Multilevel Cooperative Coevolution for Large Scale Optimization

Zhenyu Yang, Ke Tang and Xin Yao

Abstract— **In this paper, we propose a multilevel cooperative coevolution (MLCC) framework for large scale optimization problems. The motivation is to improve our previous work on grouping based cooperative coevolution (EACC-G) [1], which has a hard-to-determine parameter, group size, in tackling problem decomposition. The problem decomposer takes group size as parameter to divide the objective vector into low dimensional subcomponents with a random grouping strategy. In the MLCC, a set of problem decomposers is constructed based on the random grouping strategy with different group sizes. The evolution process is divided into a number of cycles, and at the start of each cycle MLCC uses a self-adapted mechanism to select a decomposer according to its historical performance. Since different group sizes capture different interaction levels between the original objective variables, MLCC is able to self-adapt among different levels. The efficacy of the proposed MLCC is evaluated on the set of benchmark functions provided by CEC'2008 special session [2].**

I. Introduction

EVOLUTIONARY optimization has achieved great success on many numerical and combinatorial optimization problems in recent years [3]. However, classical evolutionary algorithms (EAs) often lose their efficacy and advantages when applied to large and complex problems, e.g., those with high dimensions. Their performance deteriorates rapidly as the dimensionality of the search space increases [4]. Although cooperative coevolution [5] has been proposed as a promising framework for tackling high-dimensional optimization problems, only limited studies were reported by decomposing a high-dimensional problem into single variables or some low dimensional subcomponents [1], [6].

The cooperative coevolution (CC) [5] framework adopts a divide-and-conquer strategy to solve large and complex problems. High-dimensional optimization by CC can be summarized into three major steps [1]:

1) **Problem Decomposition:** Decompose the high dimensional objective vector into smaller subcomponents.
2) **Subcomponent Optimization:** Evolve each subcomponent separately using a certain EA.
3) **Subcomponents Coadaptation:** Execute coadaptation, i.e., coevolution, to capture interdependencies between different subcomponents.

Since Step 2) has plenty of candidate EAs and Step 3) is often embedded in the fitness evaluation operations, the problem decomposition issue becomes a critical step in the CC framework [7]. Initial efforts in tackling high-dimensional problems used two simple problem decomposition methods, i.e., the one-dimensional based and splitting-in-half strategies [6], [8], [9]. The one-dimensional based strategy decomposes a high-dimensional vector into single variables. Since it did not consider interdependencies among variables, it is unable to tackle nonseparable problems, in which interaction exists between objective variables. The splitting-in-half strategy always decompose a high-dimensional vector into two equal halves and thus reducing an n-dimensional problems into two $\frac{n}{2}$-dimensional problems. If n is large, the $\frac{n}{2}$-dimensional problems would still be very large and challenging to solve.

In [1], we proposed a grouping based CC framework (EACC-G) for high-dimensional optimization problems. It adopts a random grouping for problem decomposition. Variables in the high-dimensional objective vector are divided into several groups randomly according to predefined group size, and each group of variables is optimized by a certain EA. EACC-G is always able to decompose the original problem into small-enough subproblems, and it always provides non-zero probability that interacting variables are optimized together (by assigning them into the same group). So it is promising to perform better than the original CC framework with the one-dimensional based and splitting-in-half problem decomposition strategies.

However, it is often hard to determine an appropriate group size of EACC-G in practice, because the optimal group size is somewhat problem dependent. The small group size is proper for separable problems (the lower the dimension of subproblems, the easier it is to optimize). In contrast, large group sizes are good for nonseparable problems, since it can provide higher probabilities for grouping interacting variables together. Moreover, even for a single problem at different evolution stages, the choice of group size might be a dilemma as well. At the early stages, small group size is helpful to find good regions quickly; but in later stages, large group size, which helps subproblems contain more global information, is important for fine tuning. It would be ideal if the algorithm can adapt to an appropriate group size depending on the objective function, evolution stage and feature characteristics.

Based on the motivations above, we propose a multilevel CC framework (MLCC) in this paper. In the MLCC, we first design several problem decomposers based on different group sizes to form a decomposer pool. Each decomposer in the pool implies different interaction levels between objective variables. The evolution process is divided into a number of *cycles*. At the beginning of each cycle, MLCC selects a decomposer from the decomposer pool based on their performance records. And then, MLCC uses the selected

The authors are with the Nature Inspired Computation and Applications Laboratory, the Department of Computer Science and Technology, University of Science and Technology of China, Hefei, Anhui 230027, China. Xin Yao is also with CERCIA, the School of Computer Science, University of Birmingham, Edgbaston, Birmingham B15 2TT, U.K. (emails: zhyuyang@mail.ustc.edu.cn, ketang@ustc.edu.cn, x.yao@cs.bham.ac.uk).

Corresponding author: Ke Tang (+86–551–3600754).

978-1-4244-1823-7/08/$25.00©2008 IEEE

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

decomposer to divide the objective vector problem into several subcomponents, and evolves each of them with a certain EA. At the end of each cycle, the performance record of the selected decomposer is updated with its performance in current cycle. With such a mechanism, MLCC is able to self-adapt to proper interaction level in spite of the features objective problems and evolution stages.

The rest of this paper is organized as follows: Section II gives the preliminaries; Section III describes the proposed MLCC framework; Section IV presents the experimental results; Finally, Section V concludes this paper briefly.

II. Preliminaries on Cooperative Coevolution

"As evolutionary algorithms are applied to the solution of increasingly complex systems, explicit notions of modularity must be introduced to provide reasonable opportunities for solutions to evolve in the form of interacting coadapted subcomponents."[5] Examples of this show up in the need for rule hierarchies in classifier systems and subroutines in genetic programming [8]. CC is a general framework for applying EAs to large and complex problems using a divide-and-conquer strategy. In CC, the objective system (such as a vector) is decomposed into smaller modules and each of them is assigned to a species (i.e. subpopulation). The species are evolved mostly separately with the only cooperation happening during fitness evaluations.

