International Journal of Computer Applications (0975 - 8887)
Volume 170 - No.5, July 2017

A Survey on Metaheuristics for Solving Large Scale Optimization Problems

Atinesh Singh
Department of Computer Science and Engineering
National Institute of Technology, Durgapur

Nanda Dulal Jana
Department of Computer Science and Engineering
National Institute of Technology, Durgapur

ABSTRACT

In recent years, there has been a remarkable improvement in the computing power of computers. As a result, numerous real-world optimization problems in science and engineering, possessing very high dimensions, have appeared. In the research community, they are generally labeled as Large Scale Global Optimization (LSGO) problems. Several Metaheuristics has been proposed to tackle these problems. Broadly these algorithms can be categorized in 3 groups: Standard Evolutionary Algorithms, Cooperative Co-evolution (CC) based Evolutionary Algorithms and Memetic Algorithms. This paper gives a brief introduction of some state-of-the-art Metaheuristics used in the field of LSGO, discusses their performance in CEC Competition on LSGO and finally, future scope in this field is presented.

General Terms

Large Scale Global Optimization (LSGO), Cooperative Coevolution (CC)

Keywords

Evolutionary Computation, Large Scale Optimization, Black-Box Optimization, Computational Intelligence

1. INTRODUCTION

LSGO problems appear everywhere in business, science and engineering such as Electric motor power losses minimization, Engineering design optimization, Large scale set covering problems with application to Airline crew scheduling etc. It has become a prominent research area in the field of Evolutionary Computation. In past decades, several Metaheuristics such as Differential Evolution (DE), Particle Swarm Optimization (PSO), Ant Colony Optimization (ACO), Evolution Strategy (ES), etc have been modified to tackle these problems. But all these standard algorithms suffer from a major drawback: their performance gets deteriorated as the dimensionality of the problem increases.

There could be 3 possible reasons for the performance deterioration:

—Landscape complexity increases.

—Search space increases exponentially.

—Function evaluation becomes costly.

Many valuable research papers have been published to tackle LSGO problems. These research papers have been published in top conferences and journals such as IEEE Congress on Evolutionary Computation (CEC), Information Sciences, Lecture Notes in Computing Science, IEEE Transactions on Evolutionary Computation, etc. Numerous benchmark functions have been designed to test the robustness of these algorithms.

The purpose of this paper is to provide a broad overview of metaheuristics used in LSGO.

The remainder of this paper is organized as follows: Section 2 gives a brief description of the metaheuristics used for tackling LSGO problems; Section 3 discusses the results of "CEC Competition on LSGO"; Finally, Section 4 concludes this paper.

2. APPROACHES FOR TACKLING LSGO PROBLEMS

Several algorithms have been proposed to tackle LSGO problems. These algorithms can be roughly classified as follows:

—**Standard Evolutionary Algorithms**: Numerous standard Evolutionary Algorithms have been modified to tackle LSGO problems.

—**CC based Evolutionary Algorithms**: CC based Evolutionary Algorithms follow "divide-n-conquer" based approach. They decompose high dimensional problems into smaller sub-problems with low dimensions, and solve them separately.

—**Memetic Algorithms**: Memetic Algorithms (MAs) are a class of Evolutionary Algorithms which are combined with a problem-specific technique such as local search heuristics, approximation algorithms, etc. The hybridization of problem-specific technique with standard Evolutionary Algorithms is done to enhance the search capability of the algorithms.

