_m is slightly lower than typical to limit disruption of effective rules in the population.

D. Experimental Results

A simulated robot and environment are used to verify the algorithm. Fig. 2 shows the trajectory at the beginning of learning. Many collisions occur during learning at which time the classifier system receives high penalties that will be used to improve its performance. After the learning stage, the robot finds equilibrium in an almost circular orbit around a pilaster as shown in Fig. 3, similar to the results reported in [12].

Fig. 4 is a graph showing the number of iterations between collisions during the learning phase before the robot converges to the circular orbit and no more collisions, hence no more learning occurs: the time between collisions is increasing which indicates that the robot is learning.

IV. Enhanced Classifier System for Robot Navigation

Based on the results described in the previous section, it is apparent that the conventional classifier system provides the robot with the ability to learn. However, there are problems that significantly affect the system performance.

Local minima. The robot actions quickly converge to the same circular orbit. The problem arises because the robot ceases to learn new rules and dwells in a local minimum. Some rules dominate the rule base as the winning strengths keep growing and, consequently, the population develops monotonous rules that recognize only a few input messages.

Rule destruction. As the number of environmental messages increases, so does the complexity of the learning task. However, the population size cannot be increased without limit to address this increased complexity due to the associated increase in execution time. The resulting effect is that some good rules are replaced with new ones and therefore the system cannot distinguish between a sufficiently large number of classes [16].

Generalization. The rule-discovery process tends to produce a number of general rules that are undesirable when attempting to discriminate between many sonar inputs. If the conditional portion of a rule is too general, it may not provide an action appropriate to the input, even though the conditions are met.

A. Enhanced Classifier Systems

To address the problems mentioned above, the following enhancements are proposed and implemented.

Escaping from local minima. As mentioned earlier, the criteria to engage the genetic algorithm is the number of collisions. However, once the system converges to equilibrium (e.g. a circular orbit), the rule discovery process is not activated further since no collisions occur. To allow escape from local minima, it is proposed that the genetic algorithm be invoked after a certain number of iterations rather than only upon robot collision.

To decrease the probability of disruption of good rules, the number of iterations between applications is increased as the system learns. Because most rules at initialization are inappropriate, applying the genetic algorithm helps the system find the right rules. However, as time progresses, most rules in the rule base are correct and application of the genetic algorithm need not (and should not) be as frequent.

Rule protection. Inspired by the work of McAulay and Oh [16], "No-replace" and "No-parent" flags are applied to

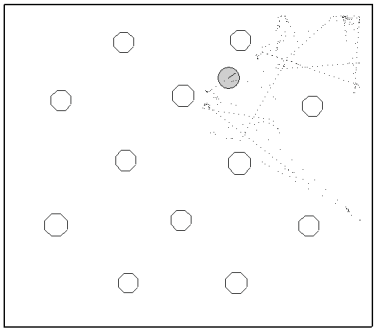

Fig. 2. The trajectory of the robot at the start of learning (pilaster environment).

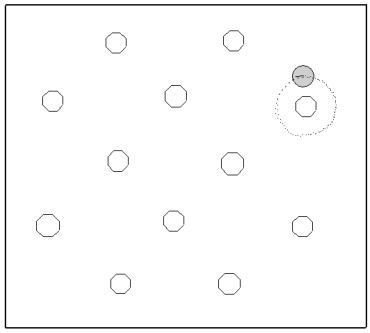

Fig. 3. The trajectory of the robot in the equilibrium state (pilaster environment).

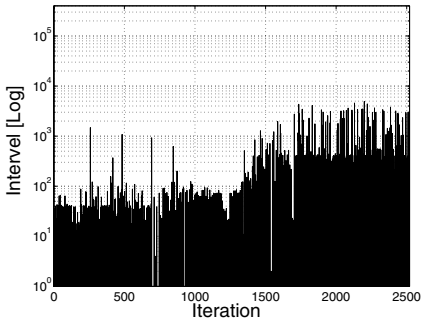

Fig. 4. Number of iterations between collisions (pilaster environment).

index classifier rules that have high strength and provide appropriate action. These rules will not be replaced in later generations and will not become a parent in the next generation, preventing good rules from being replaced and groups of similar rules from dominating the rule base.

