J. Internet Comput. Serv.
ISSN 1598-0170 (Print) / ISSN 2287-1136 (Online)
http://www.jics.or.kr

A Hybrid Estimation of Distribution Algorithm with Differential Evolution based on Self-adaptive Strategy☆

Debin Fan[1,2] Jaewan Lee[2*]

ABSTRACT

Estimation of distribution algorithm (EDA) is a popular stochastic metaheuristic algorithm. EDA has been widely utilized in various optimization problems. However, it has been shown that the diversity of the population gradually decreases during the iterations, which makes EDA easily lead to premature convergence. This article introduces a hybrid estimation of distribution algorithm (EDA) with differential evolution (DE) based on self-adaptive strategy, namely HEDADE-SA. Firstly, an alternative probability model is used in sampling to improve population diversity. Secondly, the proposed algorithm is combined with DE, and a self-adaptive strategy is adopted to improve the convergence speed of the algorithm. Finally, twenty-five benchmark problems are conducted to verify the performance of HEDADE-SA. Experimental results indicate that HEDADE-SA is a feasible and effective algorithm.

☞ keyword : Hybrid algorithm, Estimation of Distribution, Differential Evolution, Self-adaptive Strategy

1. Introduction

Continuous optimization problems appear in most fields of science and engineering, which have been received considerable attention. Their objective functions have the characteristics of continuity, noise, and other features. Therefore, it is rather difficult to solve by common methods; some metaheuristic algorithms like evolutionary algorithms (EAs) would be taken into account. Classical EAs, such as genetic algorithm (GA), ant colony optimization (ACO), particle swarm optimization (PSO), artificial bee colony (ABC), have been used to deal with these problems [1-3].

EDA first proposed by Mühlenbein and Paass [4], which has been successfully used in a set of academic and practical applications for optimization. For example, Liu et al. [5] present a copula-based EDA (cEDA) for flow-shop scheduling problem. Dong et al. [6] introduced a latent space-based EDA for large-scale global problems (LSGOs). Yang et al. [7] utilized distribution estimation and niching to implement a multimodal EDA for multimodal problems. For constrained problems, Gao et al. [8] proposed an enhanced EDA by extreme elitism selection technique, which indicates better performance compared with other EDAs.

EDA generates offspring based on global information while ignoring location information of the individual. Thus this method can improve the population diversity. However, it is easy to encounter the problem of premature convergence [9]. In addition, since the sample size is fixed, the calculation of probability distribution takes more time [9].

Differential evolution (DE) is another famous stochastic metaheuristics algorithm of EA, which was first introduced by Stron and Price [10] and had strong local search ability [11]. To solve the above issues, the hybridization of EDA and DE algorithms has attracted more and more attention and achieved superior performance. For example, Sun et al. [12] firstly present a combination of DE and EDA operator termed DE/EDA. Shao et al. [13] proposed a hybrid DE/EDA with adaptive incremental learning strategy. Inspired by literature [12], Dong et al. [14] present a hybrid mechanism of DE and EDA with local search strategy, named as EDA/DE-EIG.

1 School of Information Science and Technology, Jiujiang University, Jiujiang, China
2 Department of Information and Communication Engineering, Kunsan National University, Gunsan, Korea
* Corresponding author (jwlee@kunsan.ac.kr)
[Received 16 September 2020, Reviewed 23 November 2020, Accepted 14 January 2021]
☆ This research is supported by Institute of Information and Telecommunication Technology of Kunsan National University, Korea

Fang et al. [15] introduced a hybridization of DE and EDA termed DE/GM, which can provide a Gaussian distribution probabilistic model of EDA and crossover/mutation operators of DE to generate offspring solutions with information fusion.

Although hybridization technique does increase the performance of the algorithm, it still cannot avoid the blindness and randomness of individual evolution. In the past few years, self-adaptive strategy in EAs has aroused great attention, and various self-adaptive strategies have been introduced. For example, Wang et al. [16] introduced a self-adaptive ensemble DE algorithm (SAEDE), in which their relevant parameters were self-adaptive and ensemble. Liu et al. [17] present a self-adaptive bare-bones DE with bi-mutation strategy (SMGBDE) to solve the blindness of mutation strategy. Wang et al. [18] introduced a method combined by self-adaptive mutation DE and PSO (DEPSO). These works indicate that self-adaptive strategy can effectively reduce the blindness and randomness of individual evolution and improve the convergence speed of the algorithm.