2.1 Standard Evolutionary Algorithms

Differential Evolution is one of the state-of-the-art Evolution Algorithm used in the field of Evolutionary Computation. It was first proposed by Rainer Storn [22] in 1996. Since then, it has been used to solve many real-world optimization problems. Differential evolution is preferred over other greedy Evolutionary Algorithms like Particle Swarm Optimisation [4] for solving LSGO problems because it is more explorative in nature. As the dimension of the problem increases landscape complexity, number of local optima,

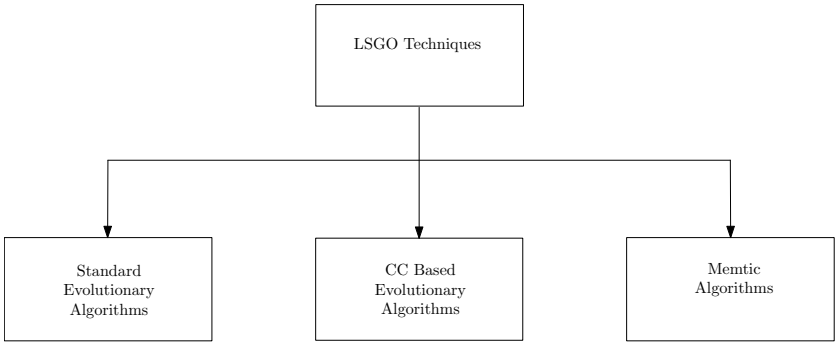

Fig. 1. Different Approaches for tackling LSGO problems.

etc. increases. Hence, Evolutionary Algorithm, which is more exploratory, would be successful for solving LSGO problems. Numerous variants for Differential Evolution have been proposed in the past decades; JADE and SaNSDE are some of the best algorithms among them.

JADE [36] is one of the recent variants of Differential Evolution which is used in high dimensional optimization. In JADE, the authors introduced a new mutation strategy called "DE/current-to-pbest" which is the generalization of "DE/current-to-best". In a new mutation strategy, any of the top p solutions can be randomly chosen to play the role of the single best solution as in "DE/current-to-best". Control parameters in JADE are dynamically adapted using feedback from the evolutionary search. Authors explained that if the control parameters are adapted carefully, then the convergence rate can be improved. They have also suggested to use the "external archive operation" which utilizes historical data to provide information of progress direction. Basically, in external archive operations, unsuccessful points are saved and further used in the future for diversifying the population.

SaNSDE [33] is another powerful variant of DE. It basically combines features of two previous algorithms, SaDE [19] and NSDE [35]. SaDE (Self adaptive differential evolution) dynamically adapts mutation strategy and crossover rate. NSDE, on the contrary, do not consider self-adaption of operators. SaNSDE calculates scale factor as follows:

$$F_i = \begin{cases} N_i(0.5, 0.3), & U_i(0,1) < f_p \\ \delta_i, & \text{otherwise} \end{cases} \quad (1)$$

where $U_i(0,1)$ denotes a uniform random number between 0 and 1, $N_i(0.5, 0.3)$ denotes a Gaussian random number with mean 0.5 and standard deviation 0.3, and δ_i denotes a Cauchy random variable with scale parameter $t = 1$. The parameter f_p will be self adapted as done in SaDE.

SaNSDE introduces SaDE's self-adaptive mechanisms into NSDE.

Mutation scheme, scale factor F and crossover rate CR are the three crucial operators in DE that determine its performance. GaDE [34] adapt these parameters as follows:

(1) Vectors are mutated in the same fashion as in SaNSDE.

(2) The Cauchy distribution with location F_m and scale parameter $t = 0.2$ is used to generate F values.

(3) The Gaussian distribution with mean CR_m and standard deviation 0.1 is used to generate CR values.

For the detailed steps of adapting F and CR, please refer [34]. The different probability distribution is used to generate control parameter F and crossover rate CR. For scale parameter, Cauchy distribution is used and for crossover rate, Gaussian distribution is used.

Sampling values from the Cauchy distribution increase the probability to generate large search step sizes, which is desired in high dimensional function optimization. It has been found that good CR values lie in a small range, so Gaussian distribution with mean 0.1 is used which is very helpful in generating such values.

