Journal of Advances in Computer Research
Quarterly pISSN: 2345-606x eISSN: 2345-6078
Sari Branch, Islamic Azad University, Sari, I.R.Iran
(Vol. 9, No. 1, February 2018), Pages: 51-70
www.jacr.iausari.ac.ir

The Introduction of a Heuristic Mutation Operator to Strengthen the Discovery Component of XCS

Ahmad Reza Pakraei✉[1], **Kamal Mirzaie**[2]
1) Group of Computer Engineering, Darab Branch, Islamic Azad University, Darab, Iran
2) Department of Computer Engineering, Maybod Branch, Islamic Azad University, Maybod, Iran
arpakraei@gmail.com, k.mirzaie@maybodiau.ac.ir

Received: 2016/11/03; Accepted: 2017/04/06

Abstract

The extended classifier systems (XCS) by producing a set of rules is (classifier) trying to solve learning problems as online. XCS is a rather complex combination of genetic algorithm and reinforcement learning that using genetic algorithm tries to discover the encouraging rules and value them by reinforcement learning. Among the important factors in the performance of XCS is the possibility to discover rules that are not only general as possible, but highly Accurate. In this paper, a new mutation operator is introduced for XCS that in addition to increasing the speed of learning will help improve performance. The purpose of speed is the amount of time that takes for the system to reach an appropriate solution and the purpose of the performance is the quality of solution that has been developed. The proposed algorithm was named XCS-KF and to evaluate its performance, it is used to solve the common problem in this area that is known as multiplexer. The results obtained showed that the speed and performance of the proposed algorithm to XCS algorithm increased significantly.

Keywords: *Classifier Systems, Genetic Algorithms, Discovery Component, Mutation Operator, Multiplexer*

1. Introduction

Learning classifier systems (LCS) [1] are population-based evolutionary algorithms. LCSs have many different types. They are used in different learning problems and have shown that they have appropriate speed and performance compared to other learning models [2].

While various implementations of the LCS algorithms exist , Holmes et al. [3] predetermine the following 4 main components for all of them : (1) a finite *Population* of classifiers that represents the current knowledge of the system, (2) *Performance* component adjusting the interaction between the environment and a classifier population, (3) *Reinforcement* component (credit assignment) which distributes the reward received from the environment to the classifiers, (4) *Discovery* component whose task is using operators for discovering better rules and improving existing rules. Figure 1 illustrates how specific mechanisms of LCS interact in the context of these major components. Two promoter engines in LCS are reinforcement and discovery components that discovering is mainly realized through a genetic algorithm.

One of the most successful types of LCS is eXtended Classifier System (XCS). XCS is a clever and smart innovation of Wilson [4, 5]. Regarding its attitude towards learning method, XCS is different from other LCS. XCS is able to derive a set of general rules using Q-learning and a genetic algorithms. XCS has been considered by many researchers due to its use in solving a wide range of learning problems such as optimization, classification and reinforcement learning [6, 7, and 8].

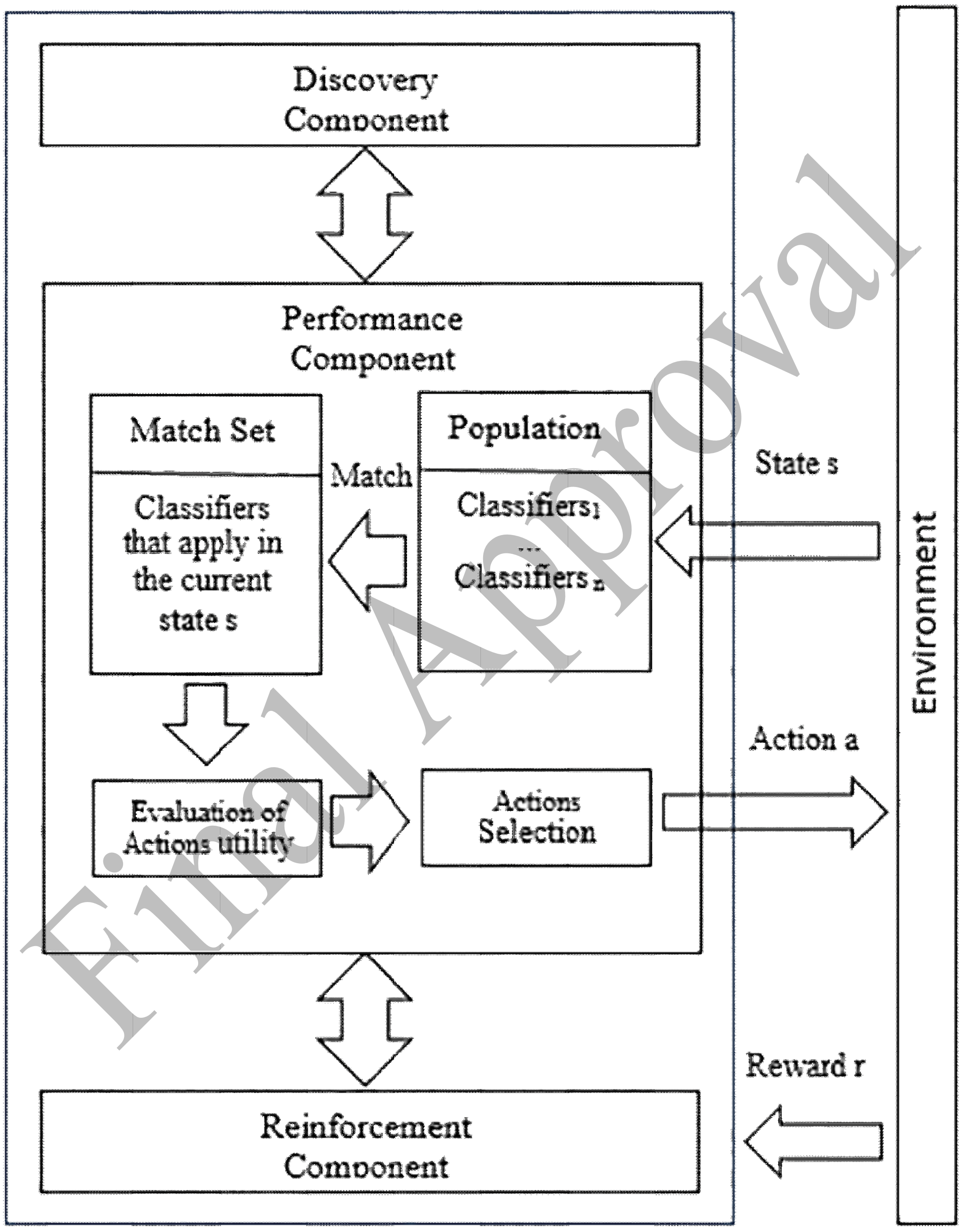


Figure 1. Learning classifier systems in interaction with the environment

During recent years, much research has been done on XCS mostly focusing on mechanisms of reinforcement learning [9, 10, and 11] and less research has been done on discovery component and especially various genetic operators that play an important role in the performance of the systems. However, Nakata et al. [12] introduced XCS with a novel rule discovery mechanism that was an extension of the compact genetic algorithm (cGA) to XCS. In detail, their rule discovery mechanism evolves classifiers and mutates classifier conditions using the probability vector as attribute feedback. Their experimental result showed that XCS with their rule discovery mechanism (or XCScGA) can reach optimal performance with fewer rule evaluations and with smaller population sizes in the Boolean multiplexer problem.

Butz et al. [13] showed that successful genetic learning depends on fitness pressure to more accurate rules. They found that proportionate selection will not lead to adequately strong fitness pressure so that XCS is not able to learn a problem. To overcome this problem, they proposed tournament selection and experimentally showed that tournament selection leads to more stable, reliable and focused pressure along with higher accuracy.

