Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles

Yaochu Jin and Bernhard Sendhoff

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

Abstract. In many real-world applications of evolutionary computation, it is essential to reduce the number of fitness evaluations. To this end, computationally efficient models can be constructed for fitness evaluations to assist the evolutionary algorithms. When approximate models are involved in evolution, it is very important to determine which individuals should be re-evaluated using the original fitness function to guarantee a faster and correct convergence of the evolutionary algorithm. In this paper, the k-means method is applied to group the individuals of a population into a number of clusters. For each cluster, only the individual that is closest to the cluster center will be evaluated using the expensive original fitness function. The fitness of other individuals are estimated using a neural network ensemble, which is also used to detect possible serious prediction errors. Simulation results from three test functions show that the proposed method exhibits better performance than the strategy where only the best individuals according to the approximate model are re-evaluated.

1 Introduction

Many difficulties may arise in applying evolutionary algorithms to solving complex real-world optimization problems. One of the main concerns is that evolutionary algorithms usually need a large number of fitness evaluations to obtain a good solution. Unfortunately, fitness evaluations are often very expensive or highly time-consuming. Take aerodynamic design optimization as an example, one evaluation of a given design based on the 3-Dimensional computational fluid dynamics (CFD) simulation will take hours on a high-performance computer.

To alleviate this problem, computationally efficient models can be constructed to approximate the fitness function. Such models are often known as approximate models, meta-models or surrogates, refer to [9] for an overview of this topic. It would be ideal if an approximate model can fully replace the original fitness function, however, researchers have come to realize that it is in general necessary to combine the approximate model with the original fitness function to ensure the evolutionary algorithm to converge correctly. To this end, re-evaluation of some individuals using the original fitness function, also termed as *evolution control* in [7], is essential.

K. Deb et al. (Eds.): GECCO 2004, LNCS 3102, pp. 688–699, 2004.

Generation-based or individual-based evolution control can be implemented. In the generation-based approach [15,2,7,8], some generations are evaluated using the approximate model and the rest using the original fitness function. In individual-based evolution control, part of the individuals of each generation are evaluated using the approximation model and the rest using the original fitness function [7,3,18,1]. Generally speaking, the generation-based approach is more suitable when the individuals are evaluated in parallel, where the duration of the optimization process depends to a large degree on the number of generations needed. By contrast, the individual-based approach is more desirable when the number of evaluations is limited, for example, when an experiment needs to be done for a fitness evaluation.

On the other hand, individual-based evolution control provides more flexibility in choosing which individuals need to be re-evaluated. In [7], it is suggested that one should choose the best individuals according to the approximate model rather than choosing the individuals randomly. In [3], not only the estimated function value but also the estimation error are taken into account. The basic idea is that individuals having a larger estimation error are more likely to be chosen for re-evaluation. Other uncertainty measures have also been proposed in [1].

In [10], the population of a genetic algorithm is grouped into a number of clusters and only one representative individual of each cluster is evaluated using the fitness function. Other individuals in the same cluster are estimated according to their Euclidean distance to the representative individuals. Obviously, this kind of estimation is very rough and the local feature of the fitness landscape is completely ignored. In this paper, we also group the population into a number of clusters, and only the individual that is closest to the cluster center is evaluated using the original fitness function. In contrast to the distance-based estimation method [10], we use the evaluated individuals (centers of the clusters) to create a neural network ensemble, which is used for estimating the fitness values of the remaining individuals. Both the structure and the parameters of the neural networks are optimized using an evolutionary algorithm with Lamarckian inheritance.

The remainder of the paper is organized as follows. Section 2 presents population clustering using the k-means algorithm. The construction of neural network ensembles using an evolutionary algorithm is described in Section 3. The proposed algorithm is applied to the optimization of three test functions in Section 4. A summary of the paper is provided in Section 5.

2 Population Clustering

A variety of clustering techniques have been proposed for grouping similar patterns (data items) [4]. Generally, they can be divided into hierarchical clustering algorithms and partitional clustering algorithms. A hierarchical algorithm yields a tree structure representing a nested grouping of patterns, whereas a partitional clustering algorithm generates a single partition of the patterns. Among the

partitional clustering methods, the k-means is the simplest and the most commonly used clustering algorithm. It employs the squared error criterion and its computational complexity is $O(n)$, where n is the number of patterns. A standard k-means algorithm is given in Fig. 1. A typical stopping criterion is that the decrease in the squared error is minimized.

