Applied Soft Computing 36 (2015) 300–314

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Cooperative differential evolution with fast variable interdependence learning and cross-cluster mutation

Hongwei Ge[a,b], Liang Sun[a,*], Xin Yang[a], Shinichi Yoshida[c], Yanchun Liang[d]

[a] College of Computer Science and Technology, Dalian University of Technology, Dalian 116023, China
[b] Key Laboratory of Advanced Control and Optimization for Chemical Processes, East China University of Science and Technology, Shanghai 200237, China
[c] School of Information, Kochi University of Technology, Kochi 7828502, Japan
[d] College of Computer Science and Technology, Jilin University, Changchun 130012, China

ARTICLE INFO

Article history:
Received 6 June 2014
Received in revised form 20 June 2015
Accepted 20 July 2015
Available online 1 August 2015

Keywords:
Cooperative optimization
Differential evolution
Large scale optimization
Cross-cluster mutation

ABSTRACT

Cooperative optimization algorithms have been applied with success to solve many optimization problems. However, many of them often lose their effectiveness and advantages when solving large scale and complex problems, e.g., those with interacted variables. A key issue involved in cooperative optimization is the task of problem decomposition. In this paper, a fast search operator is proposed to capture the interdependencies among variables. Problem decomposition is performed based on the obtained interdependencies. Another key issue involved is the optimization of the subproblems. A cross-cluster mutation strategy is proposed to further enhance exploitation and exploration. More specifically, each operator is identified as exploitation-biased or exploration-biased. The population is divided into several clusters. For the individuals within each cluster, the exploitation-biased operators are applied. For the individuals among different clusters, the exploration-biased operators are applied. The proposed operators are incorporated into the original differential evolution algorithm. The experiments were carried out on CEC2008, CEC2010, and CEC2013 benchmarks. For comparison, six algorithms that yield top ranked results in CEC competition are selected. The comparison results demonstrated that the proposed algorithm is robust and comprehensive for large scale optimization problems.

1. Introduction

Optimization problems are rife in diverse fields such as mechanical engineering, compressed sensing, natural language processing, structure control, and bio-computing [1–5]. Researchers have to determine a set of model parameters or state-variables that provide the minimum or maximum value of a predefined cost or objective [6]. With the coming of internet of things (IoT) [7], many optimization problems are becoming more difficult, i.e., the problems are characterized by more variables with complicated interactions. Research on optimization problems has attracted the attention of researchers and many algorithms have been proposed. Though the existing optimizers have shown to be successful in solving moderate scale problems, many of them still suffer from the "curse of dimensionality", which means that their performance will deteriorate as the dimensionality of the problem increases [8,9]. Thus effective and efficient algorithms for large scale optimization have become essential requirements. In this paper, we aim at solving the large scale optimization problems and providing tools for scientists and engineers when they are solving real world problems from the involved disciplines.

Generally, the natural way to address the "curse of dimensionality" is to apply cooperative optimization, which can be regarded as an automatic approach to implement the divide-and-conquer strategy. A typical cooperative optimization algorithm can be summarized as follows [10]:

1. Problem decomposition: decompose a large scale problem into smaller scale subproblems.
2. Subproblem optimization: optimize each subproblem by means of a separate optimizer.
3. Cooperative coordination: combine the subsolutions to obtain an entire solution.

A key issue with regards to the cooperative optimization is the task of problem decomposition. An appropriate decomposition algorithm should group interacted variables together so that the interdependencies among different subproblems are minimized. Based on whether the variable interdependencies are considered or not, the decomposition algorithms can be classified into two

* Corresponding author. Tel.: +86 15998564404.
E-mail address: liangsun@dlut.edu.cn (L. Sun).

http://dx.doi.org/10.1016/j.asoc.2015.07.016
1568-4946/© 2015 Elsevier B.V. All rights reserved.

categories. Generally, the algorithms performed without considering variable interdependencies are simple and effective for separable problems, but have difficulty in solving nonseparable problems [10–16]. On the other hand, the algorithms performed by considering the variable interdependencies provide opportunities to solve large scale nonseparable problems [17–24]. However, many of them either add extra computational burden to the algorithm or lack extensive variable interdependence learning.

Another key issue with regards to the cooperative optimization is the optimization of the subproblems. The widely used optimizers are inspired by nature phenomena, which include genetic algorithm (GA) [25], evolution programming (EP) [26,27], evolution strategy (ES) [28,29], differential evolution (DE) [6,30], ant colony optimization (ACO) [31], particle swarm optimization (PSO) [32–37], bacterial foraging optimization (BFO) [38], simulated annealing (SA) [39], tabu search (TS) [40], harmony search (HS) [35,36,40], etc. These optimizers facilitated research into the optimization of the subproblems. However, many of them are still not free from premature convergence for the complex multi-modal, rugged, and nonseparable problems.

