A Comparative Study of CMA-ES on Large Scale Global Optimisation

Mohammad Nabi Omidvar and Xiaodong Li

Evolutionary Computing and Machine Learning Laboratory (ECML Lab),
Royal Melbourne Institute of Technology (RMIT), Melbourne, Australia
momidvar@cs.rmit.edu.au, xiaodong.li@rmit.edu.au
http://goanna.cs.rmit.edu.au/~xiaodong/ecml

Abstract. In this paper, we investigate the performance of CMA-ES on large scale non-separable optimisation problems. CMA-ES is a robust local optimiser that has shown great performance on small-scale non-separable optimisation problems. Self-adaptation of a covariance matrix makes it rotational invariant which is a desirable property, especially for solving non-separable problems. The focus of this paper is to compare the performance of CMA-ES with Cooperative Co-evolutionary Algorithms (CCEAs) for large scale global optimisation (on problems with up to 1000 real-valued variables). Since the original CMA-ES is incapable of handling problems with more than several hundreds dimensions, sep-CMA-ES was developed using only the diagonal elements of the covariance matrix. In this paper sep-CMA-ES is compared with several existing CCEAs. Experimental results revealed that the performance of sep-CMA-ES drops significantly when the dimensionality of the problem increases. However, our results suggest that the rotational invariant property of CMA-ES can be utilised in conjunction with a CCEA to further enhance its capability to handle large scale optimisation problems.

1 Introduction

Advances in science and technology provides us with ever more options and features, however having more features makes it more difficult to find the optimum configuration of these decision variables. The rapid growth in the number of decision variables brings a grand scale challenge to optimisation techniques. In nano-technology, the properties of thousands of atoms have to be taken into account in order to produce a substance with a certain property. In aerodynamics, tens of thousands of parameters have to be tweaked in order to optimise a component of a space shuttle to a target shape. This shift in the scale of optimisation problems demands new optimisation techniques capable of dealing with thousands of decision variables.

Many Evolutionary Algorithms (EAs) [6], [3], [2] have been applied to optimisation problems, however the performance of these algorithms, like most of traditional algorithms, deteriorate as the dimensionality of the problem increases. This is referred to as the *curse of dimensionality* [4]. Divide-and-conquer is a natural approach for tackling large scale problems. Cooperative Co-evolution (CC) [15] is such a technique that decomposes a large scale problem into a set of sub-problems, each of which is optimised using a separate EA. In the original CC decomposition strategy, each variable is placed in a separate subcomponent. This new paradigm has shown great success on many optimisation problems [15]. However further investigation revealed that this CC decomposition strategy is only effective when there is no interdependency between the decision variables [22]. Shi et al. [19] proposed another technique in which the decision variables are divided into halves. This dividing-in-half strategy does not scale properly as the dimensionality of problem increases, mainly because the size of sub-problems will go beyond the optimisation capabilities of subcomponent optimisers. van den Bergh and Engelbrecht [22] proposed a new decomposition strategy where they decomposed a n-dimensional problem into m s-dimensional subcomponents. It has been shown that this new technique performs better than the original decomposition strategy when dealing with non-separable problems.

Although grouping interacting variables in a common subcomponent increases the performance of many CCEAs to a great extent, capturing the hidden dependencies between the variables is a great challenge by itself. *Random Grouping* (DECC-G, MLCC) [23] and *Delta Grouping* [14] are two major techniques that were proposed recently for capturing the interacting variables. Delta grouping in particular has shown superior performance in capturing interacting variables in grouping them in a common subcomponent [14].

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [10] is a successful optimisation algorithm that has been designed specifically for local optimisation, but it has also shown competitive results for global optimisation [8], [7]. The self-adaptation of covariance matrix makes CMA-ES rotational invariant that makes it a perfect choice for non-separable problems.