A major problem of the k-means clustering algorithm is that it may converge to a local minimum if the initial partition is not properly chosen. Besides, the number of clusters needs to be specified beforehand, which is a general problem for partitional clustering algorithms [4].

1. Choose k patterns randomly as the cluster centers
2. Assign each pattern to its closest cluster center
3. Recompute the cluster center using the current cluster members
4. If the convergence criterion is not met, go to step 2; otherwise stop

Fig. 1. The k-means algorithm.

To assess the validity of a given cluster, the silhouette method [17] can be used. For a given cluster, $X_j, j = 1, ..., k$, the silhouette technique assigns the i-th member $(x_{ij}, i = 1, ..., n_j)$ of cluster X_j a quality measure (*silhouette width*):

$$s_{ij} = \frac{b_i - a_i}{\max\{a_i, b_i\}}, \tag{1}$$

where a_i is the average distance between x_{ij} and all other members in X_j and b_i denotes the minimum of $a_i, i = 1, 2, ..., n_j$, where n_j is the number of patterns in cluster X_j and naturally, $n_1 + ... + n_k$ equals n if each pattern belongs to one and only one cluster, n is the number of patterns to be clustered. It can be seen that s_{ij} has a value between -1 and 1. If s_{ij} equals 1, it means that s_{ij} is in the proper cluster. If s_{ij} is 0, it indicates that x_{ij} may also be grouped in the nearest neighboring cluster and if x_{ij} is -1, it suggests that x_{ij} is very likely in the wrong cluster. Thus, a global silhouette width can be obtained by summing up the silhouette width over all patterns:

$$S = \frac{1}{k} \sum_{j=1}^{k} \sum_{i=1}^{n_j} s_{ij}. \tag{2}$$

Consequently, this value can be used to determine the proper number of clusters.

3 Construction of Local Neural Network Ensemble

After the population is grouped into a number of clusters, only the individual that is closest to each cluster center will be evaluated using the original fitness function. In [10], the fitness value of all other individuals are estimated based on their Euclidean distance to the cluster center. Obviously, this simplified estimation ignores the local feature of the fitness landscape which can be extracted from the evaluated cluster centers.

In our previous work [7,8], a fully connected multi-layer perceptron (MLP) neural network has been constructed using the data generated during optimization. The neural network model is trained off-line and further updated when new data are available. One problem that may occur is that as the number of samples increases, the learning efficiency may decrease. To improve the learning efficiency, weighted learning [8] and off-line structure optimization of the neural networks have been shown to be promising. In this work, we attempt to further improve the approximation quality in two aspects. First, structure optimization of the neural network is carried out on-line and only the data generated in the most recent two generations are used. This makes it possible to have an approximate model that reflects the local feature of the landscape. Second, an ensemble instead of a single neural network will be used to improve the generalization property of the neural networks.

The benefit of using a neural network ensemble originates from the diversity of the behavior of the ensemble members on unseen data. Generally, diverse behavior on unseen data can be obtained by using various initial random weights, varying the network' architecture, employing different training algorithm, supplying different training data by manipulating the given training data, generating data from different sources, or encouraging diversity [13], decorrelation [16] or negative correlation [11,12] between the ensemble members.

In this work, a genetic algorithm with local learning [6] has been used to generate the neural network ensemble, which can provide two sources of diversity: both the architecture and the final weights of the neural networks are different. Since the goal of the neural networks is to learn the local fitness landscape, we only use the data generated in the two most recent generations instead of using all data.

Assume that the λ individuals in the population are grouped into ξ clusters, thus ξ new data will be generated in each generation. Accordingly, the fitness function for evolutionary neural network generation can be expressed as follows:

$$F = \frac{1}{\xi}\left\{\alpha \cdot \sum_{i=1}^{\xi}(y_i - y_i^d(t))^2 + (1-\alpha)\cdot \sum_{i=1}^{\xi}(y_i - y_i^d(t-1))^2\right\}, \tag{3}$$

where $0.5 < \alpha \leq 1$ (set to 0.7 in this work) is a coefficient giving more importance to the newest data, $y_i^d(t), i = 1, ..., \xi$ are the data generated in the current generation and $y_i^d(t-1)$, $i = 1, ..., \xi$ are those generated in the last generation and y_i is the network output for the i-th data set.