In [14], researchers have presented theoretical models of selection pressure in XCS for proportionate and tournament selection. The models confirmed the results obtained in previous empirical research on proportionate and tournament selection methods and showed that tournament selection is stronger in making pressure to fitness. So either empirically or theoretically, it is recognized that tournament selection works stronger because it does not depend on the fitness of each classifier.

In [15], researchers introduced a new crossover operator called BLX for use in XCS and then measured its performance to two-point crossover operator common in XCS. The results of their experiments for different problems show that operator BLX on average acts better than two-point crossover operator. Finally, researchers, by examining XCS performance using different crossover operators point out the fact that different operators should be selected and used carefully and according to the problem.

In [16], a self-adapting mutation mechanism has been proposed to be used in XCS. The results of applying the algorithm in various test environments have shown that self-adapting mutation rate can reduce integrity problems in long classifiers' chain and lead to increasing the efficiency sufficiently.

This paper introduces a new mutation operator for discovery component of XCS. The genetic algorithm (GA) with the typical mutation operator is replaced with proposed mutation operator. The performance measured when using the new mutation operator in XCS is tested on solving multiplexer problems.

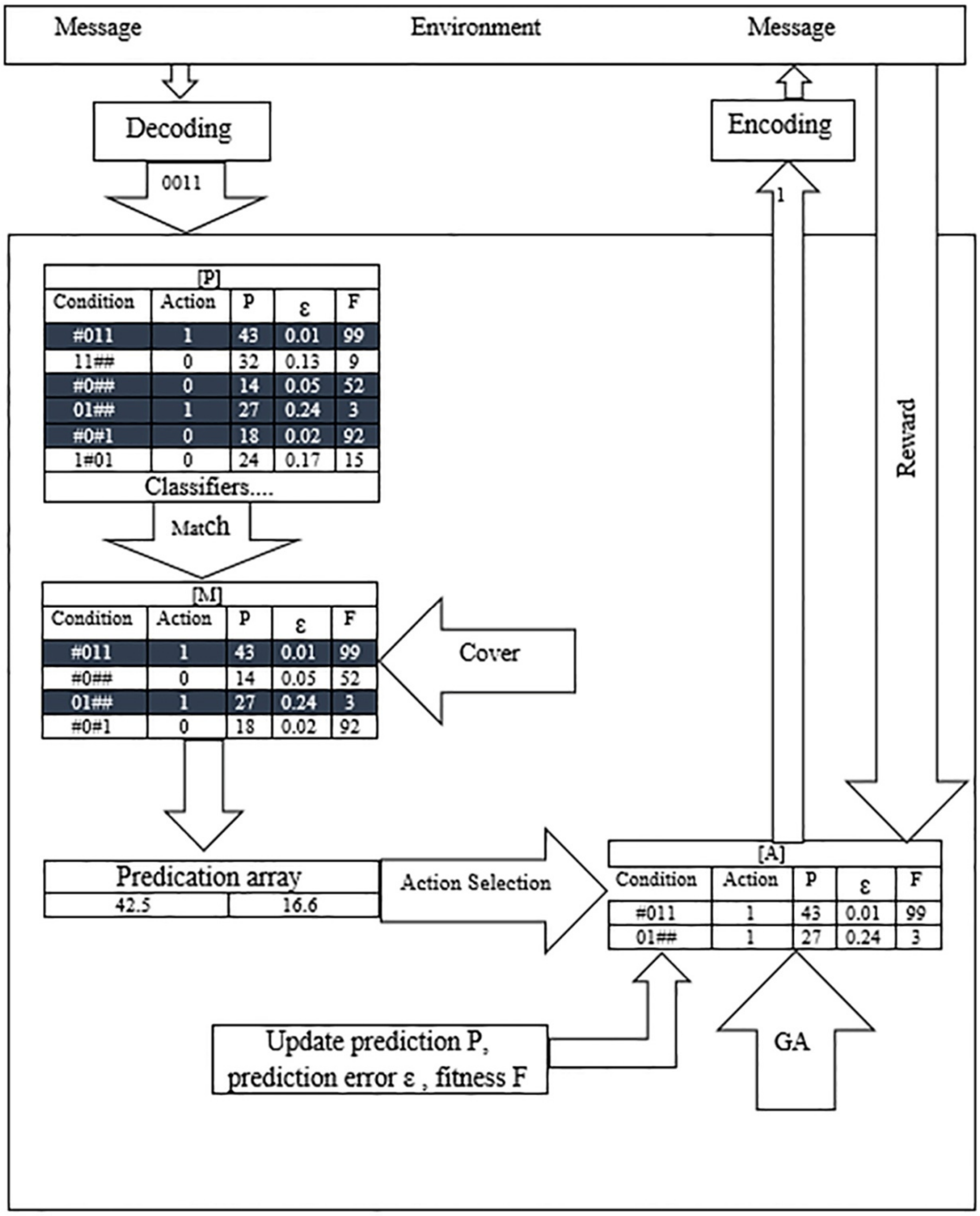
The paper is structured as follows. Section 2 provides a short overview of XCS with all details that are important for the remainder of paper. The extended XCS with the new mutation GA is introduced in Section 3. The new model is confirmed experimentally in Section 4. And finally, in Section 5 the conclusions are drawn.

2. eXtended Classifier System(XCS)

XCS, as an online learner machine, gains knowledge in interaction with an environment with little knowledge about it and evolves it over time. XCS knowledge is represented in the form of a set of rules each of which is called a classifier. So, XCS has [P] set that contains produced and evaluated classifiers so-called population. Each classifier has two main parts, condition and action, and a series of learning parameters that maintain the quality. These parameters are P prediction of the amount of reward in the future, prediction error of the past and F fitness of a classifier to other classifiers.

Also each classifier has the parameter num that keeps the number of classifiers covered by the classifier. The coverage means classifiers with the same condition and action.

According to Figure 2, the XCS's learning cycle is as fallow. It receives an input such as 0011 and then searches its current knowledge base, [p], following a set of classifiers that their condition matches to the current input such as #011 and these classifiers make a matched set [M]. It should be noted that the alphabet used in this study to show condition and action is ternary alphabet {0, 1, #} where # (don't care) means the input can be any value of 1 and 0.



[P]

Condition	Action	P	ε	F
#011	1	43	0.01	99
11##	0	32	0.13	9
#0##	0	14	0.05	52
01##	1	27	0.24	3
#0#1	0	18	0.02	92
1#01	0	24	0.17	15

[M]

Condition	Action	P	ε	F
#011	1	43	0.01	99
#0##	0	14	0.05	52
01##	1	27	0.24	3
#0#1	0	18	0.02	92

Predication array

42.5	16.6

[A]

Condition	Action	P	ε	F
#011	1	43	0.01	99
01##	1	27	0.24	3

Figure 2. Learning Cycle in XCS

Since the matched set classifiers may propose different actions therefore an action should be selected. In some cases, the proposed action is random to explore the environment and in other cases is deterministic and based on previous learning. In both cases, before applying the proposed action to the environment all classifiers advocated the selected action make a new set called action set [A]. Then the system action proposed is applied to the environment. Finally, environment provides a feedback, indicating the desirability of the proposed action. This feedback updates various parameters of learning all classifiers of the action set [A].