Specialization. To handle rules that are too general, a special genetic algorithm is applied to those that provide incorrect action. The genetic operators are only applied to the action part, while the condition part is replaced by the current sonar inputs. This way, a new specialized rule is created with exact environmental inputs and a new action.

The CS enhanced using the proposed modifications provides encouraging results. During the learning stage, the robot learns while exploring the entire environment, as shown in Fig. 5. Fig. 6 shows the robot's path after the learning stage: no collisions occur. Fig. 7 shows the number of iterations between collisions while learning. The graph in Fig. 7 is more oscillatory because of the higher complexity of the polygon environment with respect to the pilaster environment. Once a collision occurs, several other collisions follow closely while the specialization enhancement creates a new action with an updated conditional. If the action of the current winning rule is appropriate, the robot moves away from the obstacle. Otherwise, the robot will approach the obstacle again, and the specialization process restarts.

V. Conclusion and Future Directions

In this paper, a classifier system has been applied to mobile robot navigation. Experiments show that conventional classifier systems can handle only simple and symmetric environments and the system converges to a limit cycle exhibiting circular movement. To allow operation in a more complex and asymmetric environment, extensions to conventional CS are proposed to enhance its performance.

Efficient use of the rule base to match as many messages as possible will be studied to reduce the similarity of the conditional parts of rules in the population. In addition, reduction of the learning period by introduction of a-priori knowledge into the initial classifier population will be considered.

Fusion of learning classifier systems and neural networks has been examined by Smith [17]. This work shows that a genetic algorithm is able to evolve a compact, functional, LCS-like neural network with co-adapted, cooperative hidden layer nodes. The co-adaptive approach seems to have many computational advantages. Future work will focus on how to map the learning classifier systems into neural networks to harness a combination of the structural learning ability of classifier systems and the continuous adaptive characteristics of neural networks.

References

[1] P. Husbands, I. Harvey, N. Jakobi, A. Thompson, and D. Cliff, "Evolutionary Robotics in Handbook of Evolutionary Computation", Chap. G3.7, T. Back, D. Fogel, and Z. Michalewicz (Eds.) *Oxford University Press/Institute of Physics* 1997.

[2] D. Dumitrescu, B. Lazzerino, L. C. Jain and A. Dumitrescu, "Evolutionary Computation", *CRC press*, LLC, 2000.

[3] S. W. Wilson, "Classifier Systems and Animat Problem", *Machine Learning 2:*, 199- 228, 1987.

[4] M. Rylatt, C. Czarnecki and T. Routen, "Connectionist Learning in Behaviour-Based Mobile Robots: A Survey", *Artificial Intelligence Review*, 12: 445-468, 1998.

[5] K. Hornik, M. Stinchcombe and H. White,"Multiplayer Feedforward Networks are Universal Approximators", *Neural Networks*, 2(5):359-366,1989.

[6] A. Schmidt and Z. Bandar, "A modular Neural Network Architecture with Additional Generalisation Abilities for Large Input Vectors", in *Proceedings of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANGA97)*, Springer-Verlag, pp 40-43, April 1997.

[7] M. Sugeno and M. Nishida, "Fuzzy Control of a Model Car", *Fuzzy Sets and Systems* 16, pp. 103-113, 1985b.

[8] S. Thongchai, S. Suksakulchai, D. M. Wilkes and N. Sarkar, "Sonar Behavior-based Fuzzy Control for a Mobile Robot", *Proceedings of the IEEE International Conference on Systems, Man, and Cyber.*, Nashville, Tennessee, Oct. 2000.