Most studies on CMA-ES have been carried out with functions with up to 200 dimensions. There are limited studies in which functions with up to 1000 dimensions have been used [17], however these studies were confined with very few simple test functions. Furthermore, since the introduction of the new test functions such as CEC'2008 [21] and CEC'2010 [20], which are specifically designed for benchmarking of large scale optimisation algorithms, no systematic studies have been carried out on evaluating CMA-ES using these newly proposed functions. In this research we benchmark the performance of CMA-ES on large scale problems proposed in CEC'2008 Special Session and Competition on Large Scale Global Optimisation [21]. In particular we have the following research objectives:

- Benchmarking the performance of standard CMA-ES on large scale problems.
- Comparing the performance of cooperative co-evolutionary algorithms for large scale optimisation with CMA-ES on the same problem set.
- Verifying the performance of a rotational invariant algorithm such as CMA-ES on large scale non-separable optimisation problems.

The organisation of the rest of this paper is as follows. Section 2 explains the preliminaries, such as CC and CMA-ES. Section 3 presents the experimental results and their analysis. Finally, Section 4 summarises this paper and gives directions to future works.

2 Background

This section is dedicated to background information. Section 2.1 describes the Cooperative Co-evolution [15] and different decomposition strategies proposed in the past. Section 2.2 describes CMA-ES [10] in more details.

2.1 Cooperative Co-evolution

Cooperative Co-evolution has been proposed by Potter and De Jong [15], explicitly introduces the notion of modularity in EAs. This notion of modularity is essential in order to solve complex problems. CC works in the form of co-adapted subcomponents. In the context of an optimisation problem, a n-dimensional problem is divided into n 1-dimensional problems, each of which is optimised using a separate EA in a round-robin fashion. This decomposition scheme works well only when there is no interaction between the decision variables. It has been shown that the performance of original CC framework deteriorates when there are interactions between the decision variables [12]. van den Bergh and Engelbrecht [22] used a different decomposition strategy in which they divided a n-dimensional problem into m s-dimensional subproblems. It has been shown that this new decomposition strategy has a better performance compared to the original CC framework [22].

CC framework has been applied to many EAs for large scale optimisation problems. However, the performance of these algorithms deteriorate as the dimensionality of the problem increases. Fast Evolution Programming with Cooperative Co-evolution (FEPCC) [12] is one of the early techniques that has been applied to problems with up to 1000 dimensions. The experimental results revealed that FEPCC performed poorly on one of non-separable problems [12]. The reason for poor performance of FEPCC on such problems is due to grouping of interacting variables in separate subcomponents. As a result, to increase the performance of CCEAs on large scale non-separable problems, the interacting variables have to be identified and be grouped in a common subcomponent.

Random grouping has been proposed by Yang et al. [23] in order to increase the probability of grouping interacting variables in a subcomponent. Although random grouping has shown better performance compared to other algorithms, its performance degrades as the number of interacting variables increases [13]. In order to cope with the increased number of interacting variables, more intelligent and systematic techniques are required to capture interacting variables. Ray and Yao [16] calculated a correlation coefficient matrix from the current population and divided the decision variables into two subcomponents, based on a threshold value on correlation coefficients. This technique does not scale properly when the dimensionality of the problem increases. This is because the decision variables are divided into halves and as the dimensionality of the problem increases the complexity of sub-problems will go beyond the capabilities of CCEA-AVP. Another disadvantage of CCEA-AVP is the use of correlation coefficients as a measurement for degree of separability (or non-separability) of decision variables. Correlation coefficient measures such as Pearson's correlation matrix measures the linear dependence between two variables which is not a proper estimation for separability (or non-separability) of decision variables.

Delta grouping has been proposed by Omidvar et al. [14] as the first systematic technique for capturing interacting variables. Delta grouping is inspired by the idea of improvement interval under coordinate rotation explained in detailed in [18]. In delta grouping, the average amount of change in every dimension is measured between two consecutive cycles to form a delta vector. Then the variables are sorted based on the absolute magnitude of their delta values. The motivation behind delta grouping is that, in a non-separable problem, when a small delta value is observed in one of the dimensions, there is a high probability to find another decision variable with relatively small delta value. As a result, grouping the variables based on the magnitude of their delta values increases the probability of grouping two interacting variables in one subcomponent. Delta grouping has shown great performance [14] on a wide range of benchmark functions [20], [21] that were proposed especially for large scale global optimisation.

