A Memetic Learning Classifier System for Describing Continuous-Valued Problem Spaces

David Wyatt[1] and Larry Bull[2]

[1] University of the West of England, Frenchay Campus, Bristol BS16 1QY
david2.wyatt@uwe.ac.uk
[2] University of the West of England, Frenchay Campus, Bristol BS16 1QY
larry.bull@uwe.ac.uk

Summary. Learning Classifier Systems have previously been shown to have some application in deducing the characteristics of complex multi-modal test environments to a suitable level of accuracy. In this study, an accuracy-based Learning Classifier System, XCS, is used. The system has the capability of inducing a set of general rules from a sample of data points using a combination of Reinforcement Learning and a Genetic Algorithm. The investigation presented here builds on earlier work in this area by considering the application of a memetic approach during learning. The motivation for this investigation is identify if any increases in learning speed and classification performance can be made. The type of memetic learning used is based on Lamarckian Evolution but has several subtle differences from the standard approach. In particular, the Learning Classifier System is based on a Reinforcement Learning paradigm that has a dynamic effect on the fitness landscape. And, the form of lifetime learning used is based on a Widrow-Hoff delta rule update procedure in which changes to an individual's genotypic description are based upon some distance measure between the individual and a "focal rule"' (analogous to a local optima in a standard MA). In addition, no distinction is made between genotype and phenotype. Initial investigations focus on the effects on performance for three different learning rates and three different "focal rule" identification options for two different test environments - a two-dimensional and a decomposable six-dimensional test environment. Results show that improvements can be made over a non-memetic approach. The study also considers the use of a self-adaptive learning mechanism. Self-adaptation has been suggested as beneficial for Genetic Algorithms where the technique is usually used for adapting the mutation rate in a time-dependant and decentralised way. However, the investigation of a self-adaptive learning mechanism presented here focuses on the benefits of adjusting the Widrow-Hoff learning rate used within the memetic-learning component of the system. The mechanism was applied to both test environments. Results show that the mechanism can provide a more robust learning system both in terms of reduction in the number of system parameters and increased generalisation and solution convergence. Further detailed analysis of experimental results for the decomposable six-dimensional test function is also performed. This would otherwise be non-trivial for a non-decomposable six-dimensional function. The classification accuracy of several different versions of the system in-

cluding those systems with and without memetic or self-adaptive memetic learning are analysed region by region showing the effects of the new learning approach at a much greater level of detail. Analysis shows that the self-adaptive memetic version of the classifier system outperforms the non-adaptive and non-memetic versions in some of the regions.

1 Introduction

There have been several published studies demonstrating the capabilities of XCS [30] for data-mining through rule-induction. XCS is a Learning Classifier System (LCS) [11] that is capable of inducing a set of general rules from a sample of data points using a combination of Reinforcement Learning [14] and a Genetic Algorithm [10]. In [4], Bernado et al. describe an experimental comparison of XCS with seven other learning schemes, including C4.5, Naive Bayes and Support Vector Machines. Fifteen UCI repository data-sets [5] were used in that study each having a mixture of attribute types and differing numbers of classes and data-set sizes. The XCS system was shown to be highly competitive when compared with the other learning schemes. Wilson [32][33] has also demonstrated the capabilities of an interval based encoding when used to induce rules describing the Wisconsin Breast Cancer data-set. In fact, XCS was shown to improve on the best known performance for that data-set. That is, the XCS classifier system can be cast as an induction engine that is trained using a reinforcement learning approach, i.e., an external agent provides a reward for each classified data instance. Once the system has completed its training, new unseen data are presented and a measure of classification accuracy made.

Initial investigations in [8] show that the XCSR system [31] (an extension of the binary-input XCS to real-inputs) is able to identify high performance regions from a continuous multi-variable search space using a sample of training data points. Parmee [25] introduced the concept of the identification of high performance regions of complex preliminary design spaces rather than the identification of single optimal design solutions. A region of high performance is any contiguous set of points in a given design space which are considered to be exceptional solutions to a particular set of possibly conflicting design criteria. The solution provided by XCSR is a complete set of simple classification rules that define orthogonal regions of the solution space with attached classification labels. Investigations continued using a new *Simplified Learning Scheme* with the aim of improving XCSR performance with respect to learning speed and ability to respond to changes in the underlying test environment (such as class relabelling). When using the *Simplified Learning Scheme* newly created rules have their expected payoff value set to that of the first training instance they experience. This value remains constant. The new system was termed sXCSR and results showed that improvements can be made under the new learning scheme. The work presented clearly demonstrated the capability of XCSR to evolve real-valued pairs to describe interval

bounds for each variable in the multi-variable problem and thereby define a set of simple classification rules for the high performance regions.

The investigation was extended in [34] by applying XCSR and sXCSR to progressively more complex multi-modal test environments each with typical search space characteristics, convex/non-convex regions of high performance and complex interplay between variables. In particular, two test environments were used to investigate the effects of different degrees of feature sampling, parameter sensitivity, training set size and rule subsumption. Both test environments are constructed using a combination of functions allowing for the simple generation of training and test points. Each sample point can be represented by a vector of continuous values and a continuous performance measure which may be discretised as appropriate. Both test environments are also used in this study. Fixed size training data-sets were used in an effort to provide some consistency in experimentation with those design problems for which the cost of an on-line evaluation per sample point is high or for which data-sets are constructed from other off-line data sources.

The study is arranged as follows: Sect. 2 presents a basic overview of memetic algorithms, and in particular, how this relates to the approach presented here; Sect. 3 describes the XCSR system used throughout; Sect. 4 describes the experimental details for this study; Sect. 5 describes and presents results for a two-dimensional test environment; Sect. 6 describes and presents results for a six-dimensional test environment; Sect. 7 presents the results from using a self-adaptive approach to tuning the memetic learning rate for both the two and six-dimensional test environments; Sect. 8 shows a more detailed analysis of a selection of six-dimensional test environment results by decomposing the test data-set into eighteen distinct regions of high performance and finally, all findings are discussed in Sect. 9. Section 10 defines several related equations that can be used to define the two and six dimensional test environments used in this study.

2 Memetic Algorithms

In [23], Moscato termed a Memetic Algorithm (MA) as "a marriage between a population-based global search and the heuristic local search made by each of the individuals" and made it clear that the global search need not be constrained to a genetic representation. However, for the purposes of this investigation, a more constrained definition provided by Krasnogor and Smith [18] is used, that is, "MAs are extensions of Evolutionary Algorithms (EAs) that apply separate local search processes to refine individuals". It should also be made clear that the local search processes used to refine individuals may include constructive and exact methods as well as iterative improvement techniques. The key feature of the memetic approach to learning is that individuals are permitted to learn during their lifetime. This type of learning is applied using the phenotypic description of an individual rather than the

genotypic description. However, for the evolutionary process to take advantage of this lifetime learning, the effects of any improvements need to be felt in succeeding generations of the genetic search. There are two basic models of evolution that can be used for this purpose. These are the Baldwin Effect [2][22][24] and Lamarckian Evolution [29].

The Baldwin Effect was discovered independently by Baldwin [2], Morgan [22] and Osborn [24] in 1896 and has since become known as the Baldwin Effect due to Baldwin's dedicated research efforts. The Baldwin Effect allows an individual's fitness to be determined as a result of the local search processes but without any resulting changes in phenotypic description being reflected in the individual's genotypic description. According to Whitley et al. [29], this technique has the effect of changing, or smoothing out, the fitness landscape while retaining the advantages of the evolutionary process. Here, it is not characteristics acquired during an individual's lifetime that are inherited, but its ability to acquire those characteristics.

Larmarckian Evolution, introduced by Jean Baptiste Larmarck in 1809, is based on an assumption that characteristics acquired during an individual's lifetime are inherited, that is, any changes to the phenotypic description will be reflected in the individual's genotypic description. One of the drawbacks of using this technique for real-world problem solving is the requirement for an inverse mapping from phenotype and environment to genotype. There is no such requirement when using the Baldwin Effect.

Despite the issues raised above, the Larmarckian approach is used as the basis for memetic learning in this investigation. However, there are several subtle differences between the standard approach to Larmarckian Evolution and that employed here. Firstly, the underlying Evolutionary Algorithm used is based on a Reinforcement Learning paradigm that has a dynamic effect on the fitness landscape. Secondly, the form of lifetime learning used is based on a Widrow-Hoff update procedure in which changes to an individual's genotypic description are based upon some distance measure between the individual and a "focal rule". Here, the "focal rule" is analogous to a local optima in a standard MA. It should be noted that no distinction is made between genotype and phenotype in this study.

In particular, this involves identifying a "focal rule" together with any members of the current *Action Set* that qualify for update. In order to qualify for update, a rule must have been a member of a sufficient number of previous *Action Sets*, that is, the rule is expected to have been updated enough times to show its true potential. This threshold is termed as the update qualification threshold and is denoted by the symbol ξ. The "focal rule" is a single rule used in a Widrow-Hoff update procedure applied to all qualifying *Action Set* members. In this investigation, three different approaches are used to identify the "focal rule": *Best Fitness*, *Most Numerate* and *Accurate*. Each approach involves sorting the *Action Set* members in descending order with the first member in the sorted list being defined as the "focal rule". It is possible in each of these approaches, and likely in the case of *Accurate*, for several

members of a given *Action Set* to be competing for the title of "focal rule", in which case, one of them is selected at random. Once the "focal rule" and members qualifying for update have been identified, the following update rule is applied, $x_i^j = x_i^j + \eta\left[F_i - x_i^j\right], \forall i, j$, where represents gene i of qualifying member j, F_i represents gene i of the "focal rule" F, and η is a learning rate between 0 and 1 applied to the update. The new system is termed mXCSR. Initial investigations focused on the effects on performance for three different learning rates and three different "focal rule" identification options over those for the standard non-memetic approach.

3 XCSR

In [31], Wilson presents a version of XCS [30] for problems which can be defined by a vector of bounded continuous real-coded variables – XCSR. In that system, each rule in the classifier system population consists of the following parameters: $< condition >$: $< action >$: prediction (p) : prediction error (ε) : fitness (F) : experience (exp) : time-stamp (ts) : action set size (as) : numerosity (n). Given that XCSR is an accuracy-based classifier system, the three parameters p, ε and F are used to assess the accuracy of the rule's prediction in relation to its experiences over time, that is, how accurately the rule predicts the actual reward or payoff from its use for a given environment input. The other parameters, exp, ts, as and n, are used by the classifier system to maintain the internal dynamics of the system, such as balancing resources across environmental niches, genetic algorithm invocation and computational issues.

Figure 1 shows a schematic illustration of the architecture of a single-step version of the XCSR system with a particular emphasis on the data-mining capabilities of the system. The following description of the process actions of the XCSR system can be found in algorithmic form in [9]. A sample point is selected at random from the database and is presented to the system as an *input vector*. The system defines a subset of the *Population*, called the *Match Set*, from those rules whose $< condition >$ *matches* the *input vector*, where each rule predicts one of n actions ($n = 2$ for this study). If there are no matching rules, the system generates, or *covers*, a rule for each possible action using the *input vector* as a template. The *Prediction Array* is *calculated* as a sum of the fitness-weighted prediction of each rule in the *Match Set*, that is, π represents the sum of fitness-weighted prediction for all rules advocating action 1 and ϕ for those advocating action 2. The action with the highest sum represents the systems "best guess" at the classification for the given *input vector*. There are two *action selection* regimes, explore and exploit. Assuming the system exploits its knowledge, the system defines a subset of the *Match Set*, called the *Action Set*, from those rules advocating the *selected action*. The *predicted class* is compared with the *actual class* for the given *input vector* and a *reward* is received, 1000 for correct and 0 for incorrect. The system *reinforces*

those rules in the *Action Set* using the *reward*. If the average number of time-steps since the last invocation of the *Genetic Algorithm* component is greater than some pre-defined threshold, the *Genetic Algorithm* is permitted to act upon the members of the *Action Set*.

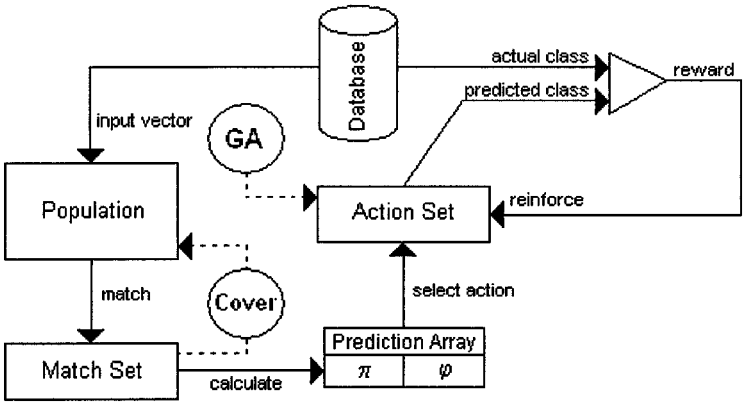

Fig. 1. The XCSR Classifier System Schematic Illustration for Single Step Data-mining Tasks

Another operator that acts upon the *Action Set* is the *subsumption* operator. One rule may subsume another if every interval predicate in the subsumee's $<condition>$ can be subsumed by the corresponding predicate in the subsumer, that is, for real-coded $<condition>$'s the subsumee's lower bound must be greater and its upper bound must be lesser than the corresponding subsuming predicate. In fact, XCSR implements two different forms of subsumption, *Action Set Subsumption* and *Genetic Algorithm Subsumption*. In the first form, a single rule is defined as the most general in a given *Action Set* and is permitted to subsume any other rule in the *Action Set* providing it is sufficiently experienced and accurate enough. In the second form of subsumption, a newly generated offspring rule may be subsumed if either of its parents are more general than it, sufficiently experienced and accurate enough.

