Swarm and Evolutionary Computation 26 (2016) 64–85

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Regular paper

# Opposition-based learning in the shuffled bidirectional differential evolution algorithm

Morteza Alinia Ahandani [a,b,*]

[a] Department of Electrical Engineering, Langaroud Branch, Islamic Azad University, Langaroud, Iran
[b] Young Researchers Club, Langaroud Branch, Islamic Azad University, Langaroud, Iran

ARTICLE INFO

*Article history:*
Received 13 March 2015
Received in revised form
20 July 2015
Accepted 24 August 2015
Available online 2 September 2015

*Keywords:*
Opposition-based learning
Shuffled bidirectional differential evolution
Search moves
Real-parameter optimization

ABSTRACT

The opposition-based learning (OBL) strategy by comparing the fitness of an individual to its opposite and retaining the fitter one in the population accelerates search process. In this paper, the OBL is employed to speed up the shuffled bidirectional differential evolution (SBDE) algorithm. The SBDE by employing the partitioning, shuffling and bidirectional optimization concepts increases the number and diversity of search moves in respect to the original differential evolution (DE). So with incorporating the SBDE and OBL strategy, we can obtain the algorithms with an ability of better exploring the promising areas of search space without occurring stagnation or premature convergence. Experiments on 25 benchmark functions and non-parametric analysis of obtained results demonstrate a better performance of our proposed algorithms than original SBDE algorithm. Also an extensive performance comparison the proposed algorithms with some modern and state-of-the-art DE algorithms reported in the literature confirms a statistically significantly better performance of proposed algorithms in most cases. In a later part of the comparative experiments, firstly proposed algorithms are compared with other evolutionary algorithms (EAs) proposed for special session CEC2005. Then a comparison against a wide variety of recently proposed EAs is performed. The obtained results show that in most cases the proposed algorithms have a statistically significantly better performance in comparable to several existing EAs.



## 1. Introduction

The differential evolution (DE) is a simple yet powerful evolutionary algorithm (EA) which was proposed by Storn and Price [1]. Ahandani and Alavi-Rad [2] mentioned the benefits and drawbacks of DE. They mentioned the following benefits for it: a few number of control parameters to be tuned i.e. amplification factor of the difference vector, crossover rate and population size, simplicity and easy implementation, speed and robustness. Also they mentioned its drawbacks as follow: stagnation or premature convergence because of its low or fast convergence speed, having a problem in accurately zooming to optimal solution, to be limited the number and diversity of search moves, utilizing greedy criterion in accepting or rejecting a new generated offspring, poor performance of it in noisy environment, requiring multiple runs for tuning parameters and to be problem dependent of the best control parameter settings. The aforementioned drawbacks, which some of them are common defects of EAs, have not prevented from widespread applications of DE in different optimization fields [3–5].

A variety of different strategies have been proposed to overcome these drawbacks. Self-adaptive settings for automatically and dynamically adjusting evolutionary parameters have been proposed as a solution for stagnation or premature convergence of the DE [6–11]. To solve the problem of accurate zooming to optimal solution, hybridizing the DE with a local search method has been proposed [12–14]. In order to increase the number and diversity of search efforts, some new operators can be added to the original DE. These new operators must be able to expand the search moves in different areas of search space [15,16]. In a greedy strategy to accept or reject a new generated member, only a member with a better quality than its makers is admissible. So it despite ensuring the fast convergence, can increase the probability of getting stuck in local minimums. Employing some strategies such as those of employed in simulated annealing, great deluge and tabu search algorithms may moderate the greedy acceptance strategy. Poor performance of DE in noisy environment is related to its deterministic choice of the scale factor. Employing some approaches to prevent for fast stagnation of DE may make successful it at handling a noisy fitness function. The DE such as other