2.2 Covariance Matrix Adaptation Evolution Strategy

CMA-ES is based on Evolution Strategy (ES) [5]. ES is a type of EA which has been extensively used for continuous optimisation tasks. The individuals in ES are real-valued vectors that are systematically changed to get better individuals. Like many EAs, ES rely on three major operations, mutation, recombination, and selection. Mutation and recombination are used for exploration of the search space and generating genetic variations, while the selection operator is for exploitation and convergence to a solution. The mutation operator is an important operator in ES and is central to understanding of CMA-ES. A detailed explanation of various selection operators can be found in [5]. Recombination is not a very common operator in state-of-the-art ES implementations, as a result the focus of this section is on mutation.

Mutation. Mutation is a key operator in ES that generates the maximum genetic variations. In real-valued continuous optimisation problems the mutation is done using a multivariate Gaussian distribution. Equation (1) shows how an individual is mutated using a Gaussian distribution.

$$\tilde{\boldsymbol{y}} = \boldsymbol{y} + \boldsymbol{z} \ , \tag{1}$$

where $\boldsymbol{y}$ is the parent, $\tilde{\boldsymbol{y}}$ is the mutant and the $\boldsymbol{z}$ is defined as follows:

$$\boldsymbol{z} = \sigma\big(N_1(0,1), ..., N_N(0,1)\big) \ , \tag{2}$$

where $N_i(0,1)$ are mutually independent random numbers, generated form a normal distribution with mean zero and standard deviation 1. σ is the strategy parameter which is called the step size. Having only one step size creates an isotropic Gaussian distribution which is symmetric about the mean value which is $\boldsymbol{y}$ in this case. This situation is depicted in Figure 1(a). Having only one step size is not very efficient in solving high dimensional problems. In an extended version of ES, instead of having only one global step size, a vector of step sizes is maintained. Each of the elements in this vector corresponds to one of the dimensions. This allows having different step sizes for every dimension. In this new scheme the mutation is performed using Equations (3).

$$\begin{aligned}\boldsymbol{z} &= \big(\sigma_1 N_1(0,1), ..., \sigma_N N_N(0,1)\big) \\ &= D\big(N_1(0,1), ..., N_N(0,1)\big)^T \\ &= DN(0,I) \ , \end{aligned} \tag{3}$$

as it can be seen form Equation (3), there are different σ values for different dimensions. This situation is depicted in Figure 1(b). D is a diagonal matrix with the σ values on the main diagonal. Although this new technique is far more flexible than the isotropic version, it still loses its efficiency when applied to non-separable functions. As it can be seen from Figure 1(b), the Gaussian distribution is scaled in the direction of the coordinate axes. In many real-world problems, the fitness landscape is not aligned with the coordinate system which makes this mutation strategy ineffective. As a result another mutation scheme is needed to work under arbitrary rotations of fitness landscape which is a desirable technique especially for non-separable problems. Covariance Matrix Adaptation ES proposes such a rotational invariant version of ES by self-adapting a rotation matrix M to align the diagonal matrix D with the principal axes of the fitness landscape [10]. So, the mutation scheme in CMA-ES is as follows:

$$\begin{aligned}\boldsymbol{z} &= M\big(\sigma_1 N_1(0,1), ..., \sigma_N N_N(0,1)\big) \\ &= MD\big(N_1(0,1), ..., N_N(0,1)\big)^T \\ &= MDN(0,I) \ , \end{aligned} \tag{4}$$

it is clear that the rotation matrix M creates correlation between the components of $\boldsymbol{z}$, thus, $C = M^T M$. The use of correlation matrix C and the effect of this new mutation scheme is depicted in Figure 1(c). As it can be seen from Figure 1(c) the step sizes are oriented towards the optimum point which is a desirable property for solving non-separable problems.