In [31], Wilson defines a $<condition>$ as consisting of interval predicates of the form $\{\{c_1, s_1\}, \ldots, \{c_n, s_n\}\}$, where c is the interval's range "centre" and s is the "spread" from that centre – termed here as the *Centre-Spread* encoding. Each interval predicate's upper and lower bounds are calculated as follows : $[c_n - s_n, c_n + s_n]$. If an interval predicate goes outside the variable's defined bounds, it is truncated. In order for a rule to match the environmental stimulus, each input vector value must sit within the interval predicate defined for that variable.

In [32], Wilson describes another version of XCS which could also be used for such multi-variable problems in which a vector of integer-coded interval predicates is used in the form $\{[l_1, u_1], \ldots, [l_n, u_n]\}$, where l and u are the intervals' lower and upper bounds, respectively – termed here as the *Interval*

encoding. It is clear that a real-coded version of the integer bounded interval predicates would be trivial to implement. For both the *Centre-Spread* and *Interval* encoded versions, mutation is implemented via a random step (range $-0.1 \leq x \leq 0.1$) and cover produces rules centred on the input value with a "spread" of s_0.

It is important to note in the case of the *Interval* encoding, a potential problem may arise through the action of the mutation operator such that it is possible for a variable predicate's upper bound to become smaller than its lower bound. There are two ways to deal with this problem, termed here as *Ordered Interval* and *Unordered Interval* [26]. The first way uses a repair operator to enforce an ordering restriction on the predicates by swapping the offending values to ensure that all interval predicates in the $< condition >$ remain feasible, i.e., in the form $\{[l_1, u_1], \ldots, [l_n, u_n]\}$. The second way lifts the ordering restriction such that an interval $[l_n, u_n]$ is equivalent to $[u_n, l_n]$. The reader is referred to [26] for a discussion of the issues related to the differences between *Interval* encodings.

4 Experimental Details

The investigation presented in [34] compared the three different real-coded interval encodings described in Sect. 3 and showed that there was little or no difference between the encodings for the two and six-dimensional test environments used. Given that these same test environments are used in this study, the choice of encoding becomes an arbitrary one. In fact, the *Unordered Interval* encoding is used throughout.

The investigation also showed results for different sized training data-sets, that is, 500 and 2000 sample points for the two-dimensional environment and 6000 and 12000 sample points for the six-dimensional environment. In both cases, the larger training data-set led to better performance. For this reason, the 2000 and 12000 sample point data-sets are used in this study. Results presented in [34] for different population sizes show that a population of 8000 rules for the two-dimensional test environment and a population of 2000 rules for the six-dimensional test environment led to the best performance.

To clarify, all experiments in this study use a system based on the *Unordered Interval* encoding. Experiments using the two-dimensional environment run for a total of 200000 trials with a population size of $N = 8000$ while those using the six-dimensional test environment run for 250000 trials with a population size of $N = 2000$. XCSR's other parameters are defined as: $\beta = 0.2$, $\alpha = 0.1$, $\varepsilon_0 = 10$, $\nu = 5$, $\theta_{GA} = 12$, $\chi = 0.8$, $\mu = 0.04$, $\theta_{del} = 20$, $\delta = 0.1$, $p_I = 10$, $\varepsilon_I = 0$, $F_I = 0.01$, $\theta_{mna} = 2$, $\theta_{sub} = 20$, plus $m = \pm 10\%$ and $s_0 = 2\%$ for the two-dimensional environment while $m = \pm 10\%$ and $s_0 = 25\%$ for the six-dimensional environment.

An approach is required to enable comparisons of performance to be made between different parameter sets. One common way to define the performance

of a classification system is to use a confusion matrix [16] of size $L \times L$, where L is the number of different classifications. The matrix contains information about the actual and predicted classifications resulting from the classification task and provides a simple format to record and analyse system performance. Figure 2 gives an example of a 2×2 confusion matrix with the two possible

		Predicted Class	
		Low	High
Actual Class	Low	**a**	**b**
	High	**c**	**d**

Fig. 2. An Example 2×2 Confusion Matrix with *High* and *Low* Classifications

Several measures of classification accuracy based on confusion matrices were developed to overcome problems associated with analysis where the number of examples in each classification is significantly different. These include Lewis and Gale's F-measure [20], the geometric mean as defined by Kubat et al. in [19], using ROC graphs to examine classifier performance [27] and Kononenko and Bratko's information-based evaluation criterion [17].

Given that all the test data-sets used in this study have been manipulated such that the number of examples per classification are nearly equal, a simple accuracy measure will suffice for basic analysis. For a two-class classification problem, the accuracy measure is defined as the number of examples correctly classified as *High* plus the number correctly classified as *Low* divided by the total number of examples classified, that is, $(a + d)/(a + b + c + d)$ according to Fig. 2. Unclassified test examples are not included in this measure, that is, the denominator may not always equal the number of sample points in the test data-set. The percentage of *High* and *Low* points correctly classified are traditionally known as sensitivity and specificity, respectively. These terms frequently appear in the medical literature and are mainly used to describe the result of medical trials for disease prevention, but have come to be used in many non-medical classification tasks including information retrieval. A similar set of performance metrics were introduced for the EpiCS [12] system.

An important aspect of the experimental method identified in [34] was the class imbalance problem [13][28] which can be defined as a problem encountered by any inductive learning system in domains for which one class is under-represented and which assume a balanced class distribution in the

training data. For a two-class problem, the class defined by the smaller set of examples is referred to as the minority class while the other class is referred to as the majority class. Initial experiments for the six-dimensional test environment used in [34], and here, showed that without applying some form of rebalancing of the class distribution, the six-dimensional test problem could not be described to an acceptable level of accuracy. The solution used in [34] for the six-dimensional test environment was an approach suggested by Ling and Li [21] that makes use of both minority over-sampling and majority under-sampling. In particular, the minority class is re-sampled with replacement until some pre-defined multiple, n, of the original sample size is achieved. The majority class is re-sampled without replacement until a number of examples equal to those sampled from the minority class have been defined. The Ling and Li re-sampling solution is restricted to the six-dimensional test environment as was the case in [34].

For all experiments presented in this study, the new mXCSR system is trained using a single training data-set and tested using a different test data-set generated from a uniform random distribution. The test data-sets have been manipulated in such a way as to provide an equal number of test points per classification. In particular, n points are sampled from a uniform random distribution and evaluated according to the given environment. The sample points are sorted in descending order of performance and the top $2m$ points are used to define the test data-set, where m equals the total number of *High* points generated. All data-sets used have two defined classes, *High* and *Low*. The training data-sets are generated from a Halton Sequence Leaped (HSL) sequence [15], where the HSL is a quasi-random sequence that provides a set of real numbers whose degree of uniformity is high. By manipulating the test data-sets to include sample points from both classifications near to the classification decision boundaries, it is hoped that clear evidence of the classifier system's capability to evolve rules that define those boundaries will be gathered.

The results for each parameter setting of the mXCSR system are averaged over ten independent runs and presented together with a standard deviation for that sample. Any conclusions made are based on the application of Mann-Whitney Rank Sum Test which makes no assumptions about the distribution of population from which the runs where sampled.

5 A Two-dimensional Test Environment

The two-dimensional test environment used in this paper is the multi-modal modified Himmelblau function [3]. The equation for the modified Himmelblau function, which is used to evaluate each sample point, is given in Sect. 10.1. There are four optima of approximately equal magnitude. This function is used to define a two-class classification task to investigate the effects of using a memetic approach to learning within the XCSR classifier system,

that is, mXCSR. In particular, an exact threshold value of $\psi = 184$, where $\psi \in [-1986, 200]$, is used to define *High/Low* class decision boundaries. Figure 3 shows a contour plot of the function, clearly indicating the four regions of high performance as defined by the threshold value given above.

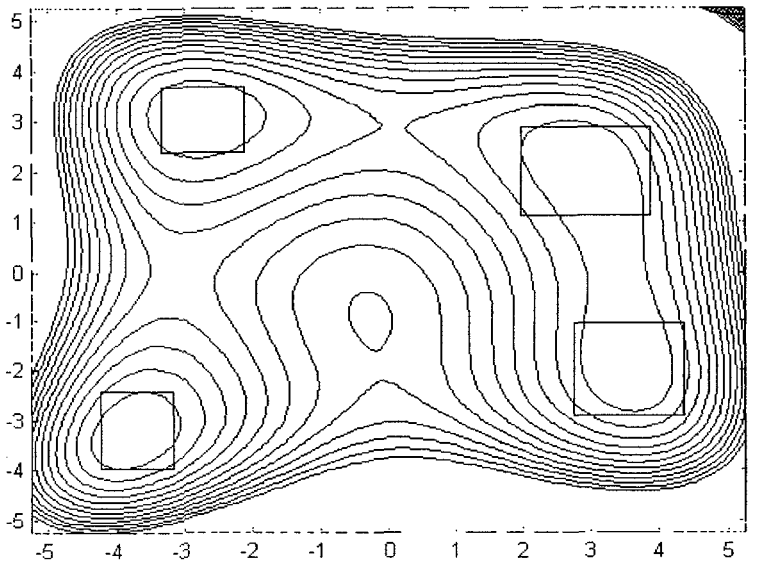

Fig. 3. The Modified Himmelblau Function Contour with Four High Performance Regions

The new mXCSR system was trained using a single training data-set, Fig. 4, generated from a HSL sequence with 2000 sample points and was tested using a different data-set generated from a uniform random distribution. This test data-set has 2116 sample points of which 1073 points are defined as *High* – shown as faint dots in Fig. 5.

Table 1 shows a single performance measure (ten run average with standard deviation) for each parameter combination using the Uniform Random test data-set. The parameter combinations used include running the new mXCSR system both with and without *Action Set Subsumption* and *Simplified Learning Scheme* using three different approaches to "focal rule" identification together with three different learning rates for each approach.

It is clear from Table 1 that the system performed well on the two-dimensional Himmelblau test problem in terms of correct classification of unseen data, between 72.8% and 90.1% depending on "focal rule" identification approach, learning rate, subsumption type and whether or not the *Simplified Learning Scheme* was used. The performance gain for mXCSR when *Action Set Subsumption* is turned off is remarkably clear in Table 1. In fact, the difference between mXCSR with and without *Action Set Subsumption* is statistically significant ($> 99.9\%$) for all parameter settings shown. Although, a

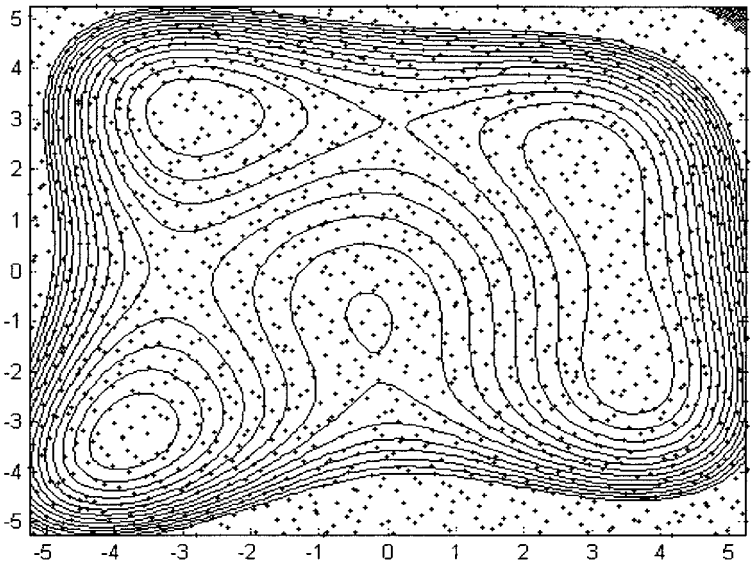

Fig. 4. Training Data-set with 2000 HSL-generated Sample Points

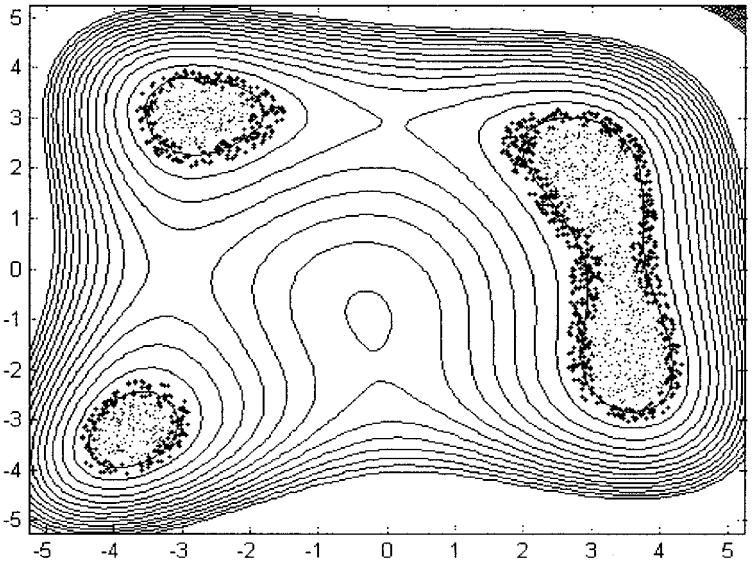

Fig. 5. Test Data-set with 2116 Sample Points Generated from a Uniform Random Distribution

Table 1. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Two-dimensional Test Environment

