A Memetic Cooperative Co-evolution Model for Large Scale Continuous Optimization

Yuan Sun[1(✉)], Michael Kirley[2], and Saman K. Halgamuge[3]

[1] Department of Mechanical Engineering, The University of Melbourne, Parkville, VIC 3010, Australia
yuans2@student.unimelb.edu.au

[2] Department of Computing and Information Systems, The University of Melbourne, Parkville, VIC 3010, Australia
mkirley@unimelb.edu.au

[3] Research School of Engineering, The Australian National University, Canberra, ACT 2601, Australia
saman.halgamuge@anu.edu.au

Abstract. Cooperative co-evolution (CC) is a framework that can be used to 'scale up' EAs to solve high dimensional optimization problems. This approach employs a divide and conquer strategy, which decomposes a high dimensional problem into sub-components that are optimized separately. However, the traditional CC framework typically employs only one EA to solve all the sub-components, which may be ineffective. In this paper, we propose a new memetic cooperative co-evolution (MCC) framework which divides a high dimensional problem into several separable and non-separable sub-components based on the underlying structure of variable interactions. Then, different local search methods are employed to enhance the search of an EA to solve the separable and non-separable sub-components. The proposed MCC model was evaluated on two benchmark sets with 35 benchmark problems. The experimental results confirmed the effectiveness of our proposed model, when compared against two traditional CC algorithms and a state-of-the-art memetic algorithm.

Keywords: Cooperative co-evolution · Memetic algorithm · Large scale global optimization · Continuous optimization problem

1 Introduction

Large scale optimization problems are very challenging for evolutionary algorithms (EAs) to solve. This in part may be attributed to the fact that (a) the search space of an optimization problem grows exponentially as the dimensionality increases [1]; (b) the complexity of an optimization problem usually grows as the dimensionality increases [2]; and (c) the computational cost of using some EAs (e.g., estimation of distribution algorithms) when solving very high-dimensional problems is extremely high [3].

Cooperative co-evolution (CC) [4] has been used with some success to 'scale up' EAs to solve high dimensional problems [5–7]. This approach employs a divide and conquer strategy, which decomposes a high dimensional problem into several sub-components that are optimized cooperatively. When optimizing each sub-component, representatives (typically the best sub-solutions found) from other sub-components are combined with individuals in the optimized sub-component, to form complete candidate solutions that can be evaluated. However, the traditional CC framework typically employs only one EA to solve all the sub-components, which may be ineffective.

In this paper, we propose a new memetic cooperative co-evolution (MCC) framework, which employs local search methods to enhance the search of an EA. The proposed MCC framework decomposes a large scale optimization problem into several sub-components based on the variable interaction structures. Then, an EA can be used to solve each sub-component cooperatively. Different local search methods (S operator [8] and R operator [9]) are selected to improve the best solution found by the EA for the separable and non-separable sub-components respectively. The S operator perturbs one decision variable at a time, therefore it is sufficient to solve separable sub-components. The R operator perturbs all the decision variables together to adapt to the local gradient of the fitness landscape, therefore, it is more appropriate to use when attempting to solve non-separable sub-components. The step sizes of the local search methods are updated using the diversity of the current population in the EA.

We have evaluated the efficacy of the proposed MCC framework using benchmark problems from the special sessions on large scale global optimization at CEC'2010 [10] and CEC'2013 [11]. Comprehensive numerical simulations showed that the proposed MCC framework achieved significantly better solution quality than the traditional CC framework. When compared against a state-of-the-art memetic algorithm, it achieved comparable or better solution quality.

The remainder of this paper is organized as follows. Section 2 describes the traditional CC framework. Section 3 describes the proposed MCC framework in detail. Section 4 describes the experiments to evaluate the proposed MCC framework, and analyzes the experimental results. Section 5 concludes the paper.

2 Cooperative Co-evolution

The standard cooperative co-evolution (CC) [4] framework consists of two stages: *decomposition* and *optimization*.