Hansen's CMA-ES [3] is one of the state-of-the-art Evolutionary Algorithm, but standard CMA-ES is inefficient for solving problems with high dimensions. So Loshchilov revised CMA-ES for tackling LSGO problems and proposed LM-CMA-ES (Limited Memory CMA-ES) [9]. It reconstructs the Cholesky factor and its inverse using $m \ll n$ direction vectors (where n is the number of decision variables). Direction vectors were helpful in optimizing high dimensional problems with large number of depended variables. Loshchilov argued that LM-CMA-ES algorithm can optimize a 1 million dimensional problem.

In [5], Korosec et al. improved the Ant Colony Optimization (ACO) algorithm to tackle LSGO problems. The new ACO-based algorithm is labeled as Differential Ant-Stigmergy Algorithm (DASA). The DASA transforms an optimization problem into a graph-search problem. It assigns parameter differences to the vertices of the graph, and then uses it to navigate through the search space.

2.2 CC based Evolutionary algorithms

Cooperative Co-evolution (CC) framework was proposed by Potter et al. [17] [18]. They adopted a "divide-n-conquer" based strategy, where problem dimension is broken down into small sub-problems and are evolved separately. Sub-problem collaboration occurs only during function evaluation.

CC framework can be summarized as follows [31]:

(1) **Problem Decomposition**: Decision vector of a high-dimensional problem is decomposed into smaller sub-components.

(2) **Subcomponent Optimization**: Each sub-component is evolved separately using certain Evolutionary Algorithm.

(3) **Subcomponents Coadaptation**: Since inter-dependencies may exist between sub-components, co-adaptation is essential in capturing such inter-dependencies during optimization.

Problem decomposition is the most crucial step in CC framework. If problem decomposition step fails to put most of the interacting variables in the same sub-component, then the performance of the algorithm will deteriorate. Previous approaches of problem decomposition include 1-dimensional and splitting-in-half based methods. 1-dimensional approach simply evolves each dimension separately. At first it seems to be a fairly straight forward approach, but it doesn't consider interdependence among variables. It fails to optimize non-separable problems. Splitting-in-half approach splits the whole dimension into two parts and evolve them separately. Though it will capture most of the interacting variables, Due to large sub-component size the performance of the algorithm using Splitting-in-half approach will be nearby the standard Evolutionary Algorithm. So, too small and too big sub-component size is not the optimal choice. Sub-component size should be adapted according to the problem.

CC framework as proposed by the Potter is given in Algorithm 1.

Algorithm 1 CCGA-1

```
gen ← 0
for each species s do
    Pop(gen) ← randomly initialized population
    evaluate fitness of each individual in Pop(gen)
end for
while termination condition = false do
    gen ← gen + 1
    for each species s do
        select Pop(gen) from Pop(gen − 1) based on fitness
        apply genetic operators to Pop(gen)
        evaluate fitness of each individual in Pop(gen)
    end for
end while
```

Yang et al. [31] proposed a CC based algorithm called DECC-G which uses a predefined group size to decompose the decision vector into multiple sub-components. Each sub-component is then optimized separately by using SaNSDE [33]. For co-adaption of the sub-components, instead of using Potters greedy collaboration method, they have used Adaptive weighting strategy. Adaptive weighting strategy applies a weight to each of the sub-component after every cycle, and then evolves the weight vector with Differential Evolution.

Basic steps of the DECC-G algorithm is as follows:

(1) Start a new cycle

(2) Randomly split the n-dimensional decision vector into sub-components of size s i.e., $n = m \times s$, where m is the number of sub-components.

(3) Optimize each sub-component one by one using SaNSDE.

(4) Apply weight vector to each optimized sub-component, and then optimize the weight vector for the best, random and worst member of the current population.

(5) Stop, if stopping criteria is met, otherwise, go to step 1.

Yang et al. proposed MLCC [32] which considers a pool of group size, also called as decomposer pool. Corresponding to each group size a performance record is maintained, which is updated at the end of every cycle.