The original CC framework for high-dimensional optimization can be summarized as follows [6], [8], [9]:

1) Decompose an objective vector into m low dimensional subcomponents.
2) Set $i = 1$ to start a new *cycle*.
3) Optimize the i-th subcomponent with a certain EA for a predefined number of fitness evaluations (FEs).
4) If $i < m$ then $i++$, and go to Step 3.
5) Stop if halting criteria are satisfied; otherwise go to Step 2 for the next *cycle*.

Here a *cycle* consists of one complete evolution of all subcomponents. The main idea is to decompose a high-dimensional problem into some low-dimensional subcomponents and evolve these subcomponents cooperatively for a predefined number of *cycles*. The cooperation occurs only during fitness evaluation. The size of each subcomponent should be within the optimization ability of the EA used.

Potter applied CC to concept learning and neural network construction [5]. García-Pedrajas *et al.* proposed COVNET, which is a new CC model for evolving artificial neural networks [10]. In the domain of learning multiagent behaviors, CC represents a natural approach to applying traditional evolutionary computation [11], [12]. Besides, CC has also been employed in real-world applications, e.g. Cao et al. adopt a CC-based approach in their pedestrian detection system [13].

High-dimensional optimization is another appropriate application of CC. Several CC EAs have been proposed to optimize high-dimensional problems. However, as pointed out in Section I, works in [6], [8], [9] only adopt two simple problem decomposition methods i.e., the one-dimensional based and splitting-in-half strategies. The one-dimensional based strategy decomposes a high-dimensional vector into single variables. Since it did not consider interdependencies among variables, it is unable to tackle nonseparable problems, in which interaction exists between objective variables. The splitting-in-half strategy always decompose a high-dimensional vector into two equal halves and thus reducing an n-dimensional problems into two $\frac{n}{2}$-dimensional problems. If n is large, the $\frac{n}{2}$-dimensional problems would still be very large and challenging to solve. Our previous work on grouping based CC framework (EACC-G) [1] adopts a random grouping strategy for problem decomposition. It divides the objective variables into several groups randomly according to a predefined group size, and then each group of variables is optimized by a certain EA. The decomposition method in EACC-G is more general and promising, but it introduces the "group size" as a hard-to-determine parameter in practice.

III. Multilevel Cooperative Coevolution

A. MLCC: The Multilevel Framework

To overcome the shortcomings of existing CC EAs on high-dimensional optimization problems, we propose a new multilevel cooperative coevolution (MLCC) framework. In the MLCC, we first design several problem decomposers based on different group size to form a decomposer pool. Each decomposer in the pool implies different interaction levels between objective variables. The evolution process is divided into a number of *cycles*. At the beginning of each cycle, MLCC selects a decomposer from the decomposer pool based on their performance records. And then, MLCC uses the selected decomposer to decompose the objective vector problem into several subcomponents, and evolves each of them with a certain EA. At the end of each cycle, the performance record of the selected decomposer is updated with its performance in current cycle. With such a mechanism, MLCC is able to self-adapt to proper interaction level in spite of features objective problems and evolution stages.

Based on the basic ideas above, the details of the framework can be summarized as follows:

1) Assign a set of group sizes, $\mathbf{S} = \{s_1, \cdots, s_t\}$. Each s_i determines a decomposer based on random grouping strategy, which means the objective variables will be divided into several groups, and each group has approximately s_i variables.
2) Create a performance record list $\mathbf{R} = \{r_1, \cdots, r_t\}$. Each $s_i \in \mathbf{S}$ is connected to a $r_i \in \mathbf{R}$. Here r_i is set to 1 initially, and will be updated according to:
$$r_i = \frac{(v - v')}{|v|} \tag{1}$$
where v is the best fitness of last cycle, while v' is the best fitnest of current cycle.
3) Assign a population to the objective vector:
$$\mathbf{U} = \{U(i,j) \mid i = (1, \cdots, NP), j = (1, \cdots, n)\}$$

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

where NP denotes the population size, and n denotes the dimension of objective vector.

4) Evaluate the population $\mathbf{U}$ to record the best fitness value v of current population.
5) If ($r < \epsilon, \forall r \in \mathbf{R}$), go to Step 2 for restart. Here ϵ is a small value to control the lower bound of performance record.
6) Compute the selection probabilities of each decomposer in the decomposer pool:

$$\mathbf{p} = \{p_1, p_2, \cdots, p_t\}$$
$$p_i = \frac{e^{7*r_i}}{\sum_{j=1}^{t} e^{7*r_j}}, i = \{1, 2, \cdots, t\} \quad (2)$$

where constant numbers e and 7 are empirical values. The $\mathbf{p}$ provides a rather high probability to select the decomposer with the best performance, while still gives some chances to the other decomposers for keeping selection diversity.

7) Select s_k from $\mathbf{S}$ based on $\mathbf{p}$, where k is the index of s_k in $\mathbf{S}$.
8) Divide the n-dimensional objective vector into several groups randomly [1] based on the group size s_k, i.e., $\mathbf{G} = \{\mathbf{G}_1, \mathbf{G}_2, \cdots, \mathbf{G}_m\}$ (assuming $n = s_k * m$). Obviously, each $\mathbf{G}_i$ represents a subcomponents of the original objective vector.
9) set $c = 1$.
10) Construct the subpopulation $\mathbf{U}'$ based on the population $\mathbf{U}$ and subcomponent $\mathbf{G}_c$:

$$\mathbf{U}' = \{U(i, j) \mid i \in \{1, \cdots, NP\}, j \in \mathbf{G}_c\}$$

11) Optimize the subcomponent $\mathbf{G}_c$ with subpopulation $\mathbf{U}'$ using a certain EA for a predefined number of FEs.
12) If $c < m$ then $c++$, and go to Step 10.
13) Evaluate population $\mathbf{U}$ to record the best fitness value v' of current population.
14) Update r_k according to Eq. (1), i.e.:

$$r_k = \frac{(v - v')}{|v|}$$

Assuming we are tackling minimization problems, and $v \neq 0$. After r_k is updated, renew v with $v = v'$.

