Distributed Classifier Migration in XCS for Classification of Electroencephalographic Signals

B. T. Skinner, H. T. Nguyen, D. K. Liu

Abstract—**This paper presents an investigation into combining migration strategies inspired by multi-deme Parallel Genetic Algorithms with the XCS Learning Classifier System to provide parallel and distributed classifier migration. Migrations occur between distributed XCS classifier sub-populations using classifiers ranked according to numerosity, fitness or randomly selected. The influence of the degree-of-connectivity introduced by Fully-Connected, Bi-directional Ring and Uni-directional Ring topologies is examined. Results indicate that classifier migration is an effective method for improving classification accuracy, improving learning speed and reducing final classifier population size, in the single-step classification of noisy, artefact-inclusive human electroencephalographic signals. The experimental results will be used as part of our larger research effort investigating the feasibility of using EEG signals as an interface to allow paralysed persons to control a powered wheelchair or other devices.**

Index Terms— **Learning classifier system (LCS), XCS, evolutionary computation, genetic-based machine learning (GBML), electroencephalogram, classifier migration.**

I. Introduction

XCS [1] is a key development in learning classifier systems (LCS) research that improves on the some of the limitations of traditional LCS, which were introduced by Holland [2]. XCS has proved to be effective in many domains, in particular data mining and single-step classification applications where it can perform better than other traditional machine learning techniques [3-5].

The parallelisation of evolutionary algorithms, in particular, parallel genetic algorithms (PGA) has become an established research area [6-8]. However, only a small number of investigators have examined migration strategies inspired by multi-deme PGAs in parallel and distributed learning classifier systems, which also employ a genetic algorithm for improving a population of classifiers.

Some early examples include Dorigo et al.'s [9] use of multiple LCS within a hierarchical structure to control an autonomous robot and Seredynski et al.'s [10] learning of Nash equilibria by coevolving distributed LCS in competitive or cooperative approaches. More recently, Cao et al [11] used distributed LCS to learn and control signalling at road traffic junctions. Recently, Bull et al [12] investigated the inclusion of a rule migration mechanism into a derivative of XCS and found it to be an effective way to improve learning speed. Here, rule migration was applied to the 20-bit multiplexer test problem. Wyatt et al [13] investigated the migration of rules for two popular multi-step test environments (maze6 and woods1). Here rules were selected from eight Zeroth Level Learning Classifier System (ZCS) sub-populations and exchanged using a unidirectional ring topology.

We have explored distributed classifier migration using a uni-directional ring topology where classifiers migrate to a single neighbouring XCS sub-population (deme) in one direction only. In addition, improved learning speed and reduction in the XCS sub-populations were achieved using fully-connected and bi-directional ring topologies.

In this paper, six classifier selection/replacement schemes were investigated, based on combinations of classifier numerosity, fitness and random. Experimental results suggest that selection/replacement policies using random and numerosity bias provide improved classification performance and that fitness bias can lead to a reduction in classification performance, resulting from the selective pressure introduced by distributed classifier migration.

Migration policies based on a range of migration rates and migration frequencies have been explored for each topology and selection/replacement scheme with some general recommendations presented to facilitate robust and improved classification performance.

Section II provides a brief description of the Wilson's XCS learning classifier system. Section III describes the multi-deme topologies and migration policy used to control the rate at which classifiers disseminate between XCS sub-populations. Section IV describes the experimental environment. Section V presents the experimental results and section VI summarises conclusions and future research.

II. Description of XCS

XCS[1, 14] is an accuracy-based classifier system. It exploits a *reinforcement learning method* [15] coupled with the robust and global search capability of a *genetic algorithm* [2, 6] to produce a population of classifiers. The classifiers form a maximally accurate and maximally general knowledge representation (*complete map*) of the target problem in single- and multi-step environments.

This section provides a brief description of XCS in single-

Manuscript received March 15, 2007. This work is supported by the Centre for Health Technologies, University of Technology, Sydney, Australia. The authors are with the Mechatronics and Intelligent Systems Group, University of Technology, Sydney, 2000, Australia.
(e-mail: brad.skinner@uts.edu.au).

step environments, since the EEG classification task was modelled as a single-step problem. Figure 1 is a schematic illustration of XCS as described in this section.

A. Knowledge Representation

XCS evolves a population [P] of classifiers, where each classifier consists of a rule and three main parameters estimating the quality of the rule. Each rule consists of a condition and an action pair [$condition \rightarrow action$]. With a binary encoding scheme the input space can be denoted by $S \subseteq \{0,1\}^L$, where L is the fixed length of a feature vector received from the environment. Thus, the condition part of the rule $C \in \{0,1,\#\}^L$ specifies the input states $s \in S$ the classifier is capable of *matching*. In this case, for example, a condition $(c_1, c_2, ..., c_n)$ matches an input $(s_1, s_2, ..., s_n)$, iff $\forall i \; c_i = s_i \vee c_i = \#$. The hash symbol # means "don't-care" and allows formation of generalisations in the condition part of a rule. The action part of the rule specifies the action $a \in A$ or *class* for which a payoff is predicted, that is, when the condition is satisfied.

Three main parameters estimate the quality of each classifier in the population [P]. The prediction p estimates (keeps an average of) the payoff expected if the classifiers condition matches the input from the environment and its advocated action (class) is selected. The prediction error ε estimates the average error between the classifiers payoff prediction and the received payoff. The fitness F estimates the accuracy of the payoff prediction p and is an inverse function of the prediction error. The fitness metric ultimately defines the superiority of rules for XCS.