Learning parameters of classifiers of the action set are updated by learning reinforcement component that is intended for the system .Therefore the prediction and prediction error will be updated such as (1).

$$P \leftarrow P + \beta\left(R - p\right),\ \epsilon \leftarrow\ \epsilon +\ \beta\left(\left|R - P\right| - \epsilon\right) \qquad (1)$$

Where $\beta(0 < \beta < 1)$ is learning rate, and accuracy k and relative accuracy k' of classifiers are calculated by (2).

$$\mathrm{k} = \begin{cases} 1 & if\ \epsilon < \epsilon_0 \\ \alpha\left(\dfrac{\epsilon}{\epsilon_0}\right)^{-\nu} & otherwise \end{cases} \quad and \quad k' = \frac{k}{\sum_{x \in [\mathrm{A}]} k_x} \qquad (2)$$

Where α , ϵ_0 and ν are learning parameters to calculate the fitness by (3).

$$F \leftarrow F + \beta\left(k' - F\right) \qquad (3)$$

2.1 The Discovery Component

A standard discovery component is comprised of a genetic algorithm and a covering mechanism. The main role of a covering mechanism is to ensure that there is at least one classifier in population [P] that handle the system input. If such a classifier does not exist, a new classifier is produced by adding some number of #'s at random to the input data and then selecting a random action [17]. For example, a new rule such as 0#110#00 might be created from the input data 00110100. The parameter Values of a new created classifier are calculated according to average parameter of the present population in the system.

A covering operator at the beginning might be used to make an initial population. The covering operator can be implemented by different methods such as the rate of using # to create the new rule, how to calculate a new classifier parameter and which covering is called. In fact, a covering operation allows the system to present a new hypothesis to solve a problem [18].

A genetic algorithm produces new rules according to the knowledge of the present population. Usually, LCSs use a genetic algorithm in a steady state as a covering mechanism. This means that at any time of using the genetic algorithm only 2 classifiers are selected. In this case, the population rules are changing with no defined understanding of a generation. This differs from a generation genetic algorithm in which all or the bulk of the population is re-changed from a generation to another [19]. Like a covering mechanism, determining how a genetic algorithm is applied in a LCS algorithm is a problem very different from a system to another. Answering 3 below questions can be the best index to classify the systems: 1-Where should an algorithm be applied? 2-When should an algorithm be applied? And 3- What operators should be used?

A set of classifiers on which a genetic algorithm is applied can have a significant effect on their evolutionary pressure. Primitive systems applied a genetic algorithm on [P] or classifiers' initial population [20]. Methods such as restricted mating and niching applied a genetic algorithm on [M] and then later to [A] that today they are used in modern systems such as XCS [21].

Mutation and crossover are 2 known operators of a genetic algorithm that both mechanisms are called by parameters determining their relative application probability. Historically, most of initial LCSs use a one-point crossover operator. But an interest in discovering complex building blocks [22, 23] led to test operators such as tow-point, informed and uniform cut in XCS [24].

2.2 The Genetic Algorithm in XCS

One of the most complex parts of XCS is discovery component of classifiers [25]. Wilson for the first time used a genetic algorithm (GA) to make new classifiers on actions set. Before that, GA has been used for new classifiers exploration on match sets or the whole population [4].

First, all action sets are checked whether GA should be apply on the Current set or not? If average time period of the last time of using discovery component for action sets is higher than threshold $\square_{GA}$, discovery component will be applied. Then, two classifiers are selected by Roulette wheel and according to their fitness to the whole population fitness then new offspring are created out of them. It is possible that new offspring are changed following crossover operations with the probability χ and/or mutation with the probability μ. If offspring are affected by crossover operations, their values of prediction, prediction error and fitness are initialized with average value of parents. Finally, new offspring after deletion operation are added to the population of classifiers. But before that if covering deletion operation is allowed each offspring is examined to determine whether it is covered by parents or not. If yes, they are not added to the population and parent's *num* value is increased.

3. The Proposed Algorithm

In this section, in addition to defining concepts such as central classifier, neighborhood and introducing the operator k-flip, a mutation operator is introduced to be used in discovery component of XCS that leads to increased speed and quality of learning. The new system is called XCS-KF.

3.1 The Central Classifier

The central classifier, F, is the best classifier of set [A] is selected as according to indicators such as the Best fitness, Most Accuracy or Numerate. In fact, the central classifier is local optima [26] that finds during a local search operation. In Figure 3, Pseudo code of IDENTIFY function is given to determine the central classifier.

```
$IDENTIFY\ Central\ Classifier\ ([A])$
1  $bestclassifierindex \leftarrow 0$
2  $MaxValue \leftarrow 0$
3  $for\ each\ classifier\ cl\ in\ [A]$
4      $if\ (cl.F > MaxValue)$
5          $MaxValue \leftarrow cl.F$
6          $bestclassifierindex \leftarrow cl_{index}$
7  $return\ cl_{bestclassifierindex}$
```

Figure 3. Pseudo code to detecting of central classifier (Best fitness)

3.2 Neighborhood

The notion of neighborhood is essential to understand the proposed algorithm. In a search space, solution s' is termed a neighbor of s if the prior can be reached from the latter in a single step. The neighborhood N(s) of a solution s is the set of all its neighbors. Defining neighborhood in problems with a combinatorial search spaces is difficult and challenging. Combinatorial search spaces are finite and are used for problems with limited size, where each solution in a problem search space is represented by a series of variables. In such a space, if solutions of problem are representation of variable v_i, each of them taking values from a domain $\mathfrak{D}_i$ and each of combined values of different variables is possible, then search space S will be equal to $S = \prod_{i=1}^{n} \mathfrak{D}_i$ In such a space, the relationships between neighbors can be defined according to modifications in one or more variables. For example, binary strings are a typical way to show solutions in XCS algorithm. In this case, solutions belong to $\mathbb{B}^n$ and a set of s' nested neighborhoods can be defined for solution s as follows $\mathcal{N}_k(s) = \{s' | H(s, s') = k\}$ Where $H(s, s')$ is s, s' Hamming distance [27].

3.3 The Operator k-flip

The concept of neighborhood can be explained with the help of the moving operator. Usually, using the operator in candidate solution s and its partial changes we can reach solution s' in its neighborhood. Transferring from solution s to another one like solution s' in its neighborhood is shown as $s' = s \oplus mv$. If $\Gamma(s)$ is a set of all possible movements considered for s, then all its neighbors are defined as $N(s) = \{s \oplus mv | mv \in \Gamma(s)\}$. One of the widely used moving operators to determine neighbors in problems of binary space is $k - flip$. The operator changes $k (k \geq 1)$ variable of a solution and in this way each s neighbor $s' \in N(s)$ has Hamming distance equal to k.

3.4 The Heuristic Mutation Operator

XCS with the help of a genetic algorithm seeks to explore answer-prone areas in problem-solving space and find optimized solutions in the areas. Exploration operation is possible mostly through using crossover operator and finds solutions in the areas by a local search with the help of mutation operator. XCS operates well in finding areas with solutions of problem-solving, but it operates slowly in approaching final solutions. Then, it was attempted to solve the problem by changing the mutation operator of a genetic algorithm. For this purpose, here a 2-step mutation operator is presented.

