Saw-tooth Algorithm Guided by the Variance of Best Individual Distributions for Designing Evolutionary Neural Networks

Pedro Antonio Gutiérrez[1], César Hervás[1], and Manuel Lozano[2]

[1] Dept. of Computer Science and Numerical Analysis, University of Córdoba
Campus de Rabanales, building C2, 14004 - Córdoba, Spain
zamarck@yahoo.es,chervas@uco.es
[2] Dept. of Computer Science and Artificial Intelligence, University of Granada
E.T.S. Ingeniería Informática, 18071 - Granada, Spain
lozano@decsai.ugr.es

Abstract. This paper proposes a diversity generating mechanism for an evolutionary algorithm that determines the basic structure of Multilayer Perceptron (MLP) classifiers and simultaneously estimates the coefficients of the models. We apply a modified version of a recently proposed diversity enhancement mechanism [1], that uses a variable population size and periodic partial reinitializations of the population in the form of a saw-tooth function. Our improvement on this standard scheme consists of guiding saw-tooth reinitializations by considering the variance of the best individuals in the population, performing the population restart when the difference of variance between two consecutive generations is lower than a percentage of the previous variance. The empirical results over six benchmark datasets show that the proposed mechanism outperforms the standard saw-tooth algorithm. Moreover, results are very promising in terms of classification accuracy, yielding a state-of-the-art performance.

Key words: Evolutionary algorithm, population reinitializations, saw-tooth algorithm, neural networks

1 Introduction

Evolutionary Algorithms (EAs) are search algorithms based on the concept of natural selection. Among them, Evolutionary Programming (EP), originally proposed by L.J. Fogel [2], is a sthocastic optimization. One of the main characteristic of an EP algorithm is the absence of a direct codification, only working with their representation. Evolutionary artificial neural networks have been a key research area in the past decade, providing an interesting platform for optimizing both the weights and architecture of the network simultaneously. The problem of finding a suitable architecture and the corresponding weights of the network is a very complex task and this difficulty justifies the use of an evolutionary algorithm to design the structure and training of the weights (for a very interesting review on this subject the reader can consult [3]).

Population size is one of the most important parameters affecting the robustness and computational efficiency of EAs. Small population sizes may result in premature convergence to nonoptimal solutions, whereas large population sizes require a considerable increase in computational effort. Although some authors try to estimate an optimal population size regarding the complexity of the problem [4], *variable population size* is a more interesting alternative [5], improving the EA capability of obtaining better solutions and reducing sensitivity in the election of different parameters. On the other hand, several methods using a constant census have been proposed in the literature that attempt to increase the diversity of the population and avoid premature convergence, including, among others, alternative mutation strategies [6], the well known Eshelman's CHC algorithm [7] or the micro Genetic Algorithm or μ-GA, suggested by Goldberg [8]. Both CHC and μ-GA algorithms are based on *periodical reinitializations* of population when the diversity drops below a threshold.

One methodology combining the effects of variable population size with periodical reinitializations is that proposed by Koumousis and Katsaras [1], which follows a saw-tooth scheme with a specific amplitude and period of variation. In each period, the population size decreases linearly and, at the beginning of the next period, randomly generated individuals are appended to the population. One of the major drawbacks of this approach is that both the amplitude and period of saw-teeth must be specified a priori.

In this paper, we propose an enhanced version of this saw-tooth EA, in which we guide the period of each saw-tooth according to the variance of the best individual fitnesses of the population. The underlying idea is that reinitializations should be performed when the best individuals have converged and no further optimization is being achieved, so randomly generated individuals can help the algorithm to explore new regions in the search space. We apply this proposed scheme over a previously implemented EP Algorithm [9], that evolves the weights and the structure of MLP Neural Network classifiers. In order to test the performance and suitability of our methodology, a comparative study using the original constant population EP algorithm, the standard saw-tooth EP algorithm and the variance guided saw-tooth EP algorithm has been applied to six datasets taken from the UCI repository [10]. The rest of the paper is organized as follows: Section 2 is dedicated to a description of the selected EP algorithm; Section 3 describes the implementation of the standard saw-tooth scheme and its application to the EP algorithm; the aim of Section 4 is to describe the proposed variance guided version of the saw-tooth scheme; Section 5 explains the experiments carried out; and finally, Section 6 shows the conclusions of our work.