Adaptation of the Covariance Matrix. The covariance matrix C that was described in the previous section is calculated based on the changes in the mean values of two successive generations. In this case, it is assumed that the current population contains enough information to successfully estimate the correlations. Equation (5) shows how the covariance matrix is updated based on the changes in two successive generations.

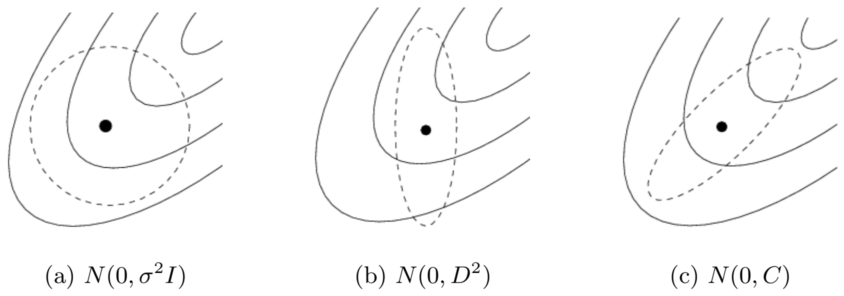

Fig. 1. Figure (a): an isotropic distribution is created when there is only one strategy parameter. Figure (b): shows the situation when there is a separate step size for each of the coordinates. Figure (c): is the case where the coordinate system is rotated by a rotation matrix which is derived from the covariance matrix C.

$$C_{\lambda}^{(g+1)} = \frac{1}{\lambda}\sum_{i=1}^{\lambda}(\boldsymbol{y}_i^{(g+1)} - \boldsymbol{m}^{(g)})(\boldsymbol{y}_i^{(g+1)} - \boldsymbol{m}^{(g)})^T \ , \tag{5}$$

where $\boldsymbol{m}^{(g)}$ is the weighted average of μ selected points from the sample of λ offspring in generation g.

In order to accurately estimate the covariance matrix, CMA-ES uses a technique called *cumulation* to utilise the information in the evolution path [10]. The idea of cumulation is simple. Instead of calculating the covariance matrix using only two consecutive generations, an archive of changes in the mean values is recorded and the covariance is updated based on this archive. It is clear that the archive contains far more information about the correlations as compared to using information from only two consecutive generations.

The next step after calculation of the covariance matrix is to find the rotation matrix M from the covariance matrix C in order to stretch the multivariate Gaussian distribution in the direction of the global optimum point. This can be achieved by performing an eigen-decomposition of the covariance matrix C to obtain an orthogonal basis for the matrix C. This orthogonal basis is essentially the matrix of eigen-vectors that can be used for transformation of the sample points. This process is shown in Equation (6).

$$\begin{aligned} N(\boldsymbol{m}, C) &= \boldsymbol{m} + N(0, C) \\ &= \boldsymbol{m} + C^{\frac{1}{2}} N(0, I) \\ &= \boldsymbol{m} + MDM^T N(0, I) \\ &= \boldsymbol{m} + MDN(0, I) \ , \end{aligned} \tag{6}$$

as it can be seen, $DN(0, I)$ is the situation described in Equation (3) and M which is the rotation matrix derived from eigen-decomposition of the covariance matrix C.

One disadvantage of CMA-ES is its relatively high time complexity. This is mainly due to self-adaptation of covariance matrix and eigen-decomposition. It has been shown that the time complexity of calculating and updating the covariance matrix is of order $O(n^3)$. This makes CMA-ES more computationally expensive compared to other EAs. A few techniques have been proposed to reduce the time complexity of CMA-ES [17], [11], [9], among which sep-CMA-Es was proposed [17]. In a sep-CMA-ES, the covariance matrix C is constrained to be diagonal. This reduces the time complexity from $O(n^3)$ to $O(n^2)$ [17]. However this modification compromises the rotational invariant property of CMA-ES. In another implementation, (1+1)-CMA-ES is proposed in which a very small population size is used [11]. It has been shown that (1+1)-CMA-ES has a time complexity of $O(n^2)$. Although the time complexity of (1+1)-CMA-ES is improved, this modification makes it less appropriate for multimodal test functions due to small population size. A restart CMA-ES has been proposed called IPOP-CMA-ES [1] which is more suitable for multimodal test functions, however IPOP-CMA-ES is only effective for up to 50 dimensions [1]. It is noteworthy that in higher dimensions, very large population sizes are required which significantly increases the computational cost of the algorithm.