Step one, both condition and action of each new classifier are mutated according to the procedure presented in the following. The mutation in the condition changes values of each classifier's variables with the probability μ to # or corresponding value in the current input $\sigma(t)$ to cover the system input. This operation is called the limited mutation. Mutation in the action changes it equal probability to the other action. Figure 4 shows Pseudo code of MUTATION function.

```
APPLY MUTATION(cl, $\sigma$)
1   $i \leftarrow 0$
2   $do\{$
3       $if\ (RandomNumber[0,1) < \mu)$
4       $if\ (cl.C[i] = \#)$
5           $cl.C[i] = \sigma[i]$
6       $else$
7           $cl.C[i] = \#$
8       $i++$
9   $\}While(\ length\ of\ cl.C)$
10  $if\ (RandomNumber[0,1) < \mu)$
11      $cl.A \leftarrow$ a randomly chosen another action
```

Figure 4. Pseudo code of Mutation operation

Step two, during FLIPPING operation, values of max k variable of the each new classifier condition is flipped using operator $k - flip$. This process is done according to corresponding value in the central classifier which is called free mutation operation. So, new classifiers are made at max distance k to the central classifier. Here, k is called neighborhood radius. Figure 5 shows Pseudo code of FLIPPING function.

```
FLIPPING(cl, F, $k$)
1   $i \leftarrow 0$
2   $do\ \{$
3       $if\ (RandomNumber[0,1) < k*0.5)$
4           $if\ (cl.C[i]! = F.C[i])$
5               $cl.C[i] = F.C[i]$
6   $i++$
7   $\}while\ (i < length\ of\ cl.C)$
```

Figure 5. Pseudo code of FLIPPING function

3.5 The New Genetic Algorithm

Figure 6 shows the flowchart of the new genetic algorithm. After initialization parameters that are needed. Two classifiers, as Parents, for reproduction are selected by the Roulette wheel. These two classifiers are selected according to the fitness to other classifiers in set [A]. After selecting parents, two classifiers cl_i and cl_j are produced, and their different parameter values are initialized according to the parents. Then, crossover operator is applied with the probability χ on cl_i and cl_j. If classifiers are changed by crossover operator, their predication parameter values are initialized according to average values of parents. Here, two-point crossover operator is used. Following selection and crossover operations, with the help of the IDENTIFY function, the central classifier is selected. After that, with the probability μ and the applying of the APPLY MUTATION and FLIPPING function, classifiers cl_i and cl_j previously made are changed in a way that are in the neighborhood of the central classifier. In other words, new classifiers are selected in the neighborhood of the central classifier. It is likely to discover better classifiers by exploring in the neighborhood of the central classifier as a local optima. So creating new classifiers in the neighborhood of the central classifier can be effective on more rapid final solutions' achievement.

Finally, new classifiers are added to the population competing their parents. But before that, if covering deletion operation is allowed during exploration operation, each of classifiers cl_i and cl_j are examined by parents or any member of the population in terms of coverage and if they are covered by any of them, new offspring are not added to the population, so *num* value of the classifier is increased. Figure 7 shows Pseudo code of the proposed genetic algorithm.

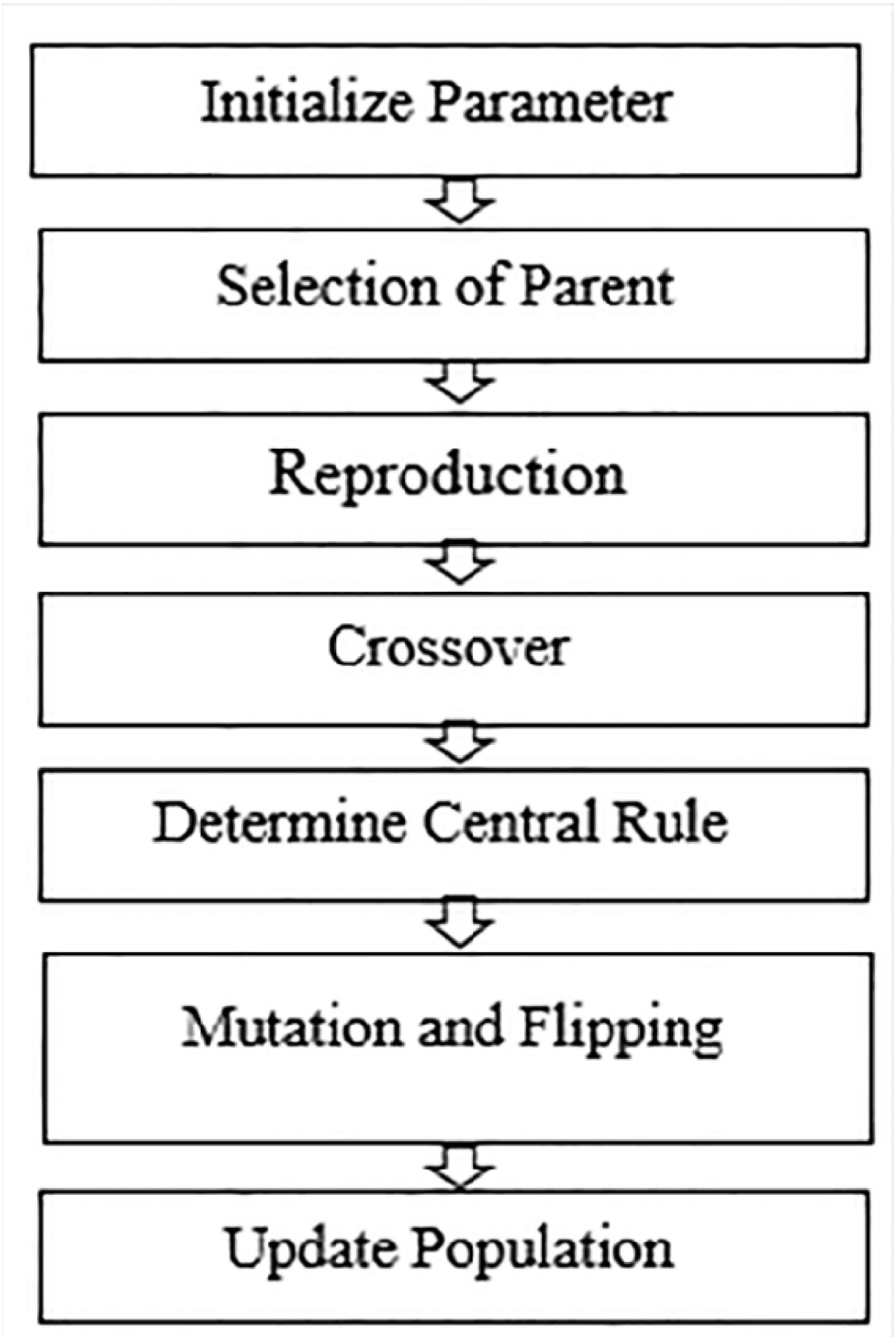


Figure 6. Simple flowchart of the proposed genetic algorithm

```
$RUN\ GA([A], \sigma, k, [p])$
1  if($actual\ time\ t - (\sum_{cl \in [A]} cl.ts * cl.n / \sum_{cl \in [A]} cl.n) > \theta_{GA}$)
2      $for\ each\ classifier\ in\ [A]$
3          $cl.ts \leftarrow actual\ time\ t$
4   $parent_1 \leftarrow SELECT\ OFFSPRING\ in [A]$
5   $parent_2 \leftarrow SELECT\ OFFSPRING\ in [A]$
6  $child_1 \leftarrow copy\ clissifier\ parent_1$
7  $child_2 \leftarrow copy\ clissifier\ parent_2$
8  $child_1.n = child_2.n \leftarrow 1$
9  $child_1.exp = child_2.exp \leftarrow 0$
10 $if\ (RandomNumber[0,1) < \chi)$
11     $applay\ CROSSOVER\ on\ child_1\ and\ child_2$
12     $child_1.p \leftarrow (parent_1.p + parent_2.p)/2$
13     $child_1.\epsilon \leftarrow (parent_1.\epsilon + parent_2.\epsilon)/2$
14     $child_1.F \leftarrow (parent_1.F + parent_2.F)/2$
15     $child_2.p \leftarrow child_1.p$
16     $child_2.\epsilon \leftarrow child_1.\epsilon$
17     $child_2.F \leftarrow child_1.F$
18  $child_1.F \leftarrow child_1.F * 0.1$
19  $child_2.F \leftarrow child_2.F * 0.1$
20  $F \leftarrow IDENTIFY\ CENTERAL\ RULE\ ([A])$
21  $for\ both\ childern\ child$
22      $APPLAY\ MUTATION\ on\ child\ according\ to\ \sigma$
23      FLIPPING child according to F $AND\ maximum\ radius\ of\ the\ neighborhood\ k$
24      $if\ (doMASubsumption)$
25          $if\ (DOES\ SUBSUM\ parent_1, child)$
26              $parent_1.n{+}{+}$
27          $else\ if\ (DOES\ SUBSUM\ parent_2, child)$
28              $parent_2.n{+}{+}$
29          $else$
30              $INSERT\ child\ IN\ POPULATION\ [P]$
31      $else$
32          $INSERT\ child\ IN\ POPULATION\ [P]$
33      $DELETE\ FROM\ POPULATION\ [P]$
```

Figure 7. The proposed genetic algorithm

4. Evaluating the Proposed Algorithm

To measure the performance of new discovery component and understand the effect on XCS performance, the both algorithms XCS and XCS-KF are used to solve a problem such as multiplexers. Both algorithms XCS and XCS-KF are compared and examined according to the speed of achieving a solution as well as final population size to solve a problem such as multiplexers. A problem such as multiplexers is one of one-step environments providing reward from outside to the system in each time step and the system state is independent of previous states at any time step.