2 Evolutionary Programming Algorithm

In this section, we present the EP algorithm used to estimate the parameters and the structure of the MLP neural networks models. The main objective of the algorithm is to design a neural network with optimal structure and weights for each classification problem tackled. The search begins with an initial population

of MLPs, to which a population-update algorithm is applied in each iteration. The algorithm shares many characteristics and properties with other previous algorithms like [11] and [12]. Individuals are subjected to the operations of replication and mutation. Crossover is not used due to its potential disadvantages in evolving artificial neural networks. With these features the algorithm falls into the class of EP.

We consider standard feed forward MLP neural networks with one hidden layer and we interpret the outputs of the neurons on the output layer from a probability point of view which considers the softmax activation function given by the following expression:

$$g_l(\mathbf{x}, \boldsymbol{\theta}_l) = \frac{\exp f_l(\mathbf{x}, \boldsymbol{\theta}_l)}{\sum_{j=1}^{J} \exp f_j(\mathbf{x}, \boldsymbol{\theta}_j)} \tag{1}$$

where J is the number of classes in the problem, $f_l(\mathbf{x}, \boldsymbol{\theta}_l)$ the output of the neuron j for pattern $\mathbf{x}$ and $g_l(\mathbf{x}, \boldsymbol{\theta}_l)$ the probability that pattern $\mathbf{x}$ has of belonging to class j. Considering this expression, the classification rule $C(\mathbf{x})$ of the MLP is the following:

$$C(\mathbf{x}) = \widehat{l}, \quad \text{where } \widehat{l} = \arg\max_l g_l(\mathbf{x}, \boldsymbol{\theta}_l), \text{ for } l = 1, 2, ...J \tag{2}$$

We define the Correctly Classified Rate by $CCR = (1/N) \sum_{n=1}^{N} (I(C(\mathbf{x}_n) = \mathbf{y}_n)$, where $I(\cdot)$ is the zero-one loss function. A good classifier tries to achieve the highest possible CCR in a given problem. The function used in the EP algorithm for obtaining the fitness of individuals is a strictly decreasing transformation of the cross entropy error function and is given by the following expression:

$$A(g) = \frac{1}{1 + l(\boldsymbol{\theta})} \tag{3}$$

where $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, ..., \boldsymbol{\theta}_J)$ and $l(\boldsymbol{\theta})$ is the cross entropy error function of the model and is obtained as:

$$l(\boldsymbol{\theta}) = -\frac{1}{N} \sum_{n=1}^{N} \sum_{j=1}^{J} y_n^{(l)} \log g_l(\mathbf{x}_n, \boldsymbol{\theta}_l) = \tag{4}$$

$$= \frac{1}{N} \sum_{n=1}^{N} \left[-\sum_{j=1}^{J} y_n^{(l)} f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) + \log \sum_{j=1}^{J} \exp f_l(\mathbf{x}_n, \boldsymbol{\theta}_l) \right]$$

The general framework of the EP algorithm is the following:

1. Generate a random population of size N_{P}.
2. Repeat until a maximum number of generations G is reached:
   (a) Apply parametric mutation to the best 10% of individuals. Apply structural mutation to the remaining 90% of individuals.
   (b) Calculate the fitness of every individual in the population.

(c) Add best fitness individual and best CCR individual of the last generation (double elitist algorithm).
(d) Rank the individuals with respect to their fitness.
(e) Best 10% of population individuals are replicated and substitute the worst 10% of individuals.

3. Select the best CCR individual and the best fitness individual in the final population and consider both as possible solutions.