Given N neural networks, the final output of the ensemble can be obtained by averaging the weighted outputs of the ensemble members:

$$y^{EN} = \sum_{k=1}^{N} w^{(k)} y^{(k)}, \tag{4}$$

where $y^{(k)}$ and $w^{(k)}$ are the output and its weight of the k-th neural network in the ensemble. If all the weights are equally set to $1/N$, it is termed basic ensemble method (BEM). Otherwise, it is termed generalized ensemble method (GEM). In this case, the expected error of the ensemble is given by:

$$E^{EN} = \sum_{i=1}^{N} \sum_{j=1}^{N} w^{(i)} w^{(j)} C_{ij}, \tag{5}$$

where C_{ij} is the error correlation matrix between network i and network j in the ensemble:

$$C_{ij} = E[(y_i - y_i^d)(y_j - y_j^d)], \tag{6}$$

where $E(\cdot)$ denotes the mathematical expectation.

It has been shown [14] that there exists an optimal set of weights that minimizes the expected prediction error of the ensemble:

$$w^{(k)} = \frac{\sum_{j=1}^{N} (C_{kj})^{-1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} (C_{ij})^{-1}}, \tag{7}$$

where $1 \leq i, j, k \leq N$.

However, a reliable estimation of the error correlation matrix is not straightforward because the prediction errors of different networks in an ensemble are often strongly correlated. A few methods have been proposed to solve this problem [5,19,20]. Genetic programming is applied to the search for an optimal ensemble size in [20] whereas the recursive least-square method is adopted to optimize the weights in [19]. In [19], a GA is also used to search for an optimal subset of the neural networks in the final population as ensemble members.

To reduce the computational complexity, only a small number of networks (three to five) has been tried in this work. A canonical evolution strategy is employed to find the optimal weights to minimize the expected error in Eq. (5).

The algorithm for constructing the neural network ensemble and the entire evolutionary optimization algorithm are sketched in Fig. 2 and Fig. 3, respectively.

4 Empirical Results

4.1 Experimental Setup

In the simulations, optimization runs are carried out on three well known test functions, the Ackley function, the Rosenbrock function and the Sphere function.

1. Prepare the training and test data
2. Generate N (ensemble size) neural networks using GA
3. Calculate the error correlation between the ensemble members
4. Determine the optimal weight for each network by using ES

Fig. 2. Algorithm for constructing neural network ensemble.

1. Initialize λ individuals, evaluate all individuals using the original fitness function
2. For each generation
   a) select the best μ individuals
   b) generate λ offspring individuals by recombination and mutation
   c) evaluate
      – clustering the λ invividuals using the k–means algorithm
      – evaluate the ξ individuals closest to the cluster centers using the original fitness function
      – construct the neural network ensemble
      – calculate the fitness of the rest $\lambda - \xi$ individuals using the neural network ensemble
3. Go to step 2 if the termination condition is not met
4. Stop

Fig. 3. The proposed evolutionary optimization algorithm.

The dimension of the test functions are set to 30. A standard $(5, 30)$ evolution strategy (ES) is used in all simulations. The maximal number of fitness evaluations is set to 2000 in the optimization for all three test functions.

Before we implement the evolutionary optimization with approximate fitness models, we need to determine a few important parameters, such as the number of clusters and the number of neural networks in the ensemble.

The first issue is the number of clusters. This number is relevant to performance of the clustering algorithm, the quality of the approximate model, and eventually the convergence property of the evolutionary algorithm.

A few preliminary optimization runs are carried out with only a single neural network being used for fitness approximation on the 30-dimensional Ackley function. It is found that with the clustering algorithm, the evolutionary algorithm is able to converge correctly when about one third of the population is re-evaluated using the original fitness function. When the number of the re-evaluated individuals is much fewer than one third of the population, the performance of the

evolutionary algorithm becomes unpredictable, that is, the evolutionary algorithm may converge to a false minimum.

