Swarm and Evolutionary Computation 32 (2017) 1–24

Contents lists available at ScienceDirect

Swarm and Evolutionary Computation

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

Survey Paper

Review of Differential Evolution population size

Adam P. Piotrowski

Institute of Geophysics, Polish Academy of Sciences, Ks. Janusza 64, 01-452 Warsaw, Poland

ARTICLE INFO

Article history:
Received 2 June 2015
Received in revised form
29 January 2016
Accepted 31 May 2016
Available online 4 June 2016

Keywords:
Differential Evolution
Population size
Adaptive control parameters
Evolutionary Algorithms
Metaheuristics

ABSTRACT

Population size of Differential Evolution (DE) algorithms is often specified by user and remains fixed during run. During the first decade since the introduction of DE the opinion that its population size should be related to the problem dimensionality prevailed, later the approaches to DE population size setting diversified. In large number of recently introduced DE algorithms the population size is considered to be problem-independent and often fixed to 100 or 50 individuals, but alongside a number of DE variants with flexible population size have been proposed.

The present paper briefly reviews the opinions regarding DE population size setting and verifies the impact of the population size on the performance of DE algorithms. Ten DE algorithms with fixed population size, each with at least five different population size settings, and four DE algorithms with flexible population size are tested on CEC2005 benchmarks and CEC2011 real-world problems. It is found that the inappropriate choice of the population size may severely hamper the performance of each DE algorithm. Although the best choice of the population size depends on the specific algorithm, number of allowed function calls and problem to be solved, some rough guidelines may be sketched. When the maximum number of function calls is set to classical values, i.e. those specified for CEC2005 and CEC2011 competitions, for low-dimensional problems (with dimensionality below 30) the population size equal to 100 individuals is suggested; population sizes smaller than 50 are rarely advised. For higher-dimensional artificial problems the population size should often depend on the problem dimensionality d and be set to $3d$–$5d$. Unfortunately, setting proper population size for higher-dimensional real-world problems ($d > 40$) turns out too problem and algorithm-dependent to give any general guide; 200 individuals may be a first guess, but many DE approaches would need a much different choice, ranging from 50 to $10d$. However, quite clear relation between the population size and the convergence speed has been found, showing that the fewer function calls are available, the lower population sizes perform better.

Based on the extensive experimental results the use of adaptive population size is highly recommended, especially for higher-dimensional and real-world problems. However, which specific algorithms with population size adaptation perform better depends on the number of function calls allowed.

1. Introduction

Among three main control parameters of Differential Evolution (DE) [128] optimization algorithms the significance of scale factor (F) and crossover rate (CR) has already been deeply researched – see the review in [31,99]. In many recently introduced DE algorithms the values of F and CR do not need to be specified by the user, but are adapted or self-adapted during run [12,48,66,75,81,82,92,118,124,171]. The impact of the third control parameter, population size (*PS*), on the performance of DE algorithms has been rarely studied so far, and to motivate the choice of *PS* in many papers the reader is referred to [50], an interesting but old and very brief study. This may be surprising, as in case of other population-based Evolutionary Algorithms (EA) *PS* attracted large attention both in empirical and theoretical studies [2,22,39,40,58,76,84,88,93]. In the overwhelming majority of DE algorithms (as well as other EAs [41]) *PS* needs to be pre-specified and is kept fixed during run [31]. Researchers that propose novel DE methods suggest setting *PS* to very different values (differences exceed an order of magnitude – see [31,98,121]), but in a few studies [3,28,47,61,96,119,138,148] these choices are backed by the analysis of the impact of *PS* on the performance of the proposed algorithm. Although in recent years a number of DE algorithms with variable *PS* have been proposed [14–16,43,51,107,131,134,136,145,149,173,175], surprisingly such approaches have never been compared with each other (even more recently introduced DE algorithms with flexible *PS* are not compared with their older counterparts in the source papers) and their superiority over DE algorithms with fixed population size has not

E-mail address: adampp@igf.edu.pl

http://dx.doi.org/10.1016/j.swevo.2016.05.003
2210-6502/

been verified. In fact, answers to the questions like 1. "how important is *PS* for the performance of DE algorithms", 2. "how large *PS* should be", 3. "should *PS* depend on the dimensionality of the problem to be solved", 4. "should *PS* be fixed or modified during run", 5. "what concepts the population size adaptation should follow" and 6. "how to relate *PS* with the number of allowed function calls" may today be only intuitive, due to the lack of detailed study on this topic. As the reader may expect, the above questions are addressed in the present paper.

To verify the impact of the population size on the performance of DE optimizers, and to find some rules how to choose DE population size, in this study 10 various DE algorithms are tested with 5–10 different but fixed population sizes on 2- to 50-dimensional CEC2005 benchmarks [130] and 1- to 216-dimensional CEC2011 real-world problems [30]. To understand and verify the efficiency of this few population size adaptation schemes that have so far been proposed, 7 variants of 4 DE algorithms with variable population size are tested on the same problems.

The scope of the study is limited to DE approaches developed for low- to moderately high-dimensional single-objective non-dynamic optimization problems. Due to the length of the manuscript, the main criterion for comparison of algorithms and their population sizes is the best objective function value found within the maximum number of function calls that has been defined for CEC2005 and CEC2011 problems in source publications [30,130]. However, in many practical applications convergence speed is an important factor; to get this into account the comparison of the results obtained after two much smaller numbers of function calls is also considered in this study.

