Computers & Operations Research 38 (2011) 394–408

Contents lists available at ScienceDirect

Computers & Operations Research

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

A differential evolution algorithm with self-adapting strategy and control parameters

Quan-Ke Pan [a], P.N. Suganthan [b,*], Ling Wang [c], Liang Gao [d], R. Mallipeddi [b]

[a] College of Computer Science, Liaocheng University, Liaocheng 252059, PR China
[b] School of Electrical and Electronic Engineering, Nanyang Technological University, 50 Nanyang Avenue, Singapore 639798, Singapore
[c] Tsinghua National Laboratory for Information Science and Technology (TNList), Department of Automation, Tsinghua University, Beijing 100084, PR China
[d] State Key Laboratory of Digital Manufacturing Equipment & Technology in Huazhong University of Science & Technology, Wuhan 430074, PR China

ARTICLE INFO

Available online 23 June 2010

Keywords:
Differential evolution
Evolutionary algorithm
Global numerical optimization
Parameter adaptation
Strategy adaptation
Continuous optimization

ABSTRACT

This paper presents a *D*ifferential *E*volution algorithm with **s**elf-adaptive trial vector generation **s**trategy and control *p*arameters (SspDE) for global numerical optimization over continuous space. In the SspDE algorithm, each target individual has an associated strategy list (SL), a mutation scaling factor *F* list (FL), and a crossover rate *CR* list (CRL). During the evolution, a trial individual is generated by using a strategy, *F*, and *CR* taken from the lists associated with the target vector. If the obtained trial individual is better than the target vector, the used strategy, *F*, and *CR* will enter a winning strategy list (*wSL*), a winning *F* list (*wFL*), and a winning *CR* list (*wCRL*), respectively. After a given number of iterations, the *FL*, *CRL* or *SL* will be refilled at a high probability by selecting elements from *wFL*, *wCRL* and *wSL* or randomly generated values. In this way, both the trial vector generation strategy and its associated parameters can be gradually self-adapted to match different phases of evolution by learning from their previous successful experience. Extensive computational simulations and comparisons are carried out by employing a set of 19 benchmark problems from the literature. The computational results show that overall the SspDE algorithm performs better than the state-of-the-art differential evolution variants.

1. Introduction

The differential evolution (DE) algorithm is a simple yet powerful search technique introduced by Storn and Price [24] for solving complex continuous nonlinear functions. Starting from a population of randomly initialized solutions, the DE algorithm employs simple mutation and crossover operators to generate new candidate solutions, and utilizes a one-to-one competition scheme to deterministically decide whether the offspring will replace their parents in the next generation. Due to its simplicity, easy implementation, and fast convergence, the DE algorithm has gained much attention with successful applications in mechanical engineering, sensor networks, scheduling, pattern recognition and in other domains [8,9,14,15,21,25,28,32].

There exist many different DE trial vector generation strategies. These strategies often possess different searching capabilities in various search phases of the evolution process. Moreover, their associated control parameters, namely the scaling factor, *F*, and the crossover rate, *CR*, may significantly influence the searching accuracy and convergence speed of the DE algorithm [3,4,10,18,30]. Therefore, it is of significance to determine suitable trial vector generation strategies and their associated parameter values for the DE algorithm when it is used to solve real problems in scientific and engineering fields. Several empirical guidelines exist in literature for choosing suitable trial vector generator strategies and parameter settings. For example, Storn and Price [17,23] stated that the control parameters of the DE algorithm were not difficult to choose, and suggested that a reasonable *NP* value should be between $5n$ and $10n$, and an effective *F* value should be in the range [0.4,1], where *NP* is the population size, and *n* is dimensionality of the problem to be solved. If the problem is near unimodal or fast convergence is desired, $CR=0.9$ is a good initial choice. Furthermore, Price [16] recommends that it is good to set $NP=20n$, $K=0.5$, and $F=0.8$ when the trial vector generation strategy DE/Current-to-rand/1 is used, where *K* is a scaling factor in DE/Current-to-rand/1. However, based on parameter settings for the DE algorithm on Sphere, Rosenbrock and Rastrigin functions, Gamperle [4] reported that choosing the proper control parameters for the DE algorithm was more difficult than expected, and they advised that *NP* should be in the range $[3n,8n]$, and *F* should be equal to 0.6 and *CR* is between 0.3 and 0.9. Recently, Ronkkonen et al. [22] suggested using *F* values in [0.4,0.95], and the *CR* value in [0,0.2] for separable functions while *CR* in [0.9,1] for dependent functions. Obviously, the rules for

* Corresponding author.
E-mail address: epnsugan@ntu.edu.sg (P.N. Suganthan).

0305-0548/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2010.06.007

choosing the control parameters of the DE algorithm are quite different as their validity is restricted to the problems, strategies, and parameters values considered in the respective investigation.