We then evaluate the clustering performance when the number of clusters is set to be one third of the population. Fig. 4 shows the global silhouette width when the cluster number is 10 and the population size is 30 on the 30-dimensional Ackley function. It can be seen that the clustering performance is acceptable according to the discussions in Section 2.

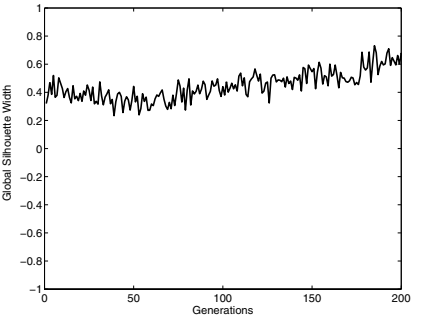
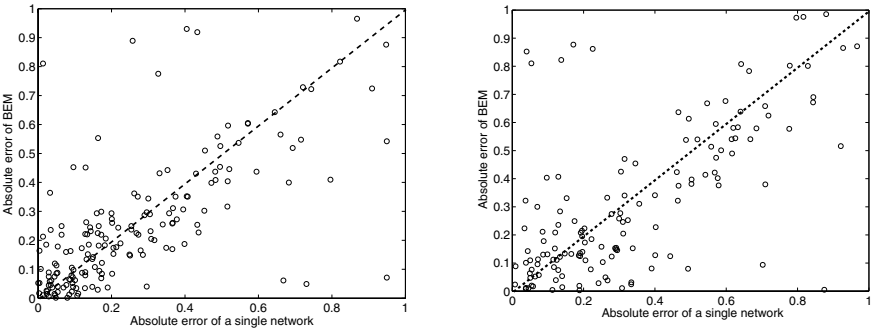


Fig. 4. Global silhouette width when the number of cluster is set to 10 and the population size is 30 on the 30-dimensional Ackley function.

Next, simulations are conducted to investigate the ensemble size. So far, the ensemble size has been determined heuristically in most applications. In [20], the optimal size turns out to be between 5 and 7. Considering the fact that a large ensemble size will increase computational cost, we compare two cases where the ensemble size is 3 and 5 on 200 samples collected in the first 20 generations of an optimization run on the 30-dimensional Ackley function. The ensemble output versus that of a single network is plotted in Fig. 5, where in Fig. 5(a) the ensemble size is 3 and in Fig. 5(b) the ensemble size is 5. Note that the more points locate in the right lower part of the figure the more effective the ensemble. It can be seen from the figure that no significant performance improvement has been achieved when the ensemble size is changed from 3 to 5. Thus, we fix the ensemble size to 3.

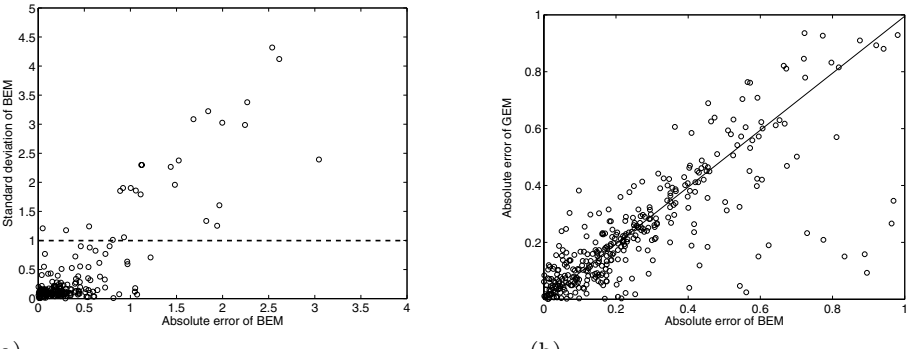
It seems that the use of an ensemble has not improved the prediction accuracy significantly. Thus, the motivation to employ an ensemble becomes questionable. In the following, we will show that an ensemble is important not only in that it is able to improve prediction. In this work, the equally important reason for introducing the ensemble is to estimate the prediction accuracy based on the different behaviors of the ensemble members, i.e., the variance of the members in the ensemble. To demonstrate this, Fig. 6(a) shows the relationship between the standard deviation of the predictions of the ensemble members and the estimation error of the ensemble. These data are also collected in the first 20 generations of an evolutionary run of the Ackley function. Additional function



(a) (b)

Fig. 5. (a) Ensemble size equals 3. (b) Ensemble size equals 5.

evaluations are carried out to get the prediction error. Of course, they are neither used in neural network training nor in optimization. It can be seen that a large standard deviation most probably indicates a large prediction error, although a small standard deviation does not guarantee a small prediction error. Encouraged by this close correlation between a large deviation and a large prediction error, we try to predict the model error. When the standard deviation is larger than a threshold (1 in this example), we replace the model prediction with the fitness of the individual closest to the cluster center, which is a very rough but feasible approximation.



