Hybridisation of GA and PSO to Optimise N-tuples

M.A. Hannan Bin Azhar
Computing Department
Canterbury College
Canterbury, Kent ,UK
m.azhar@cant-col.ac.uk

Farzin Deravi and Keith Dimond
Department of Electronics
University of Kent
Canterbury, Kent, UK
f.deravi@kent.ac.uk; krdkent@waitrose.com

***Abstract*— Among numerous pattern recognition methods the neural network approach has been the subject of much research due to its ability to learn from a given collection of representative examples. This paper is concerned with the design of a Weightless Neural Network, which decomposes a given pattern into several sets of *n* points, termed n-tuples. Considerable research has shown that by optimising the input connection mapping of such n-tuple networks classification performance can be improved significantly. This paper investigates the hybridisation of Genetic Algorithm (GA) and Particle Swarm Optimisation (PSO) techniques in search of better connection maps to the N-tuples. Experiments were conducted to evaluate the proposed method by applying the trained classifier to recognise hand-printed digits from a widely used database compiled by U.S. National Institute of Standards and Technology (NIST).**

***Keywords*—N-tuples, GA, PSO, Pattern Recognition, WNN**

I. INTRODUCTION

Pattern recognition as a field is extremely diverse and has been applied in many areas such as science, engineering, business, medicine etc. The aim of pattern recognition is to classify objects into identifiable categories or classes after extracting features from the data. This data may be numerical, pictorial, textural, linguistic or any combination of these categories. The neural classification emulates the computational paradigm of the behaviour of neurons and their interconnections in human brain. Instead of recognizing a pattern by following a set of human-designed rules, as in the structural approaches, neural nets learn the underlying rules from a given collection of representative examples. Among neural network models, the weightless or n-tuple form of network [6] stands out due to its own advantages over a variety of pattern recognition algorithms [20]. Considerable research activity has focused on the n-tuple method, both regarding theoretical issues [20][14] as well as applications to real-world tasks [21]. Several applications of n-tuple-based networks to handwritten character recognition tasks have been reported. Recognition of handwritten characters by a computer has been a topic of extensive research for many years [17]. The character recognition research can be classified based upon two major criteria: 1) the data acquisition process (on-line or off-line) and 2) the text type (machine-printed or handwritten). The off-line handwritten character recognition has been selected as the application domain of the research presented in this paper, as it is relatively more complex compared to on-line and machine-printed recognition [2].

The n-tuple method decomposes a given pattern into several sets of *n* points, termed n-tuples. The classifier stores class-specific information about the training set in a number of look-up tables .The entries in each look-up table are addressed by sampling *n* specific data locations of the input that constitutes a 'feature' of the pattern. A pattern is classified as belonging to the class for which it has the most features in common with at least one training pattern of that class. The input connection mapping of the n-tuple classifier determines the sampling and defines the locations of the pattern matrix. There will be a vast number of possible connections for a matrix with the dimension like 32 by 32. The classification and generalization performance are highly dependent on these input mappings [5][13]. A random map is suitable for an un-optimised problem [3] as it samples the point throughout the pattern matrix. Considerable research shows that by optimising the connections classification performance can be improved significantly [5][13][10]. Stochastic search algorithms like Particle Swarm [15] and Genetic Algorithm [12] were used previously to find the optimal set of connections to the n-tuple network. This paper will investigate the hybridisation of these two popular algorithms to select better connections to the network and the performance of the optimised network will be measured in recognizing handwritten characters from the NIST [28] database. Due to computational extensive nature of the simulations and also the stochastic nature of the proposed algorithms, all presented results will be taken over several test runs. Remainder of the paper has been organized as follows:

Section 2, 3 and 4 will introduce the n-tuple network, PSO and GA respectively. The hybridisation technique used to combine GA and PSO will be explained in Section 5. Section 6 will presents the experimental results. Finally Section 7 will conclude the paper.