B. Performance Component

At each discrete time step, the system receives an input *s(t)* describing the current state of the environment at time *t*. Given *s(t)*, XCS forms a *match set* [M] of classifiers in [P] whose conditions are satisfied by the current input. If the match set [M] contains less than θ_{mna} classifiers with different actions, *covering* is performed. The covering mechanism creates new classifiers with a condition that matches the current input and action selected randomly from those not contained in [M]. Specifically, each attribute in the condition part of a newly created covering classifier is set to # with a probability given by $P_{\#}$ or alternatively to the corresponding input symbol {0,1}.

XCS computes the *system* or *payoff prediction, P(a)* for each action *a* in [M]. *P(a)* estimates the payoff the system will receive if action *a* is chosen. It is computed by the fitness-weighted average of all matching classifiers that specify action *a*. The different values of *P(a)* form the *prediction array* and XCS selects an winning action based upon the values stored in *P(a)*.

For the EEG signal classification task, XCS selects the winning action using either a *pure explore mode* or *pure exploit mode*. During training sessions pure explore mode was used so actions are selected randomly. During testing and validation sessions, i.e., when XCS attempts to classify new and unseen instances based on the knowledge it has gained, pure exploit mode was used so actions are selected deterministically according to the highest prediction.

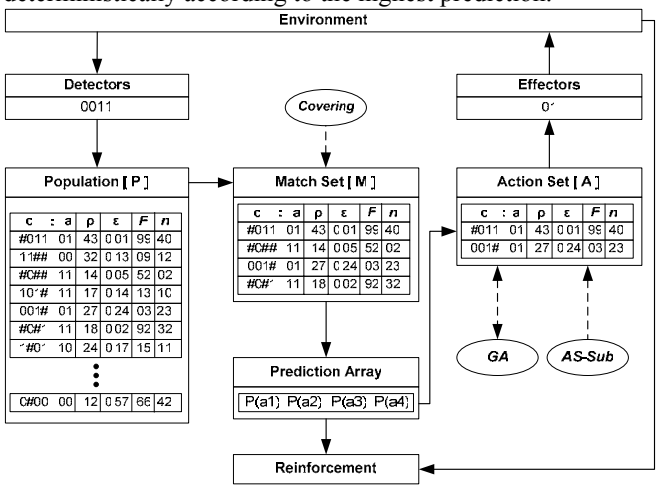

Figure 1: Block Diagram of XCS Classifier System, excluding Previous Action Set $[A]_{-1}$ (Adapted from [1])

C. Reinforcement Component

The selected action is sent to the environment and a reward R is returned, which is then used to update the parameters of classifiers currently stored in [A]. Furthermore, *only* those classifiers stored in [A] are updated in the following order. Firstly, the classifier prediction p is updated as follows:

$$p \leftarrow p + \beta\left(R - p\right) \tag{1}$$

where $\beta\left(0 \le \beta \le 1\right)$ is the *learning rate* and R is the reward received from the environment. Next, the prediction error is updated as follows:

$$\varepsilon \leftarrow \varepsilon + \beta\left(\left|R - p\right| - \varepsilon\right) \tag{2}$$

Finally, the classifiers fitness F is updated. But, first the classifiers *accuracy* κ and *relative accuracy* κ' are computed as:

$$\kappa = \begin{cases} 1, & \text{if } \varepsilon < \varepsilon_0 \\ \alpha\left(\dfrac{\varepsilon}{\varepsilon_0}\right)^{-\upsilon}, & \text{otherwise} \end{cases} \tag{3}$$

$$\kappa' = \frac{\kappa}{\sum_{cl \in [A]} \kappa_{cl}} \tag{4}$$

The parameter $\varepsilon_0 : (\varepsilon_0 > 0)$ is a constant that controls the tolerance for prediction error ε, under which a classifier is considered to be accurate; the parameters $\alpha : (0 < \alpha < 1)$ and $\upsilon (\upsilon > 0)$ are constants controlling the rate of decline in accuracy κ when ε_0 is exceeded and in this case the, classifier is considered inaccurate. The classifier accuracy κ is determined from the prediction error ε as in Eq(3). Next the accuracy values κ stored in [A] are converted to relative accuracies as in Eq(4). Finally, the classifier fitness F is updated according to the classifiers current relative accuracy as follows:

$$F \leftarrow F + \beta\left(\kappa' - F\right). \tag{5}$$

In XCS the classifier fitness F is an estimate of the classifiers accuracy relative to other classifiers in [A] and is an inverse function of the prediction error ε.

Furthermore, each time a classifier is included in [A], since its creation, the *experience* parameter *exp* is incremented tracking any previous participation. The *action set size* parameter *as* estimates the average size of the action set where the classifier participates.

D. Genetic Search Component

In XCS, a steady-state genetic algorithm (GA) is responsible for improving the set of rules. The GA attempts to discover new classifiers which contribute to existing knowledge and delete classifiers that do not offer improved contributions. Application of the GA takes place in *niches* defined by classifier in [A] iff the average time since the last GA application in [A] exceeds a constant threshold given by θ_{GA}.

Firstly, the GA selects two parental classifiers from the current [A] with probability proportional to their fitness *F*. In this paper the tournament selection method is used to select the fittest parental classifiers from a tour. The parental classifiers undergo uniform crossover with probability χ and single-bit *free* mutation with probability μ per allele. Finally, the two resulting offspring classifiers are inserted into the population to compete with their parents.

E. Classifier Deletion

The deletion of classifiers from [P] maintains a constant population. Deletion of classifiers occurs if the total number of classifiers in [P] is greater than a constant threshold. Classifiers are selected for deletion with probability proportional to an estimate of the average size of [A] in which the classifier participated. The *action set size* parameter (*as*) stores this estimate. In addition, if the classifier is sufficiently experienced ($exp > \theta_{del}$, where θ_{del} is the constant deletion threshold) and its fitness *F* is significantly lower than the average fitness of classifiers contained in [P], its probability of deletion is increased in inverse proportion to its fitness.

F. Subsumption Component

In XCS there are two *subsumption deletion* schemes aimed at broadening the generalisation capability and tending to compress [P] towards maximum generality.