(a) (b)

Fig. 6. (a) Prediction error versus the standard deviation. (b) Prediction error of the BEM versus that of the GEM.

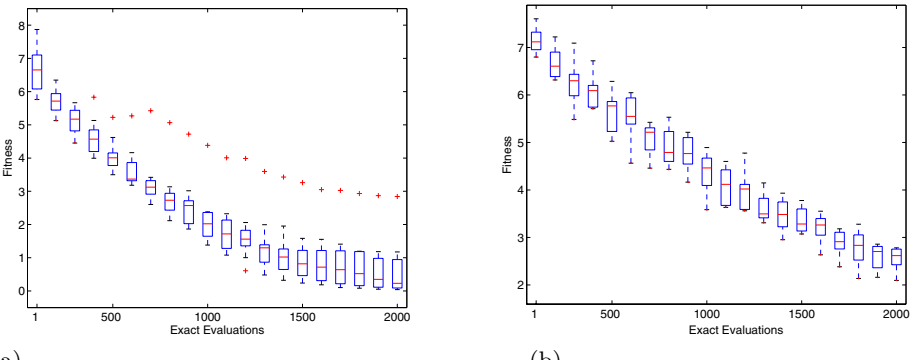
Finally, we have run a standard evolution strategy with a population size of (3,15) for 100 generations to optimize the weight of the ensemble members. The predictions of the GEM, where the weights are optimized, and that of a BEM

are shown in Fig. 6(b). It can be seen that the prediction accuracy has been improved using the GEM.

4.2 Optimization Results

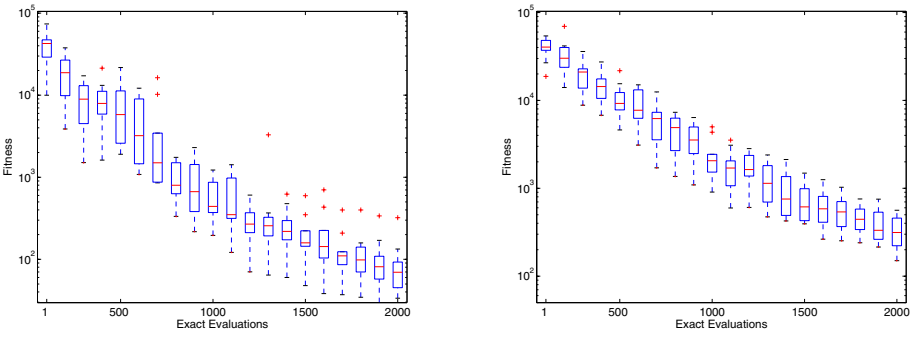
The box plots of the ten runs on the three test functions are shown in Figures 7, 8 and 9. In a box plot, the line in the box denotes the median, the lower and upper bounds of the box are the 25% and 75% lower quartiles, and the lower and upper fences are the lower and upper whiskers, respectively. The outliers are denoted by the '+' sign. For clarity, only 20 data points are shown in the figures, which are uniformly sampled from the original data. From these figures, it can clearly be seen that on average, the optimization results using the proposed algorithm are much better than those from the plain evolution strategy on all test function. Meanwhile, they are also much better than the results reported in [7], where no clustering of the population has been implemented. As we mentioned, without clustering, the evolutionary algorithm does not converge correctly if only one third of the population is re-evaluated using the original fitness function. Nevertheless, we also notice that for the Ackley function, the result from one of the 10 runs using the proposed method is much worse than the average performance, even a little worse than the average result when the plain ES is used, refer to Fig. 7(a).