II. N-TUPLE NETWORK

An n-tuple classifier is a memory-based method. It is a type of a neural network with a structure that could be easily implemented using a RAM (Random Access Memory) [1]. The n-tuple method is more specifically known as a type of Weightless Neural Networks (WNN) or RAM networks (RAM-net). Unlike conventional networks, a weightless neural network uses explicit storage elements, rather than inter-element connection weights, to keep its state. In weightless approach there is no variable weight between the nodes rather neuron functions are stored in look-up tables. The learning algorithm is very simple, the patterns are presented to the

978-1-4244-2794-9/09/$25.00 ©2009 IEEE

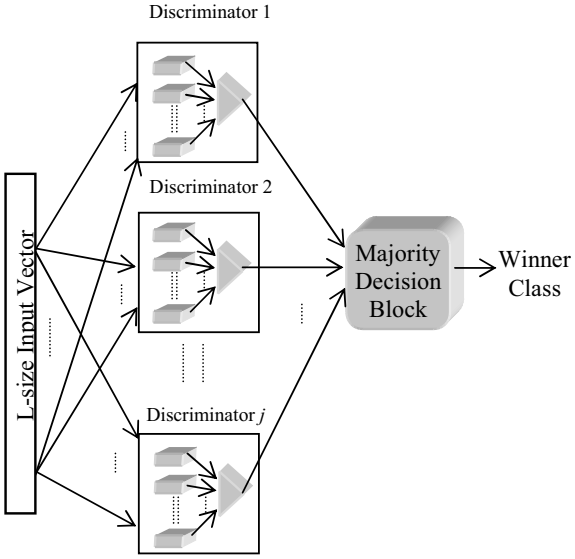

Figure 1. An n-tuple network

inputs of the network and then the patterns are stored in a certain way, which results in highly flexible and fast learning algorithms. Although the n-tuple classifier is not famously popular compared to some other methods, such as multilayer perceptrons [19], the networks based on the n-tuple method have two great strengths: they can be trained quickly and they can be implemented in conventional computers with greater ease when compared to other equation solving and minimising methods. The training of the basic classifier is a one-shot memorisation process. These advantages come at the cost of recognition robustness. It has been shown that the n-tuple method can result in quite reasonable recognition performance if used with care [20].

Fig. 1 shows an n-tuple network, which is built out of RAM nodes. The input address of the RAM unit is also known as "tuple". If the width of the address bus (also known as input connection map) is n bits then the tuple is termed as "n-tuple". The width of the address bus is also known as "tuple-size". Total number of tuples, denoted by R, is the number of tuples available to be optimised. R depends on the network's structure. A group of RAM nodes in a tree-like structure is called a discriminator. The discriminator achieves its goal by presenting to each neuron only a subset of the input pattern, and adding up the outputs of its RAM nodes. This sum can be seen as a measure of the recognition confidence of the discriminator. Therefore, when the discriminator sees a previously learned pattern, its integer output reaches the discriminator's maximum. For an input vector, of size L, the number of necessary RAM nodes R of connectivity n that should be used to cover all inputs of the input vector should satisfy: $R \times n \geq L$. L is known as the resolution of an image. A group of discriminators is used to distinguish a fixed number of classes. The number of classes, which need to be distinguished by a network, determines the number of discriminators needed in a network. The network shown in Fig. 1 can be used to distinguish a fixed number of classes. If it consists of 'j' discriminators, it can differentiate j classes. At the output of all discriminators there is a "decision block" where the winner class is chosen using some criteria such as the greatest sum, a threshold of the greatest sum, difference between sums etc. In greatest sum approach, the discriminator containing the greatest number of active RAM nodes is selected. Thus a pattern is 'recognized' as the one whose discriminator 'fired' the most, that is, the discriminator with the highest count of memorized tuples.

A. Motivation for Optimisation

The classification performance of the n-tuple classifier is highly dependent on the input bits probed [5][13][19]. The number of possible connections for a 32 by 32 binary pattern matrix is enormous. Let us consider an n-tuple classifier of 150 tuples with the tuple-size 8 bits. For an input binary image with a resolution of 32 bits by 32 bits the total number of available pixels will be 1024 bits. Now if the same pixel doesn't repeat in a tuple then the possible number of tuples that can be formed is found by the formulae of combination, $M = {}^{1024}C_8 \approx 2.91 \times 10^{19}$. If M tuples are divided into groups of 150 then total number of combinations will be $B = {}^{M}C_{150}$, which is a very large number. Therefore an exhaustive search for B mappings is impossible. The classification performance is a function of input mappings and it approximates to a normal distribution [3], where the majority of the mappings give average performance, but a small number of connection mappings give a relatively better performance. Bishop *et al.* [5] demonstrated the importance of choosing the sequence in which input is sampled to discriminate similar classes and applied a basic evolutionary technique to determine the sequence in which the input is sampled and the tuples formed. One sequence of input samples was used for the discriminators for most of the characters, but different sequences were used for the discriminators of characters which are too similar, such as c and e , i and l. Since Bishop's work genetic algorithms have been revisited by others more recently [10][4] to optimise the input connections of n-tuples. Other than GAs, algorithms like Tabu Search (TS) and Simulated Annealing (SA) were applied by Garcia and Souto [10] for this purpose. Optimisation by particle swarms was also reported for recognizing handwritten characters [4]. Noticeable improvement in performance by various algorithms [4][5][10][15] provided the motivation for our research to investigate the combination of two popular algorithms, GA and PSO, and to explore the effects of the hybridisation in the optimisation of n-tuples.

