Applied Soft Computing 34 (2015) 770–787

Contents lists available at ScienceDirect

Applied Soft Computing

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

An evolving surrogate model-based differential evolution algorithm

Rammohan Mallipeddi, Minho Lee*

School of Electronics Engineering, Kyungpook National University, 1370 Sankyuk-Dong, Puk-Gu, Taegu 702-701, Republic of Korea

ARTICLE INFO

Article history:
Received 1 May 2013
Received in revised form 20 April 2015
Accepted 10 June 2015
Available online 19 June 2015

Keywords:
Differential evolution
Global optimization
Surrogate model
Parameter adaptation
Ensemble

ABSTRACT

Differential evolution (DE) is a simple and effective approach for solving numerical optimization problems. However, the performance of DE is sensitive to the choice of mutation and crossover strategies and their associated control parameters. Therefore, to achieve optimal performance, a time-consuming parameter tuning process is required. In DE, the use of different mutation and crossover strategies with different parameter settings can be appropriate during different stages of the evolution. Therefore, to achieve optimal performance using DE, various adaptation, self-adaptation, and ensemble techniques have been proposed. Recently, a classification-assisted DE algorithm was proposed to overcome trial and error parameter tuning and efficiently solve computationally expensive problems. In this paper, we present an evolving surrogate model-based differential evolution (ESMDE) method, wherein a surrogate model constructed based on the population members of the current generation is used to assist the DE algorithm in order to generate competitive offspring using the appropriate parameter setting during different stages of the evolution. As the population evolves over generations, the surrogate model also evolves over the iterations and better represents the basin of search by the DE algorithm. The proposed method employs a simple Kriging model to construct the surrogate. The performance of ESMDE is evaluated on a set of 17 bound-constrained problems. The performance of the proposed algorithm is compared to state-of-the-art self-adaptive DE algorithms: the classification-assisted DE algorithm, regression-assisted DE algorithm, and ranking-assisted DE algorithm.

1. Introduction

Differential evolution (DE) [1] is a fast and simple population-based stochastic search technique that is inherently parallel and performs well on a wide variety of real-world problems [2]. The performance of the DE algorithm is sensitive to the population size (*NP*), mutation and crossover strategies, and their associated control parameters, such as the crossover rate (*CR*) and scale factor (*F*) [3,4]. The best combination of strategies and control parameters can be different for different optimization problems and also for the same functions with different computation times and accuracy requirements. Therefore, to successfully solve a specific optimization problem, it is generally necessary to perform a time-consuming trial-and-error search for the most appropriate combination of strategies and their associated parameter values, which results in high computational costs. In addition, when the population of DE evolves through different regions in the search space, different strategies [5] combined with different parameter settings may be more effective than a single combination of strategies and parameters for the entire search. Therefore, to overcome the trial-and-error search for the appropriate combination of strategies and parameter values and to adapt the combination to different regions in the search space, various adaptation schemes have been proposed [5–10]. Recently, a DE algorithm with ensemble of mutation strategies and parameter values (EPSDE) [10,11] was proposed to overcome the time-consuming trial-and-error procedure of the conventional DE algorithm and has shown significant improvement compared to other state-of-the-art algorithms.

In time-consuming and expensive optimization problems, surrogate models built based on the evolutionary history of the population members have been incorporated into different evolutionary algorithms (EAs) to reduce the number of original fitness evaluations [12–14]. A surrogate model is an estimation model, built by using other functions, which is used to estimate the original objective function. Surrogate models can be built using a variety of techniques, such as radial basis functions [13], artificial neural networks, Kriging models [14], support vector machines, splines, and polynomial approximation models. A classification-assisted DE that incorporates classification for pairwise comparison was proposed in [15] to solve computationally expensive problems. The performance of the support vector classification-assisted DE

* Corresponding author. Tel.: +82 1088596436.
E-mail addresses: mallipeddi.ram@gmail.com (R. Mallipeddi), mholee@knu.ac.kr (M. Lee).

http://dx.doi.org/10.1016/j.asoc.2015.06.010
1568-4946/

(SVC-DE) algorithm has been compared with both the support vector regression-assisted DE (SVR-DE) algorithm and ranking-based support vector machine assisted DE (RankSVM-DE) algorithm.

In this paper, we present an evolving surrogate model-based differential evolution algorithm (ESMDE). In the EMSDE, the DE algorithm is equipped with an evolving surrogate model constructed using the evolving population members. The evolving surrogate model enables the EMSDE algorithm to generate competitive offspring during different stages of the evolution with the assistance of the appropriate strategy and parameter combinations. In the present work, we adopt a surrogate model built using the Kriging technique based on the suggested model discussed in [14]. Kriging is a spatial prediction method based on minimizing the mean squared error. We also apply design and analysis of computer experiments (DACE), which is a parametric regression model that can handle three or more dimensions [14]. The performance of the proposed algorithm has been compared with state-of-the-art self-adaptive algorithms, such as JADE [16], CoDE [17], SaDE [5], and EPSDE [11]. The performance of the proposed algorithm has also been compared to the SVC-DE, SVR-DE, and RankSVM-DE algorithms presented in [15].

The reminder of this paper is organized as follows: Section 2 presents a literature survey regarding DE, surrogate models, and the usage of surrogate models in DE. Section 3 presents the proposed ESMDE algorithm. Section 4 presents the experimental results and discussions, and Section 5 concludes the paper.