In the decomposition stage, an optimization problem is decomposed into several sub-components. The existing decomposition methods can be classified into two different categories: *predetermined decomposition* and *automatic decomposition*. The predetermined decomposition methods determine the number of sub-components and the size of each sub-component before the decomposition stage starts, e.g., uni-variable grouping [4], S_k grouping [12], random grouping [13], delta grouping [14] and k-means grouping [15]. These methods work well when combined with EAs to solve fully separable problems. However, the performance deteriorates quickly when used to solve partially separable problems

Algorithm 1. Memetic Cooperative Co-evolution

```
1: Automatically decompose a large scale problem into several sub-components
2: while Cycle < CycleMax do
3:    for each sub-component $s_j$ do
4:       Apply an EA on the sub-component $s_j$
5:       if $s_j$ is a separable sub-component then
6:          Apply S operator on the best solution found by the EA
7:       else
8:          Apply R operator on the best solution found by the EA
9:       end if
10:   end for
11: end while
12: return the best solution ever found
```

or fully non-separable problems. The main reason is that such approaches do not take the underlying variable interaction structure into consideration.

The automatic decomposition methods automatically identify and place the interacting decision variables into the same sub-component. It is important to note that automatic decomposition caters to the underlying variable interaction structure encapsulated within the search landscape. Representative automatic decomposition methods include differential grouping [5], extended differential grouping [16], global differential grouping [17], cooperative co-evolution with variable interaction learning [18], statistical variable interdependence learning [19], and the fast variable interdependence searching [20].

In the optimization stage, an evolutionary algorithm can be used to optimize each sub-component based on a context vector. The context vector is a complete candidate solution, typically consisting of the best sub-solutions from each sub-component. When optimizing the i_{th} sub-component, the context vector (excluding the i_{th} sub-solution) is used to combine with the individuals in the i_{th} sub-component, to form complete candidate solutions that can be evaluated. It has been recently found that using only one context vector may be too greedy [21]. Therefore, the adaptive multi-context CC [21] framework is proposed, which employs more than one context vector to co-evolve sub-components.

3 Memetic Cooperative Co-evolution

In this section, the proposed memetic cooperative co-evolution (MCC) model is described in detail (Algorithm 1).

In the decomposition stage, any automatic decomposition method can be used to divide a large scale optimization problem into several sub-components. An automatic method decomposes an optimization problem based on the underlying structure of variable interactions. Taking the following objective function as an example

$$f(\boldsymbol{x}) := x_1^2 + x_2^2 + (x_3 - x_4)^2 + (x_4 - x_5)^2 + (x_6 - x_7)^2, \boldsymbol{x} \in [-1, 1]^7, \tag{1}$$

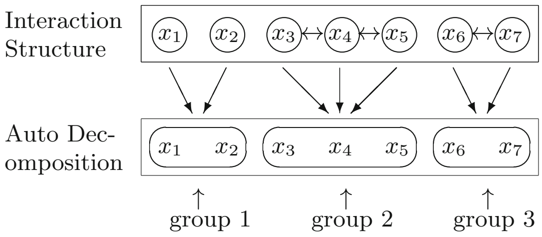
Interaction Structure
x_1 x_2 $x_3 \leftrightarrow x_4 \leftrightarrow x_5$ $x_6 \leftrightarrow x_7$
Auto Decomposition
x_1 x_2 x_3 x_4 x_5 x_6 x_7
group 1
group 2
group 3

Fig. 1. The variable interaction structure and the automatic decomposition of the objective function given in Eq. (1). The notation $x_i \leftrightarrow x_j$ denotes that decision variable x_i directly interacts with x_j.

decision variables $\{x_3, x_4, x_5\}$ interact, as well as $\{x_6, x_7\}$. Therefore, the decision variables should be divided into three sub-components $\{x_1, x_2\}$, $\{x_3, x_4, x_5\}$ and $\{x_6, x_7\}$, as shown in Fig. 1.