III. Particle Swarm on N-tuples

When particle swarm [15] is applied to n-tuple optimisation, the "tuples" of the n-tuple can be termed as "particles". Thus each particle corresponds to an input connection map of the n-tuple network. The size of an n-tuple network is defined by the total number of tuples it is built with. Total number of tuples, denoted by R, is the number of tuples available to be optimised by a particle swarm. R depends on the network's structure. The particle swarm technique makes use of a population of particles or input-maps (for n-tuples), where each particle has a position and a velocity. The PSO formulae, as shown in (1) and (2) define each particle as a potential solution in a multi-dimensional search space. The dimension of

the PSO corresponds to the bits or the tuple-size of each tuple. As the tuples are "*n*" bits, so the PSO will be *n* dimensional with the *i*-th particle represented as $X_i=(X_{i1},X_{i2},..X_{in})$. The PSO remembers the best position found by any particle which is known as global best, denoted by P_g. Additionally each particle remembers its own previously best found position designated as $P_i=(P_{i1},P_{i2},...P_{in})$ and its velocity $V_i= (V_{i1},V_{i2},...V_{in})$. Equation (1) and (2) will define the velocity and position of the *i*-th particle with *d*-th dimension [15].

$$V_{i,d}(t+1) = \omega \times V_{i,d}(t) + \psi 1 \times ran1 \times (P_{i,d} - X_{i,d}(t)) + \psi 2 \times ran2 \times (P_{gd} - X_{i,d}(t)) \quad (1)$$

$$X_{i,d}(t+1) = X_{i,d}(t) + V_{i,d}(t+1) \quad (2)$$

Search in PSO starts with the random initialisation of particles' positions and velocities within the allowed range defined by X_{max}, X_{min}, V_{max} and V_{min}. Usually V_{min} is the negative of V_{max}. Each particle keeps track of its own performance. At each iteration, the velocity of every dimension of a particle gets updated according to (1), where $V_{i,d}$, $P_{i,d}$ and P_{gd} constitute the particle's momentum. As this momentum is different for different dimension of a particle, this has effect to force the particle to change the trajectory in the search space towards the most promising areas. This momentum is essential, as it is the feature of PSO that allows particles to escape the local optima. In addition the *ran1* and *ran2* in (1) adds some random adjustments in velocities, which is essential to avoid the situation where the particle endlessly follows the exact same path. Constants $\psi 1$ and $\psi 2$ in (1) determine the relative influence of the "individuality" and "sociality" [15] traits of the particles and are usually both set the same to give each component equal weight as the individual and social learning rate.

A. Nearest Neighbour Interactions in PSO

Particle swarm optimisation like any other stochastic algorithm may prematurely converge [16]. Fast rate of information flow between particles can create similar particles resulting in less diversity in the system., which increases the possibility of being trapped in local optima [16]. Although, in general, PSO results good solutions, in high–dimensional spaces it might stumble on local minima. It may be argued that many of the particles are wasting computational effort in seeking to move in the same direction (towards the local optimum already discovered), whereas better results may be obtained if various particles explore other possible search directions. An alternative way to battle premature convergence in PSO is to consider neighbourhood interactions in PSO dynamics. A significant modification in particle dynamics is required to introduce the effects of multiple other particles in each particle. Peram *et al.* [18] proposed a method where each particle is moved towards other nearby particles with a more successful search history, instead of just the best position discovered so far. This is in addition to the terms in the original PSO update equation (1). The proposed algorithm is described as Fitness-Distance-Ratio [18] based PSO (FDR-PSO) which selects only one other particle when updating each velocity dimension and which is chosen to satisfy two following criteria:

- It must be near the particle being updated.
- It should have visited a position of higher fitness.