As can be seen from the aforementioned, as far as cooperative optimization algorithms concerned, there still exists a big room to improve their performance through deeper studies. In this paper, we propose a variant of cooperative optimization algorithm. The study concentrates on the aforementioned two issues. To solve the task of problem decomposition, we propose a fast variable interdependence searching operator, which operates by recursively partitioning decision variables into blocks and identifying the interdependences among different blocks. Then we decompose a large scale problem into small scale subproblems based on the obtained interdependencies. To solve the task of subproblem optimization, we propose a cross-cluster mutation strategy, in which each operator is identified as exploration-biased or exploitation-biased. The population is divided into several clusters. For the individuals among different clusters, the exploration-biased operators are applied, and for the individuals within each cluster, the exploitation-biased operators are applied. By favoring search in the vicinity of each cluster and in the regions among different clusters, this strategy promotes efficient exploration as well as efficient exploitation. We further incorporate the proposed strategies into the original differential algorithm to perform optimization. The reason that we adopt differential evolution as base optimizer is that it has been frequently adopted and the resulting variants have been achieving top ranks in various competitions [30].

The reminder of this paper is organized as follows. Section 2 reviews the related works with regard to cooperative optimization and differential evolution. Section 3 gives the description of the proposed algorithm, which includes the fast variable interdependence learning method, and the cross-cluster mutation strategy. Section 4 presents the experimental results, followed by concluding remarks in Section 5.

2. Related works

This work is closely related to cooperative optimization and differential evolution. In this section, we firstly review the prevailing algorithms for cooperative optimization. And then review the state of the art algorithms for differential evolution.

2.1. Cooperative optimization

The cooperative optimization algorithm addresses the “curse of dimensionality” by applying the divide-and-conquer strategy. One of the most important issue with regards to divide-and-conquer is the task of problem decomposition, the process of partitioning a large scale problem into subproblems. The decomposition decision regarding variable interdependencies plays a significant role in the algorithm’s performance. Based on whether the variable interdependencies are considered during the execution process, the existing algorithms can be classified into two categories. The algorithms belonging to the first category are performed without considering the variable interdependencies. For example, the algorithms reported in [10–15] operate by decomposing problems arbitrarily, with each subproblem being optimized by a separate optimizer. When a subproblem is being optimized, the variables belonging to other subproblems are kept unchanged. The solution to the problem is obtained by combining the subsolutions found by each of the separate optimizer. During the optimization process, the decomposition strategy is kept unchanged. The advanced algorithms belonging to this category operates by decomposing problem dynamically [8,16]. During the optimization process, the algorithms decompose problem based on a measured stagnation, and eliminate previous decomposition strategies which are no longer making contributions. In this way, the algorithm has the opportunity to optimize interacting variables. Generally, algorithms that do not consider variable interdependencies are effective for separable problems, but have difficulty solving nonseparable problems, especially the problems with tightly interacted variables.

The algorithms belonging to the second category are performed by considering the variable interdependencies implicitly or explicitly. For example, the algorithms reported in [17–19] use a random grouping scheme to decompose a large scale problem into subproblems, and then optimize each of the subproblem by a separate optimizer. For co-adaptation, the algorithms apply weights to the subproblems, and optimize the weight vectors with a certain optimizer. In [20–22], the variable interdependencies are captured during the optimization process. Initially, the system does not have any knowledge with regards to the variable interdependencies. With the optimization process going on, the system captures the variable interdependencies by investigating the relationship between the objective functions and the candidate solutions. While optimizing, the decomposition and the optimization strategies are adapted according to the obtained interdependencies. In our previous work [24], we propose a statistical model to explore the interdependencies among variables. At this point, the degree of interdependence between each pair of variables is quantified. With these interdependencies, the problem is decomposed and optimized using a cooperative particle swarm optimization (CPSO) framework. Generally, the algorithms that consider variable interdependencies provide opportunities to solve nonseparable problems. However, many of them require extensive computational burden to capture the variable interdependencies. For example, in our previous work [24], the computational efforts incurred by the variable interdependence learning take about 30% of the total computational efforts.

A critical issue in the application of cooperative optimization is to determine the learning strategy allowing efficient exploration of the interdependencies among variables, as well as avoiding intensive computational efforts consumption. Motivated by these findings, we developed a fast variable interdependence search operator to mitigate the problems encountered in the aforementioned algorithms. The proposed operator works as follows. The subsets of variables are referred as variable blocks. The interdependencies among variable blocks are further defined. We advocate a fast search operator which operates by recursively partitioning decision variables into blocks and then identify the interdependencies among the blocks. By favoring search the interdependencies among variable blocks recursively, this operator promotes efficient exploration of the interdependencies, without substantially adding computational burden.