Inspired by the above considerations, we design a hybrid EDA with DE based on self-adaptive strategy (HEDADE-SA) in this article. The proposed HEDADE-SA utilizes the self-adaptive strategy to adjust the proportion of EDA and DE operators. In the early stage of the algorithm, HEDADE-SA can use more EDA operator for exploring. In the later stage, HEDADE-SA can use more DE operator to exploit. Moreover, an alternative probability model for sampling is used in HEDADE-SA. Twenty-five test problems of CEC2005 are used to validate the performance of HEDADE-SA.

The remaining article is structured as follows: Section 2 shows a description of EDA and DE. Then, section 3 is devoted to HEDADE-SA. In Section 4, we conduct an experimental analysis of HEDADE-SA. Finally, Section 5 sums up this article.

2. Description of Algorithms

2.1 Estimation of Distribution Algorithm

EDA [4] is a kind of stochastic metaheuristics algorithm. Compared with classical GA, EDA has no crossover or mutation. It uses global statistical information to learn and sample, and then builds a probabilistic model and obtains a promising solution.

The basic steps of EDA are given as follows.

Step 1: Population initialization.

Step 2: Calculating and evaluating the fitness of individuals.

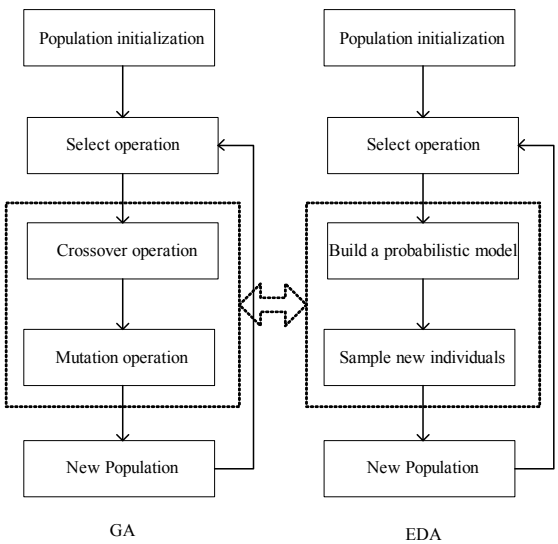
Step 3: Selecting m elite individuals from the population.

Step 4: Generating the probabilistic model according to the elite population.

Step 5: Sampling to create a new population by the probabilistic model from step 4.

Step 6: If algorithm met the termination criterion, output the global optimum; otherwise, goto Step 2.

The flowcharts of GA and EDA are as shown in Figure 1.



(Figure 1) The flowchats of GA and EDA

2.2 Differential Evolution

DE [10] is another kind of population-based random search method, which was proposed in 1995.

The details of classical DE are shown as follows.

Step 1. Population Initialization.

DE usually uses a randomly generated method to initialize the population, which is shown as follows:

$$x_{i,j}(0) = x_{i,j}^{L} + rand(0,1) \cdot (x_{i,j}^{U} - x_{i,j}^{L}) \quad (1)$$

where i = 1,2,…, NP, j = 1,2,…, D. $x_{i,j}^{U}$ indicates the upper constraint, and $x_{i,j}^{L}$ indicates the lower constraint.

Step 2. Mutation.

The frequently utilized mutation strategies are shown as follows：

DE/rand/1:

$$V_i(t) = X_{r_1}(t) + F \cdot (X_{r_2}(t) - X_{r_3}(t)) \quad (2)$$

DE/best/1:

$$V_i(t) = X_{best}(t) + F \cdot (X_{r_1}(t) - X_{r_2}(t)) \quad (3)$$

DE/current-to-best/1:

$$V_i(t) = X_i(t) + F \cdot (X_{best}(t) - X_i(t)) + F \cdot (X_{r_1}(t) - X_{r_2}(t)) \quad (4)$$

DE/rand/2:

$$V_i(t) = X_{r_1}(t) + F \cdot (X_{r_2}(t) - X_{r_3}(t)) + F \cdot (X_{r_4}(t) - X_{r_5}(t)) \quad (5)$$

DE/best/2:

$$V_i(t) = X_{best}(t) + F \cdot (X_{r_1}(t) - X_{r_2}(t)) + F \cdot (X_{r_3}(t) - X_{r_4}(t)) \quad (6)$$

where t denotes the number of evolution, F is the scaling factor, $r1$, $r2$, $r3$, $r4$, $r5$ are different integers within the range [1, NP] and $r_1 \neq r_2 \neq r_3 \neq r_4 \neq r_5 \neq i$, $X_{best}(t)$ denotes the best vector.