To show the benefit of using the neural network ensemble, the box plots of results using only a single neural network (where no remedy of large prediction errors is included) on the three test functions are provided in Figures 10, 11 and 12. Similarly, only 20 data points are presented for the clarity of the figures. Compared with the results shown in Figures 7, 8 and 9, they are much worse. In the Rosenbrock function, some runs even have diverged, mainly due to the bad performance of the model prediction.



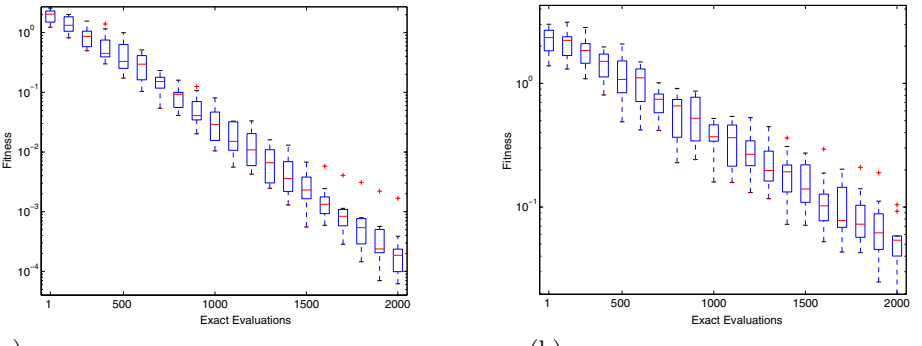
(a) (b)

Fig. 7. Box plot of the results for the 30-dimensional Ackley function. (a) The proposed algorithm. (b) Plain ES. Notice that the scales in (a) and (b) are not the same.



(a) (b)

Fig. 8. Box plot of the results for the 30-dimensional Rosenbrock function. (a) The proposed algorithm. (b) Plain ES.



(a) (b)

Fig. 9. Box plot of the results for the 30-dimensional Sphere function. (a) The proposed algorithm. (b) Plain ES. Notice that the scales in (a) and (b) are not the same.

5 Conclusions

A new method for reducing fitness evaluations in evolutionary computation has been proposed. In each generation, the population is clustered into a number of groups and only the individuals closest to each cluster center will be evaluated. Then a neural network ensemble is constructed using the data from the evaluated individuals. To further improve the prediction quality, the weights of the ensemble are optimized using a standard ES. We further exploit information contained in the ensemble by taking advantage of the standard deviation of the output of the ensemble members. When the ensemble members disagree significantly, the prediction error is very likely to be large and thus the ensemble prediction is replaced by the fitness value of the cluster center of the individual. Simulation results on the test functions suggest that the proposed algorithm is very promising.

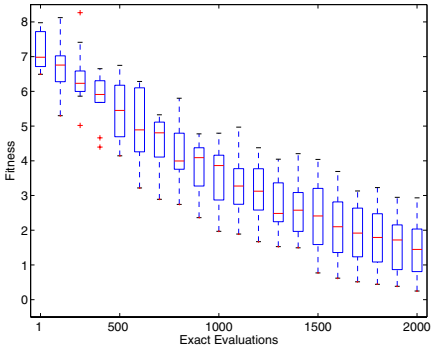


Fig. 10. Results for the 30-dimensional Ackley function with a single network.

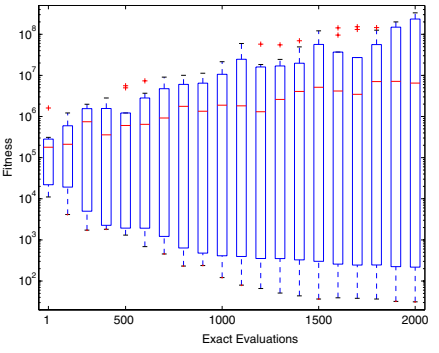


Fig. 11. Results for the 30-dimensional Rosenbrock function with a single network.

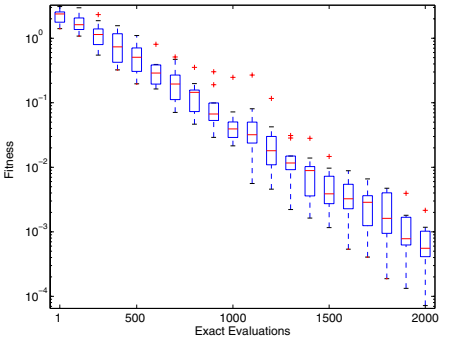


Fig. 12. Results for the 30-dimensional Sphere function with a single network.