At the start of a new cycle, a decomposer or group size is selected from the decomposer pool on the basis of its past performance, and then the problem is decomposed into small components by using this decomposer. Each sub-component is evolved in round robin fashion for certain fitness evaluations and then at the end of a cycle, group size performance is stored. The process is repeated until termination criteria is met.

MLCC [32] is further improved by Yang et al. [14]. They introduced 3 techniques to improve its performance. These 3 techniques are summarized as follows:

—More frequent random grouping.

—Removing Adaptive weighting strategy for co-adaption.

—Self-adaption of sub-component sizes.

Authors realized that Adaptive weighting strategy is computationally expensive, and wastes lots of fitness evaluation. So, they suggested that instead of using lots of fitness evaluations in Adaptive weighting, if these fitness evaluations are used to increase the number of cycles, then it will increase the frequency of random grouping which leads to the improvement in the performance of an algorithm.

CC based variant of CMA-ES called CC-CMA-ES for Large Scale Global Optimization problems has been proposed by Liu et al. [8]. In CC-CMA-ES, authors have used two new decomposition strategies that are based on diagonal of the co-variance matrix of CMA-ES.

—Min-Variance decomposition strategy (MiVD)

—Max-Variance decomposition strategy (MaVD)

These new decomposition strategies maintain a balance between exploration and exploitation. An adaptive decomposition strategy scheme is adopted which selects the appropriate decomposition strategy.

The performance of CC based algorithms for tackling LSGO problems depends heavily on optimal group size which is used to decompose the problem into smaller sub-components. Optimal group size for a particular problem tries to group most of the interacting variables into one group so that they can be evolved together. However, finding optimal group size for a particular problem a priori is not feasible as, most of the time, problems are black box. Dependency Identification techniques will be very helpful here to find the optimal group size by identifying the interacting variables. But work done in this area is very limited.

Wicker [30] proposed a simple technique for identifying interacting variables which is based on Potters non greedy collaboration strategy [16].

Wicker's method for recognizing dependency is based on the

observation that, if, by changing the value of dimensions j & k in a decision vector, It achieves better fitness than by changing the value of any one dimension, then there is strong indication that dimensions j & k have dependency and these dimensions should be merged together into the same sub-component.

A new individual for dimension i i.e., $newval_i$ is evaluated by the collaboration of two different candidate solutions $cand_{best} = \langle d_1, ..., d_n \rangle$ and $cand_{random} = \langle d'_1, ..., d'_n \rangle$ which are defined by:

$$d_j = \begin{cases} newval_i, & i = j \\ bestval_j, & \text{otherwise} \end{cases} \quad (2)$$

$$d'_j = \begin{cases} newval_i, & i = j \\ randval_k, & k = j \\ bestval_j, & \text{otherwise} \end{cases} \quad (3)$$

where $bestval_j$ is the value of the best individual in the population for dimension j and $randval_k$ is the value of a randomly chosen individual in the population for dimension k.

Now, the fitness values of the two candidate solutions are used for determining the dependency between variables. If the fitness of $cand_{random}$ is better than the fitness of $cand_{best}$, then a counter for the link (j, k) is increased. At the end of each cycle, counters are analyzed and the dimensions having maximum counters are merged.

For solving Large-Scale Global Optimization problems, Chen et al. [2] proposed Cooperative Coevolution with Variable Interaction Learning (CCVIL). They have used Wicker's [30] dependency identification technique to identify interacting variables so that they can be placed in one group for next evolutionary cycle. CCVIL, dynamically, finds interacting variables through the evolution of the algorithm. It is one of the state-of-the-art algorithms in LSGO field. Though CCVIL is successful in finding optimal group size for most of the problems, it is very computationally expensive. More than 60% of total FE's is consumed in learning stage which leaves only less than 40% for optimization stage.

Omidvar et al. have used a gradient approximation technique in Differential grouping [13] to find the interacting variables so that they can be put in the same sub-component. It is successful in identifying direct interaction among decision variable but fails to identify indirect interaction among decision variable.