Step 3. Crossover.

DE usually uses binomial crossover to produce a trial vector $U_i(t)$, which is shown in Eq. (7).

$$u_{i,j}(t) = \begin{cases} v_{i,j}(t), & if\ rand \le CR\ or\ j = j_{rand} \\ x_{i,j}(t), & otherwise \end{cases} \quad (7)$$

where $CR \in [0,1]$ denotes the crossover probability and j_{rand} is a random selected integer in [1, D], which is ensured that not less than one element of the target vector $X_i(t)$ can inherit from the mutation vector $V_i(t)$.

Step 4. Selection.

DE uses the greedy strategy to select better individual according to the fitness of $U_i(t)$ and $X_i(t)$. The selection operation is shown in Eq. (8):

$$X_i(t+1) = \begin{cases} U_i(t), & if\ f(U_i(t)) \le f(X_i(t)) \\ X_i(t), & otherwise \end{cases} \quad (8)$$

3. Hybrid Estimation of Distribution Algorithm with Differential Evolution based on Self-adaptive Strategy (HEDADE-SA)

3.1 Alternative Probability Model

The evolution of EDA is achieved by sampling according to a probability model. In continuous EDA, Gaussian distribution and Cauchy distribution are the most commonly used probability models [7].

Gaussian distribution, named as a normal distribution, random variable X of univariate Gaussian distribution denotes as $X \sim N(\mu, \delta^2)$, and its density function is shown as follows:

$$f(x;\mu,\delta) = \frac{1}{\delta\sqrt{2\pi}} e^{-\frac{1}{2}(\frac{x-\mu}{\delta})^2} \quad (9)$$

where μ denotes the mean of X, δ denotes the standard deviation of X.

Cauchy distribution is similar to Gaussian distribution, and its probability density function is defined in Eq. (10):

$$f(x;x_0,\gamma) = \frac{1}{\pi\gamma\left[1 + (\frac{x - x_0}{\gamma})^2\right]} \quad (10)$$

where x_0 denotes the location value and γ denotes the scale value.

The probability model is essential because its employment directly impacts the quality of offspring. However, it is quite challenging to generate a superior offspring under only one probability distribution and many works adopted more than one probability distribution [7]. In literature [7], an alternative usage method of the Gaussian and Cauchy probability model is utilized. In other words, there is a certain probability to choose Gaussian or Cauchy distribution to generate offspring. By employing this method, the algorithms can potentially obtain a promising offspring successfully. The alternative probability model is also employed in our proposal.

A new sampled individual of EDA can be generated in Eq. (11).

$$E_i(t) = \begin{cases} Gaussian(\mu_i,\delta_i), & if\ rand_i(0,1) < 0.5 \\ Cauchy(\mu_i,\delta_i), & otherwise \end{cases} \quad (11)$$

3.2 Self-adaptive Strategy

The proposed HEDADE-SA utilizes EDA to improve global exploration capability, meanwhile uses DE/best/1 to increase local search capability. Inspired by literature [17][18], in HEDADE-SA, the selection probability $SP_i(t)$ is dynamically modified in each evolution and is generated as follows:

$$SP_i(t) = SP\_L + (SP\_U - SP\_L) * LRF1_i(t) \quad (12)$$

where SP_U is the maximum value of $SP_i(t)$, and SP_L is the minimum value of $SP_i(t)$, $LRF1$ is a linear reduction factor (LRF) as below [17]:

$$LRF1_i(t) = \frac{1-\exp(-\left|f(X_i(t)) - f(X_{gbest})\right|)}{1+\exp(-\left|f(X_i(t)) - f(X_{gbest})\right|)} \quad (13)$$

The value of $LRF1$ is the comparison between the fitness of current individual $f(X_i)$ and global best individual $f(X_{gbest})$.

