Applied Soft Computing 30 (2015) 529–548

Contents lists available at ScienceDirect

Applied Soft Computing

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

A directional mutation operator for differential evolution algorithms

Xin Zhang[a], Shiu Yin Yuen[b,*]

[a] College of Electronic and Communication Engineering, Tianjin Normal University, Tianjin, China
[b] Department of Electronic Engineering, City University of Hong Kong, Hong Kong, China

ARTICLE INFO

Article history:
Received 26 June 2012
Received in revised form 3 February 2015
Accepted 3 February 2015
Available online 11 February 2015

Keywords:
Differential evolution
Directional mutation
Generic mutation operator
Global numerical optimization

ABSTRACT

Differential evolution (DE) is widely studied in the past decade. In its mutation operator, the random variations are derived from the difference of two randomly selected different individuals. Difference vector plays an important role in evolution. It is observed that the best fitness found so far by DE cannot be improved in every generation. In this article, a directional mutation operator is proposed. It attempts to recognize good variation directions and increase the number of generations having fitness improvement. The idea is to construct a pool of difference vectors calculated when fitness is improved at a generation. The difference vector pool will guide the mutation search in the next generation once only. The directional mutation operator can be applied into any DE mutation strategy. The purpose is to speed up the convergence of DE and improve its performance. The proposed method is evaluated experimentally on CEC 2005 test set with dimension 30 and on CEC 2008 test set with dimensions 100 and 1000. It is demonstrated that the proposed method can result in a larger number of generations having fitness improvement than classic DE. It is combined with eleven DE algorithms as examples of how to combine with other algorithms. After its incorporation, the performance of most of these DE algorithms is significantly improved. Moreover, simulation results show that the directional mutation operator is helpful for balancing the exploration and exploitation capacity of the tested DE algorithms. Furthermore, the directional mutation operator modifications can save computational time compared to the original algorithms. The proposed approach is compared with the proximity based mutation operator as both are claimed to be applicable to any DE mutation strategy. The directional mutation operator is shown to be better than the proximity based mutation operator on the five variants in the DE family. Finally, the applications of two real world engineering optimization problems verify the usefulness of the proposed method.

© 2015 Elsevier B.V. All rights reserved.

1. Introduction

Evolutionary algorithms (EAs)[1,2] are inspired from natural evolution of species. The procedures of EAs parallel those in the evolutionary process of species. Typically, EAs are population-based and depend on variation operators and survivor selection to realize the evolutionary process. They only assume that function values can be obtained given a feasible solution. There is no assumption about the explicit expression or differentiability of the function. In EAs, the function to be optimized is often called fitness functions; the domain of variables called search space; a feasible solution called individual and the function value of a feasible solution called fitness or fitness value. In practice, EAs have been applied to many fields such as engineering design[3], energy management[4], financial strategies [5] and computer vision [6] etc. These applications justify the usefulness of EAs.

Generally, the study of EAs is targeted to propose an algorithm that is applicable to a class of problems, is computationally efficient and converges quickly to the global optimum. Several popular EAs are genetic algorithm (GA)[7], genetic programming (GP), evolution strategies (ES), evolutionary programming (EP) and differential evolution (DE). Note that some researchers consider DE as a swarm intelligence algorithm. The classic version of DE is simple to implement, ea sy to use and fast. Although some classic EAs are easy to be programmed and computationally efficient, yet the classic version of EA is often stuck in a local optimum of fitness functions. Hence, numerous researches are proposed to balance the exploitation and exploration search process of EAs. A good and robust EA should not only have a fast convergence rate, but can also reach the global optimum for complex fitness function with many local optima.