4.1 Multiplexers

A problem such as multiplexers and using them in LCSs research is very common like using them in other fields of machine learning. For many years, they have been used in developing LCSs and have several versions. One kind of classifying multiplexers is according to their size. Their standard size is usually n+2^n (n>=1). Often multiplexers are shown as 3- MP(1+2^1), 6-MP(2+2^2) and 11- MP(3+2^3). By increasing the size of multiplexers, they will be solved more difficult. In all machine learning fields using multiplexers, the system performance depends on multiplexer's size. For example, using small multiplexers (3 or 6 MP) no good benchmarks is provided to measure a system performance. Instead, using large multiplexers (11 or 20 MP) can provide a good degree of difficulty for a learner. Then, a standard multiplexer performance is given.

A Boolean multiplexer defined for a binary space produces a list of n+2^n bit messages, where messages' size is equal to the multiplexer size. A learner should receive a message produced by a multiplexer and produce a proper response, 0 or 1. First, a multiplexer separates the first n bit of a message (address bit) equal to an integer coded to binary then converts a bit address into i, an integer coded. Then, the multiplexer considers the remaining bit (2^n) of a message as data, containing possible solutions of a problem. The value of ith bit extracted from a message address is a solution. Finally, if a learner provides a response equal to ith bit then the multiplexer gives a positive message otherwise it gives a negative message. For example, a multiplexer 3+2^3 is consisted of 3 bits in address and 8 bits in data. If a message $\frac{111}{\text{addres}\ bits}\frac{11111110}{data\ bits}$ is received, 3 address bits show No. 7, so 7th bit of data bits contains a problem solution, here it is 0. If the answer is right, the learner receives a positive message.

4.2 Experimental Parameters

In XCS, each classifier has different parameters such as p , ε , F and numused by an algorithm during learning process. In addition, an algorithm uses other different parameters to have a proper performance and their values differ from a problem to another. In each problem using an algorithm, the user should find proper settings according to given problem that should be done carefully because will have important effects on an algorithm performance. For example, learning parameters β , α , ε_0 and υrules discovery parameters χ , μ and $P_\#$and population parameters N , θ_{GA} *and* $\theta_{subsumption}$ can be mentioned. Other important case of LCSs is that reward that in one-step test environments the reward is immediately provided, 1000 for correct and 0 for a wrong answer. In addition, XCS-KF has a special parameter K (neighborhood radius). It should be noted that in all tests, covering deletion operation is allowed and classifiers are controlled before being added to the population in terms of coverage by parents and other classifiers of the population.

4.3 Implementation

To perform all tests, XCS implemented version by Butz has been used [28]. The version has also been used as the base of XCS-KF. Values considered for the algorithm different parameters in this study are proposed in [28] $\alpha = 0.1, \beta = 0.2, \upsilon = 0.1, \Theta_{GA} = 25,\ \varepsilon_0 = 10,\ \chi = 0.8, \mu = 0.04, P_\# = 0.5, \delta = 0.1, \Theta_{sub} = 20$. The population size N for multiplexer 11 is 800 and for multiplexer 20 is 2000.

4.4 Performing Experiment

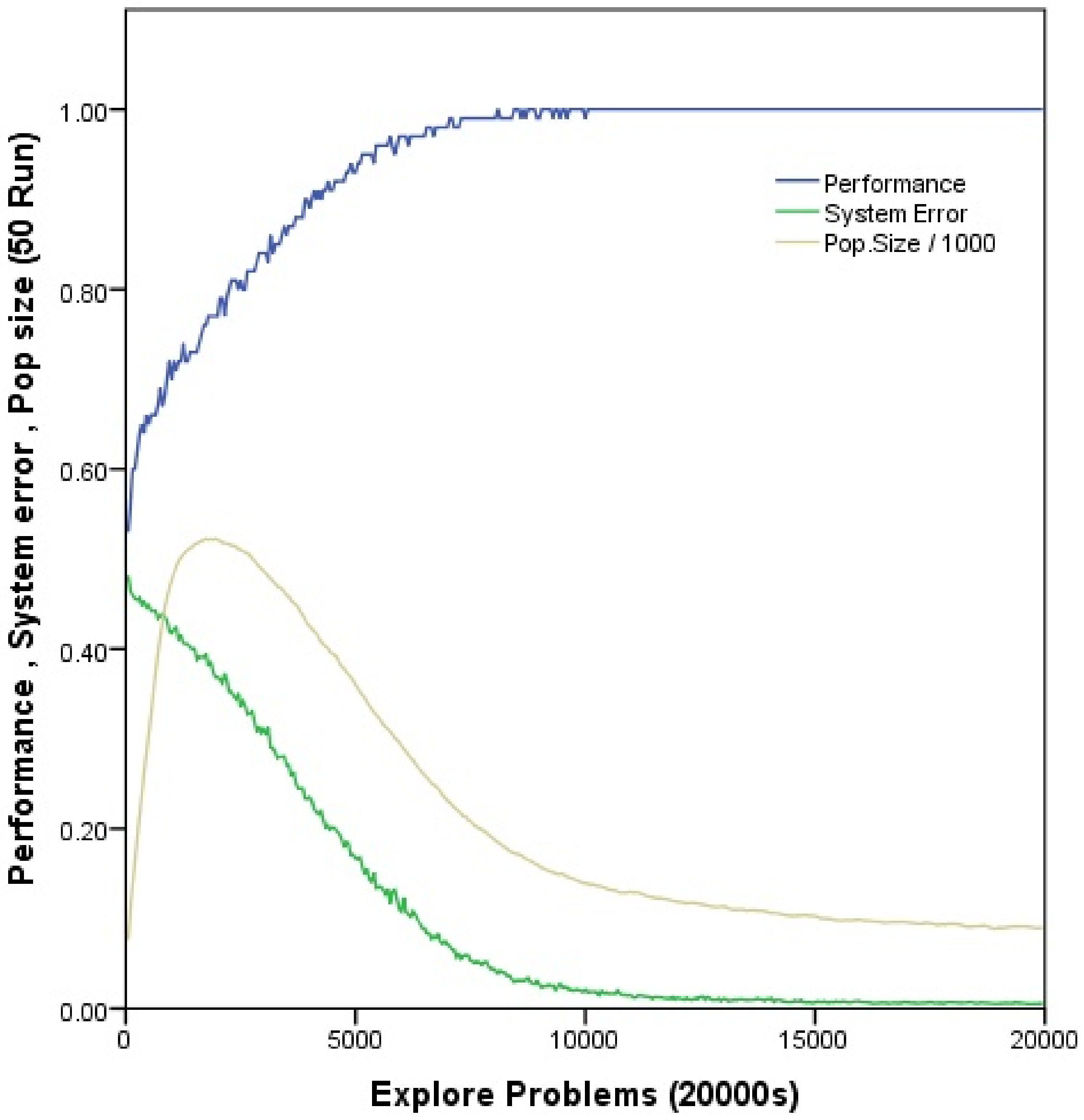
To compare the performance of two algorithms XCS and XCS-KF, the 2 algorithms are used in an experiment to solve a problem such as 11-MP and 20-MP. Then, the results obtained are compared to each other and analyzed statistically.