Therefore, the mutation individual of the proposed algorithm is generated according to Eq. (14):

$$V_i(t) = \begin{cases} E_i(t), & if\ rand_i(0,1) < SP_i(t) \\ X_{best}(t) + F \cdot (X_{r_1}(t) - X_{r_2}(t)), & otherwise \end{cases} \quad (14)$$

In the early process of HEDADE-SA, the value of $SP_i(t)$ is bigger. The probability of $rand_i(0,1) < SP_i(t)$ is also very high, and the proposed algorithm can make more use of EDA for exploring. With the evolution process, the value of $SP_i(t)$ is smaller. Consequently, the probability of $rand_i(0,1) < SP_i(t)$ is also very low, and the proposed approach utilizes more DE for exploiting promising solutions.

In order to avoid stagnation in the evolution process, an indicator is set to observe the update of the population. Once the following condition is reached, the value of $SP_i(t)$ will automatically reset to 0.5. Furthermore, the search space of the algorithm can be expanded to get rid of the stagnant agitation [17].

$$SP_i(t+1) = \begin{cases} 0.5, & if\ \mathrm{r} \le L \\ SP_i(t), & otherwise \end{cases} \quad (15)$$

where r is the proportion value of updating individuals, L is a constant from 0 to 1.

In classical EDA, the number of the sample (NS) is fixed. Using a fixed value for sampling will inevitably lead to time-consuming, which is not conducive to searching for the best solution. Inspired by literature [19][20], in HEDADE-SA, NS is adaptively changed instead of taking fixed value:

$$NS = round(NS\_L + (NS\_U - NS\_L) * LRF2) \quad (16)$$

where NS_U is the maximum value of NS, NS_L is the minimum value of NS, round denotes a rounding function, and $LRF2$ is another LRF which is inspired by literature [21]:

$$LRF2 = \exp(1 - \frac{M\max FEs}{M\max FEs - FEs + 1}) \quad (17)$$

where FEs is the number of evaluations, and $MaxFEs$ is the maximum value of evaluations.

From Eq. (17), it is easy to see that the value of $LRF2$ decreases from 1 to 0. Furthermore, NS decreases from NS_U to NS_L according to Eq. (16).

3.3 The implementation of the proposed HEDADE-SA algorithm

With the above approaches, the framework of HEDADE-A can be described in Table 1.

(Table 1) The framework of HEDADE-SA

Algorithm: HEDADE-SA
Input: *NP*: the number of population; *SP*: the self-adaptive selection probability; *NS*: the number of the sample; *FEs*: the number of evaluations; *MaxFEs*: the maximum number of evaluations; *LRF*1: the number of linear reduction factor; *LRF*2: the number of linear reduction factor. **Output**: The global optimum.
Step 1: Randomly initialize population *p(t)*. **Step 2:** Calculating and evaluating the fitness values of each individual. **Step 3:** Calculating *LRF*1 according to Eq. (13). **Step 4:** According to Eq. (12) to calculate *SP* of each individual. **Step 5: While** *FEs* 〈 *MaxFEs* **Step 5.1:** Sorting individuals in ascending order by the fitness values. **Step 5.2:** Calculating *LRF*2 according to Eq. (17). **Step 5.3:** Selecting *NS* best elite individuals for sampling according to Eq. (16). **Step 5.4: For** *i* = 1 **to** *NP* **Step 5.5:** **If** *Rand*(0,1) 〈 *SP* **Step 5.6:** Generating a mutant vector of EDA according to Eq. (11). **Step 5.7:** **Else** **Step 5.8:** Generating a mutant vector of DE according to Eq. (3). **Step 5.9:** **End if** **Step 5.10:** Generating a trial vector according to Eq. (7). **Step 5.11:** According to Eq. (8) to generate an offspring using the greedy Strategy. **Step 5.12:** *FEs* = *FEs* + 1. **Step 5.13:** Calculating *LRF*1 according to Eq. (13). **Step 5.14:** According to Eq. (12) to calculate *SP* of each individual. **Step 5.15: End For** **Step 5.16:** Updating *SP* of each individual according to Eq. (15). **Step 6: End While**