	With *AS-Sub*		Without *AS-Sub*	
	mXCSR	smXCSR	mXCSR	smXCSR
No Memetic	$74.2_{(6.8)}$	$67.8_{(4.0)}$	$90.9_{(0.6)}$	$90.3_{(0.8)}$
Best Fitness$_{(\eta=0.1)}$	$80.6_{(4.7)}$	$75.7_{(6.3)}$	$89.4_{(1.0)}$	$89.0_{(1.1)}$
Best Fitness$_{(\eta=0.2)}$	$83.0_{(2.6)}$	$77.4_{(3.2)}$	$89.5_{(1.0)}$	$88.7_{(1.4)}$
Best Fitness$_{(\eta=0.4)}$	$80.9_{(4.5)}$	$75.4_{(3.4)}$	$89.5_{(1.2)}$	$88.5_{(1.9)}$
Most Numerate$_{(\eta=0.1)}$	$78.0_{(7.6)}$	$72.8_{(7.4)}$	$89.7_{(1.5)}$	$89.5_{(1.2)}$
Most Numerate$_{(\eta=0.2)}$	$81.3_{(1.8)}$	$78.5_{(4.4)}$	$89.3_{(0.7)}$	$88.6_{(1.2)}$
Most Numerate$_{(\eta=0.4)}$	$75.5_{(4.4)}$	$75.6_{(2.2)}$	$89.4_{(0.9)}$	$89.4_{(0.8)}$
Accurate$_{(\eta=0.1)}$	$85.4_{(1.8)}$	$82.5_{(3.2)}$	$90.1_{(1.4)}$	$88.9_{(1.8)}$
Accurate$_{(\eta=0.2)}$	$83.0_{(3.2)}$	$77.7_{(4.5)}$	$89.1_{(0.8)}$	$88.8_{(1.4)}$
Accurate$_{(\eta=0.4)}$	$82.4_{(4.5)}$	$77.0_{(4.3)}$	$89.6_{(1.2)}$	$88.6_{(1.1)}$

difference has been identified in previous work [34], it is clear from the results shown here that the difference is significantly smaller, that is, the memetic learning approach has help to mitigate some of the problems associated with using *Action Set Subsumption* in the two-dimensional test domains. Figures 6 and 7 show the performance gain for the mXCSR system using the *Accurate*$_{(\eta=0.1)}$ "focal rule" identification approach when *Action Set Subsumption* is turned off.

Regarding different "focal rule" identification approaches, Table 1 shows a clear difference in performance between *Accurate*$_{(\eta=0.1)}$ and the other two approaches, *Most Numerate*$_{(\eta=0.1)}$ and *Best Fitness*$_{(\eta=0.1)}$, when *Action Set Subsumption* is turned on. This effect is a statistically significant ($> 99\%$) improvement in performance. In fact, a learning rate of 0.1 can also be shown to be statistically significant ($> 95\%$) when compared with the other two settings for the Accurate "focal rule" identification approach.

Figures 6 and 8 show a comparison of performance based on the *Simplified Learning Scheme*, that is, Fig. 6 does not use it and Fig. 8 does. Both systems are based on the *Accurate*$_{(\eta=0.1)}$ approach with *Action Set Subsumption* turned on. Results suggest that there may be a slight degradation in performance when the *Simplified Learning Scheme* is used. In fact, this result is statistically significant ($> 97.5\%$). Figures 9 and 10 show the learning speed and system error for the same two systems. These figures show that despite final performance being significantly different, there is no increase in learning speed as demonstrated in [8].

Overall, results suggest that when *Action Set Subsumption* is turned on, the memetic learning helps to improve system performance significantly above

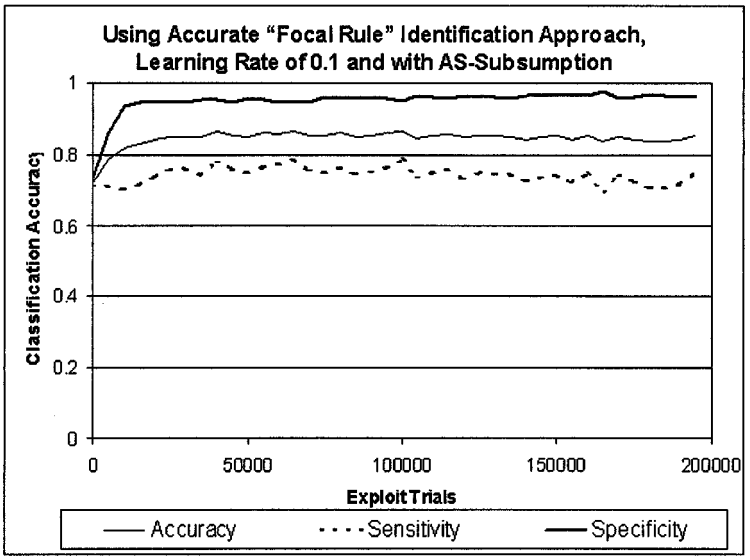

Fig. 6. Classification Accuracy for mXCSR with *Action Set Subsumption* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

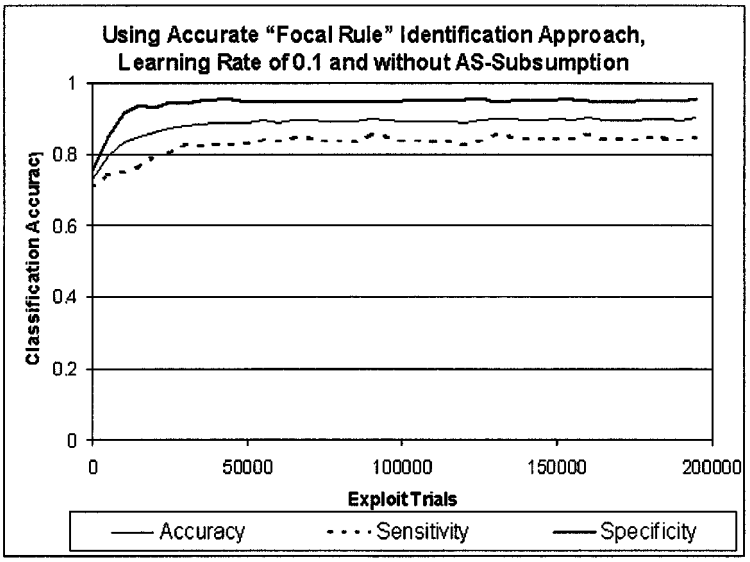

Fig. 7. Classification Accuracy for mXCSR without *Action Set Subsumption* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

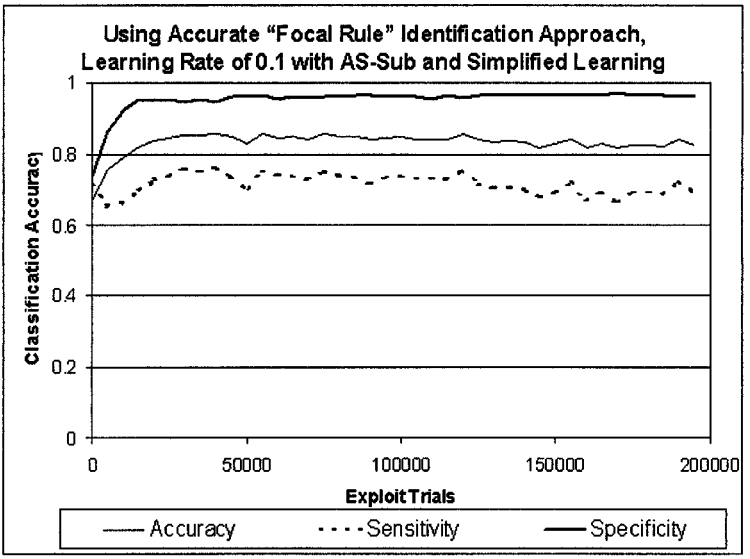

Fig. 8. Classification Accuracy for mXCSR with *Action Set Subsumption* and *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

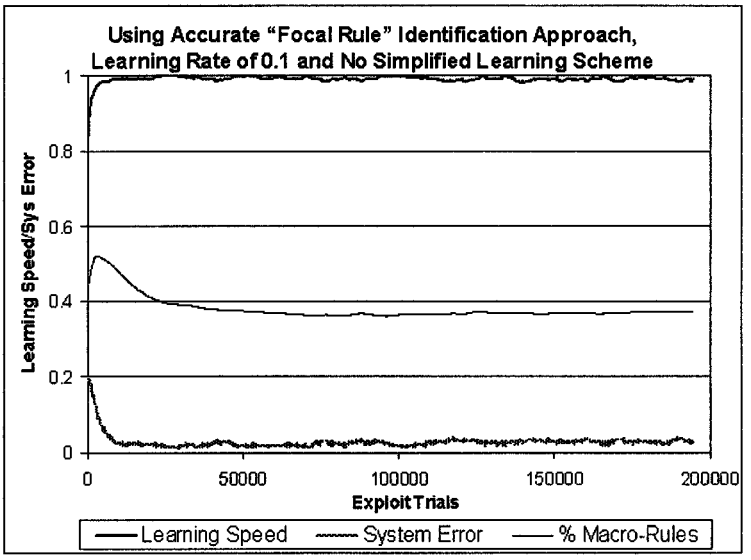

Fig. 9. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

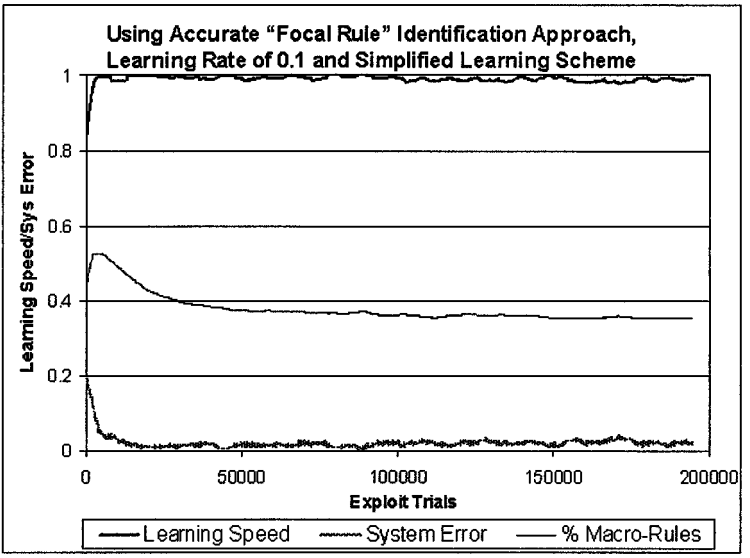

Fig. 10. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and with *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

that of a non-memetic system except for the $Most\ Numerate_{(\eta=0.1)}$ approach. In fact, this improvement is statistically significant at a level of $> 95\%$. However, it also clear that the memetic learning degrades performance for those systems that do not use *Action Set Subsumption*. Although this degradation is very small (around 2%), it is statistically significant ($> 95\%$).

6 A Six-dimensional Test Environment

The six-dimensional test environment used in this paper is a multi-modal function developed by Bonham and Parmee, [6] and [7], and is described in Sect. 10.2. It is defined by the additive effect of three different two-dimensional planes, as shown in contour plot form in Figs. 11–13 . Each plane has an associated "local" fitness value and the "global" fitness value of the six-dimensional function is defined by adding each of these "local" fitness values together, that is, $fitness_{global} = fitness_{plane1} + fitness_{plane2} + fitness_{plane3}$. Each sample point is defined by a six-dimensional vector of the form $\{a, b, c, d, e, f\}$, where $a \ldots f \in [0, 1]$.