The best trial vector generation strategy and the associated parameter values can be different for different problems and even for the same problem during different stages of the search process. Therefore, researchers have developed some DE variants with adaptive control parameters and trial vector generation strategies to suit various requirements during the evolution process. Liu and Lampinen [12] proposed a fuzzy adaptive differential evolution (FADE) algorithm, where a fuzzy logic controller was used to dynamically adapt the mutation and crossover parameters. In order to retain population diversity, Zaharie [30] proposed an adaptive DE (ADE) algorithm with multiple populations, and an adaptive Pareto DE algorithm for solving multi-objective optimization problems [31]. By encoding a crossover rate into each individual, Abbass [1] developed another self-adaptive DE algorithm for solving multi-objective optimization problems. Omran et al. [13] presented a self-adaptation method to control the scaling factor F analogous to the adaptation of the crossover rate CR in [1]. This approach (called SDE) was tested on four benchmark functions and performed better than other versions of the DE algorithm. Teo [27] proposed a DE algorithm with self-adapting populations, based on Abbass's self-adaptive Pareto DE algorithm [1]. Recently, Brest et al. [2] presented a DE algorithm, called JDE, with self-adapting parameters F and CR by encoding the parameters into each individual and adapting them by means of evolution. The authors' experimental results showed that the JDE algorithm was better or at least comparable to the standard DE algorithm and evolutionary algorithms from the literature in terms of the quality of the solutions found. In addition, the JDE algorithm gave better results in comparison with the FADE algorithm. Based on their former work [20], Qin et al. [19] developed a self-adaptive DE (SaDE) algorithm for constrained real-parameter optimization, in which both trial vector generation strategies and the associated control parameter values were gradually self-adapted according to the learning experiences. Later, the authors extended the SaDE algorithm to solve unconstrained optimization problems [7], and their experiments demonstrated [19] that the SaDE algorithm performed much better than both the conventional DE algorithm and several state-of-art adaptive parameter DE variants including the ADE, SDE and JDE algorithms.

In this paper, we propose a self-adaptive DE algorithm, namely SspDE, with each target individual having its own trial vector generation strategy, scaling factor F and crossover rate CR. During the evolution, both strategy and control parameters of each target individual can be gradually self-adapted from their previous experience in generating promising solutions. Computational experiments and comparisons show that the proposed algorithm overall performs better than the state-of-the-art DE variants such as JDE and SaDE, when applied to optimize 19 benchmark global optimization problems.

The rest of the paper is organized as follows: in Section 2, the traditional DE algorithm is introduced. In Section 3, the SspDE algorithm is described in detail. Section 4 lists the benchmark problems. Experimental design and comparisons are presented in Section 5. Finally, Section 6 gives the concluding remarks.

2. The DE algorithm

The traditional DE algorithm starts with initializing a population of NP target individuals $P_G=\{X_{1,G},X_{2,G},\ldots,X_{NP,G}\}$, where individual $X_{i,G}=(x^1_{i,G},x^2_{i,G},\ldots,x^n_{i,G})$, $i=1,2,\ldots,NP$, is an n-dimensional vector with parameter values determined randomly and uniformly between predefined search ranges $[X_{\min}, X_{\max}]$, where $X_{\min}=(x^1_{\min},x^2_{\min},\ldots,x^n_{\min})$ and $X_{\max}=(x^1_{\max},x^2_{\max},\ldots,x^n_{\max})$. Then mutation and crossover operators are employed to generate new candidate vectors, and a selection scheme is applied to determine whether the offspring or the parent survives to the next generation. The above process is repeated until a termination criterion is reached.

2.1. Mutation

A mutant individual, denoted as $V_{i,G}=(v^1_{i,G},v^2_{i,G},\ldots,v^n_{i,G})$, $i=1,2,\ldots,NP$, is generated by using a mutation operator. There are many mutation strategies in the literature [6]. Among them, the commonly used operator is 'DE/rand/1', which is described as

$$V_{i,G}=X_{a,G}+F\times(X_{b,G}-X_{c,G}) \tag{1}$$

where a, b and c are three randomly chosen indices in the range $[1, NP]$ such that a, b, c and i are pairwise different ($a\neq b\neq c\neq i\in\{1,\ldots,NP\}$). $F>0$ is a mutation scaling factor which affects the differential variation between two individuals.

2.2. Crossover

After the mutation phase, a crossover operator is applied to each mutant individual and its corresponding target individual to yield a trial vector, $U_{i,G}=(u^1_{i,G},u^2_{i,G},\ldots,u^n_{i,G})$. Binomial and exponential crossovers are two commonly used crossover schemes [18]. The binomial crossover is represented as follows:

$$u^j_{i,G}=\begin{cases} v^j_{i,G} & \text{if or } j=n_j \\ x^j_{i,G} & \text{otherwise}\end{cases} \tag{2}$$

where the index n_j refers to a randomly chosen dimension in the set $\{1,2,\ldots,n\}$, which is used to ensure that at least one dimension of the trial individual, $U_{i,G}$, differs from its target vector, $X_{i,G}$. CR is a crossover rate in the range [0,1], and $r_j\in[0,1]$ is a uniform random number. If the parameter values of the obtained trial individuals exceed the pre-specified upper bound or lower bound, we can set them equal to upper bound or lower bound, respectively.

2.3. Selection

In order to decide whether or not the trial individual $U_{i,G}$ should become a member of the target population in the next generation, a one-to-one greedy selection between a parent and its corresponding offspring is employed in DE as this strategy enhances diversity in comparison to other selection strategies such as tournament selection, rank based selection and fitness proportional selection. The one-to-one selection scheme is based on the survival of the fitter between the trial individual $U_{i,G}$ and its target counterpart $X_{i,G}$. For minimization problems, it can be formulated as follows [18]:

$$X_{i,G+1}=\begin{cases} U_{i,G} & \text{if } f(U_{i,G})\le f(X_{i,G}) \\ X_{i,G} & \text{otherwise}\end{cases} \tag{3}$$

where $f(U_{i,G})$ and $f(X_{i,G})$ are the objectives of $U_{i,G}$ and $X_{i,G}$, respectively.

2.4. The algorithmic description

Based on the above initialization, mutation, crossover and selection operations, the algorithmic description of the conventional DE algorithm is summarized in Table 1.