(Table 2) The benchmark functions of CEC2005

Type	Function	Function Name	Range	$f_i(x^*)=f\_bias_i$
Unimodal functions	f_1	Shifted Sphere Function	$[-100,100]^D$	-450
	f_2	Shifted Schwefel's Problem 1.2	$[-100,100]^D$	-450
	f_3	Shifted Rotated High Conditioned Elliptic Function	$[-100,100]^D$	-450
	f_4	Shifted Schwefel's Problem 1.2 with Noise in Fitness	$[-100,100]^D$	-450
	f_5	Schwefel's Problem 2.6 with Global Optimum on Bounds	$[-100,100]^D$	-310
Multimodal functions	f_6	Shifted Rosenbrock's Function	$[-100,100]^D$	390
	f_7	Shifted Rotated Griewank's Function without Bound	$[0,600]^D$	-180
	f_8	Shifted Rotated Ackley's Function with Global Optimum on Bounds	$[-32,32]^D$	-140
	f_9	Shifted Rastrigin's Function	$[-5,5]^D$	-330
	f_{10}	Shifted Rotated Rastrigin's Function	$[-5,5]^D$	-330
	f_{11}	Shifted Rotated Weierstrass Function	$[-0.5,0.5]^D$	90
	f_{12}	Schwefel's Problem 2.13	$[-\pi,\pi]^D$	-460
Expanded functions	f_{13}	Expanded Extended Griewank's plus Rosenbrock's Function (F8F2)	$[-3,1]^D$	-130
	f_{14}	Shifted Rotated Expanded Scaffer's F6	$[-100,100]^D$	-300
Hybrid composition functions	f_{15}	Hybrid Composition Function	$[-5,5]^D$	120
	f_{16}	Rotated Hybrid Composition Function	$[-5,5]^D$	120
	f_{17}	Rotated Hybrid Composition Function with Noise in Fitness	$[-5,5]^D$	120
	f_{18}	Rotated Hybrid Composition Function	$[-5,5]^D$	10
	f_{19}	Rotated Hybrid Composition Function with a Narrow Basin for the Global Optimum	$[-5,5]^D$	10
	f_{20}	Rotated Hybrid Composition Function with the Global Optimum on the Bounds	$[-5,5]^D$	10
	f_{21}	Rotated Hybrid Composition Function	$[-5,5]^D$	360
	f_{22}	Rotated Hybrid Composition Function with High Condition Number Matrix	$[-5,5]^D$	360
	f_{23}	Non-Continuous Rotated Hybrid Composition Function	$[-5,5]^D$	360
	f_{24}	Rotated Hybrid Composition Function	$[-5,5]^D$	260
	f_{25}	Rotated Hybrid Composition Function without Bounds	$[2,5]^D$	260

The time complexity of the proposed HEDADE-SA algorithm is mainly concentrated in Step 5.1. The main function of Step 5.1 is to sort *NP* individuals in ascending, and its time complexity is O(*NP*×*NP*). Consequently, the time complexity of HEDADE-SA is O(*NP*×*NP*×*MaxFEs*).

4. Experiments and Analysis

4.1 Benchmark functions and experimental setting

The algorithm is conducted on a set of twenty-five benchmark functions of CEC2005 [22]. Table 2 depicts the function types, the function names, the initialization ranges, and the bias values of these functions.

For a fair comparison, the following parameters are the same as follows: problem dimension *D* is equal to 30; *MaxFEs* is set to 3×10^5; each is for 30 independent runs per function. The experiments were conducted on a pc of 64 bit Core i7-4770 3.40 GHz CPUs and 8 GB RAM. Moreover, all algorithms were implemented in Eclipse SDK 4.3.2 and executed in Ubuntu 16.04 (64bit).