In this two-class classification problem, a sample point is classified as either *High* or *Low*. It is classified as *High* only when each "local" fitness value is greater than the exact threshold value $\psi = 0.35$, where $\psi \in [0, 0.5]$, and the "global" fitness value is greater than exact threshold value $\psi_G = 1.20$, where

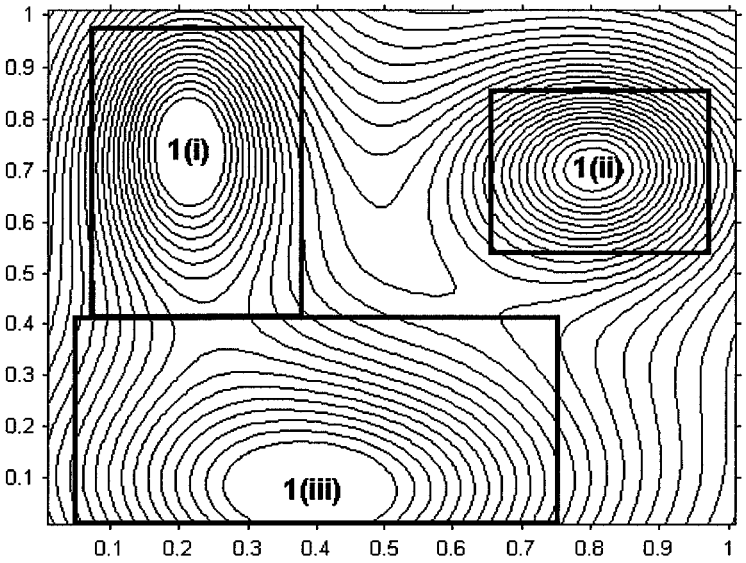

Fig. 11. Plane 1 of Bonham and Parmee's Six-dimensional Function showing Three Local Regions of High Performance

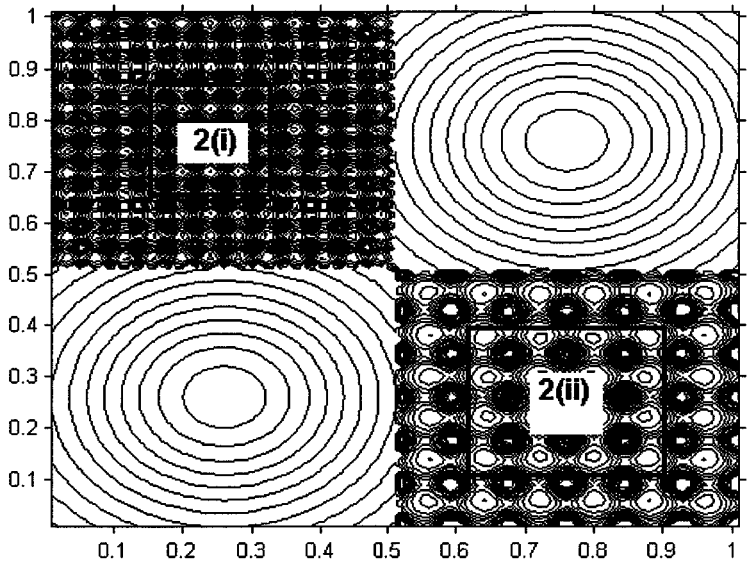

Fig. 12. Plane 2 of Bonham and Parmee's Six-dimensional Function showing Two Local Regions of High Performance

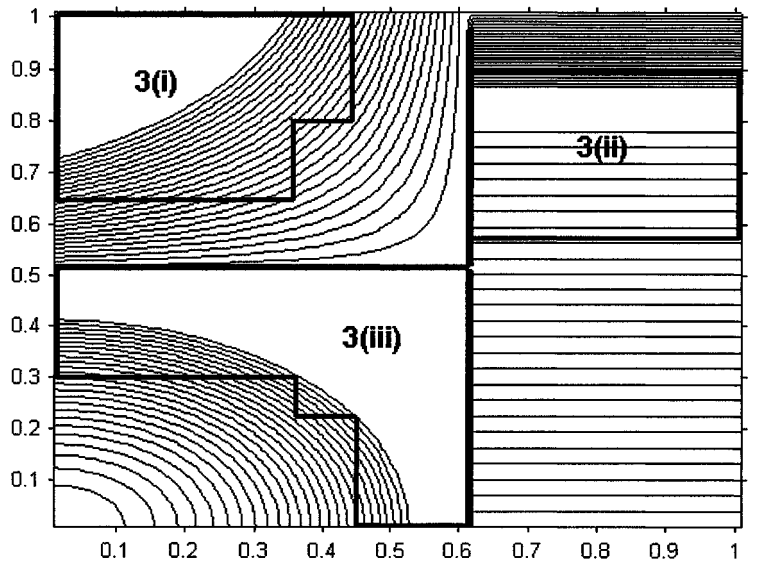

Fig. 13. Plane 3 of Bonham and Parmee's Six-dimensional Function showing Three Local Regions of High Performance

$\psi_G \in [0, 1.5]$, otherwise the point is classified as *Low*. By combining local regions of high performance, as shown in Table 2, an environment of eighteen unique regions of globally high performance are defined, that is, three local high performance regions in Plane 1, two in Plane 2 and three in Plane 3. In fact, Sect. 8 presents an analysis of a selection of results for this environment emphasising its decomposable nature.

The XCSR system was trained using a single 12000 sample point training data-set (not shown) generated from a HSL sequence which is re-balanced using the approach suggested by Ling and Li [21] with a predefined multiple of $n = 32$. The new data-set consists of 11968 sample points, that is, within 3% of the original size. The system was tested using a data-set generated from a uniform random distribution with 1693 sample points of which 813 points are defined as *High*. The test data-set is manipulated to include sample points from both classifications near to the classification decision boundaries as was the case in the two-dimensional test environment.

In general, Table 3 shows no significant difference in performance between "focal rule" identification approaches except where the standard deviation figure is high. However, there is a statistically significant difference (> 97.5%) between $Accurate_{(\eta=0.2)}$ and the other approaches as well as between $Accurate_{(\eta=0.4)}$ and *Most Numerate*. The memetic learning appears to provide little or no advantage in performance over the non-memetic system except for a slight degradation when *Action Set Subsumption* is turned off. This degradation is statistically significant (> 95%). Figures 14 and 15 show a comparison

Table 2. Eighteen Unique Regions of Globally High Performance (HPR_{region}) based on the Six-dimensional Test Function

	Plane 1	Plane 2	Plane 3
HPR_1	(i)	(i)	(i)
HPR_2	(i)	(i)	(ii)
HPR_3	(i)	(i)	(iii)
HPR_4	(i)	(ii)	(i)
HPR_5	(i)	(ii)	(ii)
HPR_6	(i)	(ii)	(iii)
HPR_7	(ii)	(i)	(i)
HPR_8	(ii)	(i)	(ii)
HPR_9	(ii)	(i)	(iii)
HPR_{10}	(ii)	(ii)	(i)
HPR_{11}	(ii)	(ii)	(ii)
HPR_{12}	(ii)	(ii)	(iii)
HPR_{13}	(iii)	(i)	(i)
HPR_{14}	(iii)	(i)	(ii)
HPR_{15}	(iii)	(i)	(iii)
HPR_{16}	(iii)	(ii)	(i)
HPR_{17}	(iii)	(ii)	(ii)
HPR_{18}	(iii)	(ii)	(iii)

of performance between a memetic learning and non-memetic system where both system have *Action Set Subsumption* turned off. It is clear from the figure that there is little difference between these systems for the given parameters settings.

Results also show that there is no clear difference in performance between using and not using the *Simplified Learning Scheme*. However, there seems to be the potential for further increases in learning speed when using the *Simplified Learning Scheme* as shown in Figs. 16 and 17, where *Action Set Subsumption* was also used. It is clear from the figure that the *Simplified Learning Scheme* does provide the same level of performance some 25000–30000 exploit trials quicker than the standard scheme. Given that the speed-up was not apparent in the simpler two-dimensional test environment, this provides some strength to an argument put forth in [8] in which it was hypothesized that the *Simplified Learning Scheme* may have a greater effect on performance as the complexity of the test environment increases.

In order to qualify for update, a rule must have been a member of a sufficient number of previous *Action Sets*, that is, the rule is expected to have been updated enough times to show its true potential. Given the similarities

Table 3. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Six-dimensional Test Environment

	With *AS-Sub*		Without *AS-Sub*	
	mXCSR	smXCSR	mXCSR	smXCSR
No Memetic	$71.3_{(20.9)}$	$80.6_{(1.2)}$	$82.2_{(0.8)}$	$82.4_{(0.6)}$
Best Fitness$_{(\eta=0.1)}$	$77.9_{(9.8)}$	$81.1_{(0.7)}$	$69.3_{(23.9)}$	$80.2_{(1.5)}$
Best Fitness$_{(\eta=0.2)}$	$81.3_{(0.9)}$	$81.0_{(1.1)}$	$81.0_{(1.3)}$	$80.0_{(1.7)}$
Best Fitness$_{(\eta=0.4)}$	$71.3_{(21.0)}$	$81.0_{(0.9)}$	$80.8_{(1.6)}$	$81.4_{(1.2)}$
Most Numerate$_{(\eta=0.1)}$	$80.8_{(1.2)}$	$80.5_{(1.1)}$	$80.4_{(1.5)}$	$79.9_{(1.1)}$
Most Numerate$_{(\eta=0.2)}$	$81.6_{(1.1)}$	$80.9_{(0.8)}$	$75.7_{(15.6)}$	$80.4_{(1.1)}$
Most Numerate$_{(\eta=0.4)}$	$81.7_{(0.9)}$	$81.8_{(1.0)}$	$71.2_{(21.4)}$	$80.6_{(1.3)}$
Accurate$_{(\eta=0.1)}$	$80.6_{(1.2)}$	$79.8_{(1.6)}$	$80.8_{(0.9)}$	$78.8_{(0.7)}$
Accurate$_{(\eta=0.2)}$	$73.9_{(18.1)}$	$78.9_{(1.8)}$	$77.3_{(7.0)}$	$78.8_{(2.2)}$
Accurate$_{(\eta=0.4)}$	$80.0_{(1.5)}$	$79.9_{(1.4)}$	$79.6_{(1.2)}$	$79.6_{(1.3)}$

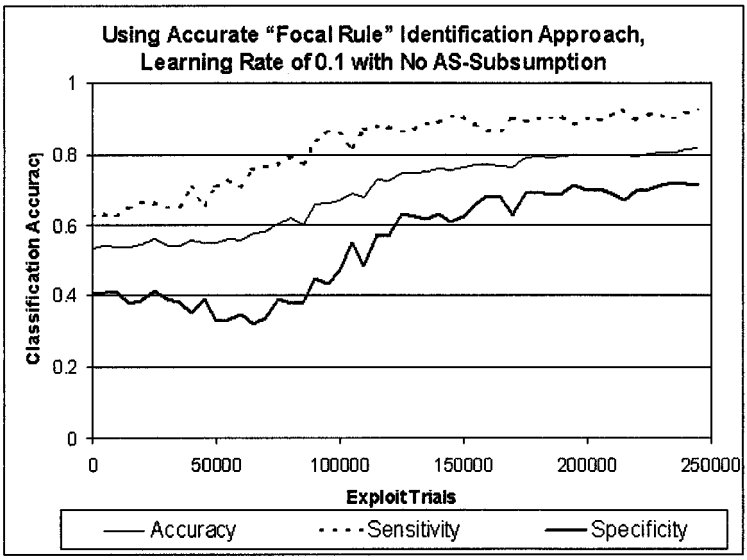

Fig. 14. Classification Accuracy for mXCSR without *Action Set Subsumption* using the *Accurate*$_{(\eta=0.1)}$ "focal rule" identification approach

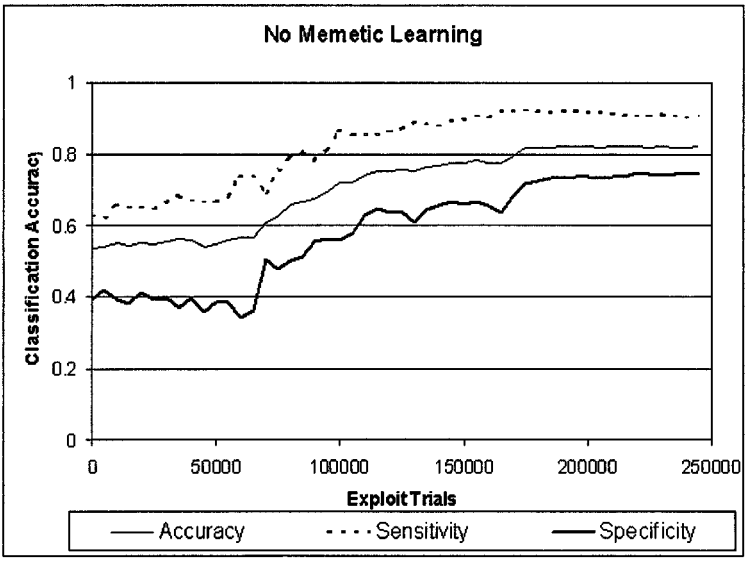

Fig. 15. Classification Accuracy for XCSR without *Action Set Subsumption*

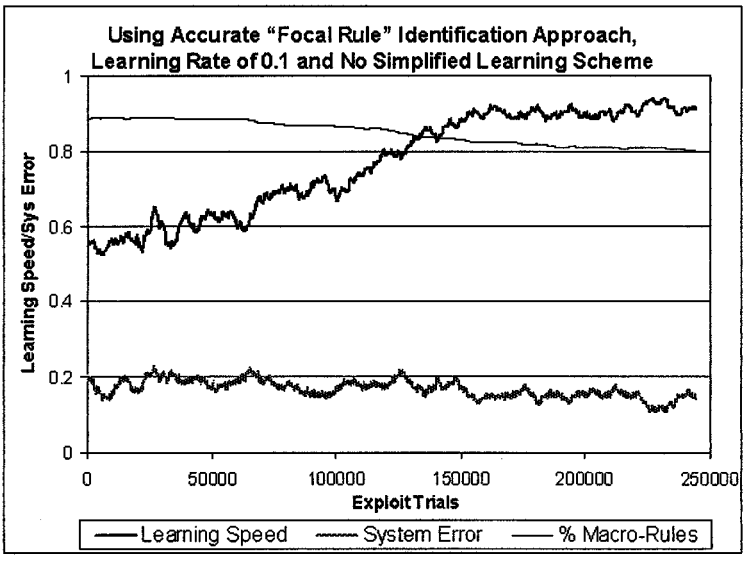

Fig. 16. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

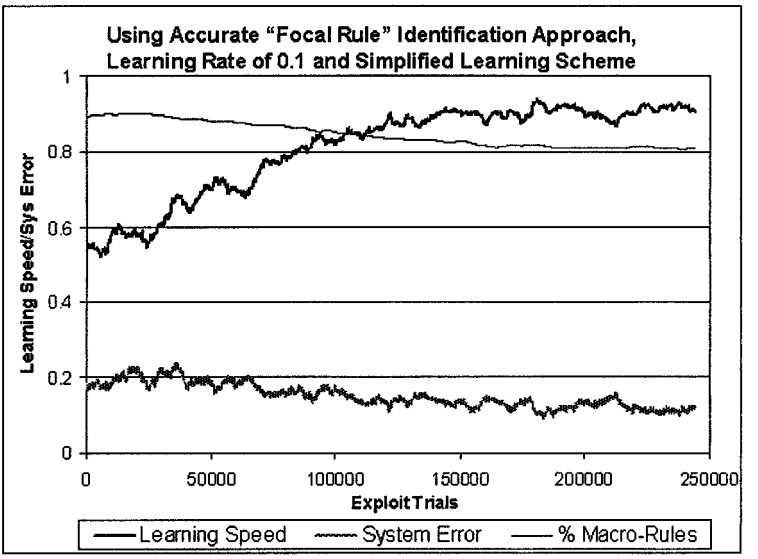

Fig. 17. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and with *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

in nature between the subsumption operator and the memetic learning suggested in this study, the update qualification value, ξ, has been fixed at 20 to this point. However, there was some evidence in [8] and [34] that permitting initially weaker rules enough time to show their true potential by reducing the early domination of more numerate rules in a given *Action Set* can lead to improvements. Based on this insight, another set of experiments was performed using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach where the threshold for update qualification varies between 20 and 200. The choice of identification approach was somewhat arbitrary given the results in Table 3 above.