* Corresponding author at: Islamic Azad University, Langaroud Branch, Electrical Engineering Department, Langaroud, Iran. Tel.: +98 142 5244411; mobile: +98 911 9465164; fax: +98 142 5244413.
*E-mail address:* alinia@iaul.ac.ir

http://dx.doi.org/10.1016/j.swevo.2015.08.002
2210-6502/

***Algorithm 1* (the SBDE algorithm)**
**Begin SBDE**
*Step 1*: generate and evaluate initial population of size $N_{pop}$.
*Step 2*: generate $m$ memeplexes with $n$ members where $N_{pop} = m \times n$.
*Step 3*: apply the BDE (*Step3.* 0 to *Step* 3.4) to improve each memeplex for $k_{\max}$ iterations:
*Step* 3.0: set counter $k = 1$.
*Step 3.1*: **While** $k \le k_{\max}$
Improve each member of memeplex ($x_i$) as follow:
*Step 3.2*: set counter $i = 1$.
*Step 3.3*: **While** $i \le n$
*Step 3.3.1*: determine the best member of each memeplex, $x_{g1}$ and the best member of population, $x_{g2}$.
*Step 3.3.2*: apply mutation and crossover operators for $x_i$ according to Eq. (1) and Eq. (3) with $x_{best} = x_{g1}$ and generate new member of $u$.
*Step 3.3.3*: evaluate value of cost function in the point $u$ and if $f(u) \le f(x_i)$, replace $x_i$ with $u$ and set $i = i + 1$, then go to step 3.3.
*Step 3.3.4*: replace Eq. (1) with Eq. (2) and repeat steps 3.3.2 and 3.3.3.
*Step 3.3.5*: repeat steps 3.3.2 to 3.3.4 with $x_{best} = x_{g2}$.
**End While**
*Step 3.4*: set $k = k + 1$.
**End While**
*Step 4*: shuffle the population.
*Step 5*: update the best member found so far.
*Step 6*: check the stopping criteria if are not met go to *Step 2*.
**End SBDE**

**Fig. 1.** The steps of SBDE.

***Algorithm 2* (the SOBBDE algorithm)**
**Begin SOBBDE**
*Step 1*: generate and evaluate initial population of size $N_{pop}$ based on **opposition-based population initialization** strategy.
*Step 2*: generate $m$ memeplexes with $n$ members where $N_{pop} = m \times n$.
*Step 3*: apply the OBBDE (*Step3.* 0 to *Step* 3.5) to improve each memeplex for $k_{\max}$ iterations:
*Step* 3.0: set counter $k = 1$.
*Step 3.1*: **While** $k \le k_{\max}$
Improve each member of memeplex ($x_i$) as follow:
*Step 3.2*: set counter $i = 1$.
*Step 3.3*: **While** $i \le n$
*Step 3.3.1*: determine the best member of each memeplex, $x_{g1}$ and the best member of population, $x_{g2}$.
*Step 3.3.2*: apply mutation and crossover operators for $x_i$ according to Eq. (1) and Eq. (3) with $x_{best} = x_{g1}$ and generate new member of $u$.
*Step 3.3.3*: evaluate value of cost function in the point $u$ and if $f(u) \le f(x_i)$, replace $x_i$ with $u$ and set $i = i + 1$, then go to step 3.3.
*Step 3.3.4*: replace Eq. (1) with Eq. (2) and repeat steps 3.3.2 and 3.3.3.
*Step 3.3.5*: repeat steps 3.3.2 to 3.3.4 with $x_{best} = x_{g2}$.
**End While**
*Step 3.4*: apply **opposition-based generation jumping** strategy based on a jumping rate of $J_r$:
**If** $rand(0,1) < J_r$
**For** $i = 1 : n$
  **For** $j = 1 : D$
    $Omem_{i,j} = MINmem_j^p + MAXmem_j^p - mem_{i,j}$;
  **End For**
**End For**
Select $n$ fittest members from the set of $\{mem \cup Omem\}$ as current memeplex.
*Step 3.5*: set $k = k + 1$.
**End While**
*Step 4*: shuffle the population.
*Step 5*: update the best member found so far.
*Step 6*: check the stopping criteria if are not met go to *Step 2*.
**End SOBBDE**

**Fig. 2.** The steps of SOBBDE.

EAs requires multiple time-consuming trial-and-error runs for tuning parameters and the best control parameter settings of it are problem dependent. However the adaptive or self-adaptive control of parameters have been proposed as a solution to overcome these disadvantages, but these strategies apply some new parameters to DE to be tuned (see [8,17,18]). To minimize the effects of the control parameters, Wang et al. [19] proposed a Gaussian bare-bones DE which is almost parameter free. A Gaussian mutation