In FDR-PSO each velocity dimension is updated by selecting a particle that maximizes the ratio of the fitness difference to the one-dimensional distance. In other words, the *d*-th dimension of the *i*-th particle's velocity is updated using a particle called the P_{fdr}, with prior best position P_b, chosen to maximize the following ratio:

$$FDR\ (b,i,d) = \frac{Fitness(P_b) - Fitness(X_i)}{\left|P_{bd} - X_{i,d}\right|} \quad (3)$$

where $b \neq i$ and |...| denotes the absolute value. A new term was introduced into the velocity update equation with a new coefficient '$\psi 3$' and a new stochastic weight factor *'ran3'*. Like in the original PSO '*ran3*' can be uniformly distributed in {0,1} or can have a constant value of 1. Note that the FDR-PSO with $\psi 3 = 0$ is the same as the usual PSO algorithm described by [15]. The modified velocity equation for FDR-PSO is presented below:

$$V_{i,d}(t+1) = \omega \times V_{i,d}(t) + \psi 1 \times ran1 \times (P_{i,d} - X_{i,d}(t)) + \psi 2 \times ran2 \times (P_{gd} - X_{i,d}(t)) + \psi 3 \times ran3 \times (P_{fdr} - X_{i,d}(t)) \quad (4)$$

Like other stochastic search algorithms, PSO is also very much problem dependent. No single parameter setting exists which can be applied to all problems [16]. For example choosing a value for the inertia weight, ω in (1), could be critical. A large inertia weight favours exploration (global search), while a small inertia weight favours local search [22].

IV. Genetic Algorithm on N-tuples

The Genetic Algorithm introduced by Holland [12] is an adaptive search strategy based on a highly abstract model of biological evolution to find an optimal solution in a given problem space. It consists of a population of individuals, representing possible solutions, which evolve through interaction and adaptation. The individual is represented as a 'chromosome'. For an n-tuple network each chromosome corresponds to each input map. If each map points to "*n"* locations of the input matrix then the chromosome will be formed with these *n* location-values called "genes" [12]. While GA is applied to the n-tuple network, a population of individual input maps is initialised and then evolved from generation *t* to generation *t + 1* by repeated applications of fitness evaluation, selection, recombination and mutation. Any map that has a fitness value greater than a threshold was considered as a fit map or tuple. The fitter a member of a population the more likely it is to produce an offspring. The partner selection strategy followed for this research was as simple as choosing the top two, three or four input-maps with high fitness values as the parent chromosomes (Table I). This selection process is known as 'elitism', which requires that the current fittest member (or members) of the population is not deleted and survives to the next generation [25]. Genetically-inspired operators like crossover and mutation are used to introduce new individuals into the population [12]. For the experiments

presented in this paper while implementing crossover, genes to offspring was copied from parent's chromosomes. One of the *g*-th (*g = 1,..,n,* where *n* is the chromosome length used for the n-tuple network) genes of the parents was selected randomly to be the *g*-th gene of the child. This is known as uniform scanning crossover. Eiben *et al.* [9] introduced gene scanning as a reproduction mechanism that generalizes classical crossovers like uniform crossover and is applicable to an arbitrary number (two or more) of parents. Mutation was realised by replacing a gene of the new offspring with a randomly selected location-value from the input matrix. Large mutation rates with elitist selection turned out to be remarkably superior to that of traditional GA to obtain the global optimum solution effectively [23]. Mutation rates used by Garcia and Souto [10] in three different topologies of n-tuple networks were 95%, 60% and 75%. A variable mutation rate would theoretically make use of a high rate to speed up evolution until a certain "fitness level" is achieved, and then reduce mutation in order to increase the average fitness and produce a more balanced population [11]. Mutation rates used for GA in this work are listed in Table I. For one parent chromosome only mutation (asexual crossover) was used with very high mutation rate of 87.5%. Mutation rates were reduced from 87.5% to 50% as more parent chromosomes were found. Reduction in values of mutation rates also enables to find the global optima by performing local search using good solutions obtained so far [23].