15) Stop if the halting criterion is satisfied; otherwise go to Step 6 for the next cycle.

Choosing an EA as the subcomponent optimizer in Step 11, we immediately get a complete MLCC algorithm for high-dimensional optimization problems. In general, one may employ any existing EA in MLCC. In this paper, we prefer using a recent variant of Differential Evolution (DE) [14], the Self-adaptive Neighbourhood Search DE (SaNSDE) [15].

IV. Experimental Studies

A. Experimental Setup

We evaluate the performance of the proposed MLCC algorithm on the test suites provided by CEC'2008 special session on Large Scale Gobal Optimization [2]. It includes 7 scalable functions, where 2 of them are unimodal problems and other 5 are multimodal problems. Functions f_1, f_4 and f_6 are separable, while other functions are nonseparable. Details of these functions can be found in [2]. For control parameters of MLCC: the population size NP is set to 50; the performance lower bound ϵ is set to 1e-4; and the pool of group sizes are set to $\mathbf{S} = \{5, 10, 25, 50, 100\}$, which is based on the scalability of subcomponent optimizer SaNSDE [15].

Experiments are conducted on 100-D, 500-D and 1000-D of these functions. The maximal fitness evaluation numbers (FEs) of them are set to 5e+5, 2.5e+6 and 5.0+6, respectively. We conducted the experiments for 25 independent runs, and the functions error values defined in [2] are recorded. We are participating in the WCCI'2008 competition on large scale global optimization, so the experimental comparisons with other algorithms will be carried out later by the competition committee.

B. Simulation Results on CEC'2008 Benchmark Functions

The results of 25 independent runs are listed in Tables I–II for 100-D problems, Tables III–IV for 500-D problems and Tables V–VI for 1000-D problems.

TABLE I
Experimental results of 25 independent runs on $f_1 - f_4$, with dimension $D = 100$.

100D		f_1	f_2	f_3	f_4
5e+3	1^{st}	2.2129e+04	8.1707e+01	1.2170e+09	5.5031e+02
	7^{th}	4.4666e+04	1.3307e+02	3.6113e+09	7.6014e+02
	13^{th}	4.8992e+04	1.4268e+02	4.9784e+09	9.3292e+02
	19^{th}	5.4269e+04	1.5356e+02	6.7713e+09	1.0277e+03
	25^{th}	6.3880e+04	1.6532e+02	9.8543e+09	1.1246e+03
	M	4.7179e+04	1.3342e+02	5.2407e+09	8.8588e+02
	Std	1.0278e+04	2.8612e+01	2.1745e+09	1.7925e+02
5e+4	1^{st}	1.8444e-02	3.8225e+01	5.4118e+02	5.9340e+01
	7^{th}	2.8947e-02	4.4734e+01	8.6516e+02	6.5629e+01
	13^{th}	3.6860e-02	4.8974e+01	1.7987e+03	6.9058e+01
	19^{th}	5.8983e-02	5.3913e+01	6.6882e+03	7.8802e+01
	25^{th}	3.3201e-01	6.4411e+01	1.2914e+04	1.0055e+02
	M	7.1643e-02	4.9838e+01	3.8738e+03	7.3530e+01
	Std	8.8372e-02	7.0032e+00	4.1263e+03	1.1015e+01
5e+5	1^{st}	5.6843e-14	1.3481e+01	1.2258e+00	2.8422e-13
	7^{th}	5.6843e-14	1.7326e+01	1.2479e+02	3.4106e-13
	13^{th}	5.6843e-14	2.3186e+01	1.4218e+02	4.5475e-13
	19^{th}	5.6843e-14	3.3394e+01	1.8704e+02	4.5475e-13
	25^{th}	1.1369e-13	4.3485e+01	2.6005e+02	6.8212e-13
	M	6.8212e-14	2.5262e+01	1.4984e+02	4.3883e-13
	Std	2.3206e-14	8.7273e+00	5.7214e+01	9.2126e-14

From the results, we can see that the MLCC performs similarly on the same function with different dimensions, while different performance can be observed on different functions. The MLCC algorithm performs quite well for 4 out the 7 tested functions, including one nonseparable function f_5, and three separable functions f_1, f_4 and f_6. However, MLCC failed to get close enough to the optima of two other functions f_2 and f_3. Functions f_2 and f_3 are completely nonseparable functions, in which interaction exists between

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

TABLE II

EXPERIMENTAL RESULTS OF 25 INDEPENDENT RUNS ON $f_5 - f_7$, WITH DIMENSION $D = 100$.