It is important to note that the level of interaction between given decision variables may be different. For example, in Eq. (1), both (x_3, x_4) and (x_3, x_5) interact with each other. However, x_3 and x_4 interact directly; x_3 and x_5 are linked by x_4. The former is called *direct interaction* and the latter is called *indirect interaction*. The formal definitions of direct interaction and indirect interaction are described in [16,22].

In the optimization stage, an EA can be used to solve each sub-component cooperatively. If the sub-component consists of a group of separable decision variables, a local search method – S operator [8] is used to further improve the best solution found by the EA. The S operator perturbs one decision variable at a time, therefore it is sufficient to solve separable problems. If the sub-component consists of a group of non-separable decision variables, the R operator [9] is used to further improve the best solution found by the EA. The R operator perturbs all the decision variables together to adapt to the local gradient of the fitness landscape, therefore, it is more appropriate to use when attempting to solve non-separable problems. The differences between the S and R operators are illustrated in Fig. 2.

The step sizes of the S and R operators are updated using the diversity of the current population in the EA:

$$\mathit{Step\_S} = \min\big(r, 0.1(\boldsymbol{ub} - \boldsymbol{lb})\big),\ \mathit{Step\_R} = \min\big(r, 0.04(\boldsymbol{ub} - \boldsymbol{lb})\big), \tag{2}$$

where $\boldsymbol{ub}$ and $\boldsymbol{lb}$ are the upper bounds and lower bounds of the search space, and r is the diversity of the current population, which is estimated as follows:

$$r = \frac{1}{N}\sum_{i=1}^{N} ||\boldsymbol{x}_{best}, pop(i,:)||_2, \tag{3}$$

where pop is the current population, $\boldsymbol{x}_{best}$ is the best solution in the current population, and N is the population size. Please note that the initial step size for S operator is larger than the initial step size for R operator. The reason for

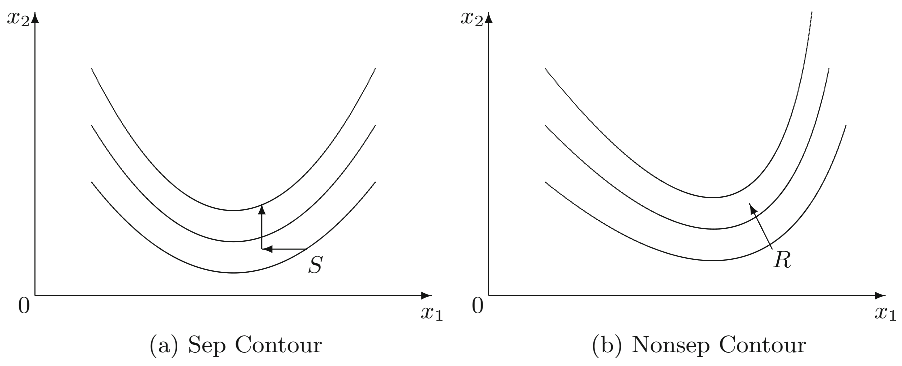

Fig. 2. The search approaches of the S and R operators. Figure (a) and (b) are the contours of a separable and non-separable fitness landscapes respectively. The S operator searches in the direction of each decision variable. The R operator searches in the direction of the local derivative.

this is that S operator always decreases the step size during the search, while the R operator can increase or decrease the step size later on.

4 Experiments

4.1 Methodology

The decomposition method – extended differential grouping (XDG) [16] and the EA – Self-adaptive Differential Evolution with Neighborhood Search (SaNSDE) [23] were embedded into the proposed memetic cooperative co-evolution (MCC) model to evaluate its efficacy. The XDG method was used, as it can identify both direct and indirect variable interactions. The SaNSDE algorithm was selected for its good performance and wide usage to solve large scale optimization problems. We denote the proposed memetic algorithm as MCC-XDG.

The proposed MCC-XDG algorithm was used to solve the CEC'2010 [10] and CEC'2013 [11] large scale benchmark problems. The maximum number of function evaluations was set to 3×10^6, divided between the decomposition stage and optimization stage. The threshold value for XDG was set to 0.1, and the population size for SaNSDE was set to 50. In each cycle, the maximal number of iterations for SaNSDE was set to 200, and the maximal number of function evaluations for local search methods was set to $10d$, where d is the dimensionality. For each benchmark problem, the median, mean and standard deviation of the best solutions found by the MCC-XDG algorithm based on 30 independent runs were recorded.