2. Differential Evolution

The first population-based DE optimization method has been introduced in [128,129]. Although there are plenty of DE algorithms today [31], most of them follows basically the similar scheme. After initial random generation from the uniform distribution, in every generation g individuals $\mathbf{x}_{i,g} = \left\{x_{i,g}^{1}, \ldots, x_{i,g}^{d}\right\}$, $i = 1, \ldots, PS$ are evolved in order to find the vector $\mathbf{x}^{*}$ such that

$$f(\mathbf{x}^{*}) = \min_{\mathbf{x}\in\Omega\subseteq\mathbf{R}^{d}} f(\mathbf{x}) \tag{1}$$

when f: $\mathbf{R}^{d} \to \mathbf{R}$. The search is often restricted within the subset $\prod_{j=1}^{d}[L^{j}, U^{j}]$. The non-classical "*d*" notation, instead of "*D*", is used in this study to facilitate noting visually the difference between "100" and "10*d*" that will frequently be used through the paper.

In each generation DE performs three steps called mutation, crossover and selection. Each individual, or parent ($\mathbf{x}_{i,g}$) creates first so-called donor vector ($\mathbf{v}_{i,g}$) by means of some mutation strategy. Plenty mutation schemes has been proposed so far [6,18,46,62,64,75,78,117,151,153], for example

DE/rand/1

$$\mathbf{v}_{i,g} = \mathbf{x}_{r1,g} + F\cdot(\mathbf{x}_{r2,g} - \mathbf{x}_{r3,g}) \tag{2}$$

DE/best/2

$$\mathbf{v}_{i,g} = \mathbf{x}_{best,g} + F\cdot(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) + F\cdot(\mathbf{x}_{r3,g} - \mathbf{x}_{r4,g}) \tag{3}$$

DE/current-to-*p*best/1

$$\mathbf{v}_{i,g} = \mathbf{x}_{i,g} + F\cdot(\mathbf{x}_{best,g}^{p} - \mathbf{x}_{i,g}) + F\cdot(\mathbf{x}_{r1,g} - \mathbf{x}_{r2,g}) \tag{4}$$

In above equations F is the control parameter called scaling factor, $r1$, $r2$, $r3$, and $r4$ are randomly selected integers from the range [1,*PS*], such that $r1 \neq r2 \neq r3 \neq r4 \neq i$, $\mathbf{x}_{\mathrm{best,g}}$ is the best individual in the current population and $\mathbf{x}_{\mathrm{best,g}}^{p}$ is a randomly selected individual from the top *p*% best individuals of the current population.

After mutation a crossover between donor ($\mathbf{v}_{i,g}$) and parent ($\mathbf{x}_{i,g}$) vector is performed to generate an offspring (or trial vector) ($\mathbf{u}_{i,g}$). Although there are few crossover schemes (see the detailed discussion in [5,63,86,148,159,167,172]), in the vast majority of DE algorithms a binomial crossover is used

$$u_{i,g}^{j} = \begin{cases} v_{i,g}^{j} & \text{if} \quad rand_{i}^{j}(0,1) \le CR \quad or \quad j = j_{rand,i} \\ x_{i,g}^{j} & \text{otherwise} \end{cases} \tag{5}$$

where the value of the control parameter CR should be set within [0,1] interval, $rand_{i}^{j}(0,1)$ means a random number generated within [0,1] interval from the uniform distribution and $j_{rand,i}$ is a randomly selected integer from [1,*d*] range. At this point often some constraints or bounds handling approaches are applied.

After crossover the objective function is evaluated for $\mathbf{u}_{i,g}$, and according to the greedy selection only the better of the offspring-parent pair is passed to the next generation

$$\mathbf{x}_{i,g+1} = \begin{cases} \mathbf{u}_{i,g} & \text{if} \quad f(\mathbf{u}_{i,g}) \le f(\mathbf{x}_{i,g}) \\ \mathbf{x}_{i,g} & \text{otherwise} \end{cases} \tag{6}$$

DE continues the search until the stopping criteria are met (that are frequently defined by setting the maximum number of function calls).

Many ideas how to improve DE algorithms have been proposed – they are not discussed here, for a review the reader is referred to [31,99]. However, some of DE modifications led to approaches that do not strictly follow the "classical" scheme given above. For example, distributed DE algorithms [1,7,32,109,110,155] divide the total DE population into sub-populations that may behave differently, and set the rules of communication, or migration of individuals, between them. Memetic DE methods combine DE paradigm with some local search procedures to speed up exploitation without loosing the global search capabilities [20,77,97,100,111]. DE has also been hybridized with many other heuristic algorithms [1,11,25,57,59,81,85,137,164,174], and was implemented as a part of multialgorithms [106,132,142–144]. As such approaches often do not follow DE scheme discussed above, the impact of the population size on their performance may depend on many specific features. Such "non-classical" DE concepts are not researched in this study.

3. Population size in Differential Evolution

Population size of DE must be set larger than the number of different randomly selected integers in the chosen mutation operation, otherwise difference vectors could not be constructed (consult Eqs. (2)–(4)). To understand the importance of the population size, some insight into the behavior of DE algorithms is needed. According to [49,155] DE does not require maintaining high population diversity during the whole search, although this opinion is not always acknowledged [165]. As in DE the step size depends mainly on the magnitude of difference vectors, and hence on the distance between individuals in the search space, DE diversity should decrease during run to allow ultimate convergence to some local optima. The dependence of DE on magnitude of difference vectors is frequently considered a main advantage of DE by practitioners, but on the other hand it does not allow proving the convergence of classical DE with probability 1 without adding some extra terms [71,72]. The scheme of DE behavior may be summarized as follows [49,95,155]. Initially individuals are randomly generated in the search space, hence distances between