It is clear from the result of these experiments, shown in Table 4, that a small improvement can be made in performance when $\xi = 50$ or $\xi = 100$, except for any highly variant results. In fact, this effect is statistically significant at the level of $> 95\%$ for differences between $\xi = 20$ and $\xi = 50$. There is also significant difference ($> 95\%$) in performance between those systems using the *Simplified Learning Scheme* with and without *Action Set Subsumption*. It should be noted that those settings of ξ greater than 100 lead to a performance that is not statistically different from the original settings of $\xi = 20$.

On closer examination, the differences in performance for those systems using the memetic learning but with *Action Set Subsumption* turned off, as shown in Table 3, are mitigated by the alteration of the threshold ξ, as shown

Table 4. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Six-dimensional Test Environment

	With *AS-Sub*		Without *AS-Sub*	
	mXCSR	smXCSR	mXCSR	smXCSR
No Memetic	$71.3_{(20.9)}$	$80.6_{(1.2)}$	$82.2_{(0.8)}$	$82.4_{(0.6)}$
$Accurate_{(\eta=0.1,\xi=20)}$	$80.6_{(1.2)}$	$79.8_{(1.6)}$	$80.8_{(0.9)}$	$78.8_{(0.7)}$
$Accurate_{(\eta=0.1,\xi=50)}$	$76.8_{(18.0)}$	$81.7_{(1.2)}$	$81.4_{(1.8)}$	$82.7_{(0.7)}$
$Accurate_{(\eta=0.1,\xi=100)}$	$81.4_{(1.4)}$	$80.2_{(1.8)}$	$81.8_{(1.2)}$	$82.1_{(0.9)}$
$Accurate_{(\eta=0.1,\xi=200)}$	$80.7_{(1.4)}$	$80.0_{(1.2)}$	$75.5_{(18.9)}$	$82.4_{(1.0)}$

in Table 4. That is to say that when the qualification threshold is between 50 and 100 for systems with *Action Set Subsumption* turned off, performance is comparable with a standard non-memetic system. Figures 18–21 illustrate this effect and also show that the number of rules required to achieve this level of performance is some 20% larger in the case of the memetic learning version of the system. This is likely to be one side effect of performing a Widrow-Hoff update on the qualifying members of a given *Action Set*, that is, there may be several rules in the rule-base with almost identical $<condition>$ parts but which cannot be subsumed in the normal manner.

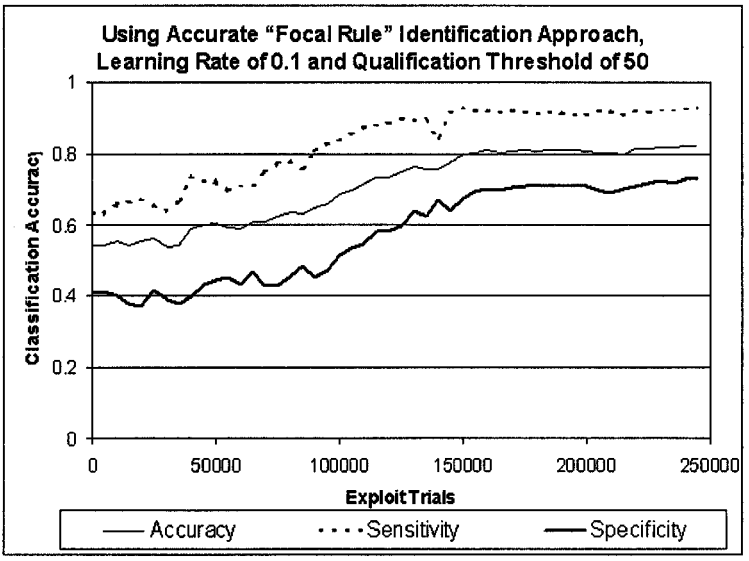

Fig. 18. Classification Accuracy for mXCSR without *Action Set Subsumption* using the $Accurate_{(\eta=0.1,\xi=50)}$ "focal rule" identification approach

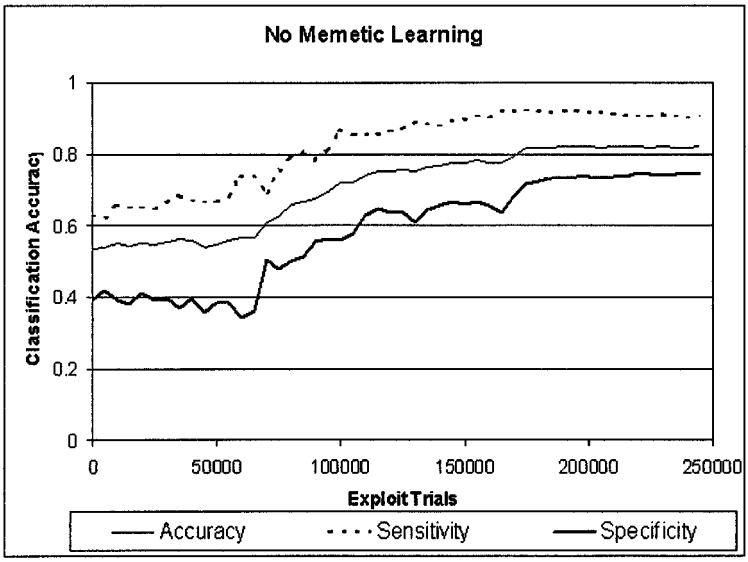

Fig. 19. Classification Accuracy for XCSR without *Action Set Subsumption*

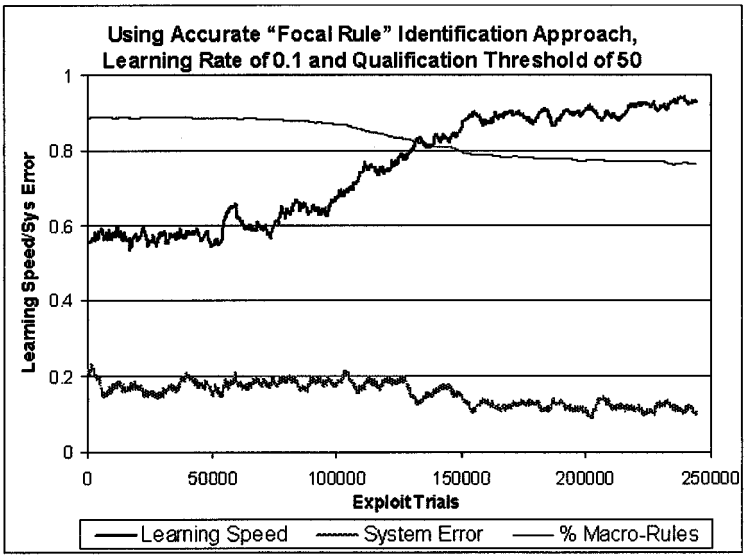

Fig. 20. Learning Speed and System Error for mXCSR without *Action Set Subsumption* using the $Accurate_{(\eta=0.1,\xi=50)}$ "focal rule" identification approach

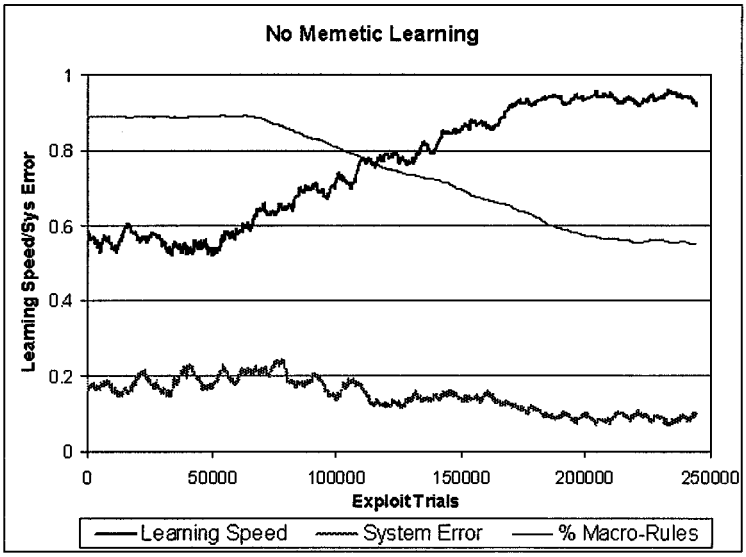

Fig. 21. Learning Speed and System Error for XCSR without *Action Set Subsumption*

7 Self-Adaptive Memetic Learning

Self-adaptation was suggested as beneficial for Genetic Algorithms in [1] where the author investigated the use of a technique for adapting the GA's mutation rate in a time-dependant and decentralised way. One advantage of using this technique is that the external global rate becomes an internal time-dependant rate, that is, there is a reduction in the number of parameters that require careful, possibly problem-dependant, setting by the user. Given this study, the investigation of a self-adaptive learning mechanism presented here focuses on the benefits of using this technique to adjust the Widrow-Hoff learning rate used within the memetic learning component of the new system. It is hoped that providing the classifier system with this capability will allow positive selective pressure toward a learning rate that is in proportion to the degree of solution convergence aiding both learning speed and solution quality in the process.

The self-adaptation is implemented by adding an extra real-coded gene to each rule in the population of the classifier system representing that particular rule's Widrow-Hoff learning rate. The approach used is similar to that presented in [1] except that the gene is affected by the mutation operator in the same way as the other genes, plus it is not included in the Widrow-Hoff update. Learning rates are initially assigned a random value in the range $[0.0, 1.0]$.

This section has been divided into two subsections, that is, results for the two-dimensional and six-dimensional test environments, respectively.

7.1 Two-dimensional Test Environment

The results shown in Sect. 5 suggest that, under certain circumstances, the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach provides the best level of performance and as such it has been used for the following set of experiments. In particular, four self-adaptive versions of the system (with and without *Action Set Subsumption* and *Simplified Learning Scheme*) are compared with the same non-adaptive versions (η fixed at 0.1).

Table 5. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Two-dimensional Test Environment

	With *AS-Sub*		Without *AS-Sub*	
	mXCSR	smXCSR	mXCSR	smXCSR
$Accurate_{(\eta=0.1,\xi=20)}$	$85.4_{(1.8)}$	$82.5_{(3.2)}$	$90.1_{(1.4)}$	$88.9_{(1.8)}$
$Accurate_{(\eta=Adaptive,\xi=20)}$	$68.6_{(4.5)}$	$67.2_{(4.1)}$	$90.8_{(0.7)}$	$90.7_{(0.7)}$

Table 5 shows a clear degradation in performance, of around 16%, for the self-adaptive versions when *Action Set Subsumption* is turned on. In fact, this difference is statistically significant at the level of greater than 99.9%. However, the table also shows that when *Action Set Subsumption* is turned off there is an improvement of between 0.7% and 2.8% for the self-adaptive versions of the system. This difference is statistically significant at the level of greater than 95%. Figures 22 and 23 shows a comparison of performance between self-adaptive and non-adaptive versions of the system where *Action Set Subsumption* is turned off and the *Simplified Learning Scheme* is not used. This comparison provides some evidence of the degree of convergence shown by both versions of the system and, in particular, how that level is greater in the self-adaptive version. Another apparent advantage of using the self-adaptive technique is an increase in generalisation as indicated by fall in % Macro-Rules during the experiment as seen in Figs. 24 and 25.

Finally, Fig. 26 shows the average learning rate for all four self-adaptive versions of the system, that is, with and without both *Action Set Subsumption* and *Simplified Learning Scheme.* This figure clearly shows that average learning rates are very noisy when *Action Set Subsumption* is turned on, but that the average learning rates for the other two systems have much in common with both the falling % Macro-Rules and rising degree of convergence, that is, positive selective pressure toward lower learning rates for systems without *Action Set Subsumption* exists.