Sun et al. [24] improved Differential Grouping technique and proposed Extended Differential Grouping. This new technique successfully identifies direct and indirect interaction among decision variables.

Differential grouping is further improved by Mei et al. [10]. They adopted a modified CMA-ES as the base optimizer for solving sub-problems.

Omidvar et al. [15] proposed DECC-D (Differential Evolution with Cooperative Coevolution using Delta-Grouping) which uses delta value to identify interacting variables. Delta value corresponding to a decision variable is calculated by measuring the amount of change in it, in successive iterations. DECC-D sorts the delta values and then decision variables corresponding to two smallest delta values are merged in one subcomponent.

Some other CC based algorithms for LSGO are Cooperative Co-evolutionary Differential Evolution algorithm with Correlation Identification Grouping (DECC-CIG) [23], Cooperative Coevolution with Variable Grouping and Filled Function (CCVF) [28], and Variable Grouping based Differential Evolution algorithm [29].

2.3 Memetic Algorithms

Multiple Trajectory Search (MTS) [25] initially generates M solutions uniformly distributed over the solution space using Simulated Orthogonal Array (SOA).

During the first iterations MTS conducts a local search on all the M solutions. For further iterations, MTS conducts a local search only on the best solutions among M solutions. MTS uses 3 local search methods. Before applying a local search method to a solution, MTS first checks which local search method better fits the landscape of the solutions neighborhood to do the search. On the basis of the performance of the local search method, the best for a particular solution is chosen.

MTS has been used to solve numerous real-world optimization problems. Aside from its outstanding performance on LSGO problems, it has also given remarkable results on multiobjective optimization problems.

MOS framework [6], rather than relying on a single algorithm, considers a pool of algorithms which can be population-based, local searches, etc. and combines them dynamically.

It uses a High-level Relay Hybrid (HRH) approach which means that algorithms are used in sequence. Each algorithm uses the population generated by the previous algorithm and participation of each algorithm is adjusted dynamically, according to some quality measures.

MA-SW-Chains [12] is another great algorithm for tackling LSGO problems. It uses Solis Wets algorithm as its local search procedure. Throughout the evolution, MA-SW-Chains manages the LS Chains, which helps continuous LS algorithms to better exploit the most promising regions. In this way, the continuous LS algorithms, adaptively, fit its strategy parameters.

Authors participated in CEC competition on LSGO in 2010, where MA-SW-Chains stood 1^{st} among 10 algorithms, proving it to be one of the strongest algorithms for LSGO problems.

Every Evolutionary Algorithm revolves around two important factors: exploration and exploitation. Two-stage based Ensemble Optimization Evolutionary Algorithm, EOEA [26], balances these two factors by using a search procedure which is divided into two stages explained as follows:

(1) **Stage 1** (Global Shrinking Stage): Search region is shrunk to a more promising region by using a search technique which possesses high convergence speed.

(2) **Stage 2** (Local Exploration Stage): A CC based search technique is used to explore the shrunk region to get better solutions.

The objective of the first stage is to find the most promising region in a big solution space and the objective of the second stage is to extensively explore the region shrunk by the previous stage to find a better solution as soon as possible.

For the first stage, an Estimation of Distribution Algorithm (EDA) based on mixed Gaussian and Cauchy models (MUEDA) [27] is used. As the problem dimension increases, and search space becomes large and complex, MUEDA can be used first to locate the most promising region in the big solution space which can be later exploited using some other algorithms. For the second stage CC based algorithm is adopted.

Dynamic Multi-Swarm Particle Swarm Optimizer (DMS-PSO) [7] uses a dynamic and randomized neighborhood topology. It has shown better performance on multi-modal problems but fails on making an efficient local search. A new DMS-PSO, incorporated with a quasi-newton method, is proposed here [37] for Large Scale Global Optimization problems. This quasi-newton method improves the local search ability of DMS-PSO.