In each experiment, number of input is entered the system randomly and for each input the system should provide a proper response as its proposed action. Since LCSs perform in two exploration and exploiting modes, performance, population size and system error are recorded in exploiting mode.

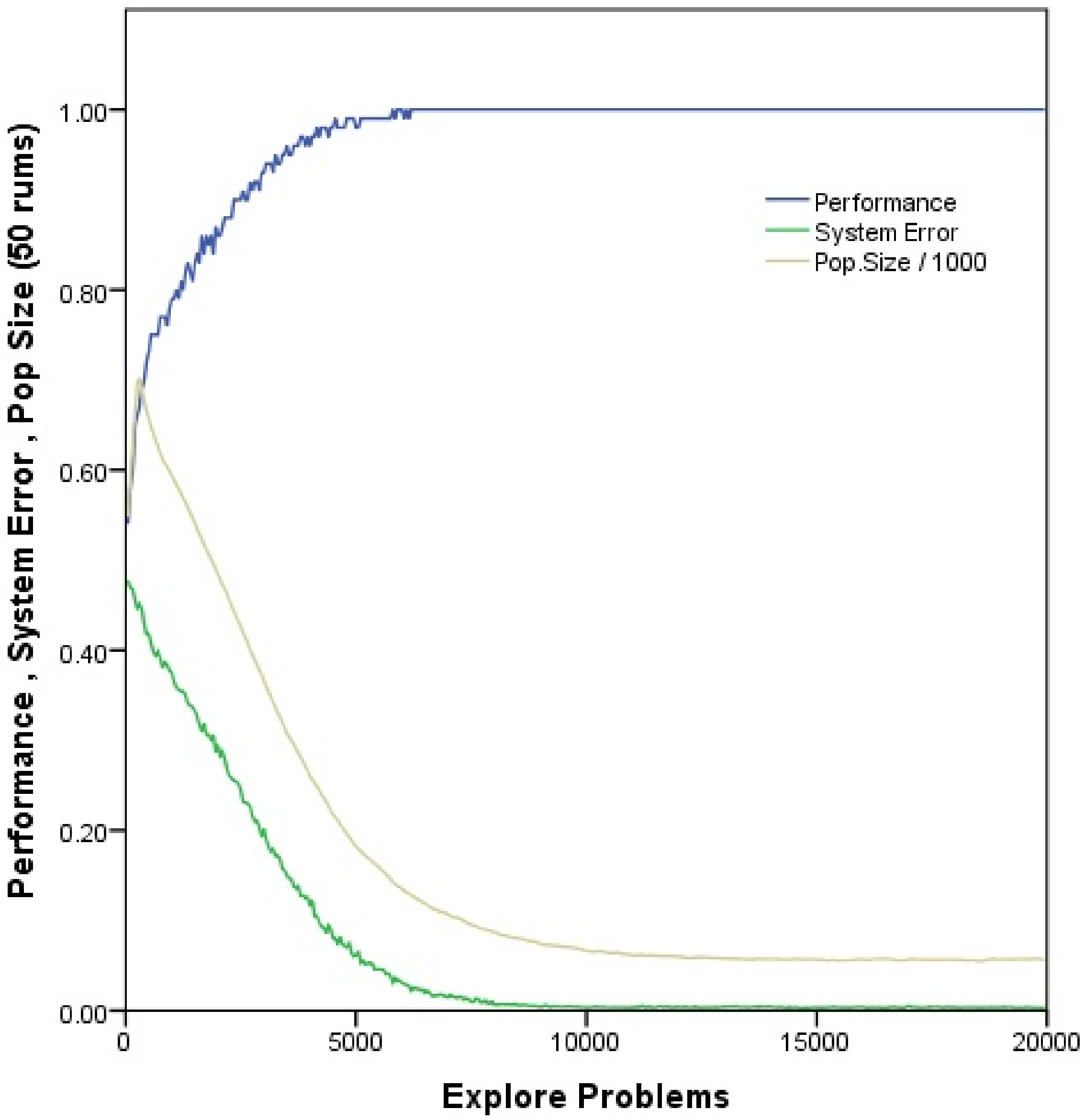
Performance is the number of correct answers of a system in the last 50 times of performing an operation. The system error is average absolute value of the difference between the system prediction of selected action and external reward received divided by 1000. The population size is existing classifiers. Each experiment has been repeated 50 runs to ensure the results and finally average performance, error and population size has been calculated.

4.4.1 Multiplexer Problem Analyses

First, XCS has been used to solve 11-MP average performance, error and population size has been calculated. Figure 8a shows that XCS with about 10000 input samples, half of them for exploration, has reached 100% performance. Also in this case, the system error has approached 0. The Figure 8a of changed population size shows changes in the number of the population classifiers. The population size began from 0, and after reaching its max it has returned to less than half of the max, about 105 classifiers. Then, algorithm XCS-KF was used to solve the problem of 11-MP with k= 4,5,6,8 and selecting the central classifier, i.e. the Best fitness, Most Accuracy or Numerate and the results were analyzed. The results showed that the algorithm had the highest performance with k= 5 as well as selecting the Best fitness classifier as the central classifier. Figure 8b shows average performance, error and population size of XCS-KF after repeating the experiment 50 runs. According to the Figure 8 it is found that the system had about 6500 inputs with 100% performance and 0 error. Also final average population size has been reduced from 105 to 65 classifiers. This means that the system produced a set of more general and high-quality classifiers. So XCS-KF to XCS 35% has reached 100% performance faster and experienced 36% more general rules.


Performance , System error , Pop size (50 Run)
1.00
0.80
0.60
0.40
0.20
0.00
0
5000
10000
15000
20000
Explore Problems (20000s)
Performance
System Error
Pop.Size / 1000