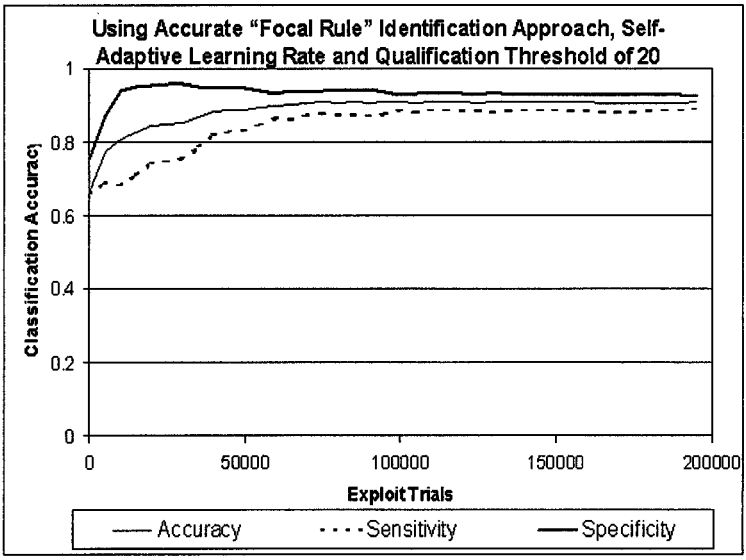

Fig. 22. Classification Accuracy for mXCSR without *Action Set Subsumption* using the *Accurate* "focal rule" identification approach with Self-Adaptive Memetic Learning

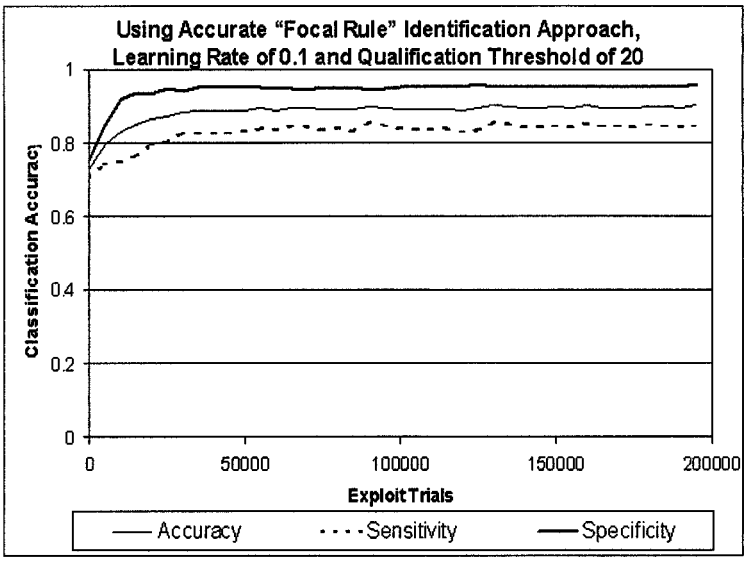

Fig. 23. Classification Accuracy for mXCSR without *Action Set Subsumption* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

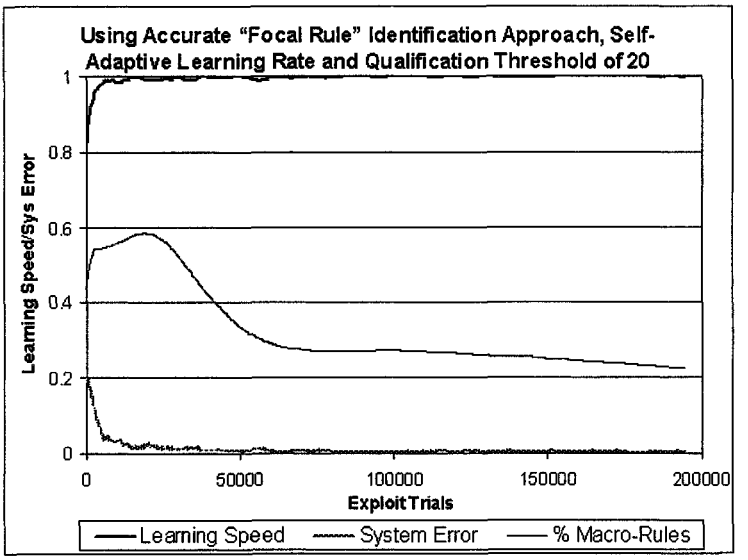

Fig. 24. Learning Speed and System Error for mXCSR without *Action Set Subsumption* using the *Accurate* "focal rule" identification approach with Self-Adaptive Memetic Learning

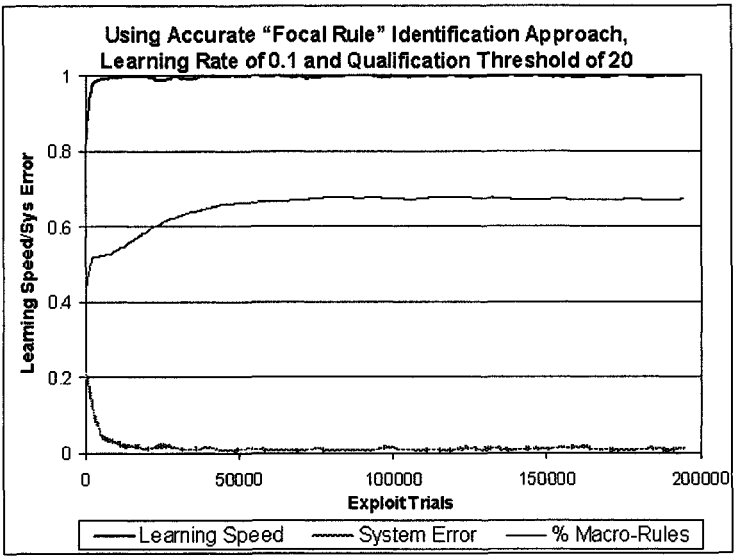

Fig. 25. Learning Speed and System Error for mXCSR without *Action Set Subsumption* using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach

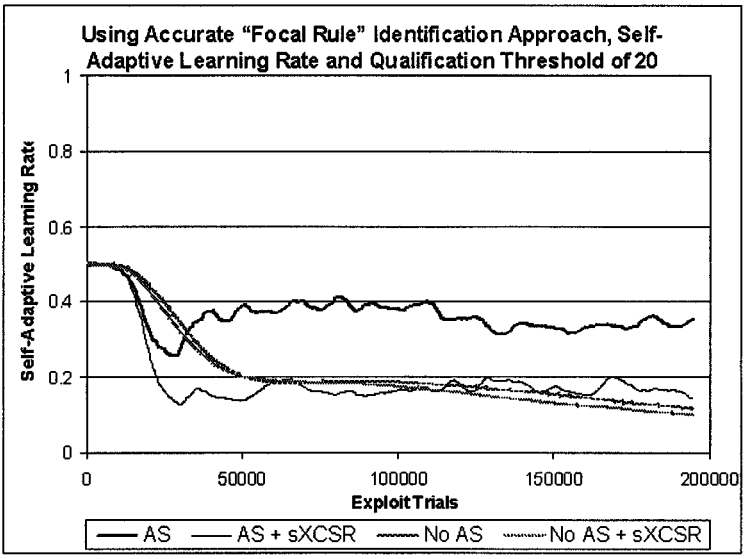

Fig. 26. Self-Adaptive Learning Rates for mXCSR with and without *Action Set Subsumption* and *Simplified Learning Scheme* using the *Accurate* "focal rule" identification approach

7.2 Six-dimensional Test Environment

Although results from using the $Accurate_{(\eta=0.1)}$ "focal rule" identification approach in the six-dimensional test environment were less clear than was the case in the two-dimensional environment, this identification approach is used throughout this subsection. It is hoped that using this approach together with the use of two different update qualification values, $\xi = 20$ and $\xi = 50$, will provide a consistent picture of the effectiveness of the self-adaptive approach. In this subsection, eight self-adaptive versions of the system (with and without *Action Set Subsumption* and *Simplified Learning Scheme* for two different update qualification values) are compared with the same non-adaptive versions.

Table 6 presents results for both update qualification values. It is clear from the results that improvements in performance have been made for all but two of the self-adaptive versions of the system. In fact, the improvements shown for three of the four systems using $\xi = 20$ are statistically significant at the level of greater than 95% and, in particular, at a level greater than 97.5% for the two systems using the *Simplified Learning Scheme*. Although the other four systems using $\xi = 50$ do not show significant improvements, they do provide evidence of the robustness of the self-adaptive technique, that is, there is no significant degradation in performance even when changes to other sensitive parameters (such as ξ) are made.

Table 6. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Six-dimensional Test Environment

	With *AS-Sub*		Without *AS-Sub*	
	mXCSR	smXCSR	mXCSR	smXCSR
$Accurate_{(\eta=0.1,\xi=20)}$	$80.6_{(1.2)}$	$79.8_{(1.6)}$	$80.8_{(0.9)}$	$78.8_{(0.7)}$
$Accurate_{(\eta=Adaptive,\xi=20)}$	$81.8_{(1.1)}$	$81.2_{(1.0)}$	$80.2_{(5.5)}$	$82.5_{(0.7)}$
$Accurate_{(\eta=0.1,\xi=50)}$	$76.8_{(18.0)}$	$81.7_{(1.2)}$	$81.4_{(1.8)}$	$82.7_{(0.7)}$
$Accurate_{(\eta=Adaptive,\xi=50)}$	$81.8_{(1.0)}$	$82.2_{(1.0)}$	$82.1_{(0.9)}$	$82.1_{(1.5)}$

Figures 27 and 28 show a comparison of performance between self-adaptive and non-adaptive versions of the system ($\xi = 20$) where *Action Set Subsumption* is turned on and the *Simplified Learning Scheme* is not used. The increase in generalisation as well as in the degree of convergence seen in the two-dimensional test environment is also evident here – although the magnitudes of both are reduced. It is also clear from Figs. 29 and 30 that there is some evidence of a speed-up in learning when using the self-adaptive version of the system, where the adaptive system reaches an accuracy of 0.8 at around 20000 exploit trials earlier than the non-adaptive version of the system. Results for the self-adaptive version of the system using an update qualification value of 50 (not shown) do not indicate any learning speed-up nor do they show an increase in generalisation, although there is evidence of a greater degree of convergence during the last 20000–30000 exploit trials of the experimental run.

Figure 31 shows the average learning rate for four self-adaptive ($\xi = 20$) versions of the system, that is, with and without both *Action Set Subsumption* and *Simplified Learning Scheme*. This figure clearly shows a positive selective pressure for lower average learning rates after an initial period of around 50000 exploit trials. The initial period of static average learning rate is linked to the effects of the covering operator and the way it initialises the learning rates of newly generated rules uniformly in the interval $[0.0, 1.0]$ leading to an average learning rate of 0.5.

Figure 32 shows the average learning rate for four self-adaptive ($\xi = 50$) versions of the system. This figure shows that there is little positive selective pressure for lower average learning rates. The reason for this lack of selective pressure may be a result of the increasing the delay before the memetic learning is applied thereby providing evolution with less opportunity to test a rule's learning rate. A simple experiment was performed to see if there was selective pressure toward a higher average learning rate, that is, initial learning rates were restricted to the interval $[0.0, 0.2]$. Results (not shown) indicate that there is little selective pressure even when the interval is restricted providing some evidence to support the hypothesis given above.

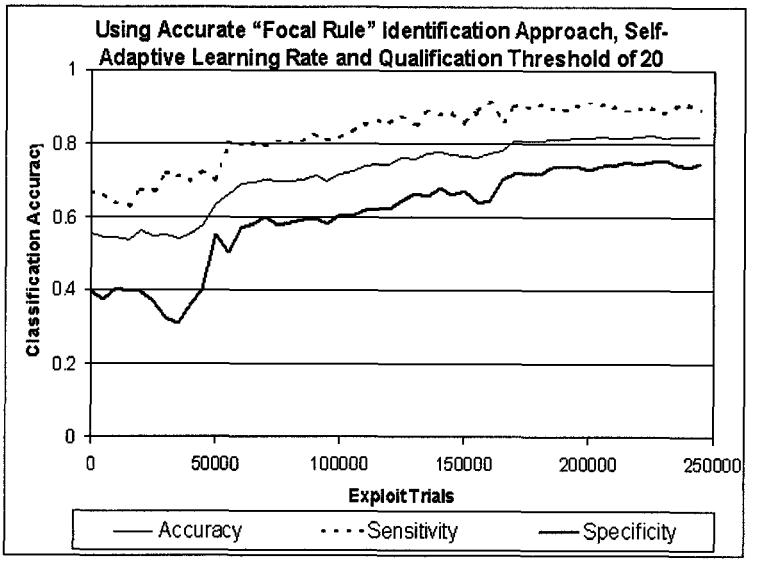

Fig. 27. Classification Accuracy for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\xi=20)}$ "focal rule" identification approach with Self-Adaptive Memetic Learning

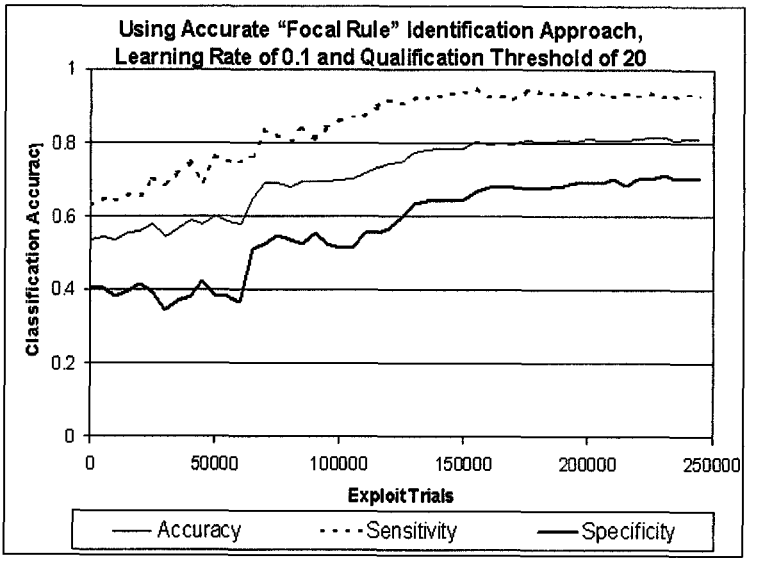

Fig. 28. Classification Accuracy for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1,\xi=20)}$ "focal rule" identification approach