100D		f_5	f_6	f_7
5.0e+03	1^{st}	1.9758e+02	1.5667e+01	-1.1698e+03
	7^{th}	2.5907e+02	1.8582e+01	-1.0633e+03
	13^{th}	3.6682e+02	1.9070e+01	-9.7477e+02
	19^{th}	4.2131e+02	1.9752e+01	-8.7255e+02
	25^{th}	4.7317e+02	2.0377e+01	-8.4297e+02
	M	3.5036e+02	1.8929e+01	-9.8691e+02
	Std	9.2022e+01	1.1758e+00	1.1181e+02
5.0e+04	1^{st}	6.6698e-03	1.5374e-02	-1.3806e+03
	7^{th}	1.1329e-02	2.6049e-02	-1.3555e+03
	13^{th}	1.7136e-02	4.1944e-02	-1.3483e+03
	19^{th}	3.5749e-02	6.5952e-02	-1.3277e+03
	25^{th}	1.4486e-01	1.1556e+00	-1.2692e+03
	M	2.9245e-02	9.3159e-02	-1.3393e+03
	Std	2.9457e-02	2.2333e-01	2.5128e+01
5.0e+05	1^{st}	2.8422e-14	8.5265e-14	-1.5480e+03
	7^{th}	2.8422e-14	1.1369e-13	-1.5451e+03
	13^{th}	2.8422e-14	1.1369e-13	-1.5448e+03
	19^{th}	2.8422e-14	1.1369e-13	-1.5425e+03
	25^{th}	5.6843e-14	1.1369e-13	-1.5372e+03
	M	3.4106e-14	1.1141e-13	-1.5439e+03
	Std	1.1603e-14	7.8697e-15	2.5283e+00

TABLE III

EXPERIMENTAL RESULTS OF 25 INDEPENDENT RUNS ON $f_1 - f_4$, WITH DIMENSION $D = 500$.

500D		f_1	f_2	f_3	f_4
2.5e+04	1^{st}	1.2481e+05	1.7139e+02	8.1379e+09	2.8353e+03
	7^{th}	1.8909e+05	1.7866e+02	2.6216e+10	4.0651e+03
	13^{th}	2.3669e+05	1.7945e+02	3.0158e+10	4.6832e+03
	19^{th}	2.5030e+05	1.8201e+02	4.0499e+10	5.0131e+03
	25^{th}	2.9364e+05	1.8486e+02	5.2804e+10	5.2395e+03
	M	2.2164e+05	1.7954e+02	3.1363e+10	4.4834e+03
	Std	5.0277e+04	3.5011e+00	1.1718e+10	7.4321e+02
2.5e+05	1^{st}	4.8597e-02	1.5112e+02	2.9761e+03	3.0562e+02
	7^{th}	1.4446e-01	1.5638e+02	3.5668e+03	3.5974e+02
	13^{th}	1.9936e-01	1.5989e+02	3.8968e+03	3.8171e+02
	19^{th}	3.7172e-01	1.6092e+02	4.5706e+03	3.9312e+02
	25^{th}	1.2894e+00	1.6504e+02	9.8119e+03	4.6318e+02
	M	3.2160e-01	1.5882e+02	4.3684e+03	3.8089e+02
	Std	2.9735e-01	3.5741e+00	1.5756e+03	3.8054e+01
2.5e+06	1^{st}	3.9790e-13	5.8375e+01	6.8225e+02	1.2506e-12
	7^{th}	3.9790e-13	6.2115e+01	8.3771e+02	1.4211e-12
	13^{th}	4.5475e-13	6.4780e+01	8.9215e+02	1.5916e-12
	19^{th}	4.5475e-13	7.1790e+01	9.7592e+02	2.0464e-12
	25^{th}	5.1159e-13	7.8811e+01	1.5924e+03	3.0383e-10
	M	4.2974e-13	6.6663e+01	9.2466e+02	1.7933e-11
	Std	3.3145e-14	5.6992e+00	1.7263e+02	6.3110e-11

TABLE IV

EXPERIMENTAL RESULTS OF 25 INDEPENDENT RUNS ON $f_5 - f_7$, WITH DIMENSION $D = 500$.

500D		f_5	f_6	f_7
2.5e+04	1^{st}	9.2379e+02	1.8470e+01	-5.5215e+03
	7^{th}	1.2122e+03	1.9192e+01	-4.8561e+03
	13^{th}	1.9680e+03	1.9599e+01	-4.4780e+03
	19^{th}	2.3047e+03	1.9808e+01	-4.2241e+03
	25^{th}	2.5094e+03	2.0563e+01	-3.9551e+03
	M	1.7640e+03	1.9515e+01	-4.5761e+03
	Std	5.1614e+02	5.1069e-01	4.7230e+02
2.5e+05	1^{st}	5.9812e-03	2.6027e-02	-6.6159e+03
	7^{th}	1.9208e-02	8.2382e-02	-6.5421e+03
	13^{th}	3.0395e-02	1.7418e-01	-6.4616e+03
	19^{th}	5.6041e-02	8.2173e-01	-6.3814e+03
	25^{th}	1.6378e-01	1.3966e+00	-6.1436e+03
	M	4.0365e-02	4.1481e-01	-6.4473e+03
	Std	3.5373e-02	4.7439e-01	1.2893e+02
2.5e+06	1^{st}	1.7053e-13	4.2633e-13	-7.4503e+03
	7^{th}	1.9895e-13	4.8317e-13	-7.4420e+03
	13^{th}	1.9895e-13	5.1159e-13	-7.4342e+03
	19^{th}	2.2737e-13	5.6843e-13	-7.4299e+03
	25^{th}	2.5580e-13	6.8212e-13	-7.4195e+03
	M	2.1259e-13	5.3433e-13	-7.4350e+03
	Std	2.4777e-14	7.0100e-14	8.0308e+00

TABLE V

EXPERIMENTAL RESULTS OF 25 INDEPENDENT RUNS ON $f_1 - f_4$, WITH DIMENSION $D = 1000$.