3 Experimental Results and Analysis

In this section we present the experimental results of running sep-CMA-ES on CEC'2008 benchmark functions [21]. Tables 1, 2, and 3 contain comparative results of different algorithms on the same benchmark functions. The mean of 25 independent runs are recorded and the best performing algorithms are highlighted in bold.

As it can be seen from Table 1, sep-CMA-ES outperformed other algorithms on 2 out of 6 benchmark functions with 100 decision variables. It is interesting that both of these functions (f_2, f_5) are non-separable. It is also noteworthy that all other algorithms are benefiting from a co-evolutionary framework, and yet sep-CMA-ES performed better on non-separable functions. Another interesting observation is that, sep-CMA-ES performed reasonably better than other algorithms, except for DECC-ML, and DECC-DML, on f_3 which is also a non-separable function. The unique characteristic of DECC-ML, and DECC-DML is that both of them use a uniform random number generator for self-adapting subcomponent sizes. Another observation is that sep-CMA-ES performed poorly on multimodal functions such as f_4, and f_7.

Tables 2, and 3 compare the performance of sep-CMA-ES with other algorithms on 500, and 1000 dimensions respectively. It can be seen that sep-CMA-ES does not have the same relative performance on higher dimensions, in fact, sep-CMA-ES has the best performance only on f_3, but CCEAs are better on all other functions. Overall, the experimental results over all dimensions have shown that sep-CMA-ES does not scale properly as the dimensionality of problem increases. According to Tables 1, 2, and 3 most of the CC algorithms outperformed sep-CMA-ES on almost all of the functions and sep-CMA-ES is often placed last especially on problems with 1000 decision variables.

Table 1. Results of CEC'08 Function on 100 dimensions

Function	DECC	DECC-ML	DECC-D	DECC-DML	MLCC	sep-CMA-ES
f_1	**2.7263e-29**	5.7254e-28	2.9283e-29	4.7379e-28	6.8212e-14	3.1918e-24
f_2	5.4471e+01	2.7974e-04	5.2479e+01	**2.4811e-04**	2.5262e+01	1.3202e+01
f_3	1.4244e+02	1.8871e+02	1.4077e+02	1.9233e+02	1.4984e+02	**4.3300e+00**
f_4	5.3370e+01	**0.0000e+00**	5.4444e+01	**0.0000e+00**	4.3883e-13	2.6324e+02
f_5	2.7589e-03	3.6415e-03	8.8753e-04	7.8858e-04	3.4106e-14	**8.8818e-18**
f_6	2.3646e-01	3.3822e-14	1.2270e-01	**3.1548e-14**	1.1141e-13	6.6495e-01
f_7	-9.9413e+02	-1.5476e+03	-9.8976e+02	**-1.5480e+03**	-1.5439e+03	-1.3625e+03

Table 2. Results of CEC'08 Function on 500 dimensions

Function	DECC	DECC-ML	DECC-D	DECC-DML	MLCC	sep-CMA-ES
f_1	**8.0779e-30**	1.6688e-27	3.8370e-29	1.7117e-27	4.2974e-13	4.2256e-22
f_2	4.0904e+01	1.3396e+00	3.8009e+01	**1.0232e+00**	6.6663e+01	4.8619e+01
f_3	6.6822e+02	5.9341e+02	5.6941e+02	6.8292e+02	9.2466e+02	**3.0788e+02**
f_4	1.3114e+02	**0.0000e+00**	1.4631e+02	**0.0000e+00**	1.7933e-11	1.8262e+03
f_5	2.9584e-04	1.4788e-03	2.9584e-04	2.9584e-04	**2.1259e-13**	9.4260e-02
f_6	6.6507e-14	1.2818e-13	**5.9828e-14**	1.2051e-13	5.3433e-13	8.0505e+00
f_7	-5.5707e+03	**-7.4582e+03**	-4.7796e+03	-7.4579e+03	-7.4350e+03	-6.3868e+03