TABLE I. GA OPERATORS

No. of fit tuples or connection maps	Parameters		
	No. of parent chromosome	*Crossover probability*	*Mutation rates*
1	1	1	87.5%
2	2	0.5	75%
3	3	0.33	62.5%
4	3	0.33	62.5%
5 or more	4	0.25	50%

V. HYBRIDISATION OF GA AND PSO

Hybridisation was achieved by switching between the GA and PSO at key stages of the search [29]. Individuals starting out as PSO particles, then switching to GA individuals, then back to particles and so on. The algorithm starts with the *Q* particles, where *Q* is the total number of particles (population size) in any iteration and they are initially distributed randomly over the whole pattern matrix. The unique strategy in the search algorithm was to reserve more tuples to a more critical class group [3]. The classes with high error rates were termed as critical classes. By using more class-specific tuples for a critical class the search algorithm would allow more time to be given to find features for a critical class. The target is to find *R* number of class-specific tuples in total. To calculate the number of class-specific tuples for a class at first the error rate of that class was divided by the total error rate and then the result of the division was multiplied with the total available tuples (*R*). The result of the multiplication was rounded to the nearest integer. No normalisation was used in the calculation of class-specific tuples. Providing more tuples to a class with a high error rate ensures that the extra care has been taken for a critical class group. Tuples engaged to a specific class best learn the features of that class and also learn some features for other classes to an extent [4].

Fitness of each particle is measured according to a reward and punishment based scheme [3], where a reward is associated with the correct recognition of the pattern and the penalties for misclassification and rejection. Based on fitness results each particle's best positional values are updated. '$P_{i,d}$' defines the location along the dimension *d* of the best positional value of each particle in the history. So '$P_{i,d}$'s represent best positions of all particles so far. Fitness of all $P_{i,d}$ particles are compared with a fitness threshold described by (5), where $\max_i(O_{ji}(t))$ is the score of the best-performed tuple among all the tuples in the current iteration.

$$Threshold = \max_i(O_{ji}(t)) \times (1 - exp(-\tau / t)) \quad (5)$$

Fitness threshold exponentially decays over iterations according to the above equation, where τ should be carefully chosen and varied throughout the search as a trade-off between the performance and the speed. A solution falling within the threshold distance of a specified value would be considered as an acceptable solution. It was found in the experiments that dropping the value of τ during unproductive iterations can substantially increase the speed of the search. In any iteration when PSO couldn't find any new solution then the search algorithm was switched to GA. The best-performed tuples for PSO were selected as parents to produce offspring in the next generation. GA and PSO used the same fitness and threshold functions (5) for evaluating maps. If in any generation GA couldn't find any new fit tuples then the algorithm was switched back to PSO. So GA and PSO together were searching for the required number of class-specific tuples (*R*). During search the new particles for the PSO were found by updating the particles' velocities and positions, according to (4) and (2). For GA genetic operators like mutation and crossover are applied to create new tuples. While switching form GA to PSO particles velocities were initialised with the value of the minimum velocity.

VI. EXPERIMENTAL RESULTS

Experiments were conducted to search for an optimal set of input connections maps by the hybrid GA and PSO algorithm. The network was built out of 150 tuples (*R*) with tuple-size 8. Due to *n*=8 the dimensionality of the hybrid PSO algorithm was 8. The available tuples were distributed among classes according to the difficulty associated in recognizing the patterns [3]. The task was to use hybrid GA-PSO algorithms to selectively choose tuples (as opposed to hill-climbing [3] and random selection [6]) that describe the classes better and later use these tuples to recognize a test data set. The NIST [28] database consists of handwritten digits (0,1…9) was used in the experiments. NIST released Special Database 3 (SD3) in February 1992 as the official training materials for the First Census Optical Character Recognition (OCR) Conference [28]. There are several partitions, denoted by *hsf_{0,1,2,3}*, in SD3 containing digits, upper and lowercase character images. The

writers of the SD3 partitions were Census Bureau field personnel stationed throughout the United States. A separate partition of images, denoted by *hsf_{4}*, was released as the testing materials for the OCR conference and it was named as Special Database 7 (SD7). Writers of SD7 were high school students in Bethesda, Maryland. In our experiment the partition *hsf_{0}* of SD3 was used for training and the partition *hsf_{4}* from the SD7 was used for testing. NIST recommends using images of *hsf_{4}* for testing as they are more difficult from other partitions and this ensures the heterogeneity between the training and testing set, a fact which is reflected in our results. There are 1000 images of each class in the training data set and half of those images were used to evaluate the solutions during the search. Each character used in the experiment was a binary image with the dimension 32 by 32. All digits were scaled into same dimension and centred.