When an offspring classifier is created through the application of the GA, it is evaluated prior to its insertion into [P]. If the condition and action of the offspring classifier can be logically subsumed by the condition and action of an accurate ($\varepsilon < \varepsilon_0$), sufficiently experienced ($exp > \theta_{sub}$) and formally more general parent classifier, then the offspring classifier is discarded and the parents *numerosity* (*num)* is incremented. This process is called *GA subsumption*. If the offspring classifier is not GA subsumed it is inserted into the population, deleting another classifier.

The second method, *action set subsumption* searches the current [A] for the most general classifier that is both accurate and sufficiently experienced. Then, all the remaining classifiers in the [A] are evaluated against the most general classifier for possible subsumption. If a classifier is subsumed it is deleted from the population. Action set subsumption tends to create a stronger subsumption pressure than the GA subsumption method [16].

G. Macroclassifiers

Macroclassifiers logically represent a set of classifiers with identical condition and action using the *numerosity* (*num*) parameter. In XCS, whenever a new classifier is generated, either at initialisation, by the GA or due to covering, the population [P] is searched to find an existing classifier with an identical condition and action. If a classifier is found its numerosity is incremented and the new classifier is discarded. If not, the new classifier is inserted into [P] with *num*=1. As a result, [P] contains structurally unique classifiers with numerosity ≥ 1. These classifiers are termed *macroclassifiers.* A macroclassifier with numerosity *n* is structurally equivalent to *n* individual classifiers.

III. Multi-Deme XCS Models

This section describes the parallel and distributed topologies and the migration policy used to control the rate at which classifiers disseminate between XCS demes.

A. Multi-Deme Topology

The simplest example of distributed populations in evolutionary algorithms is modelled on a network of isolated islands, each being panmictic. The Island Model [17] is derived from the evolutionary biological theory known as Punctuated Equilibria [18], where individuals can periodically migrate to a single destination sub-population. Cantu-paz [19] performed an exhaustive investigation on the impact of migration pressure in uni-directional ring, bi-directional ring and fully-connected topologies using PGAs and found that at any given time, topologies where the demes have more neighbours reach higher quality solutions more rapidly than sparse topologies.

Given that XCS consists of a population of co-evolving classifiers, we can similarly explore the uni-, bi-directional and fully-connected topologies and the influence of their explicit degree-of-connectivity (φ) on classification accuracy, learning speed and classifier population size. Here, topologies uses eight XCS demes with a fixed population size (*N*=8000), as illustrated in Figure 2. Two and four deme topologies (not shown) where also examined, but improvements where less marked.

B. Migration Policy

Tanese [20] was one of the first to initiate a systematic study of migration policies and its effects on the efficiency and quality of loosely-coupled, coarse-grained PGAs. Her experimental method consisted in dividing a fixed number of individuals into equally-sized demes, and to vary the migration rates and migration frequency. The findings suggest that, infrequent migrations with a small number of individuals provide an efficient and accurate PGA.

Recent studies have considered the effect of migration due to selection/replacement pressure and compared several frequently-used configurations of migration policies in PGAs [19]. More recently, the authors carried out a performance study of a multi-deme PGA with adaptive mutation on a common set of highly multi-modal test problems [21].

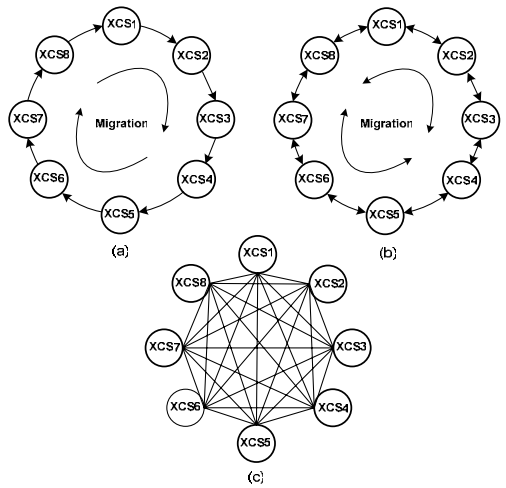

Figure 2: (a) Uni-directional Ring, **(b)** bi-directional ring and **(c)** fully-connected topologies using eight XCS demes, with degree-of-connectivity φ=1, 2, and 7 respectively.

The migration policy for the multi-deme XCS model can be conveniently defined by six parameters:

$$M = \left(m_R, m_F, t_H, \psi_S, \psi_R, s\right) \tag{6}$$

where:

- m_R: is the migration *rate* specifies the number of micro-classifiers undertaking migration as a proportion of the XCS deme population, N, $m_R \in \{0, 0.01N, 0.02N, 0.04N, 0.06N\}$.
- m_F: is the migration *frequency* and specifies how often migrations occur, $m_F \in \{50, 100, 200, 500\}$ (XCS explore iterations).
- t_H: is the *hold-off* period before migrations occur, $t_H = 0.1 \times max\_nr\_steps$
- ψ_S: is the classifier *selection* policy specified by either classifier numerosity (N), fitness (F) or random (R).
- ψ_R: is the classifier *replacement* policy specified by either classifier fitness (F) or random (R).
- *s:* is the communication type, either synchronous or asynchronous.

The selection and replacement algorithms operate on [P], as illustrated in Figure 3. Following each initiation of the GA, m_F and t_H are tested within each XCS deme. No migration occurs until the hold-off period (t_H) has elapsed, allowing demes to converge during the initial stage of learning [19]. If the number of explore trials is equal to m_F and greater than t_H, m_R emigrants are selected from [P] for migration according to ψ_S. Emigrants are selected according to ranked classifier numerosity, fitness or unranked random selection.