DE, proposed in the mid-1990s, has been extensively studied. Many variants of DE are reported in the past decade. The paradigm of DE is shown to be very powerful. For example, it secures competitive rankings in all IEEE Congress on Evolutionary Computation

* Corresponding author. Tel.: +852 34427717.
E-mail addresses: xinzhang9-c@my.cityu.edu.hk (X. Zhang), kelviny.ee@cityu.edu.hk (S.Y. Yuen).

http://dx.doi.org/10.1016/j.asoc.2015.02.005
1568-4946/© 2015 Elsevier B.V. All rights reserved.

(CEC) competitions, whereas no other search paradigm presents such a good performance [8]. The procedure that makes DE stand out is the mutation operator. Specifically, three mutually different individuals are randomly selected from the population. One of them is set as a base, and then it is added with the scaled difference of the other two individuals. To scale the difference is to control the search neighborhood and thus to increase the exploitation and exploration capacity. In this paper, without loss of generality, we assume that the optimization is a minimization problem; fitness improvement means the fitness of a generated individual is less than or equal to the minimal fitness found so far by the algorithm. Why a new individual having equal minimal fitness is seen as an improvement? The reason is to escape from a plateau.

Through the above analysis, it is known that difference vectors are helpful for fitness improvement. It is important to point out that these differences only provide a possibility of improvement. It is not guaranteed that fitness would be improved at each mutation operation, or even after a number of mutation operations, say in one generation. Actually, in computer simulations, we observe that the number of generations that can obtain fitness improvement is less than half of the total number of generations (see Section 4.3). Moreover, not only is improving the fitness imperative, but the quantity of improvement is also crucial; otherwise, if only a slight improvement is obtained at every generation, the ultimate performance would not be good either, especially when the computational time or the total number of function evaluations is limited.

On the one hand, the difference of two individuals is helpful for improving fitness; one the other hand, fitness improvement cannot happen at every generation. In the case that fitness improvement happens, the child individual should contain some better components than its parent individual. In this case, the difference vector of the child and parent individual can be seen as a good direction. Note that this direction may not be a descent direction. Because this direction leads to a child having better fitness than its parent, a further fitness improvement is possible if continuously searching along this direction. This difference vector should have a higher probability to be a descent direction resulting in better fitness value than the difference of two randomly selected individuals. Hence, we come up with the idea that uses the difference between the child and parent individuals in the case that fitness value is improved to guide the search in the next generation. It is expected to increase the number of generations that improve the fitness, and thus accelerate the convergence process and achieve a better performance. Note that our discussion is based on fitness improvement between two consecutive generations. It is on the population level. The difference information is not immediately fed back to the current generation. In other words, we discuss whether fitness value is improved after a new population of individuals is generated. Once fitness improvement happens between two consecutive generations, the differences will be used to guide the search of the next generation.

The character of this idea is similar to the swarm behavior in a swarm system. In swarm intelligence, individuals are sent out to search for a good fitness and share their collected information. Then, more efforts are paid to search the neighborhood of the location where high fitness is obtained; whereas less effort is expended to search the neighborhood of the location with low fitness obtained. In our case, once fitness value is improved at a generation, the difference vectors are collected. Clearly, they contain direction information toward to high fitness. In the next generation, more effort is made to search along these difference vectors.

The most important character of this idea is that the idea is *generic*; it can be combined with any variants of DE. Because it works on the mutation operator and mutation is an essential part of DE, hence, the proposed method can be incorporated into any variants of DE. It is expected that this idea is an intrinsic improvement to DE.

The rest of the paper is organized as follows. Section 2 reports a summary of DE and related works. Section 3 presents the proposed approach, its pseudo-code and an analysis of this approach. Section 4 reports the experimental results compared with classic DE, and results after combining the proposed method with eleven DE algorithms. The proposed approach is also compared with the proximity-based mutation operator. It is tested on two real-world engineering optimization problems in Section 5. Section 6 gives the conclusion.

2. Differential evolution and related works

This section describes the classic DE method and mutation variants in the DE family of Storn and Price [8,9]. Some related works since 2005 are given in three categories.