To compare results, the N-tuple network was trained with various methods. The overall recognition rates, the average of all recognition rates of all classes, were found in the experiments. Initially an empirical value of 100 was chosen for τ (5) in all experiments, which provided enough time to run the search for exploring solutions. A high value of 100 for τ resulted in a slowly convergent system. It was found that dropping the value of τ during unproductive iterations could substantially increase the speed of search. In the experiment a 5% drop in the value of τ from 100 provided around four times quicker results with no noticeable difference in the performance. Selecting the right population size, *Q,* was crucial in terms of speed and performance. If the hybrid GA and PSO stops after a fixed number of iterations, a choice has to be made: either choosing a larger population or having more iterations [27]. For the experiment the termination criteria was not based on iterations. The goal was to find a fixed number of tuples whose fitness values were higher than a threshold value defined by (5). Thus the proposed algorithm can stop at any iteration as soon as it finds the required number of class-specific tuples. Now for an 8 dimensional problem (n=8) to have more possibility that the swarm can pass over the entire input vector of 1024 bits even in a few iterations a suitable population size would be 200. The cognitive ($\psi 1$) and social ($\psi 2$) parameters are not critical for PSO's convergence. Typically both $\psi 1$ and $\psi 2$ are set to a value of 2 [8], although assigning different values sometimes leads to improved performance [24]. In our experiment both values were set to 1. Setting the values to 2 made no noticeable difference. The value of $\psi 3$ in (4) was set to 2 as FDR-PSO ($\psi 1 = \psi 2 = 1, \psi 3 = 2$) outperformed original PSO and several other variations of PSO in different tested benchmark problems [18]. Experimental settings for GA were described in Section 4. For PSO the inertia parameter, ω, can be decremented with the number of iterations from 0.7 to 0.4 as in [16]. In our approach as the PSO can terminate at any iteration, the inertia parameter was chosen to be a constant value of 0.7. The value of *Vmax* proved to be crucial, because large values could result in particles moving past good solutions and create excessive crowding or bumping around the best fit particle. In the experiments the *Vmax* of 2 was observed to be a good value to fine tune the entire search space with 200 particles.

TABLE II. RESULTS OF T-TEST FOR THE HYBRID GA-PSO

2nd Algorithm	Statistical Values	
	t-value	*p-value*
Random Selection [6]	12.30	1.00
Hill-climbing type [3]	3.51	0.99
Genetic Algorithm	2.19	0.97
PSO ($\psi 1$=1, $\psi 2$=1, V_{max}=2)	-1.83	0.04

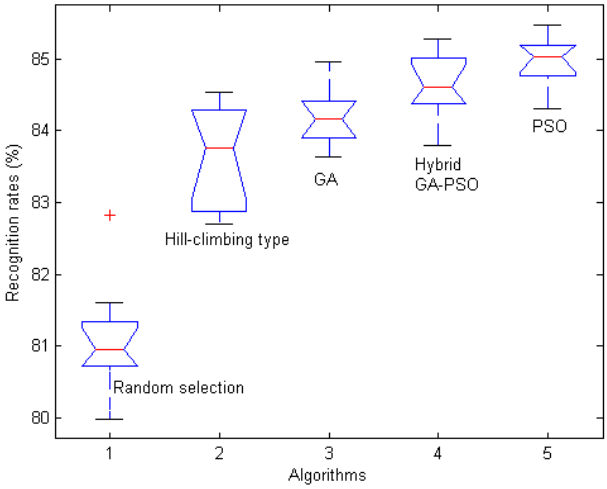

Figure 2. Box plot of training algorithms

The hybrid GA and PSO based search exhibited 3.68% higher recognition rate when compared to a conventionally trained n-tuple network [6]. The improvement by the hybrid algorithm over a hill-climbing type approach [3] and GA were 0.93% and 0.44% respectively. Statistical significance of the results was analysed by the student's t-test [7]. The hybrid GA-PSO was compared against other algorithms. The null hypothesis for the test was "average recognition rate by the hybrid GA-PSO is higher than a second algorithm". For 10 trials of each algorithm the degrees of freedom [7] was 18. In the test, a t-value was calculated from the experimental results and compared against the theoretical t-values at different confidence levels [7] and 18 degrees of freedom. Theoretical t-values for 90%, 95%, 99% and 99.9% confidence level and 18 degrees of freedom were 1.73, 2.10, 2.88, and 3.92. The t-values found in the experiment against the null hypothesis are presented in Table II. Results show that the increases in recognition rates by the hybrid GA-PSO algorithm over conventional random selection and hill-climbing type approach [3] are statistically "very highly significant" [7] because the experimental t-values for these cases were greater than the theoretical t-value (3.92) at 18 degrees of freedom. The observed t-value against the original GA was 2.19 (greater than 2.10) and this implies that the improved results by the hybrid algorithm over the GA alone were statistically significant at 95% confidence level. The t-value against the original PSO is negative in the table, which means that the null hypothesis should be rejected and rather the alternative is true. The p-values in the table indicate the probability of observing the

