Efficient evolutionary optimization using individual-based evolution control and neural networks – a comparative Study

Lars Gräning, Yaochu Jin, Bernhard Sendhoff

2005

Preprint:

This is an accepted article published in European Symposium on Artificial Neural Networks (ESANN). The final authenticated version is available online at: https://doi.org/[DOI not available]

European Symposium on Artificial Neural Networks (ESANN'2005). pp.273-278, Bruges, April 2005

Efficient Evolutionary Optimization Using Individual-based Evolution Control and Neural Networks: A Comparative Study

Lars Gräning[1] and Yaochu Jin[2] and Bernhard Sendhoff[2]

1- Department of Neuroinformatics, Technical University of Ilmenau
P.O.Box 100565, 98684 Ilmenau, Germany

2- Honda Research Institute Europe
Carl-Legien-Str. 30, 63073 Offenbach/Main, Germany
yaochu.jin@honda-ri.de

Abstract. To reduce the number of expensive fitness function evaluations in evolutionary optimization, several individual-based and generation-based evolution control methods have been suggested. This paper compares four individual-based evolution control frameworks on three widely used test functions. Feedforward neural networks are employed for fitness estimation. Two conclusions can be drawn from our simulation results. First, the pre-selection strategy seems to be the most stable individual-based evolution control method. Second, structure optimization of neural networks mostly improves the performance of all compared algorithms.

1 Introduction

It has been shown that evolutionary algorithms are very powerful in solving many real-world optimization tasks such as design optimization, see e.g., [1]. In order to reduce the number of time-consuming fitness evaluations, one idea is to estimate the fitness using computationally efficient meta-models [2]. In this work, artificial neural networks are used due to their strong approximation capability [3]. One problem to deal with in real-world optimization problems is that it is difficult to acquire enough training data to achieve sufficiently good approximation, which could result in false convergence [4]. Therefore it is not advisable to use only the neural network as a surrogate for the original fitness function. To avoid false convergence, the neural network model should be used in conjunction with the original fitness function. This is termed evolution control or model management [4]. If evolution control is used, new data become available, which can then be used for on-line neural network training during optimization. In this paper, we compare four individual-based evolution control methods which will be described in the next section.

2 Individual-Based Evolution Control Methods

In individual-based evolution control methods, the main issue is to determine in each generation which individuals should be evaluated using the expensive fitness function and which should be estimated. In the following, four individual-based

strategies are described. As illustrated in Fig. 1, the four methods can be described in a common framework. First, λ' offspring individuals are generated from μ parents using recombination and mutation. After that, the individual-based control method determines which λ^* offspring are evaluated with the original fitness function. The evaluation results are used to train the neural network before the fitness of the remaining $\lambda'-\lambda^*$ offspring is estimated by the neural network. Finally, μ parents are selected from the best λ individuals according to their fitness. This procedure repeats until a termination condition is satisfied.

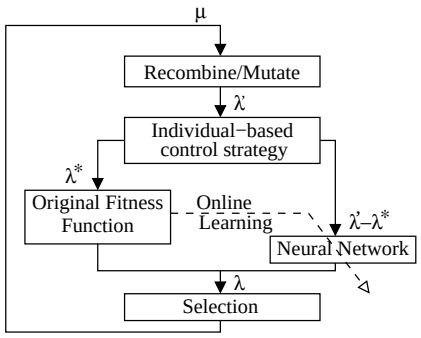

Fig. 1: A generic framework for evolutionary optimization using individual-based control methods.

2.1 Best Strategy (BS)

In the best strategy [4], $\lambda' = \lambda$ offspring are evaluated with the neural network and the λ^* best ones are re-evaluated with the original fitness function. After training the neural network the remaining $\lambda'-\lambda^*$ individuals are evaluated again with the neural network. The μ best individuals from the λ individuals become parents of the next generation.

2.2 Pre-Selection (PreS)

In the pre-selection strategy [5], $\lambda' > \lambda$ offspring individuals are generated through recombination and mutation, and the neural network is used to estimate the fitness value of the offspring. The $\lambda^* = \lambda$ most promising individuals are pre-selected from the λ' offspring and re-evaluated using the original fitness function. The main difference to the best strategy is that the μ parents are selected only from the λ^* individuals evaluated using the original fitness function.

2.3 Clustering Technique (CT)

In [6], a different approach is presented to determine which individuals are to be evaluated with the original fitness function. Using the k-means clustering technique, the $\lambda' = \lambda$ offspring individuals are grouped into λ^* clusters. Now

the λ^* individuals closest to the cluster center are evaluated using the original fitness function. The results of the fitness evaluations, as in other methods, are used to train the neural network. The fitness value of the remaining $\lambda' - \lambda^*$ individuals is estimated using the neural network. Finally, μ individuals are selected from all λ' offspring as parents of the next generation.

2.4 Clustering Technique with Best Strategy (CTBS)

The clustering technique with best strategy is extended from the clustering technique. The offspring individuals are grouped into λ^* clusters. After clustering, the neural network is used to estimate the fitness of individuals. Instead of the individual closest to each cluster center, the best individual in each cluster will be evaluated with the original fitness function.