As mentioned previously, fitness is a decreasing transformation of cross-entropy error. In general, the relationship between CCR and cross-entropy error strongly depends on the data base structure. Hence, regarding experimental results, using cross-entropy elitism is more suitable for some databases to result in a higher generalization accuracy, but using CCR elitism can be more appropriate for some other databases. For this reason, the EP algorithm returns both the best CCR and the best fitness individuals as solutions, the best approach for each problem being difficult to ascertain a priori. Parametric mutation is accomplished for each coefficient of the model with Gaussian noise, applying a standard simulated annealing process for accepting or rejecting modifications. On the other hand, we use five different structural mutations, similar to the mutations in the GNARL model [11]. The severity of mutations depends on the temperature $T(g)$ of the neural network model, defined by $T(g) = 1 - A(g)$ $0 \leq T(g) \leq 1$.

In order to define the topology of the neural networks, we consider three parameters: m, M_{E} and M_{I} . They correspond, respectively, to the minimum and maximum number of hidden nodes in the whole evolutionary process and the maximum number of hidden nodes in the initialization process. In order to obtain an initial population formed by models simpler than the most complex model possible, parameters must fulfil the condition $m \leq M_{\mathrm{E}} \leq M_{\mathrm{I}}$. More details about the EP algorithm can be consulted in [9], [13] and [14]. The EP algorithm was implemented using the Evolutionary Computation framework JCLEC [15] (`http://jclec.sourceforge.net`) and is available in the non-commercial java tool named KEEL (`http://www.keel.es`).

3 Standard Saw-tooth Evolutionary Programming Algorithm

The scheme proposed in [1] is briefly summarized in this section. The Standard Saw-tooth algorithm utilizes a variable population size following a periodic scheme where a mean population size $\overline{n}$, an amplitude D and a period of variation T define the saw-tooth shape. Thus, at a specific generation t, the population size $n(t)$ is determined as:

$$n(t) = \mathrm{int}\left\{\overline{n} + D - \frac{2D}{T-1}\left[t - T \cdot \mathrm{int}\left(\frac{t-1}{T}\right) - 1\right]\right\} \tag{5}$$

where $int(\cdot)$ is the floor function. Therefore, $n(1) = \overline{n} + D$, $n(T) = \overline{n} - D$, $n(T+1) = \overline{n} + D$, etc.The selection of the $\overline{n}$, T and D parameters affects the

performance of the algorithm. In the paper cited, the optimum values of the T and D parameters of the saw-tooth are obtained experimentally, the optimum normalized amplitude $D/\overline{n}$ being from 0.9 to 0.96 and the optimum normalized period ranging from $T/\overline{n} = 0.5$ to 0.7 for multimodal optimization problems and a standard real coded Genetic Algorithm.

In this paper, we have adapted this scheme, defining what we call Standard Saw-tooth Evolutionary Programming (SSEP). Instead of using 5, population size in each generation is calculated from the last generation population size as:

$$n(t) = n(t-1) - N \tag{6}$$

N being the saw-tooth slope and $n(0) = N_{\mathrm{P}} = 1000$ being the number of individuals in the initial population. The value of D is calculated according to the guidelines previously described. For the parameter T, we estimate its value from the maximum number of generations G and a parameter r that defines the maximum number of restarts, $T = \mathrm{int}(G/r)$. The use of this new parameter r is justified, since a better control of the increase in the diversity of the algorithm is achieved by defining the maximum number of restarts (that is, the maximum number of saw-tooth oscillations) than by defining the amplitude T of each saw-tooth. With these two parameters, the saw-tooth slope is obtained as $N = T/2D$. Figure 3(a) is an example of a standard saw-tooth scheme for the Vehicle dataset, including all the parameters mentioned.

4 Saw-tooth Evolutionary Programming Algorithm Guided by the Variance of Best Individual Distributions

The previously defined scheme has been enhanced considering the variance of the best individual distributions, forcing the beginning of a new saw-tooth oscillation when the best individuals have converged and the performance of the algorithm decreases. Consequently, if the variance of best individual fitnesses has significantly decreased, the population size falls. Otherwise a restart is performed, new individuals being added to the current population. This methodology has been called Guided Saw-tooth Evolutionary Programming (GSEP).