result by chance given that the null hypothesis is true. Small values of probabilities cast doubt on the validity of the null hypothesis.

Fig. 2 displays the side-by-side box plots [26] of the results found in the experiment. Each box in the figure was constructed with the recognition rates of ten trials. The box plot conveys location and variation information in data sets, particularly for detecting and illustrating location and variation changes between different data groups of algorithms. The notches in Fig. 2 are drawn about the median so that notches that don't overlap represent significant differences between medians (with 95% confidence). It can be noted that the recognition rates by conventional random approach were clustered at the bottom of the plot whereas data for PSO were clustered at the very top. The median of recognition rates for the hybrid algorithm was just below 85%, for PSO at 85%, for the hill-climbing type [3] was just below 84%, for GA was just above 84% and for randomly selected approach was near 81%. Box plots also show if there are unusual observations (outliers) in the dataset. Outliers are individually identified with a plus symbol in Fig. 2. where a single unusual observation for the random selection was observed.

VII. CONCLUSION

This paper described how GA and PSO could be combined and applied to optimise an n-tuple network for recognizing binary handwritten characters from the NIST database. Results found by the hybrid method showed that the improvement over original GA, Hill climber and conventional randomly selected approach [6] was statistically significant. But the hybridisation of GA and PSO couldn't outperform the performance of the original PSO approach. It was clear that by using the hybrid approach the connectivity pattern is rearranged in such a way that the more relevant features of the input patterns in the training set, for each class, tend to have more connections to the nodes in the group of tuples specific to that class, leading to an improvement of the performance of n-tuple classifiers.

REFERENCES

[1] Aleksander, I., Thomas, W. V., and Bowden, P. A. (1984). WISARD: a radical step forward in image recognition, Sensor Review, 4(3), pp. 120-124.

[2] Arıca, N. and Yarman-Vural, F.T. (2001). An Overview of Character Recognition Focused on Off-line Handwriting. IEEE Transactions on Systems, Man, and Cybernetics, 31, pp.216-233.

[3] Azhar, M. A. H. B. and Dimond, K.R. (2004). A Stochastic Search Algorithm to Optimize an N-tuple Classifier by Selecting Its Inputs, International Conference on Image Analysis and Recognition, Porto, Portugal, Springer-Verlag, September 29 - October 1, pp. 556-563.

[4] Azhar, M.A.H.B., Deravi, F. and Dimond, K.R.(2008). Criticality Dispersion in Swarms to Optimize N-tuples, Genetic and Evolutionary Computation Conference (GECCO 2008), July 12-16, Atlanta, Georgia, USA, pp.1-8.

[5] Bishop, J.M., Crowe, A.A., Minchinton, P.R. and Mitchell R.J. (1990). Evolutionary Learning to Optimise Mapping in n-Tuple Networks, IEE Colloquium on "Machine Learning", 28 June, Digest 1990/117.

[6] Bledsoe, W. and Browning, I. (1959). Pattern recognition and reading by machine, Proceedings of Eastern Joint Computer Conference, Birmingham, pp.225-232.

[7] Deacon, J. (2006). The Really Easy Statistics Site, Biology Teaching Organisation, University of Edinburgh, http://www.biology.ed.ac.uk/research/groups/jdeacon/statistics/tress1.html

[8] Eberhart, R., Simpson, P. K., and Dobbins, R. W. (1996). Computational Intelligence PC tools, 1st ed., Academic press professional, Boston, MA.

[9] Eiben, A.E., Kemenade, C.H.M.V. and Kok, J.N. (1995). Orgy in the computer: Multi-parent reproduction in genetic algorithms. In F. Moran, A. Moreno, J.J. Merelo, and P. Chacon, editors, Proceedings of the 3rd European Conference on Artificial Life, number 929 in LNAI, Springer-Verlag, pp. 934-945.