(8a) XCS

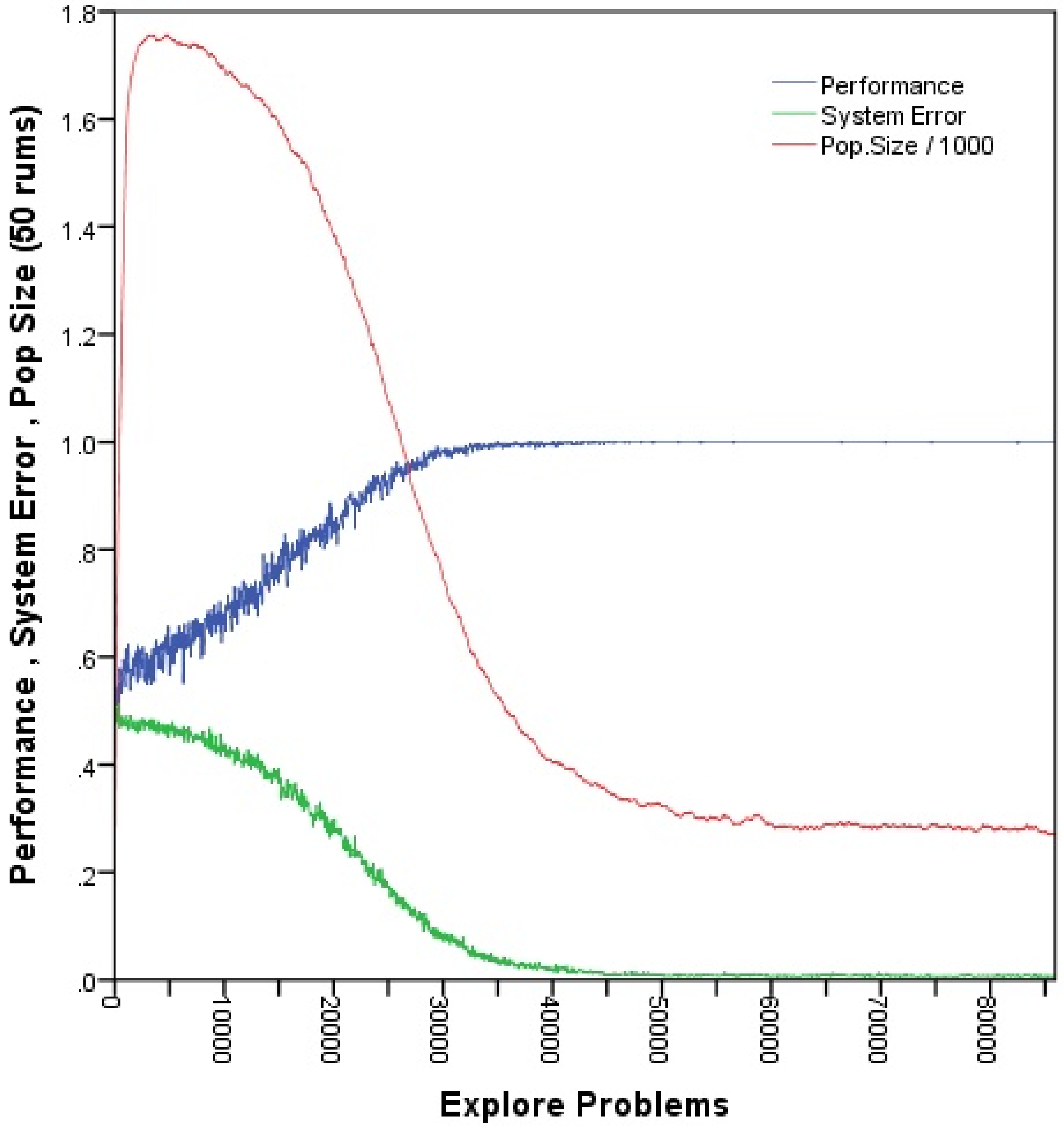


(8b) XCS-KF

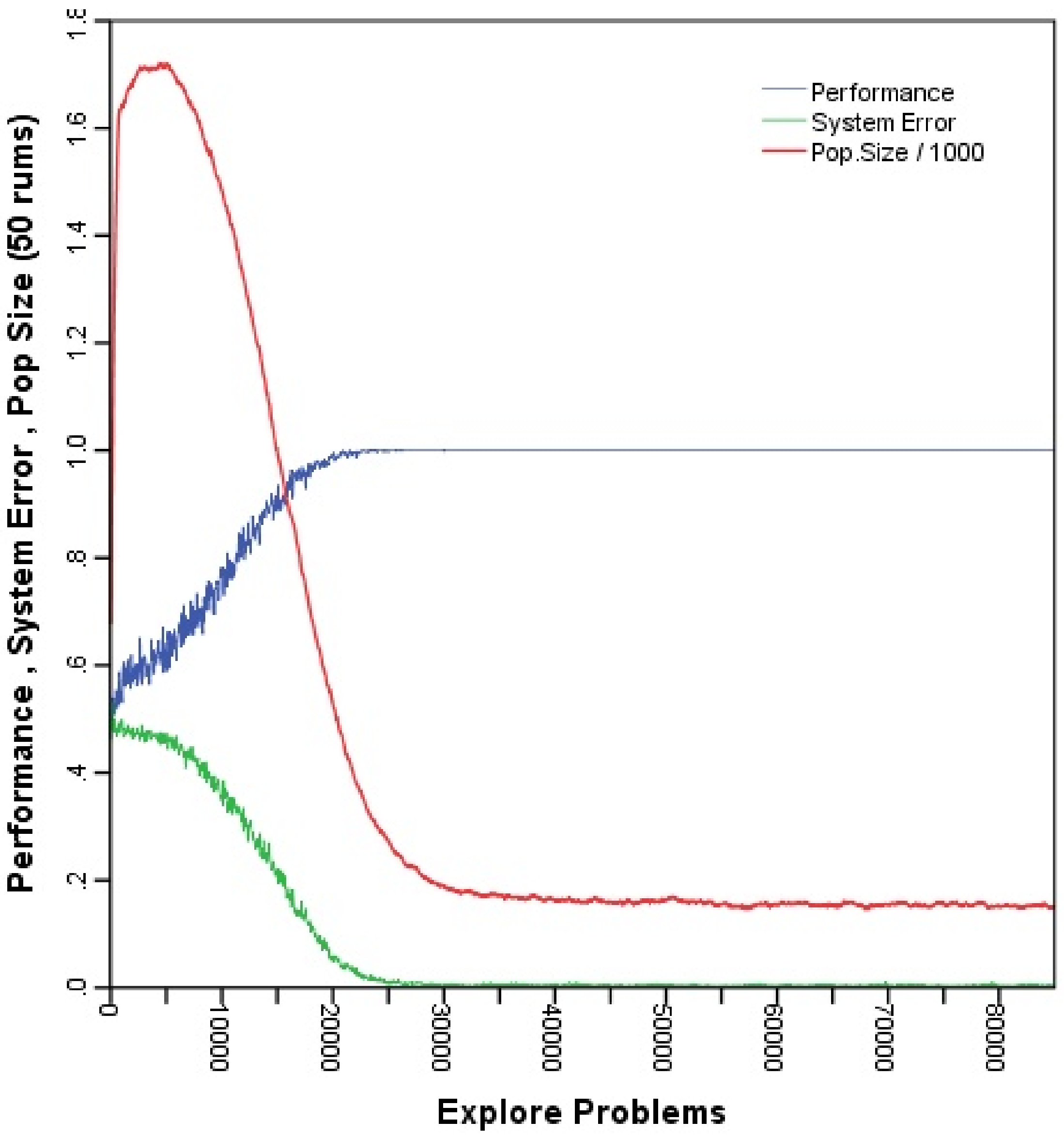
Figure 8. Results in a 11-multiplexer experiment. Blue curve: Performance, the fraction of last 50 exploit problems correct. Green curve: System error as a fraction of total reward range. Gray curve: Population size in classifiers (divided by 1,000). Are averages of 50 runs.

4.4.2 Larger Multiplexer Problem

In the following, XCS and XCS-KF were used with the same parameters to solve a problem of 20-MP. Figure 9a shows that XCS with about 61000 inputs has reached 99.8% performance. The population size began from 0, and after reaching its max it has returned to 290 classifiers at the end of the experiment. Figure 9b shows average performance, error and population size for XCS-KF. According to the Figure 9b it is found that the system had about 30000 inputs with 100% performance and 0 error. Also final average population size has been reduced from 290 to 160 classifiers. So XCS-KF to XCS 50% has reached 100% performance faster and experienced 44% more general rules.


Performance , System Error , Pop Size (50 rums)
1.8
1.6
1.4
1.2
1.0
.8
.6
.4
.2
.0
0
10000
20000
30000
40000
50000
60000
70000
80000
Explore Problems
Performance
System Error
Pop.Size / 1000

(9a) XCS



(9b) XCS-KF

Figure 9. Results in a 20-multiplexer experiment. Blue curve: Performance, the fraction of last 50 exploit problems correct. Green curve: System error as a fraction of total reward range. Red curve: Population size in classifiers (divided by 1,000). Are averages of 50 runs.