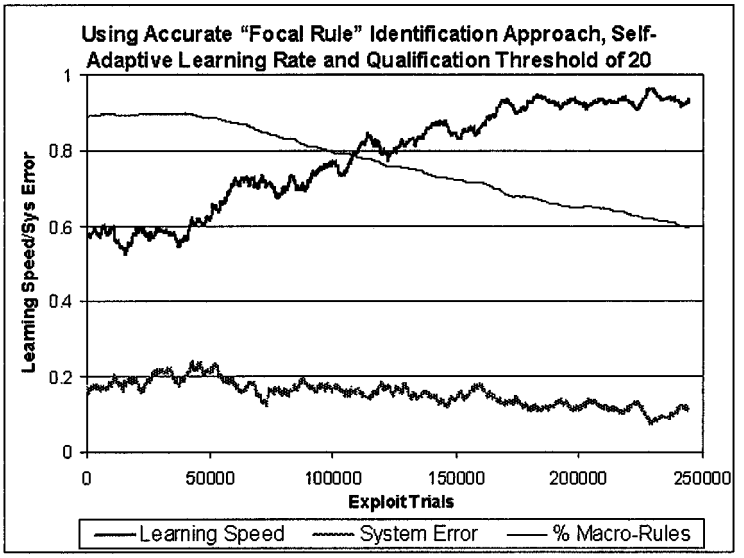

Fig. 29. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\xi=20)}$ "focal rule" identification approach with Self-Adaptive Memetic Learning

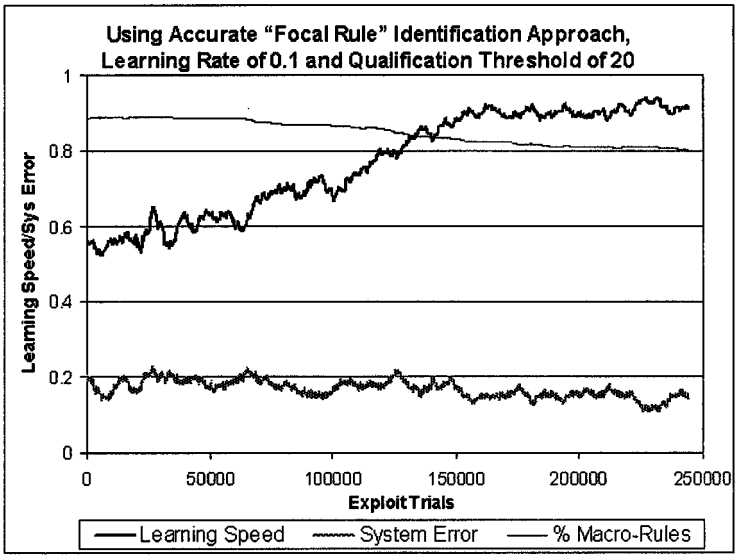

Fig. 30. Learning Speed and System Error for mXCSR with *Action Set Subsumption* and without *Simplified Learning Scheme* using the $Accurate_{(\eta=0.1,\xi=20)}$ "focal rule" identification approach

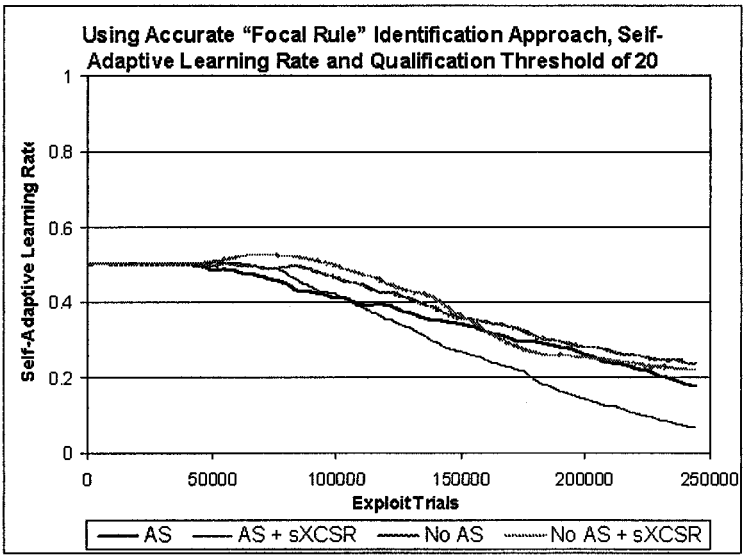

Fig. 31. Self-Adaptive Learning Rates for mXCSR with and without *Action Set Subsumption* and *Simplified Learning Scheme* using the $Accurate_{(\xi=20)}$ "focal rule" identification approach

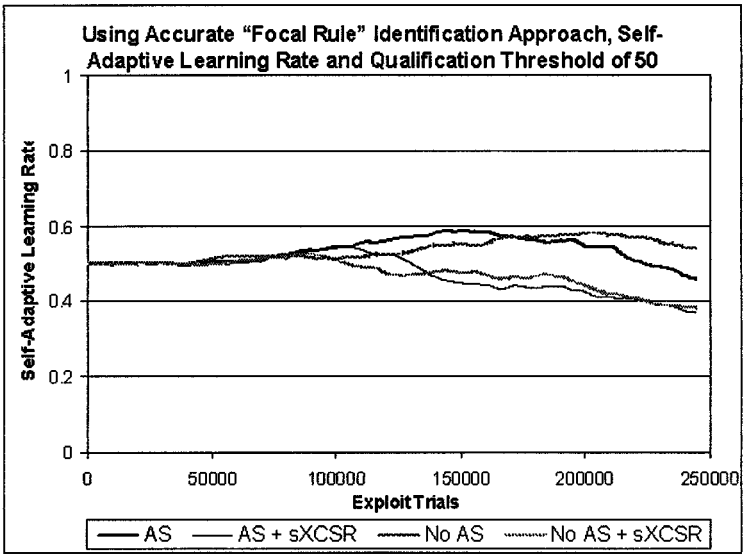

Fig. 32. Self-Adaptive Learning Rates for mXCSR with and without *Action Set Subsumption* and *Simplified Learning Scheme* using the $Accurate_{(\xi=50)}$ "focal rule" identification approach

8 Decomposing the Six-dimensional Test Environment

The six-dimensional test environment used in this study is defined by the additive effect of three different two-dimensional planes, shown in Figs. 11–13, allowing detailed analysis of experimental results that would otherwise be non-trivial for a non-decomposable six-dimensional function. By combining local regions of high performance, as shown in Table 2, an environment of eighteen unique regions of globally high performance is defined.

Table 7 shows the classification accuracy of twelve different versions of the system ($\xi = 20$) that include those systems with and without memetic or self-adaptive memetic learning, termed *Memetic*, *Non-Memetic* and *Self-Adaptive Memetic* (or *SA-Memetic*) accordingly. Comparisons between these systems show that for all parameter settings the *SA-Memetic* version is statistically equivalent to the strongest of the *Non-Memetic* and *Memetic* versions of the system and is significantly ($> 95\%$) better in 1 or 2 regions.

It is clear from the results that the performance of the system in each of the eighteen regions is at least 60% (except for 5 regions in Table 7) and for many regions performance is greater than 70%. Figure 33 represents an overview of Table 7 with respect to the number of regions falling into the performance intervals $[0\%, 70\%]$ and $[70\%, 100\%]$ for the *Memetic*, *Non-Memetic* and *SA-Memetic* versions of the system. It is clear from Fig. 33 that in terms of performance across the regions that the *SA-Memetic* version outperforms the other two versions and it is also clear that the number of regions above 70% performance rises from *Non-Memetic* through *Memetic* to *SA-Memetic*.

9 Conclusion

The motivation for this work was to investigate the effects of applying a memetic learning paradigm to the normal operations of the XCSR classifier system. Previous work showed how the classifier system is able to describe high performance regions in a design-oriented environment. It was hoped that the new memetic learning method would provide results that were at least comparable with the original system. In fact, the new method showed clear improvements for some parameter combinations in the two test environments studied here.

Results for the two-dimensional test environment show a clear improvement in performance for those parameter combinations that have *Action Set Subsumption* turned on and the system is still performing at a significantly higher level when *Action Set Subsumption* is turned off. Results for this environment highlighted a clear statistically significant improvement ($> 99\%$) in performance for $Accurate_{(\eta=0.1)}$ over the other two approaches when *Action Set Subsumption* is turned on and only slightly weaker performance when turned off. The $Accurate_{(\eta=0.1)}$ "focal rule" identification approach appears to be the best one for this test environment. However, the *Simplified Learning*

Table 7. Classification Accuracy on Test Data for Several Different Versions of XCSR when applied to a Six-dimensional Test Environment by High Performance Region (HPR_{region}), where (a)*Non-Memetic*, (b)*Memetic* or (c)*SA-Memetic*

	With *AS-Sub*						Without *AS-Sub*					
	XCSR			sXCSR			XCSR			sXCSR		
	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)	(a)	(b)	(c)
HPR_1	63.7	63.5	66.0	65.6	65.4	66.5	67.3	65.4	66.0	68.3	60.0	68.3
HPR_2	73.1	83.1	83.6	80.8	79.4	80.8	85.3	84.4	83.1	88.1	81.4	87.5
HPR_3	67.1	70.7	73.4	68.0	69.3	69.2	74.6	70.1	73.1	76.3	69.7	75.6
HPR_4	72.0	71.6	78.7	82.1	80.3	81.5	79.7	78.9	79.0	80.3	76.6	80.3
HPR_5	69.0	73.3	79.6	80.8	79.6	79.2	78.7	78.3	78.3	80.0	81.0	79.0
HPR_6	68.4	69.2	77.8	75.3	77.2	77.0	76.9	76.0	76.1	78.3	75.4	77.1
HPR_7	72.0	76.0	76.0	74.5	76.5	79.0	78.5	75.5	79.5	76.1	76.5	82.5
HPR_8	71.0	77.9	80.3	69.0	78.6	74.5	77.9	78.6	76.9	79.5	78.6	83.1
HPR_9	63.4	65.4	69.4	65.6	66.4	65.9	71.1	67.6	69.3	76.9	64.1	73.0
HPR_{10}	56.3	57.0	60.0	62.2	63.0	64.1	69.3	65.9	65.2	65.6	63.7	61.9
HPR_{11}	58.8	71.5	68.5	58.1	75.4	64.6	76.9	74.6	74.6	71.2	77.7	73.1
HPR_{12}	54.8	57.5	61.6	62.7	64.9	64.4	63.7	63.0	62.4	63.5	63.3	63.5
HPR_{13}	68.2	69.8	73.2	75.0	70.3	76.1	74.2	69.7	71.6	75.0	70.3	72.6
HPR_{14}	70.2	73.8	79.7	77.5	73.2	77.2	78.6	75.9	77.9	79.9	73.2	80.4
HPR_{15}	73.1	75.8	81.7	78.8	76.8	79.1	80.8	76.0	77.9	80.0	75.3	80.8
HPR_{16}	74.1	76.8	83.5	83.2	80.5	82.2	82.6	77.3	80.5	82.7	78.0	83.1
HPR_{17}	66.7	69.3	77.4	76.1	76.5	75.4	77.5	72.4	75.1	76.9	74.6	76.9
HPR_{18}	67.3	71.8	76.7	76.7	77.3	78.4	76.9	73.9	75.1	75.9	74.9	77.0

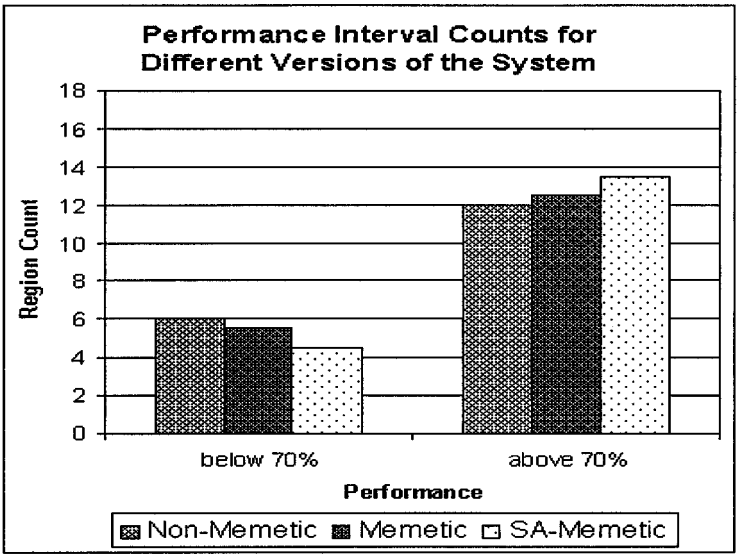

Fig. 33. Performance Interval Counts for *Memetic*, *Non-Memetic* and *SA-Memetic* Systems ($\xi = 20$)

Scheme appeared to degrade performance when *Action Set Subsumption* was used as well as failing to provide any increases in learning speed.

For the more complex six-dimensional test environment, the memetic learning appears to provide little or no advantage to performance over the non-memetic system except for a slight degradation when *Action Set Subsumption* is turned off. However, the memetic learning approach does provide a clear learning speed-up of some 25000–30000 exploit trials over non-memetic learning when the *Simplified Learning Scheme* is used. It is clear that any potential increase in learning speed must be balanced with a potential decrease in performance for a given problem.

Another important result was seen when the update qualification value, ξ, was allowed to vary between 20 and 200 in experiments on the six-dimensional test environment. Results showed that a small, but statistically significant ($> 95\%$) improvement can be made in performance when $\xi = 50$. In fact, differences in performance for those systems with *Action Set Subsumption* turned off, discussed above, are mitigated by the alteration of the threshold ξ to 50.