3 Structure Optimization of the Neural Network

To improve the approximation quality of the neural network, the structure of the neural network can also be optimized during the design optimization. A genetic algorithm has been used to optimize both the structure and the weights of the neural network. In generating offspring, specific mutation operations are employed. The mutation operations allow to insert or delete a single connection or neuron, and the weights are mutated by adding a normally distributed random number. After mutation, life-time learning of the weights is performed and the learned weights are coded back to the individuals, which is known as the Lamarckian mechanism. The EP-tournament-selection is adopted to reproduce the individuals representing the neural networks with the lowest mean squared error on the training data. See [7] for details.

4 Simulation Results

In the simulations, a (μ, λ) covariance matrix adaptation evolution strategy without recombination [8] is adopted, where μ is fixed to 3 and λ is fixed to 12. The strategy parameters of the covariance matrix are initialized between $\sigma_{min} = 0.05$ and $\sigma_{max} = 4$. The settings of λ' and λ^* listed in Table 1 are based on recommendations or findings in [5] and [6]. The maximum number of expensive fitness evaluations is 1200.

	PlainES	PreS	BS	CT	CTBS
λ'	12	24	12	12	12
λ^*	12	12	6	6	6

Table 1: Settings of the parameters λ' and λ^*.

The neural network used in the simulations consists of 10 input nodes, one hidden layer with four hidden neurons and one output. If structure optimization is used the number of hidden neurons is not fixed. An improved version of the

RProp algorithm is used to train the MLP online during optimization. To achieve good local approximation of the original fitness landscape, only data of the most recent 50 evaluations are used for training.

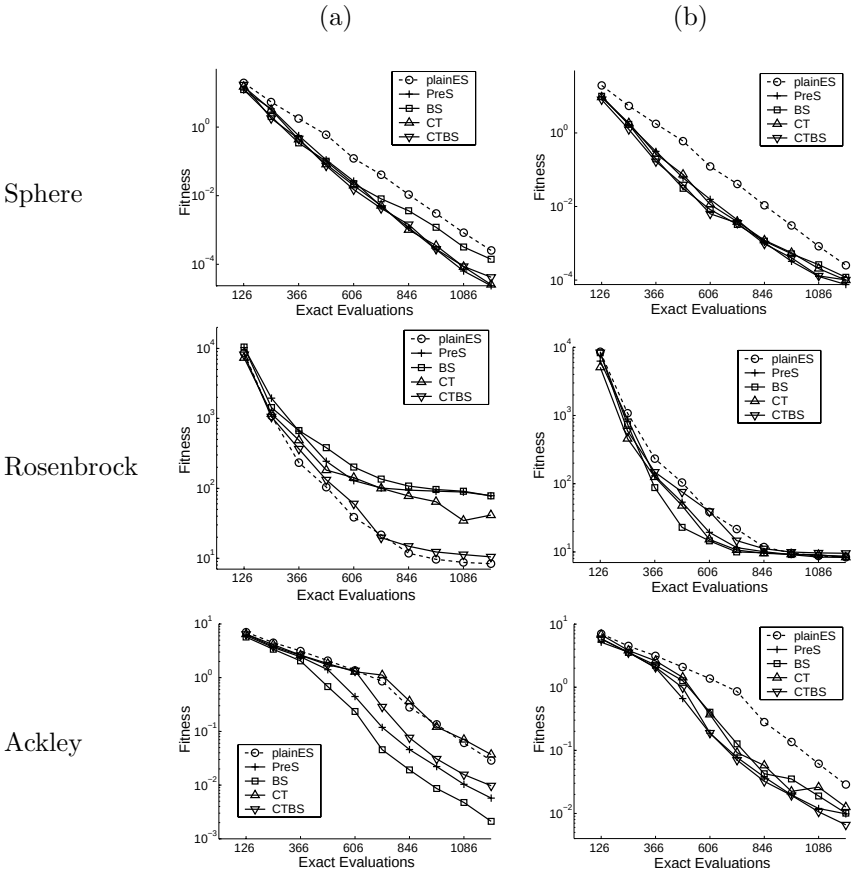

Fig. 2: Results from the three 10-dimensional test functions. (a) Without structure optimization, and (b) with structure optimization.

The results from the Sphere, Rosenbrock and Ackley function are presented in Fig. 2 and Fig. 3. The left column presents the results with and the right column without structure optimization of the neural networks. In Fig. 2, the median of the best fitness in each generation over 20 runs are plotted against the number of expensive fitness evaluations.

To show the statistical significance between the evolution control methods and the plain evolution strategy, the boxplot of the results are given in Fig. 3. The boxplot illustrates the median and the variance of the fitness value of the best individual in the final generation over 20 runs. The notches of the boxes in the plot are the graphical equivalence to the student t-test. If the notch in the

boxplot of the two strategies overlap, there is no significant difference between the medians of the strategies at a significance level of $p = 0.05$.