Zhao et al. [38] further improve the above procedure by replacing the quasi-newton method with Harmony Search method. This new algorithm is labeled as DMS-PSO-SHS (Dynamic Multi-Swarm Particle Swarm Optimizer with Sub-regional Harmony Search). Authors participated in CEC competition on LSGO in 2010, where DMS-PSO-SHS stood 3^{rd} among 10 algorithms, showing that it is one of the robust algorithms for solving LSGO problems.

There are some CC based algorithms which try to find dependency among decision variables like CCVIL, Differential Grouping, Delta Grouping, etc. but there hasn't been much work proposed that finds dependency among decision variables by using the memetic algorithms. Sayed et al. tried to fill this gap by proposing DIMA [20]. It consists of following two stages:

(1) Dependency Identification and Decomposition.
(2) Optimization and Information Exchange.

At first, DIMA uses the Dependency Identification (DI) technique to identify the dependent variables so that they can be placed into same sub-components. Sub-components are then evolved using a memetic algorithm.

Optimal grouping of variables which is used to decompose LSGO problem into small sub-components should minimize the number of interdependent variables. An interdependent variable is a variable which appears in more than two sub-components. While optimizing a sub-component having an interdependent variable, if its value changes, then all instances of it must be updated in other sub-components too. In order to achieve this, a technique based on information exchange is adopted.

Optimal group size is found by minimizing the least square difference of $F(x)$ and the summation of all $f_k(x_v), v = [1, V]$ ($V =$ dependent variables), as defined in equation 4.

$$min \ \ [F(x) - \sum_{k=1}^{m} f_k(x_v)]^2, v = [1, V] \tag{4}$$

DIMA is revised and a new updated version of DIMA, HDIMA [21], is proposed.

Table 1. 2008 Competition results

Rank	Algorithm	Authors
1	MTS	Tseng et al. [25]
2	LSEDA-gl	Wang et al.
3	jDEdynNP-F	Brest et al. [1]
4	MLCC	Yang et al. [32]

Table 2. 2010 Competition results

Rank	Algorithm	Authors
1	MA-SW-Chains	Molina et al. [12]
2	EOEA	Wang et al. [26]
3	DMS-PSO-SHS	Zhao et al. [38]
4	DASA	Korosec et al. [5]

Table 3. 2013 Competition results

Rank	Algorithm	Authors
1	MOS	LaTorre et al. [6]
2	DECC-G	Yang et al. [31]
3	CC-CMA-ES	Liu et al. [8]
4	VMODE	Ernesto Diaz Lopez

The three main stages of HDIMA are as follows:

(1) Dependency Identification and Decomposition.
(2) Optimization and Information Exchange.
(3) Sub-components aggregation and Optimization.

Dependency identification and decomposition is performed in same way as in DIMA. In the second stage, After the optimization of a sub-component, Information Exchange Mechanism (IEM) is activated so that all the instances of the interdependent variables in different sub-components are updated.

In the last stage, sub-components are aggregated and optimized using Memetic Algorithm (MA) as one large scale problem with IEM deactivated.

3. CEC COMPETITION ON LSGO RESULTS

Congress on Evolutionary Computation (CEC) is one of the most important international conference in the field of Evolutionary Computation. CEC organizes a special competition on Large Scale Global Optimization, where authors compete with their algorithms specially designed for tackling Large scale optimization problems. Competition results of past sessions are mentioned in Table 1, Table 2, Table 3 and Table 4.

4. CONCLUSION AND FUTURE DIRECTIONS

Optimization problems with high dimensions frequently appear in science, engineering, and other disciplines. Over the past decade, various metaheuristics or their modifications have been proposed to tackle these problems. In this paper, we surveyed some state-of-the-art metaheuristics in the field of LSGO. We found two promising approaches for tackling these problems one is CC based algorithms and the other is memetic algorithms.