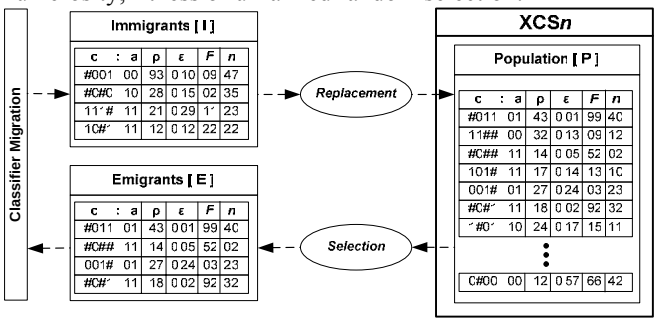

Figure 3: Classifier selection and replacement mechanisms operate on the local XCS population [P]. Producing [I] and [E] classifier populations.

Classifiers migrated from different XCS demes are received as immigrants. Immigrants are inserted into the recipient [P] with replacement of local classifiers according to ψ_R, which is random or fitness biased. Insertion and deletion of classifiers into the local [P] performed using the same algorithms as for GA deletion. Furthermore, all distributed classifier migrations occur synchronously as specified by *s*.

IV. Experimental Environment

This section provides a description of the experimental environment for the single-step classification of multi-channel human EEG signals. Training and testing was performed using a *subset* of EEG data taken from the authors previous study. The study was approved by the institutional research ethics committee and participants were only entered into the study after informed consent.

A. Experimental Tasks

The participants performed two mental tasks, chosen to invoke changes in brainwave activity across different hemispheres and across different electrodes [22]. Each task was performed for a continuous period of 10 seconds and repeated for 10 independent trials during a single EEG recording session. The participants were asked to remain still, not make any overt movements or vocalise any task aimed at minimising muscular artefacts. The two subset tasks studied in this paper were performed with eyes opened as follows:

- *Mental Counting (MC):* The participant was instructed to imagine counting consecutive numbers.
- *Figure Rotation (FR):* The participant was instructed to visualise a complex object being rotated about one of its axis.

B. Data Acquisition and Pre-processing

The Biosemi™ Active-Two System (www.biosemi.com) was used in this study for recording EEG signals from the participant. Raw data was acquired using 32-channels at a sampling rate of 1024Hz with 24-bit digital resolution per channel, using the biopotential measurement system and active electrodes. The EEG electrode montage was adapted from the extended 10-20 electrode system [23] and all 32 channels were referenced to electrically-linked electrodes located on the ear lobes of the participant.

A 32-electrode set of Ag-AgCl, Pin-type Active electrodes were used to measure the EEG signal. To record brain activity

from a participants scalp, the electrodes were mounted into electrode holders located on the Biosemi head cap. To provide sufficient electrical contact between the participants scalp and EEG electrodes, electrode holders were filled with low impedance, highly conductive electrode gel before attaching the electrodes. The recording of the EEG signals was conducted in a dedicated temperature controlled research laboratory.

Several pre-processing steps were performed prior to applying parametric transform methods to the raw EEG data. Firstly, the number of channels was reduced from 32- to 6-channels. As in previous research work [24], this study focused on EEG channels (O_2, P_4, C_4) for the right hemisphere (R) and (O_1, P_3, C_3) for the left hemisphere (L). The remaining six channels were down-sampled to 256Hz, removing frequencies not typically found in human EEG signals, which can range from 0.1Hz to 100Hz, as shown in Figure 4(a). Each 10 second recording was reduced to 8 seconds by removing 1 second of data from the beginning and end of the EEG window. This was aimed at removing transitional effects introduced prior to or following each recording experiment as a participant initiates the mental task or concludes early.

Finally, the remaining time-domain EEG data was partitioned into separate *segments* of 2 seconds, allowing the parametric method to operate on smaller portions of EEG data as illustrated in Figure 4(b). For each mental task this produces 240 samples for 2 second segments respectively.

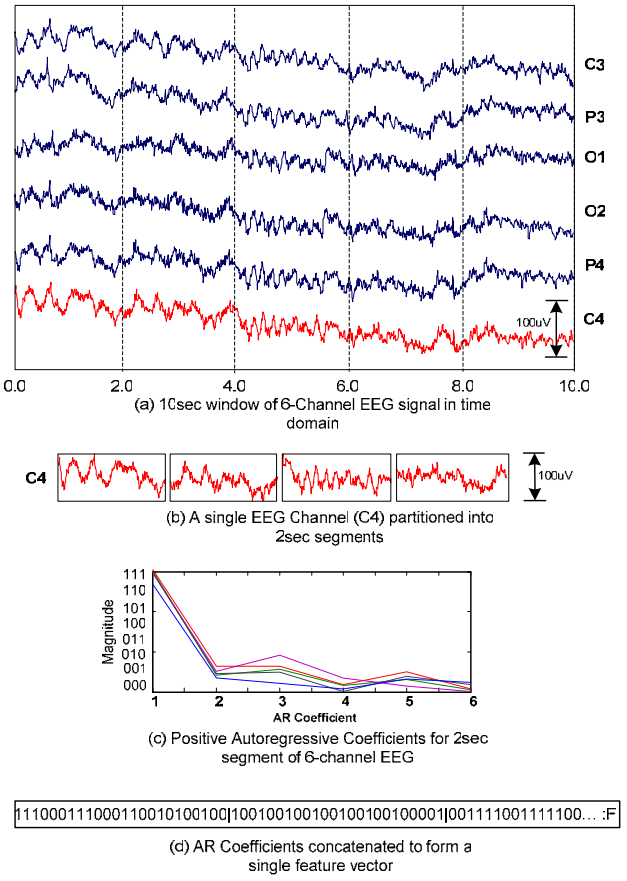

Figure 4: Feature vector creation – from multi-channel EEG signal to binary string for input into XCS.