The condition that must be fulfilled in order to force a restart of the population is that the difference of variance of the best individual fitnesses between two consecutive generations decreases by a specific percentage (λ) with respect to the previous generation variance, previously establishing the most appropriate distribution to represent the best individual fitnesses. As a first approximation, the best individual fitness values have been characterized using two distinct distributions, uniform distribution and normal distribution. Considering the uniform distribution, the variance is estimated by:

$$\widehat{\sigma}^2 = \frac{1}{12}\left(f_{max} - f_{min}\right)^2 \tag{7}$$

f_{max} and f_{min} being, respectively, the maximum and minimum fitness value of the best individual set B, and, considering the normal distribution, the variance

is estimated as:

$$\widehat{\sigma}^2 = \frac{1}{|B|} \sum_{i=1}^{|B|} \left(f_i - \overline{f}\right)^2 \tag{8}$$

f_i being the fitness of the individual i of the set of best individuals B and $\overline{f}$ the mean fitness value of all individuals in B.

Finally, a minimum number of individuals m_{n} in the population is defined based on the amplitude D of saw-teeth, $m_{\mathrm{n}} = 1000 - 2D$. Restart will also be forced if the population size achieves this minimum. The number of the best individuals whose variance is studied in order to guide the algorithm ($|B|$) is constant during the evolution process and has been fixed to m_{n}. As a result, the number of individuals in each generation is obtained as:

- If $[(\widehat{\sigma}_t^2 < \widehat{\sigma}_{t-1}^2)$ and $(\widehat{\sigma}_{t-1}^2 - \widehat{\sigma}_t^2) < \lambda \widehat{\sigma}_{t-1}^2]$ or $[(n(t-1) - N) \leq m_{\mathrm{n}}]$,
  Perform a restart, generating new individuals until filling initial population size, $n(t) = 1000$.
- Otherwise,
  Decrease population size, $n(t) = n(t-1) - N$, or keep population size to the minimum value if the maximum number of restarts has been achieved, $n(t) = m_{\mathrm{n}}$.

5 Experiments

The proposed GSEP algorithm is applied to six datasets taken from the UCI repository, [10], to test its overall performance as compared to EP and SSEP algorithms. The selected databases include both binary classification problems (German, Heart-statlog, Ionosphere) and multiclass classification problems (Balance, Glass and Vehicle). The experimental design was conducted using a holdout cross-validation procedure, with 30 runs and 70% of instances for the training set and 30% for the generalization set.

Regarding the configuration of the different experiments, the specific parameters for each database have been fine tuned by a trial and error process and summarized in Table 5. Considering an estimated mean population size $\overline{n}$ of 500 and the guidelines proposed in [1], D parameter value should range between 480 and 490. Consequently, we have considered two different values of the m_{n} parameter , 20 and 40, for SSEP and GSEP algorithms. The period T of saw-teeth is determined by the maximum number of restarts r and maximum number of generations G, both specified in Table 5. For GSEP experiments, r value has been increased in one additional restart, in order to avoid an excessive number of generations with a minimum population size.

In order to graphically evaluate the performance of the proposed methodology, the mean and the maximum fitness of the population have been plotted versus the number of generation in Fig. 1 and Fig 2. The depicted values correspond to the average over 30 runs of Vehicle experiment, using both SSEP and GSEP schemes. The dynamic guided Saw-tooth algorithm achieves an overall

Table 1. Non-common parameters values.

Dataset	Balance	German	Glass	Heart	Ionos	Vehicle
G	300	150	200	100	350	1500
$[m, M_{\mathrm{E}}, M_{\mathrm{I}}]$	[3,4,5]	[2,3,4]	[7,8,9]	[1,1,2]	[3,4,5]	[6,7,8]
r	5	1	2	2	2	4
λ	0.005	0.0075	0.0075	0.0075	0.005	0.001

better performance (0.6% better final fitness). At the final generations, mean fitness is higher using the GSEP scheme than using the SSEP and, throughout the evolution process, the guided application of the saw-tooth reinitialization leads to better fitness peaks.