The study also considered the use of a self-adaptive learning mechanism for adjusting the Widrow-Hoff learning rate used in the memetic learning. The mechanism was applied to both test environments. Results for the two-dimensional test environment showed that although performance varies (depending on *Action Set Subsumption*) there is clear evidence of increased generalisation as well as solution convergence when the self-adaptive technique is

used. Results for the six-dimensional test environment also showed evidence of increased generalisation and solution convergence in addition to a learning speed-up of some 20000 exploit trials. One disadvantage of using the memetic learning approach is an increase in rule-set size. This is clearly at odds with the requirement for a compact rule-set. However, using the self-adaptive mechanism seems to reverse the bloating effects of the standard memetic learning approach on rule-set size for both test environments.

Finally, experimental results for the decomposable six-dimensional test function were analysed region by region. Three different versions of the classifier system, *Non-Memetic*, *Memetic* and *Self-Adaptive Memetic*, were compared clearly showing that for all parameter settings the *Self-Adaptive Memetic* version is statistically equivalent to the strongest of the *Non-Memetic* and *Memetic* versions and is significantly ($> 95\%$) better in 1 or 2 regions. It was also shown that the *Self-Adaptive Memetic* version outperformed the other two by simply counting the number of regions for which each version achieved 70% or greater performance.

10 Mathematical Description of Test Environments

This section defines several related equations that can be used to define the two and six dimensional test environments used in this study. In order to estimate the fitness of a given sample point, a vector of real values is needed, where each value is within some pre-defined interval. For the two dimensional test environment, $x_1, x_2 \in [-6.0, 6.0]$, and for the six dimensional test environment, $a \ldots f \in [0.0, 1.0]$.

10.1 Two-dimensional Test Environment

The two-dimensional test environment is the multi-modal modified Himmelblau function defined by the equation:

$$f(x_1, x_2) = 200 - \left(x_1^2 + x_2 - 11\right)^2 - \left(x_2^2 + x_1 - 7\right)^2$$

10.2 Six-dimensional Test Environment

The six-dimensional test environment is defined by the additive effect of three different two-dimensional planes. Each plane has an associated "local" fitness value and the "global" fitness value of the six-dimensional function is defined by adding each of these "local" fitness values together:

$$fitness_{global} = fitness_{plane1} + fitness_{plane2} + fitness_{plane3}$$

Plane 1

$$z_1 = \frac{0.41}{\frac{(0.8-a)^2}{0.04} + \frac{(0.7-b)^2}{0.04} + 1}$$

$$z_2 = \frac{0.42}{\frac{(0.2-a)^2}{0.0225} + \frac{(0.75-b)^2}{0.09} + 1}$$

$$z_3 = \frac{0.44}{\frac{(0.4-a)^2}{0.2025} + \frac{(0.05-b)^2}{0.09} + 1}$$

$$fitness_{plane1} = \min \begin{cases} 0.5 \\ z_1 + z_2 + z_3 \end{cases}$$

Plane 2

$$z_1 = 0.5 - \frac{0.5}{\frac{(0.25-c)^2}{0.09} + \frac{(0.75-d)^2}{0.09} + 1}$$

$$z_2 = 0.5 - e^{(\cos(48\pi c)+\cos(48\pi d))/14.8}$$

$$z_3 = 0.5 - \frac{0.5}{\frac{(0.75-c)^2}{0.09} + \frac{(0.25-d)^2}{0.09} + 1}$$

$$z_4 = 0.5 - e^{(\cos(24\pi c)+\cos(24\pi d))/14.8}$$

$$z_5 = \max \begin{cases} 0.0 \\ z_2 - z_1 \end{cases}$$

$$z_6 = \max \begin{cases} 0.0 \\ z_4 - z_3 \end{cases}$$

$$z_7 = \begin{cases} z_5 & \text{if } (c < 0.5) \text{ AND } (d > 0.5) \\ 0.0 & \text{otherwise} \end{cases}$$

$$z_8 = \begin{cases} z_6 & \text{if } (c > 0.5) \text{ AND } (d < 0.5) \\ 0.0 & \text{otherwise} \end{cases}$$

$$z_9 = \begin{cases} 0.0 & \text{if } (c < 0.5) \text{ AND } (d < 0.5) \\ \frac{0.35}{\frac{(0.75-c)^2}{0.09} + \frac{(0.75-d)^2}{0.09} + 1} & \text{otherwise} \end{cases}$$

$$z_{10} = \begin{cases} 0.0 & \text{if } (c > 0.5) \text{ AND } (d > 0.5) \\ \dfrac{0.35}{\frac{(0.25-c)^2}{0.09} + \frac{(0.25-d)^2}{0.09} + 1} & \text{otherwise} \end{cases}$$

$$fitness_{plane2} = z_7 + z_8 + z_9 + z_{10}$$

Plane 3

$$z_1 = \min \begin{cases} 0.5 \\ 1.8e^2 + 3f^2 \end{cases}$$

$$z_2 = \begin{cases} 0.0 & \text{if } (e > 0.6) \text{ OR } (f > 0.5) \\ z_1 & \text{otherwise} \end{cases}$$

$$z_3 = \begin{cases} 0.625f & \text{if } (f < 0.8) \\ 0.5 & \text{if } (f \geq 0.8) \text{ AND } (f < 0.85) \\ \frac{10(1-f)}{3} & \text{if } (f \geq 0.85) \end{cases}$$

$$z_4 = \begin{cases} z_3 & \text{if } (e > 0.6) \\ 0.0 & \text{otherwise} \end{cases}$$

$$z_5 = \min \begin{cases} 0.5 \\ z_4 \end{cases}$$

$$z_6 = \min \begin{cases} 0.5 \\ 1.5(0.6 - e) \times 2.5(f - 0.5) \end{cases}$$

$$z_7 = \begin{cases} z_6 & \text{if } (e \leq 0.6) \text{ AND } (f > 0.5) \\ 0.0 & \text{otherwise} \end{cases}$$

$$fitness_{plane3} = z_2 + z_5 + z_7$$

Acknowledgements

The first author is funded by the Engineering and Physical Sciences Research Council. The authors would also like to thank Professor Ian Parmee for his continued support.

References

1. Bäck T (1992) Self-Adaptation in Genetic Algorithms. In: Proceedings of the First European Conference on Artificial Life. MIT Press, Cambridge. pp. 263–271
2. Baldwin J (1896) A New Factor in Evolution. American Naturalist 30: 441–451
3. Beasley D, Bull D, Martin R (1993) A Sequential Niche Technique for Multimodal Function Optimisation. Evolutionary Computation, 1(2): 101–125
4. Bernadó E, Llorà X, Garrell J (2001) XCS and GALE: a Comparative Study of Two Learning Classifier Systems with Six Other Learning Algorithms on Classification Tasks. In: Lanzi PL, Stolzmann W, and Wilson SW (Eds.), Advances in Learning Classifier Systems. Fourth International Workshop (IWLCS-2001), Lecture Notes in Artificial Intelligence (LNAI-2321). Springer-Verlag:Berlin, pp. 115–133
5. Blake C, Merz C (1998) UCI Repository of Machine Learning Databases. University of California, Irvine
   Available at `http://www.ics.uci.edu/~mlearn/MLRepository.html`
6. Bonham C (2000) Evolutionary Decomposition of Complex Design Spaces. PhD Thesis, University of Plymouth
7. Bonham C, Parmee I (1999) An Investigation of Exploration and Exploitation Within Cluster-Oriented Genetic Algorithms (COGAs). In: Banzhaf W, Daida J, Eiben A, Garzon M, Honavar V, Jakiela M, Smith R (Eds), Proceedings of the Genetic and Evolutionary Computation Conference 1999, Morgan Kaufmann, pp. 1491–1497
   Available at `http://www.ad-comtech.co.uk/Parmee-Publications.htm`
8. Bull L, Wyatt D, Parmee I (2002) Initial Modifications to XCS for use in Interactive Evolutionary Design. In: Merelo J, Adamidis P, Beyer HG, Fernandez-Villacanas JL, Schwefel HP (eds) Parallel Problem Solving From Nature - PPSN VII, Springer Verlag, pp. 568–577
9. Butz M, Wilson S (2001) An Algorithmic Description of XCS. In: Lanzi PL, Stolzmann W, Wilson SW (Eds.), Advances in Learning Classifier Systems. Third International Workshop (IWLCS-2000), Lecture Notes in Artificial Intelligence (LNAI-1996). Springer-Verlag: Berlin, pp. 253–272
10. Holland J (1975) Adaptation in Natural and Artificial Systems, The University of Michigan Press, Ann Arbor
11. Holland J (1986) Escaping Brittleness: the Possibilities of General-Purpose Learning Algorithms Applied to Parallel Rule-based Systems. In: Michalski RS, Carbonell JG, Mitchell TM (Eds.), Machine Learning, An Artificial Intelligence Approach. Morgan Kaufmann: Los Altos, California
12. Holmes J (1996) A Genetics-Based Machine Learning Approach to Knowledge Discovery in Clinical Data. Journal of the American Medical Informatics Association Supplement 883
13. Japkowicz N, Stephen S (2002) The Class Imbalance Problem: A Systematic Study. Intelligent Data Analysis, Volume 6(5): 429–450
14. Kaelbling L, Littman M, Moore A (1996) Reinforcement Learning: A Survey. Journal of Artificial Intelligence Research 4: 237–285
15. Kocis L, Whiten WJ (1997) Computational Investigations in Low Discrepancy Sequences, ACM Transactions on Mathematical Software, 23(2): 266–294
16. Kohavi R, Provost F (1998) Glossary of Terms. Machine Learning 30: 271–274

17. Kononenko I, Bratko I (1991) Information-Based Evaluation Criterion for Classifier's Performance. Machine Learning 6: 67–80
18. Krasnogor N, Smith J (2000) Memetic Algorithms: Syntactic Model and Taxonomy. Technical Report, Intelligent Computer Systems Centre, University of the West of England, Bristol, U.K. in 2000 and Automated Scheduling, Optimisation and Planning Group, University of Nottingham, U.K. in 2002
19. Kubat M, Holte R, Matwin S (1997) Learning when Negative Examples Abound. In: Proceedings of the Ninth European Conference on Machine Learning, LNCS 1224, Springer-Verlag, pp. 146–153
20. Lewis D, Gale W (1994) A Sequential Algorithm for Training Text Classifiers. In: Proceedings of SIGIR-94, 17th ACM International Conference on Research and Development in Information Retrieval, ACM/Springer, pp. 3–12
21. Ling C, Li C (1998) Data Mining for Direct Marketing: Problems and Solutions. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-98), AAAI, pp. 73–79
22. Morgan C (1896) On Modification and Variation, Science 4: 733–740
23. Moscato P (1989) On Evolution, Search, Optimization, Genetic Algorithms and Martial Arts - Towards Memetic Algorithms. Technical Report 826, California Institute of Technology, Pasadena, California.
24. Osborn H (1896) Ontogenic and Phylogenic Variation, Science 4: 786–789
25. Parmee I (1996) The Maintenance of Search Diversity for Effective Design Space Decomposition using Cluster-Oriented Genetic Algorithms (COGAs) and Multi-Agent Strategies (GAANT). In: Proceedings of 2nd International Conference on Adaptive Computing in Engineering Design and Control, PEDC, University of Plymouth, pp. 128–138
Available at `http://www.ad-comtech.co.uk/Parmee-Publications.htm`
26. Stone C, Bull L (2003) For Real! XCS with Continuous-Valued Inputs. Evolutionary Computation 11(3): 299–336
Also available at `http://www.cems.uwe.ac.uk/lcsg`
27. Swets JA (1988) Measuring the Accuracy of Diagnostic Systems. Science 240: 1285–1293
28. Weiss G, Provost F (2001) The Effect of Class Distribution on Classifier Learning: An Empirical Study. Technical Report ML-TR-44, Department of Computer Science, Rutgers University
29. Whitley D, Gordon S, Mathias K (1994) Larmarckian Evolution, the Baldwin Effect and Function Optimization. In: Davidor Y, Schwefel H, Manner R (Eds.) Parallel Problem Solving From Nature - PPSN III, Springer-Verlag, pp. 6–15
30. Wilson S (1995) Classifier Fitness Based on Accuracy. Evolutionary Computation 3(2): 149–175
31. Wilson S (2000) Get real! XCS with Continuous-valued inputs. In: Lanzi PL, Stolzmann W, Wilson SW, (Eds.) Learning Classifier Systems. From Foundations to Applications Lecture Notes in Artificial Intelligence (LNAI-1813) Springer-Verlag: Berlin, pp. 209–222
32. Wilson S (2001) Compact Rulesets for XCSI. In: Lanzi PL, Stolzmann W, Wilson SW, (Eds.) Advances in Learning Classifier Systems. Fourth International Workshop (IWLCS-2001), Lecture Notes in Artificial Intelligence (LNAI-2321). Springer-Verlag: Berlin, pp. 197–210
33. Wilson S (2001) Mining Oblique Data with XCS, In: Lanzi PL, Stolzmann W, Wilson SW, (Eds.) Advances in Learning Classifier Systems. Third Interna-

tional Workshop (IWLCS-2000), Lecture Notes in Artificial Intelligence (LNAI-1996). Springer-Verlag: Berlin, pp. 158–177
34. Wyatt D, Bull L (2003) Using XCS to Describe Continuous-Valued Problem Spaces. Technical Report UWELCSG03-004.
Available at `http://www.cems.uwe.ac.uk/lcsg`