Table 3. Results of CEC'08 Function on 1000 dimensions

Function	DECC	DECC-ML	DECC-D	DECC-DML	MLCC	sep-CMA-ES
f_1	1.2117e-29	5.1750e-28	**1.0097e-29**	3.3391e-27	8.4583e-13	1.2288e-21
f_2	4.2729e+01	**3.4272e+00**	3.8673e+01	5.81133e+00	1.0871e+02	6.5811e+01
f_3	1.2673e+03	1.0990e+03	1.1597e+03	1.22537e+03	1.7986e+03	**7.9644e+02**
f_4	2.4498e+02	**0.0000e+00**	2.7406e+02	**0.0000e+00**	1.3744e-10	4.2148e+03
f_5	2.9584e-04	9.8489e-04	**1.0392e-15**	1.4611e-15	4.1837e-13	3.6758e-02
f_6	1.3117e-13	2.5295e-13	**1.1866e-13**	2.2908e-13	1.0607e-12	1.9632e+01
f_7^*	-1.4339e+04	**-1.4757e+04**	-1.1035e+04	-1.4750e+04	-1.4703e+04	-1.2419e+04

As it was mentioned earlier in Section 2, sep-CMA-ES has a better time and space complexity as compared to CMA-ES [17], and it has been shown that it outperforms CMA-ES on partially separable problems when the dimension is above 100. On the other hand CMA-ES outperforms sep-CMA-ES on fully non-separable problems, however the full calculation of covariance matrix at the heart of CMA-ES substantially increases its computational cost. The experimental results presented in this paper shows that even sep-CMA-ES performed poorly on benchmark functions especially designed for large scale global optimisation such as CEC'2008 benchmark functions. We speculate that a CC implementation of CMA-ES may hold a great promise and has the benefits of both worlds. CMA-ES has shown superior performance on non-separable functions due to its rotational invariant property and CC is an efficient framework for breaking down a large scale problem into more manageable sub-problems. Using CMA-ES, as the subcomponent optimiser in a CC framework brings the rotational invariant property of CMA-ES with the scalability strength of CC together in one algorithm. Since the covariance matrix adaptation only happens after the full evaluation of all subcomponents in a CC framework, the CMA part happens less frequently, compensating the high computational cost of CMA-ES.

4 Conclusion and Future Works

In this paper we investigated the use of CMA-ES on large scale non-separable problems. CMA-ES is designed as a highly competitive and robust local optimiser. We have shown that the performance of CMA-ES degrades significantly when the dimensionality of the problem increases. This is true for all of EAs, however CMA-ES suffers from the cures of dimensionality more than other CCEA algorithms, since CMA-ES needs a large population size in order to maintain diversity. This is clearly evident from the performance of CMA-ES on multimodal test functions. Having a large population size is not practical in higher dimensions due to exponential growth in computational cost. Eigen-decomposition of the covariance matrix at the heart of CMA-ES is another source of performance degradation on large scale problem.

Despite its shortcomings on large scale problems, CMA-ES remains a competent solution for optimising small to medium-sized non-separable problems. Experimental results in Table 1 confirms this on non-separable functions with 100 decision variables. This property of CMA-ES makes it ideal to be incorporated into a CC framework as a subcomponent optimiser.

In the future, we intend to develop a CC implementation of CMA-ES for further investigation and comparison on large scale non-separable problems, especially the newly proposed CEC'2010 [20] benchmark function.