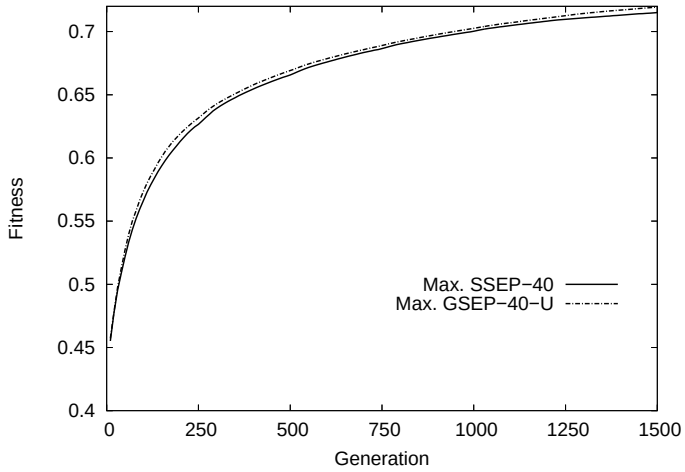


Fig. 1. Maximum fitness average for 30 runs using SSEP-40 and GSEP-40-U algorithms.

Table 2 shows the mean value and standard deviation of results of the Correctly Classified Rate (CCR) for training and generalization sets and the corresponding number of net connections in 30 runs of the experiment, for the EP algorithm and the best performing variant of SSEP and GSEP algorithms. As the algorithms returned both the best CCR and cross-entropy individuals, Table 2 only includes the most efficient approach in each given problem. The algorithm that yields better generalization results is represented in bold print. The saw-tooth algorithms (SSEP and GSEP) can be observed to outperform the original EP algorithm in four out of the six problems analyzed. Moreover, the models obtained have a lower number of connections in five out of six databases, which suggests that including periodical reinitializations allows the evolutionary

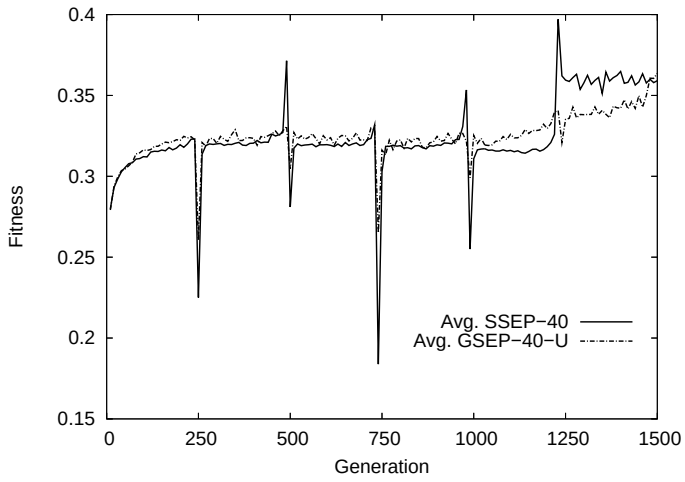


Fig. 2. Mean fitness average for 30 runs using SSEP-40 and GSEP-40-U algorithms.

process to explore simpler structures that offer better generalization results. Finally, the GSEP scheme proposed yielded better results than SSEP in all the experiments evaluated. On the other hand, the efficiency of all the algorithms has been estimated by obtaining the mean size of the population throughout all the generations (also presented in Table 2). The values represented correspond to the average and standard deviation of this mean population size during the 30 runs of the different experiments. Both EP and SSEP algorithms have a fix population scheme, their standard deviation being equal to 0. In Fig. 3 average population size over the 30 runs has been plotted for SSEP and GSEP, together with two example runs for GSEP. In general, saw-tooth schemes (SSEP and GSEP) result in a lower computational cost. Moreover, for Glass, Heartlog and Ionosphere experiments, GSEP obtains a lower population size than SSEP.