The performance of the MCC-XDG algorithm was compared with the performance of two traditional CC algorithms: DECC-XDG (SaNSDE with XDG) and DECC-G (SaNSDE with random grouping [13]), as well as a state-of-the-art memetic algorithm: MA-SW-Chains [24]. The MA-SW-Chains algorithm assigns to each individual a local search intensity that depends on its features, by chaining different local search applications. It achieved the best performance in the CEC 2010 special session and competition on large scale global optimization. The parameter settings for the three algorithms were consistent with the original papers. The Wilcoxon rank-sum test (significance level $\alpha = 0.05$) with Holm p-value correction [25] was conducted in a pairwise fashion to find the best performing algorithm.

4.2 Results

The experimental results of the proposed MCC-XDG algorithm when used to solve the CEC'2010 and CEC'2013 benchmark problems are presented in Table 1 and Table 2 respectively. It achieved the best solution quality on 17 out of 20 CEC'2010 benchmark problems and 8 out of 15 CEC'2013 benchmark problems when compared against three other algorithms. The experimental results showed that the MCC-XDG algorithm can solve some of the benchmark problems to great accuracy (median), e.g., CEC'2010 f_1, f_3, f_6 to f_8, f_{11}, f_{12}, f_{16} and f_{17}.

Comparison with DECC-XDG. The proposed MCC-XDG algorithm achieved equal or better results across all the CEC'2010 and CEC'2013 benchmark problems compared against the DECC-XDG algorithm. In some cases, the median of the best solution found by the MCC-XDG is much better than that found by the DECC-XDG algorithm. Taking CEC'2010 f_7 as an example, the median of the best solution found by MCC-XDG is 5.86×10^{-21}, which is much smaller (better) than the median of the best solution found by DECC-XDG (2.34×10^{2}). It is important to note that the only difference between the MCC-XDG and DECC-XDG algorithms is that MCC-XDG uses local search methods to enhance the search of the EA – SaNSDE, while DECC-XDG only uses SaNSDE to solve each sub-component. Therefore, the experimental results confirmed the effectiveness of the MCC model and the local search methods.

Comparison with DECC-G. The proposed MCC-XDG algorithm achieved equal or better results than the DECC-G algorithm across all the benchmark problems except for CEC'2010 f_2 and CEC'2013 f_2. The DECC-G algorithm uses a predetermined decomposition method – random grouping. On CEC'2010 f_2 and CEC'2013 f_2, the DECC-G achieved the best solution quality when compared against the other three algorithms. However, on other benchmark problems especially partially non-separable problems (CEC'2010 f_4 to f_{18} and CEC'2013 f_4 to f_{11}), the DECC-G algorithm was outperformed by the other three algorithms. The reason for this is that the DECC-G algorithm (random grouping) decomposes an optimization problem without considering any variable interaction.

Comparison with MA-SW-Chains. The proposed MCC-XDG algorithm achieved comparable or better results than a state-of-the-art memetic algorithm

Table 1. The results of the proposed MCC-XDG algorithm when used to solve the CEC'2010 benchmark problems. The MCC-XDG algorithm is compared with DECC-XDG, DECC-G and MA-SW-Chains. The best performances are highlighted in bold (Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction).