1000D		f_1	f_2	f_3	f_4
5e+04	1^{st}	2.5464e+05	1.7673e+02	1.6059e+10	5.5748e+03
	7^{th}	3.4969e+05	1.8347e+02	5.4648e+10	7.6248e+03
	13^{th}	4.7100e+05	1.8534e+02	6.3491e+10	9.9253e+03
	19^{th}	4.8789e+05	1.8681e+02	7.9356e+10	1.0320e+04
	25^{th}	5.7902e+05	1.8932e+02	1.0032e+11	1.0540e+04
	M	4.2257e+05	1.8482e+02	6.3426e+10	8.7958e+03
	Std	9.9485e+04	2.7763e+00	2.3158e+10	1.8520e+03
5e+05	1^{st}	1.0956e-01	1.5938e+02	5.8458e+03	5.8990e+02
	7^{th}	1.8695e-01	1.7115e+02	7.2477e+03	6.9701e+02
	13^{th}	3.5893e-01	1.7308e+02	8.0254e+03	7.4242e+02
	19^{th}	6.8615e-01	1.7401e+02	8.6413e+03	8.2982e+02
	25^{th}	2.4695e+00	1.7922e+02	1.0406e+04	1.2198e+03
	M	6.1471e-01	1.7207e+02	7.9146e+03	7.8290e+02
	Std	6.5773e-01	4.0024e+00	1.1156e+03	1.4708e+02
5e+06	1^{st}	7.3896e-13	9.9719e+01	1.4618e+03	2.3874e-12
	7^{th}	7.9581e-13	1.0517e+02	1.7275e+03	2.6148e-12
	13^{th}	8.5265e-13	1.0812e+02	1.8342e+03	2.8422e-12
	19^{th}	9.0950e-13	1.1123e+02	1.8952e+03	4.1496e-12
	25^{th}	9.0950e-13	1.1888e+02	2.0531e+03	1.1415e-09
	M	8.4583e-13	1.0871e+02	1.7986e+03	1.3744e-10
	Std	5.0095e-14	4.7544e+00	1.5809e+02	3.3716e-10

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

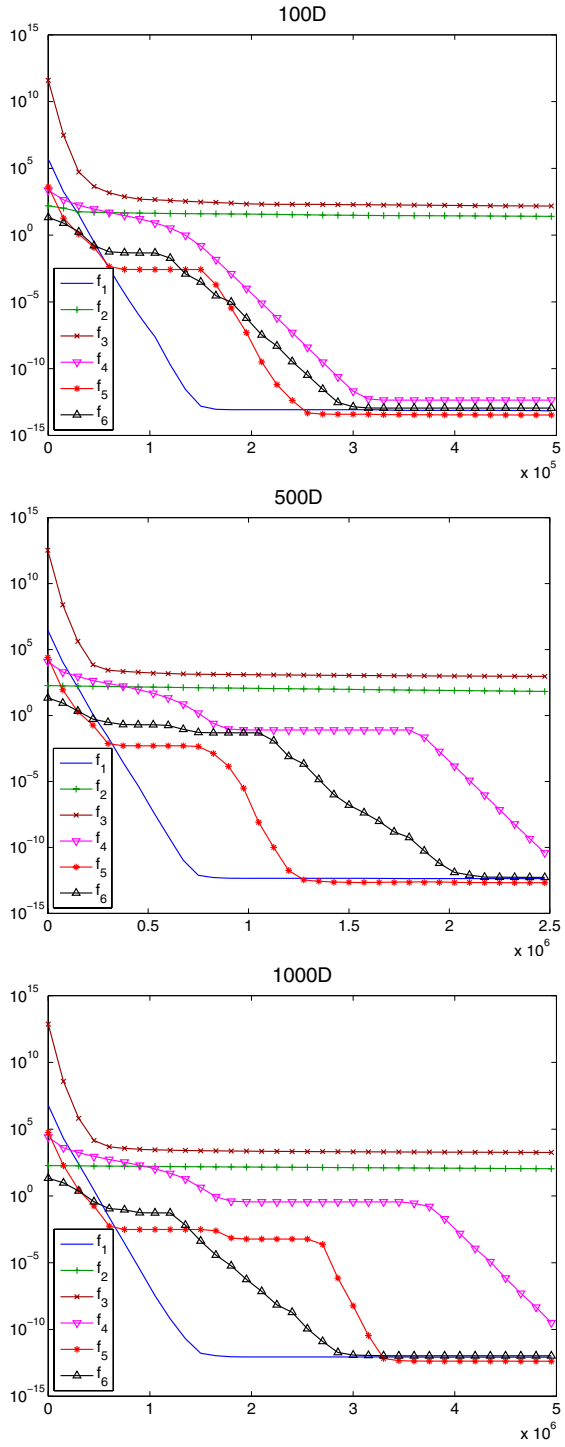

Fig. 1. The evolution curves for functions f_1–f_6 with dimensions D = 100, 500, 1000. The results were averaged over 25 runs. The vertical axis is the function error value and the horizontal axis is the number of FEs.

TABLE VI

EXPERIMENTAL RESULTS OF 25 INDEPENDENT RUNS ON $f_5 - f_7$, WITH DIMENSION $D = 1000$.