2. Literature review

2.1. Differential evolution

Differential evolution (DE) is a parallel direct search method that utilizes NP D-dimensional parameter vectors, the so-called individuals, which encode the candidate solutions, *i.e.* $\mathbf{X}_{i,G} = \{x^1_{i,G}, \ldots, x^D_{i,G}\}, \quad i = 1, \ldots, NP$. The initial population needs to cover as much of the entire search space as possible by uniformly randomizing the initial individuals within the search space constrained by the prescribed minimum and maximum parameter bounds $\mathbf{X}_{\min} = \{x^1_{\min}, \ldots, x^D_{\min}\}$ and $\mathbf{X}_{\max} = \{x^1_{\max}, \ldots, x^D_{\max}\}$. For example, the initial value of the jth parameter of the ith individual at generation $G = 0$ is generated by:

$$x^j_{i,0} = x^j_{\min} + rand(0,1).(x^j_{\max} - x^j_{\min}) \quad j = 1, 2, \ldots, D \tag{1}$$

where $rand(0,1)$ represents a uniformly distributed random variable within the range [0,1].

After initialization, the DE algorithm employs a mutation operation to produce a mutant vector $\mathbf{V}_{i,G} = \{v^1_{i,G}, v^2_{i,G}, \ldots, v^D_{i,G}\}$ corresponding to each individual $\mathbf{X}_{i,G}$, the so-called target vector, in the current population. For each target vector $\mathbf{X}_{i,G}$ at generation G, its associated mutant vector can be generated via a certain mutation strategy. The most frequently used mutation strategies are:

"DE/best/1"[18] : $$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) \tag{2}$$

"DE/best/2"[18] : $$\mathbf{V}_{i,G} = \mathbf{X}_{best,G} + F \cdot (\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) + F \cdot (\mathbf{X}_{r^i_3,G} - \mathbf{X}_{r^i_4,G}) \tag{3}$$

"DE/rand/1"[18] : $$\mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F \cdot (\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) \tag{4}$$

"DE/rand/2"[5] : $$\mathbf{V}_{i,G} = \mathbf{X}_{r^i_1,G} + F \cdot (\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) + F \cdot (\mathbf{X}_{r^i_4,G} - \mathbf{X}_{r^i_5,G}) \tag{5}$$

"DE/rand-to-best/1"[18] or"DE/target-to-best/1"[19] :

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G}) \tag{6}$$

"DE/rand-to-best/2"[5] or"DE/target-to-best/2" :

$$\mathbf{V}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{best,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r^i_1,G} - \mathbf{X}_{r^i_2,G} + \mathbf{X}_{r^i_3,G} - \mathbf{X}_{r^i_4,G}) \tag{7}$$

"DE/current-to-rand/1"[20] : $$\mathbf{U}_{i,G} = \mathbf{X}_{i,G} + K \cdot (\mathbf{X}_{r^i_1,G} - \mathbf{X}_{i,G}) + F \cdot (\mathbf{X}_{r^i_2,G} - \mathbf{X}_{r^i_3,G}) \tag{8}$$

The indices $r^i_1, r^i_2, r^i_3, r^i_4, r^i_5$ are randomly generated mutually exclusive integers different from index i and within the range of [1,NP]. These indices are randomly generated once for each mutant vector. The scale factor F is a positive control parameter for scaling the difference vector. $\mathbf{X}_{best,G}$ is the best individual vector with the best fitness value in the population at generation G. K is randomly chosen within the range [0,1].

After the mutation phase, a trial vector $\mathbf{U}_{i,G} = \{u^1_{i,G}, u^2_{i,G}, \ldots, u^D_{i,G}\}$ is generated by the crossover operation on each pair of target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$. The crossover operation speeds the convergence by a constant factor. The DE algorithm generally employs two kinds of crossover methods: binomial (or uniform) and exponential (or two-point modulo). In the basic version, the DE algorithm employs the binomial crossover defined as [1]:

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if } (rand_j[0,1] \le CR) \text{ or } (j = j_{rand}), \quad j = 1, 2, \ldots, D \\ x^j_{i,G} & \text{otherwise} \end{cases} \tag{9}$$

In (9), the crossover probability, $CR \in [0,1]$, is a user-specified constant that controls the fraction of parameter values that are copied to the trail vector from the mutant vector. j_{rand} is a randomly chosen integer in the range [1,D]. The binomial crossover operator copies the jth parameter of the mutant vector $\mathbf{V}_{i,G}$ to the corresponding element in the trial vector $\mathbf{U}_{i,G}$ if $rand_j$ [0,1] or $j = j_{rand}$. Otherwise, it is copied from the corresponding target vector $\mathbf{X}_{i,G}$. The condition $j = j_{rand}$ is introduced to ensure that trial vector $\mathbf{U}_{i,G}$ will differ from its corresponding target vector $\mathbf{X}_{i,G}$ by at least one parameter. Here, the crossover operator is uniform in the sense that each parameter of the mutant vector, regardless of its location, has the same probability CR of inheriting its value from a given to the trail vector. For this reason, the uniform crossover does not exhibit any representational bias.

In the exponential crossover, an integer $n \in [1,D]$, which acts as a starting point in the target vector, is chosen randomly from the point at which the crossover or exchange of components with the mutant vector starts. $L \in [1,D]$ denotes the number of components that are contributed by the mutant vector to the target vector. Integer L is drawn from [1,D] depending on the crossover probability (CR), according to the following pseudo-code:

```
L = 0;
DO
    L = L + 1;
WHILE (rand (0, 1) < CR and L < D);
```

After choosing n and L, the trial vector is obtained by:

$$u^j_{i,G} = \begin{cases} v^j_{i,G}, & \text{for} j = \langle n \rangle_D, \langle n+1 \rangle_D, \ldots, \langle n+L-1 \rangle_D \\ x^j_{i,G}, & \text{for all other} j \in [1, D] \end{cases} \tag{10}$$

where the angular brackets $\langle\,\rangle_D$ denote a modulo function with modulus D.