[9] J-A. Meyer, P. Husbands and I. Harvey, "Evolutionary Robotics: a Survey of Applications and Problems in Evolutionary Robotics", *First European Workshop, Evorobot'98*, P. Husbands and J.-A. Meyer (eds.). Springer Verlag 1998. ISBN: 3540649573.

[10] I. Harvey, P. Husbands, D. Cliff, A. Thompson, N. Jakobi, "Evolutionary Robotics at Sussex", In *Robotics and Manufacturing: Recent Trends in Research and Applications* vol. 6, M. Jamshidi, F. Pin and P. Dauchez (eds.), ASME Press, New York (1996) pages 293–298. *Proceedings of ISRAM96, International Symposium on Robotics and Manufacturing*, Montpellier, France May 27-30 1996.

[11] J. H. Holland, "Adaptation in natural and artificial systems", *University of Michigan Press*, An Arbor, MI, 1975.

[12] L. N. Moussi, R.R. Gudwin, F.J. Von Zuben and M. K. Madrid, "Neural Networks in classifier systems (NNCS): An application to autonomous navigation", in V.V. Kluev & N.E. Mastorakis (eds.) *Advances in Signal*

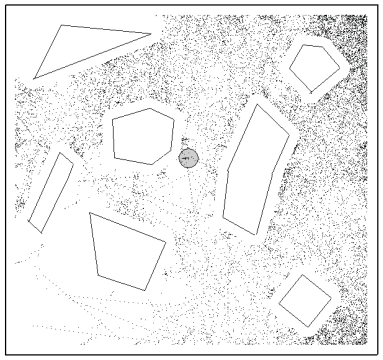

Fig. 5. Robot trajectory using enhanced CS during the learning period (polygon environment).

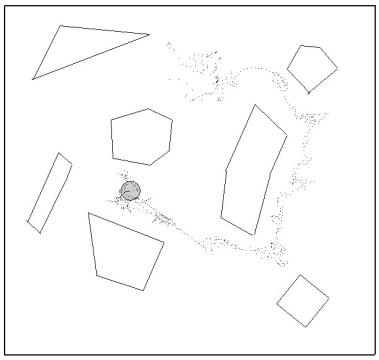

Fig. 6. Robot trajectory using enhanced CS after the learning period (polygon environment).

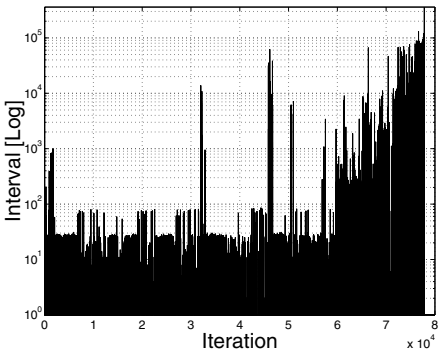

Fig. 7. Number of iterations between collisions (polygon environment).

Processing, Robotics and Communications, Electrical and Computer Engineering Series, WSES Press, pp. 256-262, 2001.

[13] L. B. Booker, D. E. Goldberg and J. H. Holland, "Classifier Systems and Genetic algorithms", *Artificial Intelligence*, 40:235-282, 1989.

[14] J. H. Holland, and J.S. Reitman, "Cognitive Systems Based on Adaptive Algorithms", in *D.A. Waterman and F. Hayes-Roth (eds.), Pattern-Directed Inference Systems*, Academic Press, NY, 1978.

[15] R. A. Richards, "Zeroth-order Shape Optimization Utilizing a Learning Classifier System", *Ph.D. Thesis*, Mechanical Eng. Dept., Stanford University, 1995.

[16] A. D. McAulay and J.C. Oh, "Improved leaning in genetic rule-based classifier system", *Proceeding of IEEE of IEEE International Conference on Systems, Man, and Cybernetics*, 2, 1393-1398, 1991.

[17] R. E. Smith and H. B. Cribbs, "Is a Learning Classifier System a Type of Neural Network?", *Evolutionary Computation 2(1)*; 19-36, 1994.