C. *Parametric Methods for EEG Signal Representation*

Parametric methods assume a description of an EEG signal can be devised from a time-series model of a random process. As such, parametric methods *model* fixed blocks of EEG data as the output of a linear filter of order p driven by a Gaussian white noise sequence with zero-mean Ch6,[25]. The output for such a filter is a p^{th} order *autoregresssive* (AR) process or maximum entropy method (MEM), given by Eq(7).

$$x(n) = -\sum_{k=1}^{p} a(k)x(n-k) + u(n) \quad (7)$$

where, $x(n)$ is the stationary time-series output sequence that models the fixed segment of EEG data, $a(k)$ are the AR coefficients and $u(n)$ is a white noise input driving sequence.

The Burg AR parameter estimation algorithm was used in this study. This method operates on a fixed *segment* of EEG data recursively yield a p^{th} order AR model of parameter estimates $a[k]$, called AR *coefficients*. It is the AR coefficients that are used to describe the EEG signal opposed to the estimate of the spectral density, as illustrated in Figure 4(c).

The computationally efficient Burg method estimates AF coefficients from the complex-valued reflection coefficient sequence, based on a least squares criterion, while satisfying the Levinson-Durbin recursion Ch8,[25].

Selection of the AR model order (p) for real-world signals, represents a trade-off between increased resolution and decreased prediction variance. Model order determination was performed using a random selection of ten 2 second EEG segments from the two mental tasks. As the model order was incrementally increased from p=1 to p=50 the rate of decrease in prediction error variance curve and minimisation of the Akaike information criterion (AIC)[25] curve suggested a model order of between $5 \leq p \leq 10$. The AR model used for all experiments were implemented with order p=6.

D. *Feature Vector Encoding*

The production of feature sets was the final stage of transforming raw EEG signals into a form that was amenable to the parallel and distributed XCS learning classifier system.

Separate training and testing sets, containing multiple feature vectors were created to train XCS and test classification accuracy, specificity and sensitivity. Single feature vectors (*instances*) were encoded from the six AR coefficients that represent a single EEG 2.0sec segment for each EEG channel (O_1, P_3, C_3, O_2, P_4, C_4). Each real-valued Burg AR coefficient was converted to a positive value, normalised and finally converted to an unsigned binary string. The final feature vector for a single EEG segment was created from the concatenation of the six EEG channels containing the six AR coefficients represented as 3-bit binary strings as illustrated in Figure 4(d). Thus, a single *instance* was represented by a 108-bit binary string and a class label (FR or MC) corresponding to the two mental tasks used in this paper.

A total of 40 instances were randomly inserted into training and testing sets, which provided 50% of the data for the training phase and the remaining 50% for the testing phase.

E. XCS Parameter Settings

The parameters for each XCS deme were set as follows: N=8000, α=0.1, β=0.2, δ=0.1, ε_o=0.01, ν=5, θ_{GA}=25, θ_{mna}=#actions, χ=0.9, μ=0.005, θ_{del}=100, θ_{sub}=100, $P_{\#}$=0.76, *doGAsubsumption*=1, *doASsubsumption*=0, *reward*=1000/0, t*oursize*=0.4, $P_{explore}$=1, *crossover_type*=uniform, *initpop*=0, *fitness_reduction*=1, *general_mutation*=0, *niche_mutation*=0, *force_different_in_tournament*=0, *select_tolerance*=0.001, *do_mam*=1, p_I=10, ε_I=0.0, f_I=0.01.

A total of 25 independent experiments were performed for a maximum of 20000 explore iterations. All remaining parameters were set to default values.

Parameter values for distributed classifier migration in XCS used the following sets: $t_H = \{2000\}$, $m_F \in \{50,100,200,500\}$, $m_R \in \{0.01N, 0.02N, 0.04N, 0.06N\}$, $\psi_S \in \{\text{N, F, R}\}$, and $\psi_R \in \{\text{F, R}\}$. XCS demes used synchronous communications.

F. Cluster Computing Environment

All experiments were performed using a computing cluster specified in Table I.

TABLE I
CLUSTER COMPUTING HARDWARE AND SOFTWARE ENVIRONMENT

Computing Environment Component	Description
Number of Node Utilised	2,4,8
Processor Type and core Speed	Pentium Core2 Duo @ 2.93Ghz
Front-side Bus Bandwidth	1066MHz
DRAM capacity and bandwidth	4GB DDR2 @ 667MHz
Network Type and Bandwidth	1000Mbps Ethernet
Network Switching Type	Gigabit Switching Fabric
Network Protocol	TCP/IP V4
OS Kernel Type and Version	RH E Linux 4.0 (2.4.20-19.8)
MPI Type and Version	LAM 7.0.6 / MPI 2
Compiler Type and Version	mpiCC and gcc (3.2.3)
Coding Language Standard	ISO C

V. RESULTS

Figure 5 to Figure 7 provide a comparison between the different selection/replacement (ψ_S:ψ_R) policies for each topology, which achieved better classification performance.

In Figure 5, the fully-connected topology maintains a significant improvement in classification accuracy and learning speed after 6K iterations, for *all* (ψ_S:ψ_R) policies compared to XCS without migration, according to a paired *t-test* at a 99.5% confidence level.

In Figure 6, the bi-directional ring topology maintains a significant improvement in classification accuracy and learning speed after 8K iterations, for *all* (ψ_S:ψ_R) policies compared to XCS without migration, according to a paired *t-test* at a 99.5% confidence level. Policies {N,R}, {N,F}, {R,R} and {R,F} provide improved classification accuracy, although those based on random selection (ψ_S=R) have a slower learning speed. The {F,R} and {F,F} policies initially provide the highest learning speed, but lower classification accuracy after the 20K explore iterations.