References

1. Auger, A., Hansen, N.: A restart CMA evolution strategy with increasing population size. In: McKay, B., et al. (eds.) The 2005 IEEE International Congress on Evolutionary Computation (CEC 2005), vol. 2, pp. 1769–1776 (2005)
2. Bäck, T.: Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms. Dover Books on Mathematics. Oxford University Press, Oxford (1996)
3. Bäck, T., Fogel, D.B., Michalewicz, Z. (eds.): Handbook of Evolutionary Computation. Institute of Physics Publishing, Bristol, and Oxford University Press, New York (1997)
4. Bellman, R.E.: Dynamic Programming. Dover Books on Mathematics. Princeton University Press, Princeton (1957)
5. Beyer, H., Schwefel, H.: Evolution strategies - a comprehensive introduction. Natural Computing 1(1), 3–52 (2002)
6. Goldberg, D.E.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley, Reading (1989)
7. Hansen, N.: Benchmarking a BI-population CMA-ES on the BBOB-2009 function testbed. In: Workshop Proceedings of the GECCO Genetic and Evolutionary Computation Conference, pp. 2389–2395. ACM, New York (July 2009)

8. Hansen, N., Kern, S.: Evaluating the CMA evolution strategy on multimodal test functions. In: Yao, X., et al. (eds.) PPSN 2004. LNCS, vol. 3242, pp. 282–291. Springer, Heidelberg (2004)
9. Hansen, N., Muller, S., Koumoutsakos, P.: Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (CMA-ES). Evolutionary Computation 11(1), 1–18 (2003)
10. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation 9(2), 159–195 (2001)
11. Igel, C., Suttorp, T., Hansen, N.: A computational efficient covariance matrix update and a (1+1)-CMA for evolution strategies. In: Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation GECCO, pp. 453–460. ACM, New York (2006)
12. Liu, Y., Yao, X., Zhao, Q., Higuchi, T.: Scaling up fast evolutionary programming with cooperative coevolution. In: Proceedings of Congress on Evolutionary Computation, pp. 1101–1108 (2001)
13. Omidvar, M.N., Li, X., Yang, Z., Yao, X.: Cooperative co-evolution for large scale optimization through more frequent random grouping. In: Proc. of IEEE World Congress on Computational Intelligence (under review) (2010)
14. Omidvar, M.N., Li, X., Yao, X.: Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: Proc. of IEEE World Congress on Computational Intelligence (under review) (2010)
15. Potter, M.A., De Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Proceedings of the Third Conference on Parallel Problem Solving from Nature, vol. 2, pp. 249–257 (1994)
16. Ray, T., Yao, X.: A cooperative coevolutionary algorithm with correlation based adaptive variable partitioning. In: Proc. of IEEE Congress on Evolutionary Computation, pp. 983–989 (May 2009)
17. Ros, R., Hansen, N.: A simple modification in cma-es achieving linear time and space complexity. In: Rudolph, G., Jansen, T., Lucas, S., Poloni, C., Beume, N. (eds.) PPSN 2008. LNCS, vol. 5199, pp. 296–305. Springer, Heidelberg (2008)
18. Salomon, R.: Reevaluating genetic algorithm performance under coordinate rotation of benchmark functions - a survey of some theoretical and practical aspects of genetic algorithms. BioSystems 39, 263–278 (1995)
19. Shi, Y., Teng, H., Li, Z.: Cooperative co-evolutionary differential evolution for function optimization. In: Proc. of the First International Conference on Natural Computation, pp. 1080–1088 (2005)
20. Tang, K., Li, X., Suganthan, P.N., Yang, Z., Weise, T.: Benchmark functions for the cec 2010 special session and competition on large-scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China (2009), `http://nical.ustc.edu.cn/cec10ss.php`
21. Tang, K., Yao, X., Suganthan, P.N., MacNish, C., Chen, Y.P., Chen, C.M., Yang, Z.: Benchmark functions for the cec 2008 special session and competition on large scale global optimization. Technical report, Nature Inspired Computation and Applications Laboratory, USTC, China (2007), `http://nical.ustc.edu.cn/cec08ss.php`
22. van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Transactions on Evolutionary Computation 8(2), 225–239 (2004)
23. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Information Sciences 178, 2986–2999 (2008)