An Effective Approach for Adapting the Size of Subcomponents in Large-Scale Optimization with Cooperative Coevolution

Giuseppe A. Trunfio
DADU, University of Sassari
P.zza Duomo, 6
Alghero, Italy
trunfio@uniss.it

ABSTRACT

The performance of cooperative co-evolutionary algorithms for large-scale global optimization (LSGO) can be significantly affected by the adopted problem decomposition. This study investigates a new adaptive Cooperative Coevolutionary algorithm in which several decompositions are *concurrently* applied during short learning phases. Moreover, the study includes some experimental results on a set of LSGO problems and a comparison with a recent approach based on reinforcement-learning. According to the numerical results, the proposed adaptive approach can provide a superior search efficiency on several benchmark functions.

CCS Concepts

•**Computing methodologies** → **Continuous space search; Randomized search;** *Reinforcement learning;*

Keywords

Cooperative coevolution, large scale optimization, differential evolution, adaptation

1. INTRODUCTION

The CC idea [2] consists of decomposing the original high-dimensional problem into a set of lower-dimensional subcomponents, which are easier to solve. Typically, to each subcomponent is assigned a subpopulation of candidate solutions, which is evolved according to the adopted optimization metaheuristic. During the process, the only cooperation takes place in the fitness evaluation, through an exchange of information between subcomponents based on a common *context vector*.

A major challenge with the CC approach, consists of grouping variables into an optimal set of subcomponents. For fully separable (i.e. without interacting variables) or fully non-separable problems, a typical decomposition approach consists of using equally sized subcomponents. However, also their common size can be a critical factor in determining the performance of the CC technique. The problem has been addressed first in [6] (MLCC approach) and, more recently, in [1] (MLSoft approach), where the authors: (*i*) showed that in several cases there exists an optimal value for the size of subcomponents; (*ii*) proposed an adaptive approach based on a reinforcement learning technique for finding such an optimal value. Unfortunately, as noted in [1], the non-stationary nature of the problem makes hard the learning of a suitable size for the subcomponents.

This study presents an alternative approach for effectively adapting the size of subcomponents within a CC algorithm. In the proposed method, the learning phases consist of a concurrent application of a pool of alternative 'decomposers'. Such an approach, compared with the activation of one decomposer at a time proposed in [6, 1], enables a more reliable comparative evaluation of the candidate decomposers.

GECCO '15 July 11-15, 2015, Madrid, Spain
© 2015 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-3488-4/15/07...$15.00
DOI: http://dx.doi.org/10.1145/2739482.2764711

2. THE CCAS APPROACH

In both the MLCC and MLSoft approaches mentioned above, a *decomposer* (i.e. a size of subcomponents) is randomly drawn at each cycle according to its current probability, which is computed on the basis of a *value function*. The latter reflects the rewards obtained by the decomposer at the end of the cycles in which it has been used. Such adaptive methods can be seen in a perspective of a *reinforcement learning* (RL) approach [3], where the improvement of fitness is the reinforcement signal and the actions consist of the choice of the decomposer.

Unfortunately, in such a learning scheme the rewards obtained by the different decomposers may be strongly affected by the state of the environment in which they have operated. This is because of the expected evolution of the population on the fitness landscape, which can be significantly complex. In other words, an hypothetical agent that has to choice a decomposer operates on a non-stationary and history-dependent environment, for which RL schemes conceived for Markovian environments are not guaranteed to converge to the optimal policy (although they can still be used with acceptable results in some cases [3]). Instead, the proposed Cooperative Coevolution with Adaptive Subcomponents (CCAS) approach consists of evaluating the different decomposers under the same conditions. More in detail, during the learning phases, the decomposers of a predefined set are applied starting from the same state of the search,

Table 1: Achieved results with CCAS (using $\epsilon = 0.05$ and $\epsilon = 0.01$) and comparison with the results achieved using our implementation of MLSoft (with $\tau = 0.5$). The Table also shows the best static decomposer tested in the numerical investigation. Standard deviations are in parentheses. The used test functions are: f_1 = Quartic function, f_2 = Rastrigin's function, f_3 = Ackley's function, f_4 = Schwefel's function, f_5 = Styblinski-Tang function, f_6 = Shifted Sphere, f_7 = Shifted Schwefel problem 2.21, f_8 = Shifted Rosenbrock function , f_9 = Shifted Griewank function

