Applied Soft Computing 61 (2017) 88–110

Contents lists available at ScienceDirect

Applied Soft Computing

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

Influence of randomization strategies and problem characteristics on the performance of Differential Search algorithm

Wadee Alhalabi [a], Elena Niculina Dragoi [b,*]

[a] King Abdulaziz University, Faculty of Computing and Information Technology, Department of Computer Science, Jeddah, Saudi Arabia
[b] "Gheorghe Asachi" Technical University of Iasi, Faculty of Chemical Engineering and Environmental Protection, Department of Chemical Engineering, Automation and Applied Informatics, Blvd. D Mangeron, 700050, Iasi, Romania

ARTICLE INFO

Article history:
Received 20 September 2016
Received in revised form 1 August 2017
Accepted 2 August 2017

Keywords:
Randomization
Dimensionality
Global optimization
Differential search
Alkylation
Heat exchanger

ABSTRACT

In this work, the influence of different random number generators, problem dimensionality, and number of function evaluations on the optimization efficiency of Differential Search algorithm is presented in detail. Two types of random number generators were taken into account: discrete and continuous. Different combinations between the dimensionality property and the number of function evaluation setting were tested on: i) a set of benchmark functions from the CEC 2013 special session on real parameter optimization; and ii) 2 chemical engineering problems (optimal operation of an alkylation unit and heat exchanger network design). Also, a comparison with other optimizers was performed. It was found that, in similar conditions, the performance of the algorithm (in terms of the best solutions) varies substantially depending on the distribution used. In case of the benchmark problems, the best solutions were obtained for Binomial and Weibull distribution. For the separable functions is was observed that, indifferent of the distribution used, the algorithm was not able to find acceptable solution within the constraint represented by the number of function evaluations. In the case of the alkylation problem the best solutions were obtained by the Weibull distribution, the performance of the Differential Search algorithm being comparable to other optimizers such as Differential Evolution. In the case of the heat exchanger, three different distribution provided near optimal solutions (Binomial, ChiSquare and Weibull).

1. Introduction

Optimization is an important aspect of all natural processes, with a tendency to consume the minimum amount of resources and to obtain the highest gain. This is also applicable to research studies, where in order to solve specific problems, the best optimizers are employed. Although an optimization state is always desirable and there is a proliferation of optimization methods, there is no universal approach for solving all the optimization problems [1]. The no-free lunch theorem states that one cannot expect a single algorithm to outperform all others in all possible problem instances and therefore researchers focus on developing new algorithms that can be efficiently applied to large areas of problems [2–4].

The inspiration for these methods varies, in the latest years, nature (biological systems) ranking on top. The field study associated to this methods is called Bio-Inspired Computing, the areas of research including Evolutionary Algorithms (EAs) – inspired from evolution, Swarm Optimization – inspired from the swarm behavior, Artificial Neural Networks – inspired from mechanisms of the mammalian brain, Artificial Immune Systems – inspired from the response of the vertebrate immune system. As it can be observed, the list is quite impressive and the multitude of algorithms proposed is rising. As each algorithm has its specific characteristics that are either inherited from the inspiration source or are consequences of the internal methods applied, an in-depth performance analysis of each approach is required in order to identify the areas where its application is most suited. The majority of these algorithms are global optimizers, a global optimization procedure implying finding the best set of parameters that optimize an objective function [5].

In this work, a relatively new algorithm, called Differential Search (DS) [6] and inspired from the migration movement of organisms was studied in detail and its performance on different types of problems assessed. The scope was to determine the influence of characteristics of the problem being solved (dimensionality, linearity, single or multi-objective, existence of multiple minima) on the algorithm's capability to find solutions in the close vicinity of the minima (considering that the problem is a minimization one).

* Corresponding author.
E-mail addresses: wsalhalabi@kau.edu.sa (W. Alhalabi), elenan.dragoi@gmail.com (E.N. Dragoi).

http://dx.doi.org/10.1016/j.asoc.2017.08.001
1568-4946/

As it was stated previously, DS is a relatively new algorithm and therefore the available literature and studies regarding its application and behavior are quite scarce. It was proposed in 2012 by Civicioglu [6] as a mean to transform the geocentric cartesian coordinates into geodetic coordinates. It simulates the Brownian like random walk movement used by migration organisms. The main idea of DS is that an artificial superorganisms migrates to the global minimum by testing if some randomly positions are temporarily suitable.

In an attempt to raise the performance, Gan and Duan [7] proposed a chaotic based DS in which the property of a chaotic variable is integrated in order to improve the search capabilities. This variant, called CDS is applied for image extraction and image enhancement and, in order to further improve its performance, a Lateral Inhibition (LI) mechanism is employed for image pre-processing, resulting in a new algorithm called CDS-LI. A comparison with the classical DS, CDS and PSO showed that for the problem at hand, the best results are obtained with the CDS-LI version. Another work in which DS is applied to solve a specific problem is the one of Goswami and Chakraborty [8] where parametric optimization of three electrochemical micromachining systems is performed by the classical variant of DS proposed in [6]. Similar to the other studies, results pointed out that DS is an acceptable global optimization tool. The capabilities of DS were also tested on a set of three antenna array synthesis problems: i) placing wide nulls on the array pattern by controlling the amplitude; ii) obtaining array patterns with individual nulls imposed at the interference directions by controlling the amplitude-only, phase-only and position-only; iii) failure correction [9]. As in the previous cases, the results were in the accepted interval, which pointed out DS as a good alternative to other antenna array synthesis algorithms.