Func	Stats	MCC-XDG	DECC-XDG	DECC-G	MA-SW-Chains
f_1	Median	**0.00e+00**	5.57e+02	6.06e-14	2.67e-14
	Mean	**6.08e-29**	1.37e+04	9.14e-14	3.80e-14
	Std	**2.21e-28**	4.11e+04	7.87e-14	4.91e-14
f_2	Median	2.96e+03	4.42e+03	**1.16e+02**	8.47e+02
	Mean	3.04e+03	4.43e+03	**1.13e+02**	8.40e+02
	Std	2.51e+02	1.56e+02	**2.64e+01**	4.88e+01
f_3	Median	**1.42e-14**	1.68e+01	1.79e+00	5.16e-13
	Mean	**7.55e-01**	1.67e+01	1.77e+00	5.76e-13
	Std	**3.77e+00**	3.53e-01	3.14e-01	2.73e-13
f_4	Median	3.55e+11	7.38e+11	1.17e+13	**3.10e+11**
	Mean	3.73e+11	7.37e+11	1.09e+13	**2.97e+11**
	Std	1.41e+11	1.44e+11	2.83e+12	**6.19e+10**
f_5	Median	**8.15e+07**	1.54e+08	2.25e+08	2.30e+08
	Mean	**8.64e+07**	1.53e+08	2.46e+08	2.18e+08
	Std	**2.55e+07**	2.27e+07	5.40e+07	5.75e+07
f_6	Median	**3.55e-09**	1.64e+01	4.94e+06	2.45e+00
	Mean	**4.10e+04**	1.63e+01	5.03e+06	1.42e+05
	Std	**2.05e+05**	3.60e-01	8.77e+05	3.96e+05
f_7	Median	**5.86e-21**	2.34e+02	4.40e+06	7.94e-03
	Mean	**6.21e-21**	7.50e+02	5.13e+06	1.17e+02
	Std	**2.02e-21**	1.62e+03	3.69e+06	2.37e+02
f_8	Median	**8.18e-19**	6.55e+00	8.71e+07	2.76e+06
	Mean	**1.43e+06**	4.78e+05	7.34e+07	6.90e+06
	Std	**1.95e+06**	1.32e+06	3.16e+07	1.90e+07
f_9	Median	**1.41e+06**	1.12e+08	2.43e+08	1.48e+07
	Mean	**1.59e+06**	1.15e+08	2.41e+08	1.49e+07
	Std	**8.42e+05**	1.33e+07	2.67e+07	1.61e+06
f_{10}	Median	2.32e+03	5.23e+03	9.47e+03	**2.02e+03**
	Mean	2.33e+03	5.23e+03	9.28e+03	**2.01e+03**
	Std	1.14e+02	1.40e+02	1.29e+03	**1.59e+02**
f_{11}	Median	**1.62e-05**	1.07e+01	2.53e+01	3.77e+01
	Mean	**3.97e-01**	1.08e+01	2.51e+01	3.86e+01
	Std	**7.17e-01**	8.89e-01	1.45e+00	8.06e+00
f_{12}	Median	**2.83e-06**	1.21e+04	4.49e+04	**3.09e-06**
	Mean	**4.40e-06**	1.23e+04	4.47e+04	**3.24e-06**
	Std	**6.20e-06**	2.50e+03	5.11e+03	**5.78e-07**
f_{13}	Median	**8.83e+00**	3.83e+03	3.12e+03	8.61e+02
	Mean	**1.86e+01**	3.76e+03	3.99e+03	9.83e+02
	Std	**3.12e+01**	1.34e+03	2.52e+03	5.66e+02
f_{14}	Median	**1.05e+07**	6.01e+08	5.88e+08	3.23e+07
	Mean	**1.07e+07**	5.97e+08	5.85e+08	3.25e+07
	Std	**3.08e+06**	3.42e+07	4.44e+07	2.46e+06
f_{15}	Median	**2.48e+03**	6.35e+03	6.63e+03	2.67e+03
	Mean	**2.46e+03**	6.34e+03	8.60e+03	2.68e+03
	Std	**9.87e+01**	9.01e+01	3.22e+03	9.95e+01
f_{16}	Median	**2.23e-12**	**1.78e-08**	7.89e+01	9.32e+01
	Mean	**5.59e-01**	**1.77e-08**	7.76e+01	9.95e+01
	Std	**8.66e-01**	**1.83e-09**	1.46e+01	1.53e+01
f_{17}	Median	**5.54e-01**	1.25e+05	1.78e+05	1.28e+00
	Mean	**8.51e-01**	1.26e+05	1.76e+05	1.27e+00
	Std	**9.23e-01**	5.34e+03	1.02e+04	1.24e-01
f_{18}	Median	**3.11e+02**	1.36e+03	2.57e+04	1.41e+03
	Mean	**3.77e+02**	1.38e+03	2.44e+04	1.57e+03
	Std	**2.61e+02**	1.39e+02	1.24e+04	6.73e+02
f_{19}	Median	**2.64e+05**	1.72e+06	7.87e+05	3.75e+05
	Mean	**2.67e+05**	1.73e+06	7.74e+05	3.80e+05
	Std	**2.56e+04**	1.14e+05	3.94e+04	2.34e+04
f_{20}	Median	**6.80e+02**	3.49e+04	3.36e+03	1.04e+03
	Mean	**6.88e+02**	2.01e+05	3.39e+03	1.06e+03
	Std	**1.66e+02**	8.09e+05	3.04e+02	9.38e+01