1000D		f_5	f_6	f_7
5.0e+04	1^{st}	2.3755e+03	1.8259e+01	-1.0767e+04
	7^{th}	3.1751e+03	1.9533e+01	-9.7903e+03
	13^{th}	4.1580e+03	2.0006e+01	-8.7385e+03
	19^{th}	4.3487e+03	2.0237e+01	-7.9443e+03
	25^{th}	5.0832e+03	2.0506e+01	-7.7460e+03
	M	3.8331e+03	1.9777e+01	-8.8983e+03
	Std	8.6845e+02	5.9326e-01	1.0637e+03
5.0e+05	1^{st}	3.8974e-03	2.5341e-02	-1.3021e+04
	7^{th}	1.1500e-02	5.6260e-02	-1.2867e+04
	13^{th}	2.5602e-02	1.0277e-01	-1.2753e+04
	19^{th}	3.9470e-02	1.7313e-01	-1.2624e+04
	25^{th}	8.7503e-02	1.3723e+00	-1.2319e+04
	M	2.9769e-02	2.0227e-01	-1.2743e+04
	Std	2.2644e-02	3.1404e-01	1.9377e+02
5.0e+06	1^{st}	3.6948e-13	9.0950e-13	-1.4733e+04
	7^{th}	3.9790e-13	1.0232e-12	-1.4712e+04
	13^{th}	4.2633e-13	1.0232e-12	-1.4702e+04
	19^{th}	4.5475e-13	1.1369e-12	-1.4691e+04
	25^{th}	4.5475e-13	1.1937e-12	-1.4680e+04
	M	4.1837e-13	1.0607e-12	-1.4703e+04
	Std	2.7847e-14	7.6844e-14	1.5175e+01

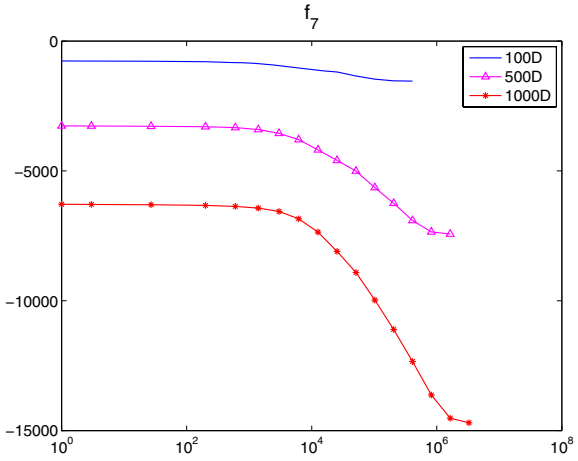

Fig. 2. The evolution curve for function f_7 with dimensions D = 100, 500, 1000. The results were averaged over 25 runs. The vertical axis is the function error value and the horizontal axis is the number of FEs.

any two variables of the objective vector [2]. It is always assumed that decomposition will not work well on such class of problems, and in such a case we can only pursue a better near-optimum with the decomposition based methods [15]. For another function f_7, it is difficult to comment MLCC's performance since the global optimum of f_7 is not known. But the decrease of fitness values partially demonstrates the efficacy of MLCC on this function The evolution process curves of MLCC on the 7 tested functions are shown in Figures 1 and 2. Besides, by comparing the curves from 100-D to 1000-D, we can see that MLCC shows quite good scalability.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

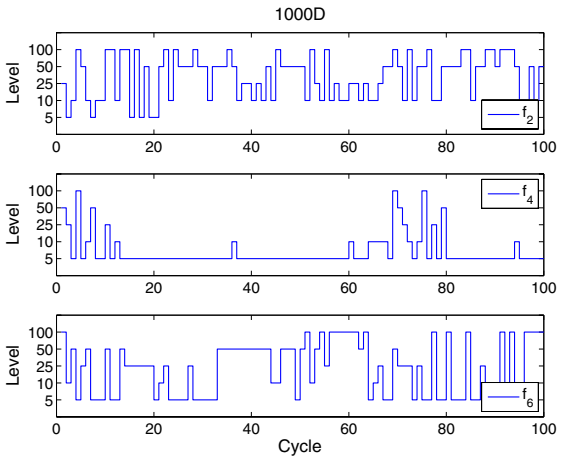

Fig. 3. The self-adaptation curves of MLCC on f_2, f_4 and f_6 with dimensions $D = 1000$. The vertical axes are values of selected level, and the horizontal axis is the number of cycles.

To gain a better understanding of how MLCC works, Figure 3 shows its self-adaptation curves on functions f_2, f_4 and f_6 with dimension D = 1000 (The situations for D = 100 and 500 were omitted to save spaces). For the nonseparable function f_2, it can be seen that MLCC keeps to rather high level during evolution. This is consistent with the assumption that nonseparable functions should adopt decomposer with high interaction level. The situation on the separable function f_4 just verifies the above assumption from the other side: low level decomposer is beneficial to optimizing separable functions. The adaptation process on f_6 is very interesting. On this function, MLCC has been self-adapted from low level to high level, and then to low level again. This might imply that even for a single problem at different evolution stages, preferred decomposers can be different as well. The efficacy of MLCC on this function evidences that the designed self-adaptation mechanism is capable of handling such complex situations.

C. Comparison with Other Algorithms

We have also compared the performance of MLCC with our previous work EACC-G [1] and a MLCC-R, which is the same to MLCC except that the self-adaptive mechanism is replaced with a random strategy to adapt between different levels. The group size used for EACC-G is $s = 5$, or $s = 50$, or $s = 100$. The results for $D = 1000$ are summarized in Table VII.

In general, MLCC has shown better overall performance on the tested set than the other algorithms. It achieved significantly better results than all other algorithms on 5 (f_2 – f_6) out 7 of the functions. For the simple sphere function f_1, all algorithms showed good performance and there is little difference between them. For the other function f_7, MLCC and EACC-G with group size $s = 5$ outperformed others. It can be found that MLCC is better than MLCC-R consistently on most all functions (except they are similar on f_1). This feature verifies the efficacy of the self-adaptation in MLCC. The efficiency of MLCC in comparison to the other algorithms can also be observed from the evolution curves for f_2–f_7 in Figures 4 and 5 (The curve for f_1 is omitted for there is little difference between the compared algorithms).

TABLE VII
COMPARISON BETWEEN MLCC, MLCC-R AND EACC-G, WITH DIMENSION $D = 1000$. ALL RESULTS HAVE BEEN AVERAGED OVER 25 INDEPENDENT RUNS.