Numerous CC based algorithms have been proposed for tackling LSGO problems. Most of them rely on random grouping techniques to group interacting variables, but these algorithms are not so powerful. The performance of CC based algorithms could be

Table 4. 2015 Competition results

Rank	Algorithm	Authors
1	MOS	LaTorre et al. [6]
2	IHDELS	Molina et al. [11]
3	CC-CMA-ES	Liu et al. [8]
4	DECC-G (Baseline Model)	Yang et al.

drastically improved by incorporating Dependency Identification (DI) technique. By using DI technique, algorithms will be able to group dependent variable together more efficiently. But very limited work has been done on this issue and more efforts are required.

Memetic algorithms are local search based algorithms. They generally revolves around designing smart local search techniques which can explore complex landscape of high dimensional solution space more efficiently. Memetic algorithms generally do not decompose the problem into smaller sub-components, rather they are applied directly to the original problem.

5. REFERENCES

[1] J. Brest, A. Zamuda, B. Boskovic, M. S. Maucec, and V. Zumer. High-dimensional real-parameter optimization using self-adaptive differential evolution algorithm with population size reduction. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2032–2039, June 2008.

[2] Wenxiang Chen, Thomas Weise, Zhenyu Yang, and Ke Tang. *Large-Scale Global Optimization Using Cooperative Coevolution with Variable Interaction Learning*, pages 300–309. Springer Berlin Heidelberg, Berlin, Heidelberg, 2010.

[3] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, June 2001.

[4] J. Kennedy and R. Eberhart. Particle swarm optimization. In *Neural Networks, 1995. Proceedings., IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948 vol.4, Nov 1995.

[5] P. Korosec, K. Tashkova, and J. Silc. The differential ant-stigmergy algorithm for large-scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[6] A. LaTorre, S. Muelas, and J. M. Pea. Large scale global optimization: Experimental results with mos-based hybrid algorithms. In *2013 IEEE Congress on Evolutionary Computation*, pages 2742–2749, June 2013.

[7] J. J. Liang and P. N. Suganthan. Dynamic multi-swarm particle swarm optimizer. In *Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005.*, pages 124–129, June 2005.

[8] Jinpeng Liu and Ke Tang. Scaling up covariance matrix adaptation evolution strategy using cooperative coevolution. In *Proceedings of the 14th International Conference on Intelligent Data Engineering and Automated Learning — IDEAL 2013 - Volume 8206*, IDEAL 2013, pages 350–357, New York, NY, USA, 2013. Springer-Verlag New York, Inc.

[9] Ilya Loshchilov. A computationally efficient limited memory cma-es for large scale optimization. In *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, GECCO '14, pages 397–404, New York, NY, USA, 2014. ACM.

[10] Yi Mei, Mohammad Nabi Omidvar, Xiaodong Li, and Xin Yao. A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. *ACM Trans. Math. Softw.*, 42(2):13:1–13:24, June 2016.

[11] D. Molina and F. Herrera. Iterative hybridization of de with local search for the cec'2015 special session on large scale global optimization. In *2015 IEEE Congress on Evolutionary Computation (CEC)*, pages 1974–1978, May 2015.

[12] D. Molina, M. Lozano, and F. Herrera. Ma-sw-chains: Memetic algorithm based on local search chains for large scale continuous global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[13] M. N. Omidvar, X. Li, Y. Mei, and X. Yao. Cooperative co-evolution with differential grouping for large scale optimization. *IEEE Transactions on Evolutionary Computation*, 18(3):378–393, June 2014.

[14] M. N. Omidvar, X. Li, Z. Yang, and X. Yao. Cooperative co-evolution for large scale optimization through more frequent random grouping. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[15] M. N. Omidvar, X. Li, and X. Yao. Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[16] Mitchell A. Potter. *The Design and Analysis of a Computational Model of Cooperative Coevolution*. PhD thesis, Fairfax, VA, USA, 1997. UMI Order No. GAX97-28573.