Table 2. The results of the proposed MCC-XDG algorithm when used to solve the CEC'2013 benchmark problems. The MCC-XDG algorithm is compared with DECC-XDG, DECC-G and MA-SW-Chains. The best performances are highlighted in bold (Wilcoxon rank-sum tests ($\alpha = 0.05$) with Holm p-value correction).

Func	Stats	MCC-XDG	DECC-XDG	DECC-G	MA-SW-Chains
f_1	Median	**0.00e+00**	5.32e-01	1.31e-11	7.12e-13
	Mean	**3.16e-29**	3.73e+01	2.58e-11	1.34e-12
	Std	**9.51e-29**	1.24e+02	3.83e-11	2.45e-12
f_2	Median	5.50e+03	1.29e+04	**8.24e+01**	1.24e+03
	Mean	5.81e+03	1.27e+04	**8.53e+01**	1.25e+03
	Std	1.29e+03	6.40e+02	**2.71e+01**	1.05e+02
f_3	Median	2.01e+01	2.13e+01	2.01e+01	**6.83e-13**
	Mean	2.01e+01	2.13e+01	2.01e+01	**6.85e-13**
	Std	1.27e-02	1.64e-02	3.10e-03	**2.12e-13**
f_4	Median	**3.28e+09**	7.87e+09	8.48e+10	**2.75e+09**
	Mean	**3.43e+09**	8.07e+09	9.00e+10	**3.81e+09**
	Std	**1.03e+09**	2.02e+09	3.63e+10	**2.73e+09**
f_5	Median	4.22e+06	4.00e+06	8.61e+06	**2.03e+06**
	Mean	4.21e+06	4.21e+06	8.27e+06	**2.25e+06**
	Std	9.80e+05	6.86e+05	1.32e+06	**1.30e+06**
f_6	Median	1.00e+06	1.06e+06	1.05e+06	**6.33e+02**
	Mean	1.00e+06	1.06e+06	1.05e+06	**1.86e+04**
	Std	1.28e+04	1.32e+03	1.44e+03	**2.54e+04**
f_7	Median	**1.27e+04**	1.40e+07	2.82e+08	4.03e+06
	Mean	**1.47e+04**	1.40e+07	3.53e+08	3.85e+06
	Std	**1.03e+04**	5.88e+06	2.35e+08	6.34e+05
f_8	Median	6.30e+13	2.77e+14	2.50e+15	**4.60e+13**
	Mean	7.72e+13	3.16e+14	2.90e+15	**4.62e+13**
	Std	4.10e+13	1.89e+14	1.31e+15	**9.02e+12**
f_9	Median	2.60e+08	4.92e+08	5.68e+08	**1.36e+08**
	Mean	2.55e+08	4.90e+08	5.94e+08	**1.44e+08**
	Std	4.69e+07	2.83e+07	1.36e+08	**2.55e+07**
f_{10}	Median	9.12e+07	9.42e+07	9.28e+07	**3.34e+02**
	Mean	9.14e+07	9.43e+07	9.29e+07	**3.72e+04**
	Std	8.53e+05	3.20e+05	5.88e+05	**6.25e+04**
f_{11}	Median	**7.09e+06**	6.08e+08	5.19e+10	2.10e+08
	Mean	**1.27e+07**	6.27e+08	5.93e+10	2.10e+08
	Std	**1.20e+07**	2.84e+08	4.23e+10	2.35e+07
f_{12}	Median	**6.63e+02**	3.82e+03	3.35e+03	1.25e+03
	Mean	**7.03e+02**	4.40e+03	3.41e+03	1.24e+03
	Std	**1.84e+02**	2.02e+03	2.85e+02	8.33e+01
f_{13}	Median	**1.99e+06**	1.01e+09	5.56e+09	1.91e+07
	Mean	**3.14e+06**	1.22e+09	5.74e+09	1.98e+07
	Std	**2.25e+06**	5.13e+08	2.37e+09	2.30e+06
f_{14}	Median	**1.20e+07**	2.34e+09	6.35e+10	1.43e+08
	Mean	**1.25e+07**	3.44e+09	7.68e+10	1.45e+08
	Std	**3.18e+06**	2.94e+09	4.96e+10	1.60e+07
f_{15}	Median	**6.63e+05**	9.65e+06	5.03e+06	5.80e+06
	Mean	**6.67e+05**	1.00e+07	5.13e+06	5.98e+06
	Std	**1.59e+05**	1.51e+06	4.36e+05	1.42e+06