In Figure 7, the uni-directional ring topology maintains a significant improvement in classification accuracy and learning speed after 13K iterations, for *all* (ψ_S:ψ_R) policies compared to XCS without migration, according to a paired *t-test* at a 99.5% confidence level. Policies {N,R} and {N,F} provide better classification accuracy. The {F,R}, {F,F}, {R,R} and {R,F} policies result in similar classification accuracies after 20K explore iterations, with those based on random selection (ψ_S=R) having less of an effect on learning speed.

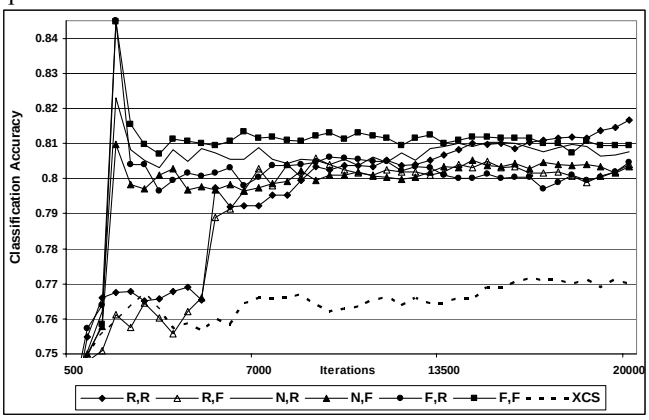

Figure 5: Best classification performance for **fully-connected** topology using different selection (ψ_S) and replacement (ψ_R) policies (m_R=0.04N, m_F=500).

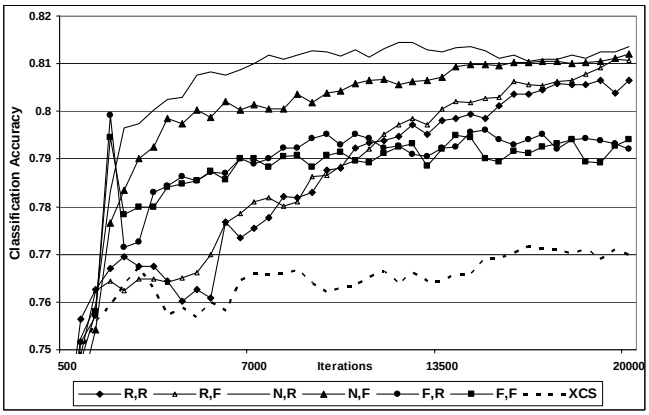

Figure 6: Best classification performance for **bi-directional** topology using different selection (ψ_S) and replacement (ψ_R) policies (m_R=0.04N, m_F=100)

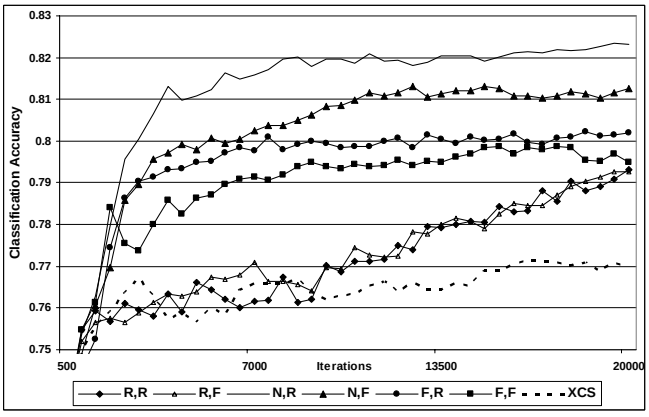

Figure 7: Best classification performance for **uni-directional** topology using different selection (ψ_S) and replacement (ψ_R) policies (m_R=0.04N, m_F=100).

The results indicate that learning speed is influenced by the selection/replacement policy and degree-of-connectivity (φ). Selection/replacement policies based on random selection typically provide slower learning speed than policies with numerosity or fitness biased selection for the ***same*** m_R and m_F

values. The fully-connected topology (φ=7) provides the highest learning speed for all selection/replacement policies. The bi-directional ring topology (φ=2) has a lower learning speed than the fully-connected topology, but is faster than the uni-directional ring topology (φ=1) for the same m_R and m_F values.

We also found, declining classification accuracy can occur for policies using fitness biased selection if m_R and m_F are not appropriately selected for the topology as illustrated by the example in Figure 8.

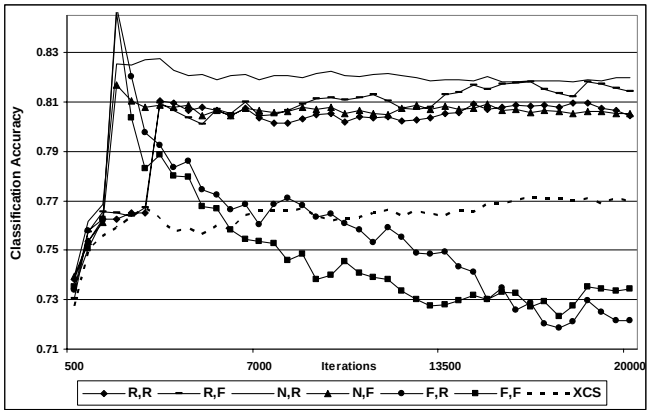

Figure 8: Classification performance can ***decline*** when $m_R \geq 0.04N$ and $m_F \leq 200$ for the fully-connected topology. This occurs for policies based on fitness biased selection.

Table II, shows the maximum value for migration rate (m_R) versus migration frequencies (m_F) that were reached, before classification accuracy declined below the baseline result achieved by XCS. The starred (*) values indicate a maximum migration rate was beyond the upper bound of m_R=0.06N, used in this study.