[17] Mitchell A. Potter and Kenneth A. De Jong. *A cooperative coevolutionary approach to function optimization*, pages 249–257. Springer Berlin Heidelberg, Berlin, Heidelberg, 1994.

[18] Mitchell A. Potter and Kenneth A. De Jong. Cooperative coevolution: An architecture for evolving coadapted subcomponents. *Evol. Comput.*, 8(1):1–29, March 2000.

[19] A. K. Qin and P. N. Suganthan. Self-adaptive differential evolution algorithm for numerical optimization. In *2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791 Vol. 2, Sept 2005.

[20] E. Sayed, D. Essam, and R. Sarker. Dependency identification technique for large scale optimization problems. In *2012 IEEE Congress on Evolutionary Computation*, pages 1–8, June 2012.

[21] Eman Sayed, Daryl Essam, and Ruhul Sarker. *Using Hybrid Dependency Identification with a Memetic Algorithm for Large Scale Optimization Problems*, pages 168–177. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[22] R. Storn. On the usage of differential evolution for function optimization. In *Proceedings of North American Fuzzy Information Processing*, pages 519–523, Jun 1996.

[23] J. Sun and H. Dong. Cooperative co-evolution with correlation identification grouping for large scale function optimization. In *2013 IEEE Third International Conference on Information Science and Technology (ICIST)*, pages 889–893, March 2013.

[24] Yuan Sun, Michael Kirley, and Saman Kumara Halgamuge. Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, pages 313–320, New York, NY, USA, 2015. ACM.

[25] Lin-Yu Tseng and Chun Chen. Multiple trajectory search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3052–3059, June 2008.

[26] Y. Wang and B. Li. Two-stage based ensemble optimization for large-scale global optimization. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.

[27] Yu Wang and Bin Li. *A Self-adaptive Mixed Distribution Based Uni-variate Estimation of Distribution Algorithm for Large Scale Global Optimization*, pages 171–198. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[28] F. Wei, Y. Wang, and T. Zong. A novel cooperative coevolution for large scale global optimization. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 738–741, Oct 2014.

[29] F. Wei, Y. Wang, and T. Zong. Variable grouping based differential evolution using an auxiliary function for large scale global optimization. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, pages 1293–1298, July 2014.

[30] K. Weicker and N. Weicker. On the improvement of coevolutionary optimizers by learning variable interdependencies. In *Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406)*, volume 3, page 1632 Vol. 3, 1999.

[31] Zhenyu Yang, Ke Tang, and Xin Yao. Large scale evolutionary optimization using cooperative coevolution. *Information Sciences*, 178(15):2985 – 2999, 2008. Nature Inspired Problem-Solving.

[32] Zhenyu Yang, Ke Tang, and Xin Yao. Multilevel cooperative coevolution for large scale optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1663–1670, June 2008.

[33] Zhenyu Yang, Ke Tang, and Xin Yao. Self-adaptive differential evolution with neighborhood search. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 1110–1116, June 2008.

[34] Zhenyu Yang, Ke Tang, and Xin Yao. Scalability of generalized adaptive differential evolution for large-scale continuous optimization. *Soft Computing*, 15(11):2141–2155, 2011.

[35] Zhenyu Yang, Xin Yao, and Jingsong He. *Making a Difference to Differential Evolution*, pages 397–414. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.

[36] J. Zhang and A. C. Sanderson. Jade: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, Oct 2009.

[37] S. Z. Zhao, J. J. Liang, P. N. Suganthan, and M. F. Tasgetiren. Dynamic multi-swarm particle swarm optimizer with local search for large scale global optimization. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 3845–3852, June 2008.

[38] S. Z. Zhao, P. N. Suganthan, and S. Das. Dynamic multi-swarm particle swarm optimizer with sub-regional harmony search. In *IEEE Congress on Evolutionary Computation*, pages 1–8, July 2010.