Function	CCAS ($\epsilon = 0.05$)	CCAS ($\epsilon = 0.01$)	MLSoft ($c = 0.5$)	Static
f_1	5.67E+00 (5.45E+00)	4.41E+00 (3.84E+00)	4.81E+00 (1.70E+00)	2.58E+00 (1.08E+00)
f_2	**6.63E−02 (2.48E−01)**	6.63E−02 (2.48E−01)	3.09E+01 (6.27E+00)	4.81E+01 (2.78E+00)
f_3	**2.02E−13 (1.17E−12)**	1.96E+00 (1.04E+00)	1.71E−12 (1.39E−12)	1.70E−13 (5.38E−15)
f_4	**5.76E+02 (2.59E+02)**	**6.17E+02 (4.26E+02)**	7.03E+03 (1.39E+03)	9.90E+02 (2.26E+02)
f_5	**1.71E+00 (4.80E+00)**	**1.49E+01 (2.09E+01)**	4.91E+01 (5.78E+01)	1.81E+01 (1.60E+01)
f_6	0.00E+00 (0.00E+00)	3.36E−30 (1.25E−29)	7.66E−22 (2.17E−21)	3.15E−26 (2.43E−26)
f_7	**6.71E+01 (7.37E+00)**	**4.46E+01 (1.22E+01)**	1.01E+02 (4.09E+00)	7.17E+01 (6.89E−01)
f_8	**6.75E+01 (5.01E+01)**	1.62E+03 (4.10E+02)	1.42E+03 (6.88E+02)	1.44E+01 (2.26E+01)
f_9	**2.03E−13 (1.87E−14)**	6.36E−02 (1.24E−01)	1.47E−03 (5.51E−03)	6.57E−04 (2.46E−03)

including the same context vector. In other words, they are concurrently executed on the same initial environment in order to estimate their value functions.

Moreover, in order to achieve a better allocation of the available computational resources, the proposed approach is devised in such a way to have a suitable number of individuals for each population associated to the different decomposers. Indeed, in spite of the fact that the size of the population should be suitably adjusted according to the problem dimension, with few exceptions (e.g. [4]), the CC algorithms typically adopt a fixed number of individuals.

In the proposed strategy, the CC optimizer can be in two different states, namely *learning* and *optimization*. Both phases keep carrying out the optimization process. However, while in the learning phase all the available decomposers are concurrently applied, in the non-learning one, only the best decomposer is actually used. The duration of each learning phase is expressed in cycles (i.e. number of invocations of the optimizer). At the first cycle, the search is put in *learning* mode. At the end of the learning phase, all the rewards obtained by each decomposer are used for the computation of a *value function* q_j. After the computation of all q_j, the algorithm ends the learning phase by selecting, for the subsequent optimization phase, the decomposer with the greatest value function. In the current implementation, the learning phase is randomly resumed with a low probability ϵ.

The CCAS approach was applied to the minimization of nine typical benchmark test functions (see Table 1). The last four non-separable functions have been taken from those proposed for the CEC'08 special session on large scale global optimization (LSGO). For all functions we used $d = 1000$ as problem dimension. The adopted optimizer was JADE [7], an adaptive Differential Evolution algorithm in which the parameter adaptation is implemented by evolving the mutation factors and crossover probabilities based on their historical record of success. We executed 25 independent runs on the adopted test functions and the learning durations were set to 3 cycles. Moreover, we investigated two different values of the resuming probability ϵ, namely 0.01 and 0.05. In addition, we implemented a version of MLSoft [1], for which we used $c = 0.5$ (see [1]). The average achieved results are shown in Table 1, where they are also compared with the best results obtained using the static (i.e. fixed) decomposition. Using the Kruskal-Wallis test with significance 0.05 we assessed the significant differences between the different approaches. Then, we carried out Mann-Whitney-Wilcoxon (MWW) tests, with Bonferroni correction, for the between-groups comparisons. In Table 1, we marked in bold the best results, when the difference was statistically significant. As can be seen, CCAS with $\epsilon = 0.05$ outperformed MLSoft with $c = 0.5$ in seven out of nine cases. In the remaining two cases, the results provided by CCAS were statistically equivalent to those of MLSoft.

According to the numerical results, the proposed approach can effectively adapt the size of subcomponents during the CC search. Thus, the method deserves to be investigated more thoroughly, especially using a more extended suite of benchmark functions.

More details on the proposed algorithm and its preliminary evaluation can be found in [5].

3. REFERENCES

[1] M. N. Omidvar, Y. Mei, and X. Li. Effective decomposition of large-scale separable continuous functions for cooperative co-evolutionary algorithms. In *Proceedings of the IEEE Congress on Evolutionary Computatio*. IEEE, 2014.

[2] M. A. Potter and K. A. De Jong. A cooperative coevolutionary approach to function optimization. In *Parallel Problem Solving from Nature*, PPSN III, pages 249–257. Springer-Verlag, 1994.

[3] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[4] G. A. Trunfio. Enhancing the firefly algorithm through a cooperative coevolutionary approach: an empirical study on benchmark optimisation problems. *IJBIC*, 6(2):108–125, 2014.

[5] G. A. Trunfio. A cooperative coevolutionary differential evolution algorithm with adaptive subcomponents. *Procedia Computer Science*, (in press), 2015.

[6] Z. Yang, K. Tang, and X. Yao. Multilevel cooperative coevolution for large scale optimization. In *IEEE Congress on Evolutionary Computation*, pages 1663–1670. IEEE, 2008.

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