[10] Garcia, L.A.C. and Souto, M. C. P. (2004). Global Optimisation Methods for Choosing the Connectivity Pattern of N-tuple Classifiers, Proc. of the IEEE International Joint Conference on Neural Networks, Budapeste, pp. 2263-2266.

[11] Harvey, I. (1993). Evolutionary Robotics and SAGA: the Case for Hill Crawling and Tournament Selection. In Artificial Life III, Langton, C. (Ed.), Publisher: Addison-Wesley, pp. 299-326.

[12] Holland, J. H. (1975). Adaptation in Natural and Artificial Systems. Ann Arbor, Mich., University of Michigan Press.

[13] Jørgensen, T. M., Christensen, S. S. and Liisberg, C. (1995). Crossvalidation and information measures for RAM based neural networks, Proc. of the Weightless Neural Networks Workshop, University of Kent at Canterbury, UK, pp. 87-92.

[14] Jørgensen, T.M., Linneberg, C. (1999). Theoretical analysis and improved decision criteria for the n-tuple classifier, IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.21, pp.336-347.

[15] Kennedy, J., and Eberhart, R. C. (1995). Particle swarm optimisation, Proc. of the 1995 IEEE Int. Conf. on Neural Networks (Perth, Australia).

[16] Løvbjerg, M. and Krink, T. (2002). Extending particle swarm optimisers with self-organized criticality, Proc. of the IEEE Congress on Evolutionary Computation, Honolulu, Hawaii USA.

[17] Mori, S., Suen, C.Y. and Yamamoto, K.(1992). Historical review of OCR research and development. Proceedings of the IEEE, July, Special Issue on Optical Character Recognition, 80(7), pp.1029-1058.

[18] Peram, T., Veeramachaneni, K. and Mohan, C.K. (2003). Fitness-Distance-Ratio based Particle Swarm Optimization. In Proceedings of the IEEE Swarm Intelligence Symposium, IEEE Press, pp. 174-181.

[19] Picton, P. (2000). Neural Networks (Grassroots Series), 2nd Edition, Palgrave Publishers Ltd.

[20] Rohwer, R. and Morciniec, M. (1998). The Theoretical and Experimental Status of the n-tuple Classifier, Neural Networks 11(1): pp. 1-14.

[21] Rohwer, R. and Cressy, D. (1989). Phoneme classification by boolean networks, Proceedings of the European Conference on Speech Communication and Technology, pp. 557-560.

[22] Shi, Y, and Eberhart, R. (1998). Parameter selection in particle swarm optimization, in Evolutionary Programming VII, pp. 591--600, Springer, Lecture Notes in Computer Science 1447.

[23] Shimodaira,H. (1996). A new genetic algorithm using large mutation rates and population-elitist selection (GALME), Proceedings of Eighth IEEE International Conference on Tools with Artificial Intelligence, 16-19 Nov. pp. 25-32.

[24] Suganthan, P. N. (1999). Particle swarm optimiser with neighbourhood operator. Proc. of the IEEE Congress on Evolutionary Computation, IEEE Service Center, Piscataway, NJ, pp. 1958-1962.

[25] Tomassini, M. (1995). A Survey of Genetic Algorithms, Annual Reviews of Computational Physics, World Scientific,Vol.III, pp.87-118.

[26] Tukey, J.W. (1977), Exploratory Data Analysis, Reading, MA: Addison-Wesley.

[27] Van den Bergh, F. and Engelbrecht, A. P. (2001). Effects of swarm size on cooperative particle swarm optimisers. Proc. of the Genetic and Evolutionary Computation Conference, San Francisco, USA.

[28] Wilkinson, R., Geist, J., Janet, S., Grother, P., Burges, C., Creecy, R., Hammond, B., Hull, J., Larsen, N., Vogl, T., and Wilson, C. (1992). The first census optical character recognition systems conference, Technical Report NISTIR 4912, National Institute of Standards and Technology (NIST), Gaithersburg, USA.

[29] Yang, B., Chen, Y., Zhao, Z. (2007). A Hybrid Evolutionary Algorithm by Combination of PSO and GA for Unconstrained and Constrained Optimization Problems, IEEE International Conference on Control and Automation (ICCA 2007). May 30 2007-June 1 2007, pp. 166 – 170.