TABLE II
MIGRATION RATES AND FREQUENCIES FOR VARYING DEGREES-OF-CONNECTIVITY FOR FITNESS BIASED SELECTION

m_F	φ=7	φ=2	φ=1
500	≤ 0.06N	*	*
200	< 0.04N	≤ 0.06N	*
100	≤ 0.02N	≤ 0.04N	*
50	≤ 0.01N	≤ 0.02N	≤ 0.06N

In addition to improved classification accuracy and learning speed the inclusion of distributed classifier migration can significantly reduce the XCS classifier population as illustrated in Figure 9 to Figure 11.

Compaction of the classifier population has two important benefits. Firstly, a compact rule set can be readable by humans, allowing off-line interpretation and understanding of what XCS has learnt, as compared, for example, with a trained neural network, that contains a weight matrix of real-valued numbers making off-line interpretation very difficult. Secondly, a smaller classifier population that maintains high classification performance is appealing for use in EEG classification applications that utilise embedded hardware for mobile or remote classification of multi-channel EEG signals, where computational power and memory capacity is limited.

In this paper, compaction of the classifier sub-populations takes place during the learning phase, unlike existing off-line compaction algorithms [26], which achieve human-readable set of high-quality classifiers. Furthermore, the experimental results show the reduction of classifier population is determined solely by the selection policy (ψ_S). The replacement policy (ψ_R) has no influence.

Fitness biased selection provides the fastest reduction in classifier population size, reaching a minimum of 0.40N after 12K iterations for the fully-connected topology as illustrated in Figure 9. Classifier selection policies using numerosity bias or random are slower to converge to a similar classifier population (0.41N), which is reached after 17K iterations. A single, XCS classifier population reaches a minimum of 0.93N.

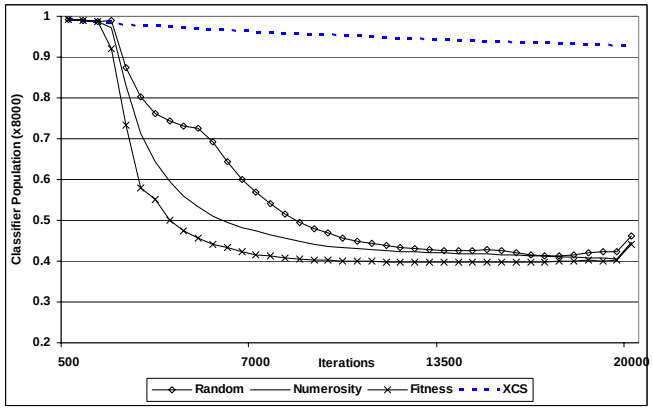

Figure 9: Classifier population for different selection policies (ψ_S) of the best performing **fully-connected** topology (m_R=0.04N, m_F=500).

For the bi-directional ring topology, random classifier selection provides the fastest reduction in classifier population size, reaching 0.28N after 10K iterations, as illustrated in Figure 10. Classifier selection policies using numerosity or fitness bias are slower to converge and reach 0.36N and 0.39N after 18K iterations respectively.

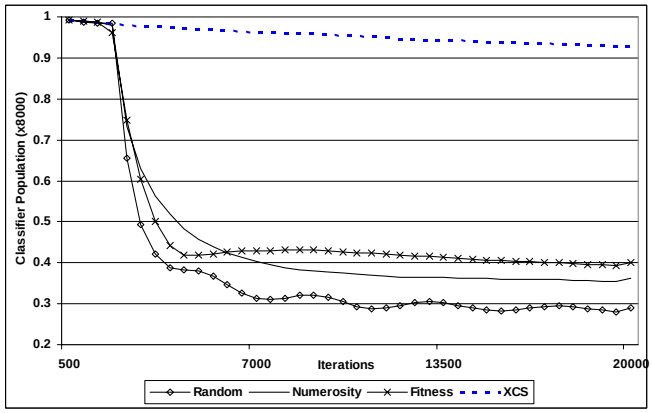

Figure 10: Classifier population for different selection policies (ψ_S) of the best performing **bi-directional** ring topology (m_R=0.04N, m_F=100).

For the uni-directional ring topology, random classifier selection also provides the fastest reduction in classifier population size, reaching a minimum of 0.27N after 7K iterations, as illustrated in Figure 11. Classifier selection policies using numerosity or fitness bias are slower to converge and reach 0.46N and 0.36N after 19K iterations respectively.

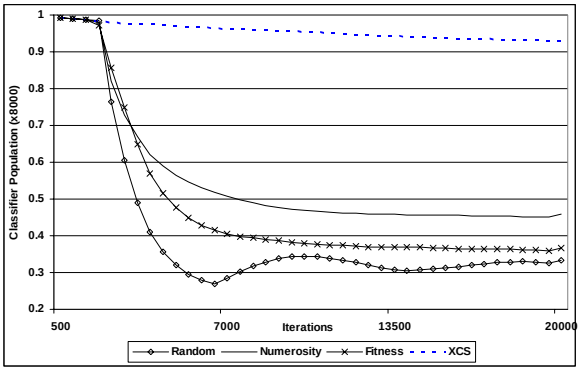

Figure 11: Classifier population for different selection policies (ψ_S) of the best performing **uni-directional ring** topology (m_R=0.04N, m_F=100).

VI. Conclusions and Future Work

This paper presented an investigation into exploiting the population-based nature of a learning classifier system by introducing a *migratory pressure* inspired by multi-deme parallel genetic algorithms. We found that distributed classifier migration between multiple XCS demes can be an effective method to improve classification accuracy, improve learning speed and reduce classifier population size when applied to the classification of multi-channel human electroencephalographic signals.

Migration policies with classifier selection/replacement based upon numerosity, fitness and random selection can provide significant improvements in classification accuracy and learning speed, according to a paired *t-test* at a 99.5% confidence level. Although, policies based on fitness bias can decline in performance if the migration rate or migration frequency are set too high. Learning speed and the rate of compaction in the classifier population is influenced by the degree-of-connectivity explicit in each of the three topologies examined, namely, the fully-connected, bi-directional ring and uni-directional ring topologies. Also, compaction of classifier population is determined exclusively by the selection policy.

