Differential Evolution with Self-Adaptation

Janez Brest
University of Maribor, Slovenia

INTRODUCTION

Many practical engineering applications can be formulated as a global optimization problem, in which objective function has many local minima, and derivatives of the objective function are unavailable. Differential Evolution (DE) is a floating-point encoding evolutionary algorithm for global optimization over continuous spaces (Storn & Price, 1997) (Liu & Lampinen, 2005) (Price, Storn & Lampinen, 2005) (Feoktistov, 2006). Nowadays it is used as a powerful global optimization method within a wide range of research areas.

Recent researches indicate that self-adaptive DE algorithms are considerably better than the original DE algorithm. The necessity of changing control parameters during the optimization process is also confirmed based on the experiments in (Brest, Greiner, Bošković, Mernik, Žumer, 2006a). DE with self-adaptive control parameters has already been presented in (Brest et al., 2006a).

This chapter presents self-adaptive approaches that were recently proposed for control parameters in DE algorithm.

BACKGROUND

Differential Evolution

DE creates new candidate solutions by combining the parent individual and several other individuals of the same population. A candidate replaces the parent only if it has better fitness value.

The population of the original DE algorithm (Storn & Price, 1995) (Storn & Price, 1997) contains *NP D*-dimensional vectors: $\mathbf{x}_{i,G}$, $i = 1, 2, \ldots, NP$. G denotes the generation. The initial population is usually selected uniform randomly between the lower and upper bounds. The bounds are specified by the user according to the nature of the problem. After initialization DE performs several vector transforms (operations): mutation, crossover, and selection.

Mutant vector $\mathbf{v}_{i,G}$ can be created by using one of the mutation strategies (Price et al., 2005). The most useful strategy is 'rand/1': $\mathbf{v}_{i,G} = \mathbf{x}_{r1,G} + \mathrm{F} \times (\mathbf{x}_{r2,G} - \mathbf{x}_{r3,G})$, where F is the mutation scale factor within range [0, 2], usually less than 1. Indexes *r1*, *r2*, *r3* represent the random and distinct integers generated within range [1, *NP*], and also different from index *i*.

After mutation, a 'binary' crossover operation forms the trial vector $\mathbf{u}_{i,G}$, according to the i^{th} population vector and its corresponding mutant vector $\mathbf{v}_{i,G}$:

$$\underline{\text{if}}\ (rand \le CR\ \underline{\text{or}}\ j = j_{rand})\ \underline{\text{then}}\ u_{i,j,G} = v_{i,j,G}\ \ \underline{\text{else}}\ u_{i,j,G} = x_{i,j,G}\ ,$$

where $i = 1, 2, \ldots, NP$ and $j = 1, 2, \ldots, D$. CR is the crossover parameter or factor within the range [0,1] and presents the probability of creating parameters for the trial vector from the mutant vector. Uniform random value *rand* is within [0, 1]. Index $j_{rand} \in [1, NP]$ is a randomly chosen index and is responsible for the trial vector containing at least one parameter from the mutant vector.

The selection operation selects, according to the objective fitness value of the population vector $\mathbf{x}_{i,G}$ and its corresponding trial vector $\mathbf{u}_{i,G}$, which vector will survive to be a member of the next generation.

The original DE has more strategies and Feoktistov (Feoktistov, 2006) proposed some general extensions to DE strategies. The question is which strategy is the most suitable to solve a particular problem. Recently some researchers used various combinations of two, three or even more strategies during the evolutionary process.

Parameter Tuning and Parameter Control

Globally, we distinguish between two major forms of setting parameter values: parameter *tuning* and parameter *control* (Eiben, Hinterding & Michalewicz, 1999). The former means the commonly practiced approach that tries to find good values for the parameters before running the algorithm, then tuning the algorithm using

these values, which remain fixed during the run. The latter means that values for the parameters are changed during the run. According to Eiben *et al.* (Eiben et al., 1999) (Eiben & Smith, 2003), the change can be categorized into three classes:

1. *Deterministic parameter control* takes place when the value of a parameter is altered by some deterministic rule.
2. *Adaptive parameter control* is used when there is some form of feedback from the search that is used to determine the direction and/or the magnitude of the change to the parameter.
3. *Self-adaptive parameter control* is the idea that "evolution of the evolution" can be used to implement the self-adaptation of parameters. Here, the parameters to be adapted are encoded into the chromosome (individuals) and undergo the actions of genetic operators. The better values of these encoded parameters lead to better individuals which, in turn, are more likely to survive and produce offspring and, hence, propagate these better parameter values.

DE has three control parameters: amplification factor of the difference vector - F, crossover control parameter - CR, and population size - NP. The original DE algorithm keeps all three control parameters fixed during the optimization process. However, there still exists a lack of knowledge about how to find reasonably good values for the control parameters of DE, for a given function (Liu & Lampinen, 2005).

Although the DE algorithm has been shown to be a simple, yet powerful, evolutionary algorithm for optimizing continuous functions, users are still faced with the problem of preliminary testing and hand-tuning of its control parameters prior to commencing the actual optimization process (Teo, 2006). As a solution, self-adaptation has proved to be highly beneficial for automatically and dynamically adjusting control parameters. Self-adaptation allows an evolutionary strategy to adapt itself to any general class of problem, by reconfiguring itself accordingly, and does this without any user interaction (Bäck, 2002) (Bäck, Fogel & Michalewicz, 1997) (Eiben, Hinterding & Michalewicz, 2003).

RELATED WORK

Work Releted to Differential Evolution

The DE (Storn & Price, 1995) (Storn & Price, 1997) algorithm was proposed by Storn and Price, and since then it has been used in many practical cases. The original DE was modified and many new versions have been proposed. Ali and Törn (Ali & Törn, 2004) proposed new versions of the DE algorithm, and also suggested some modifications to the classical DE, in order to improve its efficiency and robustness. They introduced an auxiliary population of NP individuals alongside the original population (noted in (Ali, 2004), a notation using sets is used). Next they proposed a rule for calculating the control parameter F, automatically. Jiao et al. (Jiao, Dang, Leung & Hao, 2006) proposed a modification of the DE algorithm, applying a number-theoretical method for generating the initial population, and using simplified quadratic approximation with the three best points. Mezura-Montes et al. (Mezura-Montes, Velázquez-Reyes & Coello Coello, 2006) conducted a comparative study of DE variants. They proposed a rule for changing control parameter F at random from interval [0.4, 1.0] at generation level. They used different values of control parameter CR for each problem. The best CR value for each problem was obtained by additional experimentation. Tvrdik in (Tvrdik, 2006) proposed a DE algorithm using competition between different control parameter settings. The prominence of the DE algorithm and its applications is shown in recently published books (Price et al., 2005), (Feoktistov, 2006). Feoktistov in his book (Feoktistov, 2006, p. 18) says, that "the concept of differential evolution is a spontaneous self-adaptability to the function".

Work Releted to Adaptive or Self-Adaptive DE

Liu and Lampinen (Liu & Lampinen, 2005) proposed a version of DE, where the mutation control parameter and the crossover control parameter are adaptive. A self-adaptive DE (SDE) is proposed by Omran et al. (Omran, Salman & Engelbrecht, 2005) (Salman, Engelbrecht & Omran, 2007), where parameter tuning is not required. Self-adapting was applied for control