1000D	MLCC	MLCC-R	EACC-G $s = 5$	EACC-G $s = 50$	EACC-G $s = 100$
f_1	8.46e-13	8.00e-13	4.30e-12	9.27e-13	**6.78e-13**
f_2	**1.09e+02**†	1.37e+02	1.43e+02	1.36e+02	1.24e+02
f_3	**1.80e+03**†	2.00e+03	1.98e+03	2.18e+03	2.55e+03
f_4	**1.37e-10**†	3.41e-01	1.85e-08	4.56e+02	3.09e+02
f_5	**4.18e-13**†	2.16e-03	1.68e-03	2.21e-03	3.74e-03
f_6	**1.06e-12**†	4.35e-02	1.32e-08	1.14e-12	1.46e+00
f_7	**-1.47e+04**	-1.43e+04	**-1.47e+04**	-1.18e+04	-1.19e+04

† The result is significant better than all others in the same row with a two-tailed t-test of 24 degrees of freedom, and significant at $\alpha = 0.05$.

V. CONCLUSIONS

In this paper, we proposed a multilevel cooperative coevolution (MLCC) framework for large scale optimization problems. The motivation is to improve our previous work on grouping based cooperative coevolution (EACC-G) [1]. In tackling the crucial problem decomposition issue, EACC-G adopts a random grouping strategy to divide the objective variables into several groups. However, the key parameter, i.e. the group size, of the grouping strategy is often problem dependent and thus becomes hard to choose. In the proposed MLCC, a set of group sizes are selected to construct a problem decomposer pool. Each decomposer in the pool is represented by a value indicating group size. Different decomposers are able to produce subcomponents with different interaction levels. The MLCC is designed to self-adapt between different decomposers according to their historical performance. The efficacy of the proposed MLCC algorithm is evaluated on the benchmark functions provided by CEC'2008 special session on Large Scale Global Optimization [2], and the results are encouraging.

ACKNOWLEDGMENT

The authors are grateful to Mr. Tianshi Chen for his valuable comments, which have helped to improve the quality of this paper. This work is partially supported by the National Natural Science Foundation of China (Grant No. 60428202), the Fund for Foreign Scholars in University Research and Teaching Programs (Grant No. B07033), and the Graduate Innovation Fund of University of Science and Technology of China (Grant No. KD2007044).

REFERENCES

[1] Z. Yang, K. Tang, and X. Yao, "Large Scale Evolutionary Optimization Using Cooperative Coevolution," *Submitted to Information Sciences*, 2007.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

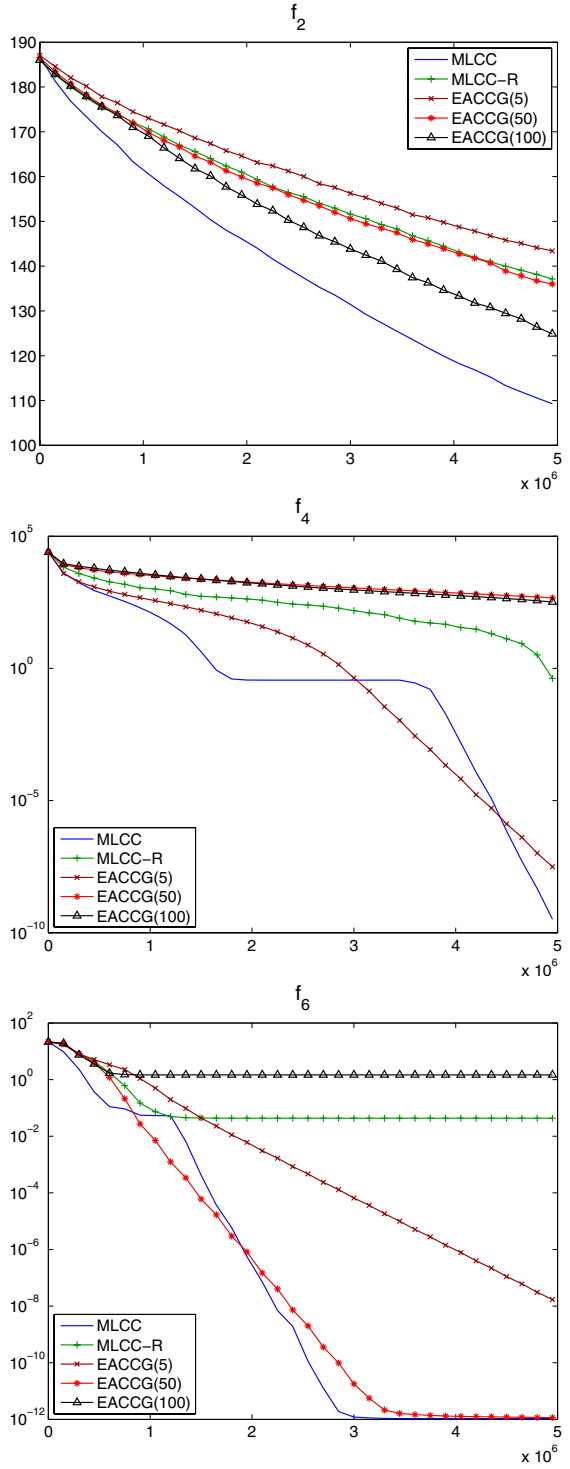

Fig. 4. The evolution curves for functions f_2, f_4 and f_6 with dimensions D = 1000. The results were averaged over 25 runs. The vertical axis is the function error value and the horizontal axis is the number of FEs.