Through the paper, suppose the real parameter optimization problem is to find the global minimization solution. The problem is represented as $f(\cdot)$. An individual $\mathbf{x}$ is represented as a $D \times 1$ column vector, e.g. $\mathbf{x} = [x_1, x_2, \cdots, x_D]^T$, where D is the dimension of the optimization problem, $x_j (1 \le j \le D)$, is the jth component (gene) of $\mathbf{x}$. $f(\mathbf{x})$ means the function value at $\mathbf{x}$. In this paper, fitness value is used as a synonym of function value. The search space Ω is delimited by two vectors $\mathbf{x}^{\min} = [x_1^{\min}, \cdots, x_j^{\min}, \cdots, x_D^{\min}]^T$ and $\mathbf{x}^{\max} = [x_1^{\max}, \cdots, x_j^{\max}, \cdots, x_D^{\max}]^T$, denoting the lower bound and upper bound of the search space.

2.1. Classic differential evolution

In general, DE is composed of four steps as first introduced by Storn and Price [10,11]. The four steps are initialization, mutation, crossover, and survivor selection. DE is a population based method. The initialization step is to randomly generate a population of NP individuals, where NP is the population size predefined by the user. Each component of an individual is randomly generated between the lower bound and upper bound. The initialized population is denoted as $G_1 = \{\mathbf{x}_{1,1}, \cdots, \mathbf{x}_{i,1}, \cdots, \mathbf{x}_{NP,1}\}$, where the two subscripts of each individual stand for the order of the individual in the population and the number of the generations, respectively. The number of generations NG is predefined by the user. Mathematically, the ith individual $\mathbf{x}_{i,1} = [x_{1,i,1}, \cdots, x_{j,i,1}, \cdots, x_{D,i,1}]^T$ in the first generation is generated as follows:

$$x_{j,i,1} = x_j^{\min} + (x_j^{\max} - x_j^{\min}) \cdot rand(0, 1), j = 1, \cdots, D$$

where $rand(0, 1)$ returns a uniformly distributed random number between 0 and 1 ($0 \le rand(0, 1) \le 1$).

Example 1. Suppose a minimization problem is given as follows:

$$\begin{aligned} \min \quad & f(\mathbf{x}) = x_1^2 + x_2^2 \\ s.t. \quad & -5 \le x_j \le 5, j = 1, 2. \end{aligned}$$

The problem dimension D is 2. The population size NP is 5. Initially five solutions are randomly created. Suppose they are $G_1 = \{\mathbf{x}_{1,g}, \cdots, \mathbf{x}_{i,g}, \cdots, \mathbf{x}_{5,g}\}$: $\mathbf{x}_{1,g} = [-4, -4]^T$, $\mathbf{x}_{2,g} = [-3, -3]^T$, $\mathbf{x}_{3,g} = [-1, -1]^T$, $\mathbf{x}_{4,g} = [2, 2]^T$, $\mathbf{x}_{5,g} = [4, 4]^T$. Accordingly, their function values are $f(\mathbf{x}_{1,g})$=32, $f(\mathbf{x}_{2,g})$=18, $f(\mathbf{x}_{3,g})$=2, $f(\mathbf{x}_{4,g})$=8, $f(\mathbf{x}_{5,g})$=32.

Denote the current generation as G_g. In the mutation step, NP mutant vectors will be generated, denoted as $V = \{\mathbf{v}_{1,g}, \cdots, \mathbf{v}_{i,g}, \cdots, \mathbf{v}_{NP,g}\}$. Now, take the generation of the ith mutant vector $\mathbf{v}_{i,g}$ for example. Select three mutually different individuals from the current population. Suppose the chosen individuals are denoted as $\mathbf{x}_{r1,g}, \mathbf{x}_{r2,g}, \mathbf{x}_{r3,g}$, where $r1$, $r2$ and $r3$ are three mutually

Download English Version:

https://daneshyari.com/en/article/495086

Download Persian Version:

https://daneshyari.com/article/495086

Daneshyari.com