Currently, the number of individuals to be controlled is fixed. As suggested in [8], an adaptation of the control frequency could provide more performance improvement. One possibility is to determine the number of individuals to optimize the performance of the clustering algorithm using the global silhouette width.

References

1. J. Branke and C. Schmidt. Fast convergence by means of fitness estimation. *Soft Computing*, 2003. To appear.
2. M.A. El-Beltagy, P.B. Nair, and A.J. Keane. Metamodeling techniques for evolutionary optimization of computationally expensive problems: promises and limitations. In *Proceedings of Genetic and Evolutionary Conference*, pages 196–203, Orlando, 1999. Morgan Kaufmann.
3. M. Emmerich, A. Giotis, M. Özdenir, T. Bäck, and K. Giannakoglou. Metamodel-assisted evolution strategies. In *Parallel Problem Solving from Nature*, number 2439 in Lecture Notes in Computer Science, pages 371–380. Springer, 2002.

4. A.K. Jain, M.N. Murty, and P.J. Flynn. Data clustering: A review. *ACM Computing Surveys*, 31(3):264–323, September 1999.
5. D. Jimenez. Dynamically weighted ensemble neural networks for classification. In *Proceedings of International Joint Conference on Neural Networks*, pages 753–756, Anchorage, 1998. IEEE Press.
6. Y. Jin, T. Okabe, and B. Sendhoff. Neural network regularization and ensembling using multi-objective evolutionary algorithms. In *Proceedings of IEEE Congress on Evolutionary Computation*, Portland, 2004. IEEE. To appear.
7. Y. Jin, M. Olhofer, and B. Sendhoff. On evolutionary optimization with approximate fitness functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 786–792. Morgan Kaufmann, 2000.
8. 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.
9. Y. Jin and B. Sendhoff. Fitness approximation in evolutionary computation - A survey. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1105–1112, New York City, NY, 2002.
10. H.-S. Kim and S.-B. Cho. An efficient genetic algorithms with less fitness evaluation by clustering. In *Proceedings of IEEE Congress on Evolutionary Computation*, pages 887–894, Piscataway, NJ, 2001. IEEE.
11. Y. Liu and X. Yao. Negatively correlated neural networks can produce best ensemble. *Australian Journal of Intelligent Information Processing System*, 4(3–4):176–185, 1997.
12. Y. Liu, X. Yao, and T. Higuchi. Evolutionary ensembles with negative correlation learning. *IEEE Transactions on Evolutionary Computation*, 4(4):380–387, 2000.
13. D.W. Opitz and J. W. Shavlik. Generating accurate and diverse members of a neural network ensemble. In *Advances in Neural Information Processing Systems*, volume 8, pages 535–541, Cambridge, MA, 1996. MIT Press.
14. M.P. Perrone and L.N. Cooper. When networks disagree: Ensemble methods for hybrid neural networks. In R. J. Mammone, editor, *Artificial Neural Networks for Speech and Vision*, pages 126–142. Chapman & Hall, London, 1993.
15. A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. Eiben, Th. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Parallel Problem Solving from Nature*, volume V, pages 87–96, 1998.
16. B. E. Rosen. Ensemble learning using decorrelated neural networks. *Connection Science*, 8(3–4):373–384, 1996.
17. P.J. Rousseeuw. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *Journal of Computational and Applied Mathematics*, 20:53–65, 1987.
18. H. Ulmer, F. Streicher, and A. Zell. Model-assisted steady-state evolution strategies. In *Proceedings of Genetic and Evolutionary Computation Conference*, LNCS 2723, pages 610–621, 2003.
19. X. Yao and Y. Liu. Making use of population information in evolutionary artificial neural networks. *IEEE Transactions on Systems, Man, and Cybernetics-Part B:Cybernetics*, 28(3):417–425, 1998.
20. B.-T. Zhang and J.G. Joung. Building optimal committee of genetic programs. In *Parallel Problem Solving from Nature*, volume VI, pages 231–240. Springer, 2000.