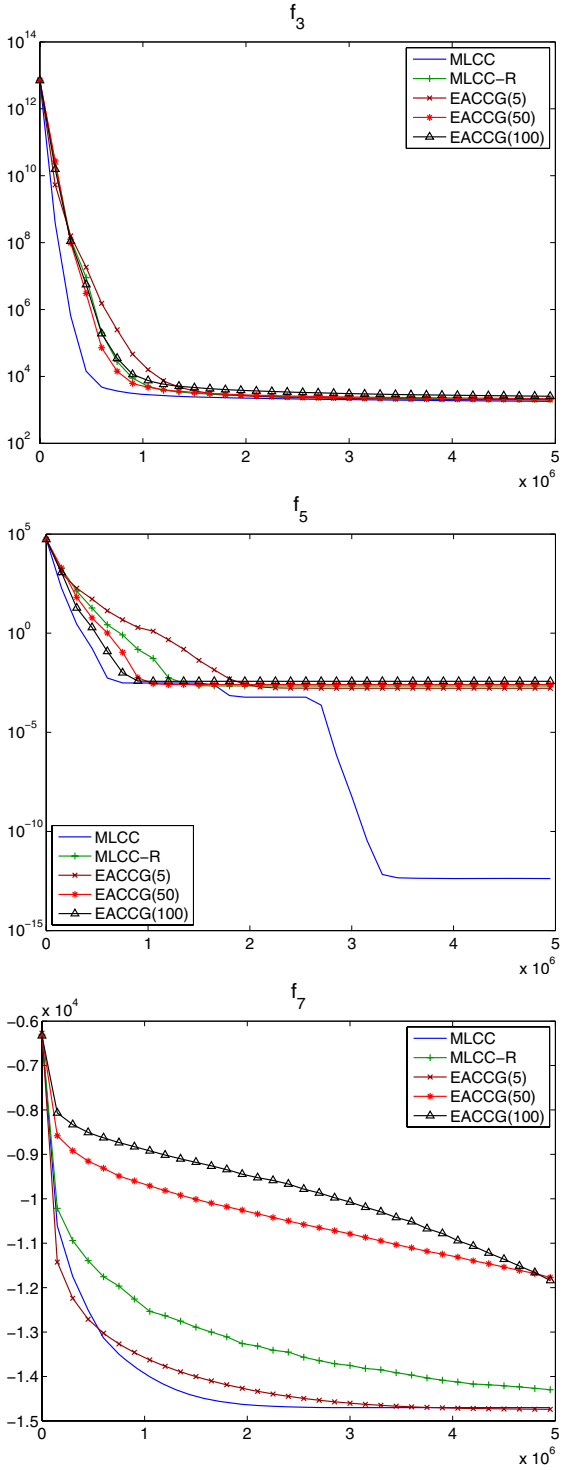

Fig. 5. The evolution curves for functions f_3, f_5 and f_7 with dimensions D = 1000. The results were averaged over 25 runs. The vertical axis is the function error value and the horizontal axis is the number of FEs.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.

[2] K. Tang, X. Yao, P. N. Suganthan, C. MacNish, Y. P. Chen, C. M. Chen, and Z. Yang, "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, http://nical.ustc.edu.cn/cec08ss.php*, 2007.
[3] R. Sarker, M. Mohammadian, and X. Yao, *Evolutionary Optimization*. Kluwer Academic Publishers Norwell, MA, USA, 2002.
[4] F. van den Bergh and A. P. Engelbrecht, "A Cooperative Approach to Particle Swarm Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.
[5] M. Potter, "The Design and Analysis of a Computational Model of Cooperative Coevolution," Ph.D. dissertation, George Mason University, 1997.
[6] Y. Liu, X. Yao, Q. Zhao, and T. Higuchi, "Scaling up Fast Evolutionary Programming with Cooperative Coevolution," *Proceedings of the 2001 Congress on Evolutionary Computation*, pp. 1101–1108, 2001.
[7] M. Potter and K. De Jong, "Cooperative Coevolution: An Architecture for Evolving Coadapted Subcomponents," *Evolutionary Computation*, vol. 8, no. 1, pp. 1–29, 2000.
[8] ——, "A Cooperative Coevolutionary Approach to Function Optimization," *Proceedings of the Third Conference on Parallel Problem Solving from Nature*, vol. 2, pp. 249–257, 1994.
[9] Y. Shi, H. Teng, and Z. Li, "Cooperative Co-evolutionary Differential Evolution for Function Optimization," *Proceedings of the First International Conference on Natural Computation*, pp. 1080–1088, 2005.
[10] N. García-Pedrajas, C. Hervás-Martínez, and J. Muñoz Pérez, "COVNET: a cooperative coevolutionary model for evolving artificial neural networks," *IEEE Transactions on Neural Networks*, vol. 14, no. 3, pp. 575–596, 2003.
[11] L. Panait and S. Luke, "Cooperative Multi-Agent Learning: The State of the Art," *Autonomous Agents and Multi-Agent Systems*, vol. 11, no. 3, pp. 387–434, 2005.
[12] L. Panait, S. Luke, and R. Wiegand, "Biasing Coevolutionary Search for Optimal Multiagent Behaviors," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp. 629–645, 2006.
[13] X. Cao, H. Qiao, and J. Keane, "A Low-Cost Pedestrian-Detection System With a Single Optical Camera," *IEEE Transactions on Intelligent Transportation Sytems*, vol. 8, no. 4, to appear, 2007.
[14] K. Price, R. Storn, and J. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*. Springer-Verlag, ISBN:3-540-20950-6, 2005.
[15] Z. Yang, K. Tang, and X. Yao, "Differential Evolution for High-Dimensional Function Optimization," *Proceedings of the 2007 IEEE Congress on Evolutionary Computation*, pp. 3523–3530, 2007.

Authorized licensed use limited to: UNIVERSITY OF NOTTINGHAM. Downloaded on December 11, 2009 at 07:31 from IEEE Xplore. Restrictions apply.