A raising trend of using DS to solve different problems was observed in the last two years, where the area of problems enlarged to include the chemical engineering domain, structure design and other engineering problems [10–24]. However, its strengths and weaknesses influenced by the characteristics of the problem being solved (linearity, dimensionality) and by the internal approaches used (stop criteria, parameter control settings, initialization procedure) are not studied in detail or not studied at all. In order to determine its efficiency, Civicioglu [6] performed a series of tests on a set of benchmark functions with a dimensionality varying from 2 to 30. The statistical analysis employed was the Wilcoxon Rank Sum Test and the results showed that in the majority of cases DS outperforms Artificial Bee Colony (ABC), different variants of Differential Evolution (DE) algorithm or Particle Swarm Optimization (PSO). As no emphasis was put on the dimensionality of the benchmark functions, initialization procedures, or stop criteria, it is clear that an extensive study related to these aspects is required.

This lack of knowledge can influence the decision of an end-user to choose another approach, even if DS has the potential to provide faster and more accurate solutions. Consequently, in this work, the influence of different types of random distributions, dimensionality of the problem and number of iterations on the performance of the DS algorithm is tested for different benchmark and real-life problems from the chemical engineering area. The scope is to determine the characteristics of the problems for which DS performs better and the areas where improvements are required in order to raise performance.

2. Differential search

DS is a population based heuristics evolutionary algorithm developed by Civicioglu [6] to transform the geocentric cartesian coordinates to geodetic coordinates [25]. It is inspired by the migration of living beings and it is based on the Brownian like random walk movement. Random solutions to the problem being solved form the population, which is assumed to correspond to an artificial superorganism migrating [6]. The goal of migration (that has as a result a new position) is to identify the global optimum of the problem. The superoganism checks some randomly selected positions (based on the cost function) and if suitable, the individuals that made the discovery settle there, the migration continuing from that point [6].

The general steps of the algorithm include initialization, evaluation, and migration. Migration is performed until the stop criteria is reached and consists of donor selection, stopover site determination, selection of individuals influencing the stopover site, evaluation of the stopover site and superogranism update. The terminology used to describe some of the operation is different, but at their core, they are similar to genetic algorithm (GA) operators. A comparison between the terminology used in GA and DS is listed in Table 1.

In the initialization phase (Step 1 from Fig. 1), the initial values of the superorganism and of the control parameters are randomly determined. The superoganism is formed from *Np* members, where each characteristic of member is defined as:

$$x_{i,j} = rand() * \left(up_j - low_j\right) + low_j \tag{1}$$

where $i = 0.Np$ represents the position of each organism in the superorganism, up_j and low_j are the upper and lower limit of the jth characteristics and *rand()* is a randomly generated number. The organism is defined as $X_i = [x_{i,j}]$ and the superorganism as $S_g = X_i$.

Each member of the superoganism is evaluated using a fertility function that can vary, depending on the objectives of the problem being solved. This function is equivalent to the fitness function used in the EAs.

In the donor selection step (Step 2), using a shuffling function that randomly reorganizes the indexes from the 0.Np interval, the donor is selected. In this manner, a set of randomly selected individuals move towards the targets of the donor to discover new stopover sites (temporary positions during the migration) [6]. The stopover site allows the organisms that discovered it to settle and continue the migration from that position.

The stopover site position (Step 3.1) is determined using the following formula:

$$stopoverSite = superorganism + scale * (donor - superorganism) \tag{2}$$

where *scale* indicates the size of the change. The value of *scale* is generated using a gamma-random number generator in the interval [0,1] [26].

After that, the organisms (that participate to the discovery of the stopover site determined using Eq. (2)) are randomly determined (Step 3.3). Two control parameters *p1* and *p2* are used to direct the selection of these organisms (Step 3.2).

After the stopover site is generated, it is evaluated in order to test its fertility compared to the sources that discovered that stopover site (Step 4). If it is more fertile, the superorganism moves to that stopover site (Step 5). This superorganism update is similar to the selection phase encountered in classical GA.

A general schema of the DS algorithm is presented in Fig. 1 where the steps of the DS algorithm are numbered based on the order of their execution. In Fig. 1, the stop criteria represents the conditions that once reached, stop the algorithm. In the initial version [6], the stop criteria is represented by the number of iterations reaching a pre-defined value (*maxgeneration*). In the current work, the number of function evaluations (FE) is counted and the algorithm stops when it reaches a specific value {10000, 50000, 100000}.

From the five main steps, three [(1) through (3)] are using randomly generated numbers. This is similar to other EAs, where