5. Data Analysis and Testing Hypothesis

The study hypotheses were analyzed according to the data obtained from implementing two algorithms XCS and XCS-KF to solve the problem of multiplexers. This analysis was carried out by applying different statistical tests such as Levene's test and independent t-test with the help of statistical software SPSS. Leven test examined variables' variance homogeneity and t-test was used to compare the average of 2 independent samples of quantitative data. In the test, hypotheses null (H_0) and 1 (H_1) are as (4):

$$\begin{cases} H_0 : \mu_1 = \mu_2 \\ H_1 : \mu_1 \neq \mu_2 \end{cases} \tag{4}$$

Null hypothesis showed that no significant difference was found between average 2 populations. While hypothesis 1 showed a significant difference, it should be noted that to compare the averages, first variances should be calculated and in fact Variance test is prior to the average test. To compare performance, the system error and average population size of the 2 algorithms, null hypothesis is as follows: No significant difference was found between performance, the system error and average population size of the 2 algorithms.

Here, T-test is used to compare the average results and prove the hypothesis and also reject the probability that the relationship between the two averages are random. The test results show that the difference between performance, the system error and average population size of XCS and XCS-KF is statistically significant at the level (99% >).

6. Conclusion

In this study, a two-step mutation operator was introduced to be used in a genetic algorithm for discovery component of XCS. The proposed model is called XCS-KF. Then to assess its performance, it was used to solve the problem of 11-mp and 20-mp. Figure 8b and 9b show how the performance XCS-KF with the new mutation operator is better than the traditional XCS. Note that there is statistically significant benefit (T-test, time taken to reach and maintain optimality over 50 subsequent exploit cycles for average of 50 runs, $p<0.01$) from the new discovery mechanism. So Experimental' results showed that XCS-KF clearly has better performance and higher learning speed to solve the problem of multiplexers than XCS. Using proposed mutation operator not only increases the speed of XCS learning but also produces more general and accurate classifiers. It is seen that increased speed learning using proposed algorithm does not lead to reduced accuracy of the system.

References

[1] Holland, J. H., Adaptation. In R. Rosen and F.Snell, editors, Progress in Theoretical Biology, vol. 4, pp. 263–293. New York, Academic Press, 1976.

[2] Bull, L., Applications of Learning Classifier Systems. Springer, Heidelberg, 2004.

[3] Holmes, J. H., Lanzi, P. L., Stolzmann, W., and Wilson, S. W., "Learning classifier systems: new models, successful applications", Information Processing Letters, vol. 82, No. 1, pp. 23–30, 2002

[4] Wilson, S.W., "Classifier fitness based on accuracy", Evolutionary Computation, vol. 3, No .2, pp. 149–175, 1995.

[5] Wilson, S.W., "Generalization in the XCS classifier system", In:Koza, J.R., et al. (eds.) Genetic Programming, pp. 665–674, Morgan Kaufmann, San Francisco, 1998.

[6] Butz, M. V., "Combining gradient-based with evolutionary online learning: an introduction to learning classifier systems", in Proceedings of the 7th International Conference on Hybrid Intelligent Systems (HIS '07), pp. 12–17, 2007.

[7] Abbasi, E., Naghavi,N. "Offline Auto-Tuning of a PID Controller Using Extended Classifier System (XCS) Algorithm." Journal of Advances in Computer Engineering and Technology 3.1 (2017): 41-50, 2017.

[8] Sabzehzar, A., and et al. "An Improved eXtended Classifier System for the Real-time-input Real-time-output (XCSRR) Stability Control of a Biped Robot." Procedia Computer Science 61 (2015): 492-499, 2015.

[9] Zhaoxiang, Z., Dehua, L., Junying, W., and Dan, X.," Learning classifier system with average reward reinforcement learning", Knowledge-Based Systems, vol. 40, pp. 58-71, 2013.

[10] Masaya, N., Lanzi, P.L, Kovacs, T.,"How should learning classifier systems cover a state-action space?", in IEEE Congress on Evolutionary Computation (CEC), pp. 3012-3019, 2015.

[11] Tatsumi, T., Komine, T., Sato, H. and Takadama, K., "Handling different level of unstable reward environment through an estimation of reward distribution in XCS". In Evolutionary Computation (CEC), 2015 IEEE Congress on (pp. 2973-2980), 2015.

[12] Nakata, M., Lanzi, P.L. and Takadama, K., June. Simple compact genetic algorithm for XCS. In Evolutionary Computation (CEC), 2013 IEEE Congress on (pp. 1718-1723), 2013.

[13] Butz, M. V., Sastry, K., And Goldberg, D. E., "Strong, stable, and reliable fitness pressure in XCS due to tournament selection" GPEM, vol. 6, No. 1, pp. 53–77, 2005.

[14] Orriols-Puig, A., Sastry, K., Lanzi, P., Goldberg, D., and Bernad´o-Mansilla, E., "Modeling selection pressure in XCS for proportionate and tournament selection", In GECCO'07, vol. 2, pp. 1846–1853, ACM Press, 2007.

[15] Morales-Ortigosa, S., Orriols-Puig, A., And Bernado Mansilla, E., "New crossover operator for evolutionary rule discovery in XCS", in 8th international conference on hybrid intelligent systems. IEEE Computer Society, pp. 867–872, 2008.

[16] Hurst, J., Bull, L., "A self-adaptive XCS" In IWLCS'01: Revised Papers from the 4th International Workshop on Advances in Learning Classifier Systems, pp. 57–73, Springer-Verlag , London, UK, 2002.

[17] Wilson, S.W., "Knowledge growth in an artificial animal", Proceedings of the 1st International Conference on Genetic Algorithms and Their Application, pp. 16–23, 1985.

[18] Olivier, O., Wilson, S.W., "Learning Classifier Systems: A Survey", Universit´e Pierre et Marie Curie, 2000.

[19] Sigaud, O., Wilson, S., "Learning classifier systems: a survey", Soft Computing, vol. 11, No. 11, pp. 1065–1078, 2007.

[20] Holland, J., Reitman, J., "Cognitive systems based on adaptive agents, in Pattern-Directed Inference Systems", Waterman, D. A., Inand, F., Eds., Hayes-Roth, 1978.

[21] Booker, L., "Intelligent behavior as an adaptation to the task environment", Doctoral Dissertation, University of Michigan Ann Arbor, MI, USA, p. 342, 1982.

[22] Goldberg, D. E., "Genetic Algorithms in Search, Optimization, and Machine Learning", Addison Wesley, Reading, MA, 1989.

[23] Butz, M. V., Pelikan,M, Llor`a,X and Goldberg, D.E., "Extracted global structure makes local building block processing effective in XCS", in Proceedings of the Genetic and Evolutionary Computation Conference, pp. 655–662, ACM, New York, USA, 2005.

[24] Butz, M.V., Pelikan,M, Llor`a,X and Goldberg, D.E.,"Automated global structure extraction for effective local building block processing in XCS", Evolutionary Computation, vol. 14, no. 3, pp. 345–380, 2006.

[25] Butz, M. V., Wilson, S. W., "An algorithmic description of XCS", In Lanzi, P. L., Stolzmann, W. and Wilson, S. W., "Classifier Fitness Based on Accuracy. Evolutionary Computation", 1995, vol. 3, No. 2, pp. 149-175, 2001.

[26] Wyat, D., Larry, B., "A Memetic Learning Classifier System for Describing Continuous-Valued Problem Spaces", Recent Advances in Memetic Algorithms, pp. 355-395, 2005.

[27] Neri, F. and Moscato, P. eds., Handbook of memetic algorithms (Vol. 379). Heidelberg: Springer,2012.

[28] Butz, M, V., "XCS Java 1.0: An Implementation of the XCS classifier system in Java", Technical Report 2000027, Illinois Genetic Algorithms Laboratory, 2000.