6 Conclusions

The application of the standard saw-tooth scheme has proved viable in the designing of Evolutionary MLP Neural Networks, providing diversity to the evolutionary process and resulting in a more efficient and accurate algorithm. The proposed GSEP algorithm improves the performance of the SSEP algorithm and could be easily introduced into any existing EA. This proposed scheme has been presented as an enhanced version of the standard saw-tooth scheme, performing the population restart when the difference of variance between two generations is lower than a percentage of previous variance. The evaluation of the algorithm for a wide, thought not exhaustive, range of problems examined showed results that are comparable to those of other classification techniques found in machine learning literature [16]. In this way, it can be affirmed that distribution of the best individual fitnesses can be considered a suitable tool for guiding restarts

Table 2. Best statistical (mean and standard deviation, SD) CCR results in training and testing sets, number of connections and population size throughout the 30 runs for: Evolutionary Programming with 1000 individuals (EP), Standard Saw-tooth EP (SSEP) and Guided Saw-tooth EP (GSEP), with 20 and 40 individuals as minimum population size and considering uniform (U) and normal (N) distributions.

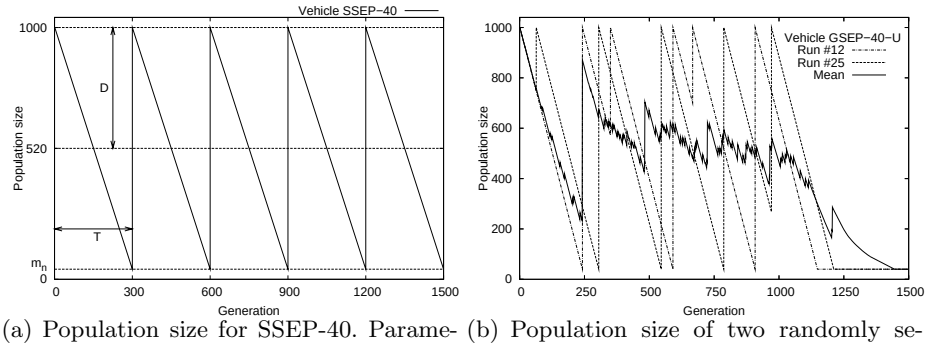
Dataset (Elitism)	Method	CCR_T Mean ± SD	CCR_G Mean ± SD	#connect Mean ± SD	Pop. Size Mean ± SD
Balance	EP	95.47 ± 1.50	**94.10** ± 1.64	30.73 ± 2.16	1000.00
(CCR)	SSEP-40	94.48 ± 1.35	93.18 ± 1.86	**30.43** ± 2.46	524.00
	GSEP-40-U	94.31 ± 1.56	93.65 ± 1.56	30.60 ± 2.25	527.05 ± 45.77
German	EP	78.66 ± 1.48	73.05 ± 1.58	95.20 ± 21.90	1000.00
(Fitness)	SSEP-40	78.48 ± 1.06	72.87 ± 1.77	88.17 ± 21.08	520.50
	GSEP-40-N	78.20 ± 1.44	**73.43** ± 2.22	**85.80** ± 19.56	553.86 ± 46.08
Glass	EP	72.61 ± 3.00	67.23 ± 4.10	77.57 ± 8.01	1000.00
(CCR)	SSEP-40	71.76 ± 2.96	67.99 ± 4.04	**73.63** ± 7.41	544.48
	GSEP-40-N	70.99 ± 2.96	**68.93** ± 4.53	76.37 ± 7.26	389.69 ± 18.70
Heart	EP	86.06 ± 0.90	**86.91** ± 2.06	17.27 ± 2.03	1000.00
(Fitness)	SSEP-20	86.34 ± 1.22	85.44 ± 1.65	**16.47** ± 2.37	520.00
	GSEP-20-N	86.50 ± 0.96	86.37 ± 1.85	17.23 ± 2.57	366.91 ± 35.53
Ionos.	EP	98.27 ± 0.93	92.53 ± 1.80	**73.77** ± 9.91	1000.00
(Fitness)	SSEP-20	98.43 ± 0.83	92.41 ± 2.08	80.83 ± 11.64	510
	GSEP-20-U	98.43 ± 0.75	**92.61** ± 1.80	77.47 ± 12.83	392.20 ± 18.73
Vehicle	EP	79.94 ± 1.45	78.33 ± 2.74	96.47 ± 6.62	1000.00
(Fitness)	SSEP-20	80.12 ± 1.86	78.50 ± 2.93	95.93 ± 6.67	510.00
	GSEP-40-U	80.55 ± 1.84	**78.66** ± 2.15	**95.03** ± 10.05	449.21 ± 47.40

and introducing diversity into the EP algorithm evaluated. Since the results show that the improvement is consistent in the selected databases, we are extending experimentation using more databases, in order to apply of statistical tests and analyze in depth the efficiency of the proposed methodologies.