Future research aims to utilise XCS classifiers evolved using distributed classifier migration in an ensemble machine format for the classification of multi-channel data derived from EEG signals of ten participants and multiple-tasks (≥4).

Acknowledgments

The authors wish to thank Professor Ashley Craig and Dr Yvonne Tran for support in conducting the EEG trials and Dr Matthew Gaston for establishing, maintaining and supporting the UTS Engineering Linux Computing Cluster Environment.

References

[1] S. Wilson, "Classifier Fitness Based on Accuracy," *Evolutionary Computation*, vol. 3, pp. 149-175, 1995.

[2] J. H. Holland, *Adaptation in natural and artificial systems*, 1st MIT Press ed. ed. Cambridge, Mass: MIT Press, 1992.

[3] L. Bull, *Applications of learning classifier systems*, vol. 1. Berlin: Springer, 2004.

[4] E. Bernado-Mansilla, X. Llora, and J. Garrell, "XCS and GALE: A Comparative Study of Two Learning Classifier Systems on Data Mining," presented at 4th International Workshop Learning Classifier Systems 2001, San Francisco, CA, USA, 2002.

[5] S. Wilson, "Mining Oblique Data with XCS," presented at Advances in Learning Classifer Systems: Third International Workshop on Learning Classifier Systems, Paris, France, 2000.

[6] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, vol. 1, 16 ed: Addison Wesley Longman, 1997.

[7] E. Alba and M. Tomassini, "Parallelism and Evolutionary Algorithms," *Evolutionary Computation, IEEE Transactions on*, vol. 6, pp. 443 - 462, 2002.

[8] Z. Konfrst, "Parallel Genetic Algorithms: Advances, Computing Trends, Applications and Perspectives," presented at Parallel and Distributed Processing Symposium, 2004. Proceedings. 18th International, Santa Fe, New Mexico, USA, 2004.

[9] M. Dorigo and U. Schnepf, " Genetics-based machine learning and behavior-based robotics: a new synthesis," *Systems, Man and Cybernetics, IEEE Transactions on* vol. 23, pp. 141-154, 1993.

[10] F. Seredynski and C. Z. Janikow, "Learning Nash equilibria by coevolving distributed classifier systems," presented at Evolutionary Computation, 1999. CEC 99. Proceedings of the 1999 Congress on Washington DC, 1999.

[11] Y. J. Cao, P. X. Zhang, S. J. Cheng, N. Ireson, L. Bull, and R. Miles, "An Evolutionary Intelligent Agents Approach to Traffic Signal Control," *International Journal of Knowledge-based Intelligent Engineering Systems*, vol. 5, pp. 279-289, 2001.

[12] L. Bull, M. Studley, A. Bagnall, and I. Whittley, "On the use of rule-sharing in learning classifier system ensembles," presented at The IEEE Congress on Evolutionary Computation, Edinburgh, UK, 2005.

[13] D. Wyatt and L. Bull, "An Investigation into Island Model Rule Migration for a Number of Mobile Autonomous Learning Classifier System Agents," presented at Genetic and Evolutionary Computation Conference, New York, 2002.

[14] M. Butz and S. Wilson, "An Algorithmic Description of XCS," presented at Advances in Learning Classifer Systems: Third Workshop on Learning Classifier Systems, Paris, 2000.

[15] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An introduction*, c1998 ed. Cambridge: MIT Press, 1998.

[16] M. Butz, T. Kovacs, P. L. Lanzi, and S. Wilson, "Toward a Theory of Generalization and Learning in XCS," *IEEE Transactions On Evolutionary Computation,*, vol. 8, pp. 28-46, 2004.

[17] J. Cahoon, W. Martin, and J. Lienig, "Parallel Genetic Algorithms Based on Punctuated Equilibria," *Handbook of Evolutionary Computation*, vol. C6.3, pp. 1-16, 1997.

[18] N. Eldridge and S. Gould "Punctuated equilibria: an alternative to phyletic gradualism," *Models in paleobiology*, pp. 82-115, 1972.

[19] E. Cantú-Paz, *Efficient and Accurate Parallel Genetic Algorithms*, 1 ed: Kluwer Academic Publishers, 2000.

[20] R. Tanese, "Distributed Genetic Algorithms," presented at Proceedings of the Third International Conference on Genetic Algorithms, George Mason University,Virginia,USA, 1989.

[21] B. T. Skinner, H. T. Nguyen, and D. K. Liu, "Performance Study of a Multi-Deme Parallel Genetic Algorithm with Adaptive Mutation," presented at The Second International Conference on Autonomous Robots and Agents, Palmerston North, New Zealand, 2004.

[22] Z. A. Keirn and J. J. Aunon, "A new mode of communication between man and his surroundings," *Biomedical Engineering, IEEE Transactions on* vol. 37, pp. 1209-1214, 1990.

[23] B. J. Fisch and R. Spehlmann, *Fisch and Spehlmann's EEG Primer: Basic Principles of Digital and Analog EEG*, vol. 1, 3rd edition (Dec, 1999) ed. New Orleans: Elsevier, 1999.

[24] C. W. Anderson, E. A. Stolz, and S. Shamsunder, "Multivariate autoregressive models for classification of spontaneous electroencephalographic signals during mental tasks," *Biomedical Engineering, IEEE Transactions on* vol. 45, pp. 277-286, 1998.

[25] S. L. Marple, *Digital Spectral Analysis with Applications*, vol. 1, 1 ed. Englewood Cliffs, N.J: Prentice-Hall, 1987.

[26] C. Fu and L. Davis, "A Modified Classifier System Compaction Algorithm," presented at Genetic and Evolutionary Computation Conference, New York, 2002.