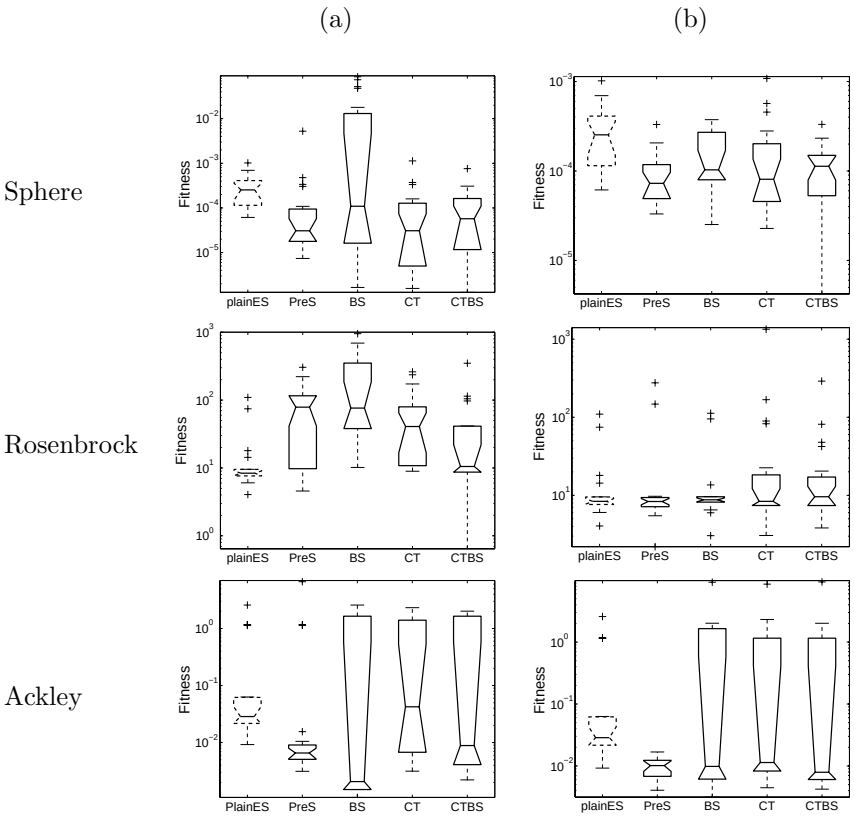

Fig. 3: Boxplot of the best fitness in the final generation over 20 runs after 1200 exact fitness evaluations are done. (a) Without structure optimization, and (b) with structure optimization.

From Fig. 2 and Fig. 3, we can see that all evolution control methods except the best strategy improve the performance of the plain evolution strategy significantly on the 10D Sphere function. But there are no statistically significant differences between the model-assisted strategies themselves, no matter whether structure optimization of the neural networks are conducted or not.

It turns out that the clustering technique with best strategy outperforms other algorithms on the 10D Rosenbrock function, when no structure optimization of the neural network is carried out. However, all algorithms fail to improve the performance of the plain evolution strategy significantly. This result may be attributed to the fact that the number of hidden neurons is not sufficiently large to approximate the Rosenbrock function. Meanwhile, the result indicates that with structure optimization, the neural networks perform locally very well

on the Rosenbrock function.

From Fig. 2, it can be seen that the individual-based evolution control methods perform well on the Ackley function. But as we can see in the boxplots in Fig. 3 the variance of the strategies are very high except the pre-selection strategy. The pre-selection strategy outperforms the plain evolution strategy in almost all of the 20 runs.

5 Conclusion

In this paper, we compared four individual-based evolution control strategies on three test functions. Neural networks with or without structure optimization are used for estimating the fitness. From our results, we showed that the pre-selection strategy is the most promising one among the compared individual-based control strategies. The clustering based approaches are not as good as in [6] mainly because a single network instead of network ensembles has been used in this study.

The future work will be to adapt the number of individuals to be controlled during optimization and to use neural network ensembles instead of a single network. The pre-selection and the clustering based strategies will be implemented in design optimization of three-dimensional turbine blades.

References

[1] M. Olhofer, T. Arima, T. Sonoda, and B. Sendhoff. Optimisation of a stator blade used in a transonic compressor cascade with evolution strategies. In *Adaptive Computation in Design and Manufacture*, pages 45–54. Springer, 2000.

[2] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Computing*, 9(1):3–12, 2005.

[3] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, 1989.

[4] Y. Jin, M. Olhofer, and B. Sendhoff. A framework for evolutionary optimization with approximate fitness functions. *IEEE Transactions on Evolutionary Computation*, 6(5):481–494, 2002.

[5] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies with controlled model assistance. In *Congress on Evolutionary Computation*, pages 1569–1576, 2004.

[6] Y. Jin and B. Sendhoff. Reducing fitness evaluations using clustering techniques and neural networks ensembles. In *Genetic and Evolutionary Computation Conference*, volume 3102 of *LNCS*, pages 688–699. Springer, 2004.

[7] M. Hüsken, Y. Jin, and B. Sendhoff. Structure optimization of neural networks for aerodynamic optimization. *Soft Computing Journal*, 9(1):21–28, 2005.

[8] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–196, 2001.