Acknowledgements. This work has been partially subsidized by the TIN 2005-08386-C05-02 project of the Spanish Inter-Ministerial Commission of Science and Tech. (MICYT) and FEDER funds. The research of P.A. Gutiérrez has been backed by the FPU Predoctoral Program (Spanish Ministry of Edu. and Sci.).

References

1. Koumousis, V.K., Katsaras, C.P.: A saw-tooth genetic algorithm combining the effects of variable population size and reinitialization to enhance performance. IEEE Transactions on Evolutionary Computation **10**(1) (2006) 19–28
2. Fogel, L.: Artificial Intelligence through Simulated Evolution. 1st edn. John Wiley & Sons, New York (1996)
3. Yao, X.: Evolving artificial neural networks. Proc. of the IEEE **87**(9) (1999) 1423–1447



(a) Population size for SSEP-40. Parameter of saw-teeth are the following: $T = 300$, $D = 480$, $m_n = 40$ and $r = 4$

(b) Population size of two randomly selected executions and mean population size average for 30 executions using GSEP-40-U algorithm.

Fig. 3. Population size for SSEP and GSEP schemes.

4. Goldberg, D., Deb, K., Clark, J.: Genetic algorithms, noise, and the sizing of populations. Complex Systems **6**(4) (1992) 333–362
5. Smith, R.: Adaptively resizing populations: An algorithm and analysis. Proceedings of the Fifth International Conference on Genetic Algorithms (1993) 653–653
6. Cobb, H., Grefenstette, J.: Genetic algorithms for tracking changing environments. Proceedings of the 5th International Conf. on Genetic Algorithms (1993) 523–530
7. Eshelman, L.: The chc adaptive search algorithm: How to have safe search when engaging in nontraditional genetic recombination. In: Proc. Foundations Genetic Algorithms-1. (1991) 257–266
8. Goldberg, D.: Sizing populations for serial and parallel genetic algorithms. Proceedings of the Third International Conference on Genetic Algorithms (1989) 70–79
9. Hervás, C., Martínez-Estudillo, F.J., Gutiérrez, P.A.: Classification by means evolutionary product-unit neural networks. In: Proc. of the 2006 International Joint Conference on Neural Networks, Vancouver, Canada (2006) 2834–2842
10. Newman, D.J., Hettich, S., Blake, C.L., Merz, C.J.: UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998)
11. Angeline, P.J., Saunders, G.M., Pollackm, J.P.: An evolutionary algorithm that constructs recurrent neural networks. IEEE Trans. on Neu. Net. **5**(1) (1994) 54–65
12. Yao, X., Liu, Y.: Evolving artificial neural networks through evolutionary programming. In L. Fogel, P. Angeline, T.B., ed.: Evolutionary Programming V: Proc. of the Fifth Annual Conference on Evolutionary Programming. (1996) 257–266
13. Martínez-Estudillo, A.C., Hervás-Martínez, C., Martínez-Estudillo, F.J., García-Pedrajas, N.: Hybridization of evolutionary algorithms and local search by means of a clustering method. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics **36**(3) (2006) 534–545
14. Martínez-Estudillo, A.C., Martínez-Estudillo, F.J., Hervás-Martínez, C., García-Pedrajas, N.: Evolutionary product unit based neural networks for regression. Neural Networks **19**(4) (2006) 477–486
15. Ventura, S., Romero, C., Zafra, A., Delgado, J., Hervás, C.: JCLEC: a java framework for evolutionary computation. Soft Computing (2007) Published online.
16. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Mach. Learn. **59**(1-2) (2005) 161–205