– MA-SW-Chains across the CEC'2010 benchmark problems. The main difference between the MCC-XDG and MA-SW-Chains algorithms is that MCC-XDG solves an optimization problem by a divide and conquer strategy, while MA-SW-Chains solves an optimization problem as a whole. Therefore, the experimental results confirmed the effectiveness of the divide and conquer strategy – cooperative co-evolution. It is important to note that on CEC'2010 f_3, the proposed MCC-XDG algorithm was reported to outperform the MA-SW-Chains algorithm, in spite of the fact that the mean of the best solution found by MCC-XDG is worse than the mean of the best solution found by MA-SW-Chains. The reason for this is that the Wilcoxon rank sum test examines whether two samples are from continuous distributions with equal medians instead of means. On the CEC'2013 benchmark problems, the proposed MCC-XDG algorithm performed equally well with the MA-SW-Chains algorithm.

5 Conclusion

In this paper, we have investigated the effectiveness of the cooperative co-evolution framework when used to 'scale up' EAs to solve large scale optimization problems. A new memetic cooperative co-evolution framework was proposed, which employs local search methods (S and R operators) to enhance the search of an EA to solve separable and non-separable sub-components. Comprehensive experimental results showed that the proposed memetic cooperative co-evolution framework improved the performance of the traditional cooperative co-evolution framework. When compared against a state-of-the-art memetic algorithm, it achieved comparable or better solution quality.

References

1. Omidvar, M.N., Li, X., Tang, K.: Designing benchmark problems for large-scale continuous optimization. Inf. Sci. **316**, 419–436 (2015)
2. Weise, T., Chiong, R., Tang, K.: Evolutionary optimization: pitfalls and booby traps. J. Comput. Sci. Technol. **27**(5), 907–936 (2012)
3. Dong, W., Chen, T., Tino, P., Yao, X.: Scaling up estimation of distribution algorithms for continuous optimization. IEEE Trans. Evol. Comput. **17**(6), 797–822 (2013)
4. Potter, M.A., Jong, K.A.: A cooperative coevolutionary approach to function optimization. In: Davidor, Y., Schwefel, H.-P., Männer, R. (eds.) PPSN 1994. LNCS, vol. 866, pp. 249–257. Springer, Heidelberg (1994). doi:10.1007/3-540-58484-6_269
5. Omidvar, M.N., Li, X., Mei, Y., Yao, X.: Cooperative co-evolution with differential grouping for large scale optimization. IEEE Trans. Evol. Comput. **18**(3), 378–393 (2014)
6. Mei, Y., Li, X., Yao, X.: Cooperative coevolution with route distance grouping for large-scale capacitated arc routing problems. IEEE Trans. Evol. Comput. **18**(3), 435–449 (2014)
7. Tan, K.C., Yang, Y., Goh, C.K.: A distributed cooperative coevolutionary algorithm for multiobjective optimization. IEEE Trans. Evol. Comput. **10**(5), 527–549 (2006)

