Applied Soft Computing 38 (2016) 501–517

Contents lists available at ScienceDirect

Applied Soft Computing

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

A memory based differential evolution algorithm for unconstrained optimization

Raghav Prasad Parouha*, Kedar Nath Das

Department of Mathematics, NIT, Silchar, Assam, India

ARTICLE INFO

Article history:
Received 12 January 2015
Received in revised form 31 July 2015
Accepted 8 October 2015
Available online 19 October 2015

Keywords:
Differential Evolution
Mutation
Crossover
Elitism
Unconstrained optimization

ABSTRACT

In optimization, the performance of differential evolution (DE) and their hybrid versions exist in the literature is highly affected by the inappropriate choice of its operators like mutation and crossover. In general practice, during simulation DE does not employ any strategy of memorizing the so-far-best results obtained in the initial part of the previous generation. In this paper, a new "Memory based DE (MBDE)" presented where two "swarm operators" have been introduced. These operators based on the *pBEST* and *gBEST* mechanism of particle swarm optimization. The proposed MBDE is employed to solve 12 basic, 25 CEC 2005, and 30 CEC 2014 unconstrained benchmark functions. In order to further test its efficacy, five different test system of model order reduction (MOR) problem for single-input and single-output system are solved by MBDE. The results of MBDE are compared with state-of-the-art algorithms that also solved those problems. Numerical, statistical, and graphical analysis reveals the competency of the proposed MBDE.

1. Introduction

Optimization is a ubiquitous and spontaneous process and frequently appears in the real world problems. In the world of optimization evolutionary algorithms (EAs) have been treated as the successful alternatives, since last few decades. Among all EAs, differential evolution (DE) [1] is an efficient, formidable, and popular ingredient [2]. Some reasons for the popularity of DE are highlighted as follows:

i. The main body of the classical DE requires 4–5 lines in any programming language. Therefore it has easy implementation, faster convergence, and stronger stability [3].
ii. It requires only a very few control parameters like CR (crossover rate), F (mutant factor) and NP (population size) to be tuned.
iii. The space complexity of DE is also low as compared to some of the most competitive real parameter optimizers [3]. In general, it has efficient global search ability and hence considered as global optimization algorithm [3].
iv. As evidenced by the recent studies on DE [4,5], it exhibits much better performance in comparison with several other EAs.

So far, DE has received extensive attention and applied to many engineering optimization problems, such as mechanical engineering design problem [6], fuzzy clustering of image pixel [7], economic load dispatch [8] and many others [3]. However, most of the time, the solution gets stuck in some local optima. As a result it leads to a premature convergence. It is because DE have some individual shortcomings such as follows:

i. The local exploitation ability and convergence rate of DE is too low [3].
ii. It loses maintaining the diversity in the population during simulation [3].
iii. The performance of DE decreases as the dimension of the problem increases [3].
iv. As other EAs it does not guarantee to find a global optimal solution in a finite time interval [3].

Therefore, in order to improve the performance of basic DE, a number of attempts are made in the literature [3–16]. A detailed survey on the variants of DE can be found in [4,5]. Moreover, in order to improve the robustness of DE, a number of mutation strategies of DE have been proposed in [3,10–12]. Basically, DE is much sensitive to choice of the mutation strategy. On the other hand, inappropriate choice of mutation strategy may lead to premature convergence, stagnation, or wastage of computational time [3]. Also, it is very difficult to recommend a fixed set of parameters for different problems [3].

* Corresponding author. Tel.: +91 8486542469.
E-mail addresses: rparouha@gmail.com (R.P. Parouha), kndnitsmaths@gmail.com (K.N. Das).

http://dx.doi.org/10.1016/j.asoc.2015.10.022
1568-4946/

Similarly, researchers mainly used two types of crossover schemes in DE, namely binomial crossover and exponential crossover [1]. In [17], Price recommended that the use of binomial crossover is better. But later, it is observed that there are no significant differences between these crossovers [18].

Unfortunately, according to "No Free Lunch Theorem [19])", no single optimization method exist, which is able to solve consistently to all global optimization problems. In spite of quite a high number of DE variants exist in the literature; DE further yields improved results while hybridizing with particle swarm optimization (PSO) [20]. Each of them is capable of dominating the shortcoming of the other to add the robustness in the resultant hybrid algorithm. The magical synergy of DE and PSO has been well established and has crossed many success milestones in recent past. The year-wise applications of DE-PSO hybrid techniques and their variants are summarized as follows:

Author	Year	Technique used	Application
Hendtlass [21]	2001	SDEA	Unconstrained global optimization
Zhang and Xie [22]	2003	DEPSO	Unconstrained global optimization
Kannan et al. [23]	2004	CPSO	Generation expansion planning
Talbi and Batouche [24]	2004	DEPSO	Medical image processing
Das et al. [25]	2005	PSO-DV	Unconstrained global optimization
Moore and Venayagamoorthy [26]	2006	DEPSO-MV	Multi-objective optimization
Hao et al. [27]	2007	DEPSO	Unconstrained global optimization
Omran et al. [28]	2008	BBDE	Unconstrained optimization problems and image processing
Das et al. [29]	2008	PSO-DV	Engineering design
Jose et al. [30]	2009	DEPSO	Noisy functions
Zhang et al. [31]	2009	DE-PSO	Unconstrained global optimization
Caponio et al. [32]	2009	SFMDE	Unconstrained and engineering design optimization
Xu and Gu [33]	2009	PSOPDE	Unconstrained global optimization
Wang and Cai [34]	2009	HMPSO	Constrained optimization problems
Khamsawang et al. [35]	2010	PSO-DE	Power systems
Liu et al. [36]	2010	PSO-DE	Constrained and engineering optimization
Wang et al. [37]	2010	DEDEPSO	Unconstrained global optimization
Niknam et al. [38]	2011	FAPSO–VDE	Power systems
Pant and Thangaraj [39]	2011	DE-PSO	Unconstrained and real life problems optimization
Thangaraj et al. [40]	2011	DE-PSO, AMPSO, GA-PSO	Unconstrained global optimization
Epitropakis et al. [41]	2012	A family of DE and PSO based hybrids	Unconstrained global optimization
Dor et al. [42]	2012	DEPSO-2S	Unconstrained real life problems
Xin et al. [43]	2012	––	Review and taxonomy of hybrid DE and PSO
Nwankwor et al. [44]	2013	HPSDE	Well placement optimization
Araújo and Uturbey [45]	2013	PSO–DE	Power systems
Sayah and Hamouda [46]	2013	DEPSO	Power systems
Kordestani et al. [47]	2014	CDEPSO	Dynamic optimization problems
Yu et al. [48]	2014	HPSO-DE	Unconstrained global optimization
Zuo and Xiao [49]	2014	Multi-DEPSO	Dynamic optimization problems
Parouha and Das [50]	2015	DPD	Constrained and engineering optimization

Though many variants of DE and its hybrid algorithms have been suggested in the literature to solve optimization problems, they are unable to provide satisfactory result. The reason behind this is DE has no mechanism to memorize the so-far best solution, but it uses only the global information about the search space [37,51]. Therefore, in spite of the increased convergence rate of DE, the algorithm mostly loses its computing power and eventually leads to premature convergence [3].

An attempt is made in this paper to employ the memory-based mechanism in DE algorithm under the PSO environment. The rest of the paper is organized as follows. Section 2 presents the traditional DE. Section 3 presents a detailed description of the proposed algorithm. Section 4 presents result and discussion. Application of proposed algorithm is presented in Section 5. Finally, Section 6 draws the conclusion with some future scopes.

2. Traditional differential evolution (DE)

DE is simple yet powerful optimization algorithm introduced by Storn and Price in 1995 [1]. It uses three operators, mutation, crossover, and selection to evolve from the randomly generated initial population to the final individual solution. In the initialization a population of NP target vectors (parents) $X_i = (x_{1i}, x_{2i}, \ldots, x_{Di})$, $i = 1, 2, \ldots, NP$ is randomly generated within user-defined bounds, where D is the dimension of the problem. This population undergoes with the cyclic processes of mutation, crossover, and selection, which are briefly explained below. In this paper, only the minimization problems are considered. However, maximization problem can easily be converted to minimization problem.

Mutation: Let $X_i(t) = (x_{1i}(t), x_{2i}(t), \ldots\ldots, x_{Di}(t))$ be the 'i^{th}' individuals at 't^{th}' generation. A mutant vector $V_i(t+1) = (v_{1i}(t+1), v_{2i}(t+1), \ldots\ldots, v_{Di}(t+1))$ is generated as follows:

$$V_i(t+1) = x_{r_1}(t) + F * (x_{r_2}(t) - x_{r_3}(t)),$$
$$r_1 \neq r_2 \neq r_3 \neq i, i = 1, 2, \ldots, NP, \quad (1)$$

where $r_1, r_2, r_3 \in \{1, 2, \ldots, NP\}$ are randomly chosen integers, different from each other and also different from the running index i. $F \in [0, 1]$ is a scaling factor which controls the amplification of the difference vector.

Crossover: According to the target vector $X_i(t)$ and the mutant vector $V_i(t+1)$, a new trial vector (offspring) $U_i(t+1) = (u_{1i}(t+1), u_{2i}(t+1), \ldots, u_{Di}(t+1))$ is created as follows:

$$U_{ji}(t+1) = \begin{cases} V_{ji}(t+1) & \text{if}(\text{rand}(0,1) \leq CR)\,\text{or}\, j = \text{rand}(i) \\ X_{ji}(t) & \text{if}(\text{rand}(0,1) > CR)\,\text{and}\, j \neq \text{rand}(i) \end{cases} \quad (2)$$

where $j \in \{1, 2, \ldots, D\}$, CR $\in [0, 1]$ is the crossover constant, $rand(i) \in [1, 2, \ldots, D]$ is a randomly chosen index, which ensures that $U_i(t+1)$ gets at least one parameter from $V_i(t+1)$ [1].

Selection: The generated trial vector $U_i(t+1)$ from the crossover operation will be compared with the target vector $X_i(t)$ based on better fitness values. The fittest between these two will survive for the next generation. Therefore the selection criteria in DE are defined as follows:

$$X_i(t+1) = \begin{cases} U_i(t+1) & \text{if}\, f(U_i(t+1)) \leq f(X_i(t)) \\ X_i(t) & \text{otherwise} \end{cases}. \quad (3)$$