8. Tseng, L., Chen, C.: Multiple trajectory search for large scale global optimization. In: IEEE Congress on Evolutionary Computation, CEC 2008, IEEE World Congress on Computational Intelligence, pp. 3052–3059. IEEE (2008)
9. Rosenbrock, H.: An automatic method for finding the greatest or least value of a function. Comput. J. **3**(3), 175–184 (1960)
10. Tang, K., Yao, X., Suganthan, P.: Benchmark functions for the CEC 2010 special session and competition on large scale global optimization. Technique report, USTC, Natrue Inspired Computation and Applications Laboratory, no. 1, pp. 1–23 (2010)
11. Li, X., Tang, K., Omidvar, M.N., Yang, Z., Qin, K.: Benchmark functions for the CEC 2013 special session and competition on large-scale global optimization. Gene **7**(33), 8 (2013)
12. Van den Bergh, F., Engelbrecht, A.P.: A cooperative approach to particle swarm optimization. IEEE Trans. Evol. Comput. **8**(3), 225–239 (2004)
13. Yang, Z., Tang, K., Yao, X.: Large scale evolutionary optimization using cooperative coevolution. Inf. Sci. **178**(15), 2985–2999 (2008)
14. Omidvar, M.N., Li, X., Yao, X.: Cooperative co-evolution with delta grouping for large scale non-separable function optimization. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
15. Mahdavi, S., Rahnamayan, S., Shiri, M.E.: Multilevel framework for large-scale global optimization. Soft Comput. 1–30 (2016)
16. Sun, Y., Kirley, M., Halgamuge, S.K.: Extended differential grouping for large scale global optimization with direct and indirect variable interactions. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, pp. 313–320. ACM (2015)
17. Mei, Y., Omidvar, M.N., Li, X., Yao, X.: A competitive divide-and-conquer algorithm for unconstrained large-scale black-box optimization. ACM Trans. Math. Softw. (TOMS) **42**(2), 13 (2016)
18. Chen, W., Weise, T., Yang, Z., Tang, K.: Large-scale global optimization using cooperative coevolution with variable interaction learning. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6239, pp. 300–309. Springer, Berlin (2010). doi:10.1007/978-3-642-15871-1_31
19. Sun, L., Yoshida, S., Cheng, X., Liang, Y.: A cooperative particle swarm optimizer with statistical variable interdependence learning. Inf. Sci. **186**(1), 20–39 (2012)
20. Ge, H., Sun, L., Yang, X., Yoshida, S., Liang, Y.: Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation. Appl. Soft Comput. **36**, 300–314 (2015)
21. Tang, R., Wu, Z., Fang, Y.: Adaptive multi-context cooperatively coevolving particle swarm optimization for large-scale problems. Soft Comput. 1–20 (2016)
22. Sun, Y., Kirley, M., Halgamuge, S.K.: Quantifying variable interactions in continuous optimization problems. IEEE Trans. Evol. Comput. (in press)
23. Yang, Z., Tang, K., Yao, X.: Self-adaptive differential evolution with neighborhood search. In: IEEE Congress on Evolutionary Computation, CEC 2008, IEEE World Congress on Computational Intelligence, pp. 1110–1116. IEEE (2008)
24. Molina, D., Lozano, M., Herrera, F.: MA-SW-chains: memetic algorithm based on local search chains for large scale continuous global optimization. In: 2010 IEEE Congress on Evolutionary Computation (CEC), pp. 1–8. IEEE (2010)
25. Sheskin, D.J.: Handbook of Parametric and Nonparametric Statistical Procedures. CRC Press, Boca Raton (2003)