4.2 Experimental study and discussion

4.2.1 Parameter study in F and *CR*

In HEDADE-SA, the choice of the scaling factor *F* and the crossover probability *CR* is quite sensitive. Therefore, in this section, we compared the proposed HEDADE-SA algorithm with different values of *F* and *CR*, namely HEDADE-SA with (*F* = 0.1, *CR* = 0.1), HEDADE-SA with (*F* = 0.1, *CR* = 0.9), HEDADE-SA with (*F* = 0.5, *CR* = 0.1),

(Table 3) Results of HEDADE-SA with different values of *F* and *CR*

Function	*F*=0.1/*CR*=0.1	*F*=0.1/*CR*=0.9	*F*=0.5/*CR*=0.1	*F*=0.5/*CR*=0.9
f_1	5.06E-09	8.91E-29	1.05E-05	2.11E-28
f_2	4.37E+03	1.67E-06	8.93E+03	3.38E-01
f_3	2.13E+07	1.11E+06	3.57E+07	1.09E+06
f_4	9.59E+03	6.44E+02	1.41E+04	1.53E+01
f_5	3.23E+03	3.70E+03	3.92E+03	1.98E+03
f_6	8.34E+01	3.11E-01	5.86E+01	1.51E+01
f_7	4.70E+03	4.70E+03	4.70E+03	4.70E+03
f_8	2.09E+01	2.09E+01	2.10E+01	2.10E+01
f_9	1.49E-01	1.28E+01	3.36E+01	1.33E+01
f_{10}	1.41E+02	4.08E+01	1.76E+02	1.80E+02
f_{11}	3.38E+01	1.42E+01	3.43E+01	3.86E+01
f_{12}	2.57E+04	5.27E+03	1.05E+05	3.78E+03
f_{13}	5.56E+00	2.35E+00	6.52E+00	1.28E+01
f_{14}	1.32E+01	1.17E+01	1.32E+01	1.31E+01
f_{15}	4.03E+02	3.08E+02	4.07E+02	3.17E+02
f_{16}	1.64E+02	9.15E+01	2.07E+02	2.00E+02
f_{17}	2.01E+02	8.26E+01	2.37E+02	2.21E+02
f_{18}	8.71E+02	8.47E+02	9.08E+02	8.48E+02
f_{19}	8.71E+02	8.61E+02	8.97E+02	8.63E+02
f_{20}	8.91E+02	8.52E+02	9.00E+02	8.66E+02
f_{21}	5.00E+02	5.00E+02	5.00E+02	5.00E+02
f_{22}	9.60E+02	9.61E+02	9.79E+02	9.31E+02
f_{23}	5.34E+02	5.34E+02	5.34E+02	5.34E+02
f_{24}	2.00E+02	2.00E+02	2.00E+02	2.00E+02
f_{25}	1.66E+03	1.65E+03	1.66E+03	1.65E+03
Ranking	2.58	**1.62**	3.54	2.26

and HEDADE-SA with (*F* = 0.5, *CR* = 0.9). Other parameters of HEDADE-SA are same and shown as follows: *NP* = 1000, *NS_U* = 10%**NP* and *NS_L* = 3, *SP_L* = 0.2 and *SP_U* = 0.9, and *L* = 0.3. Table 3 shows the experiment results for Friedman's test. The value of average rankings is smaller, the performance of the algorithm is better. It can be seen that HEDADE-SA with (*F* = 0.1, *CR* = 0.9) performs the best performance according to Table 3. In the following experiments, HEDADE-SA adopts the above parameters.

4.2.2 Comparison of HEDADE-SA and HEDADE-SA variants

To demonstrate the effectiveness of the self-adaptive strategy, HEDADE-SA is compared with four HEDADE-SA variants, namely HEDADE-SA with DE/rand/1, DE/rand/2, DE/best/1, and DE/best/2 mutation strategies, denoted by HEDADE-SA1, HEDADE-SA2, HEDADE-SA3, and HEDADE-SA4.

For HEDADE-SA1, HEDADE-SA2, HEDADE-SA3, and HEDADE-SA4, the parameter settings are the same as follows: *NP* is set to 1000, $SP_i(t)$ is set to 0.5, and *NS* is set to 10%**NP*. In Table 4, Wilcoxon's rank sum test is implemented to compare HEDADE-SA with other algorithms. Moreover, "-", "+", and "≈" represents that the performance of HEDADE-SA is better than, worse than, and similar to that of others, respectively. From Table 4, the performance of HEDADE-SA is superior to that of HEDADE-SA1, HEDADE-SA2, HEDADE-SA3, and HEDADE-SA4 on 13, 14, 12, and 11 problems, respectively, and that is similar to HEDADE-SA1, HEDADE-SA2, HEDADE-SA3, and HEDADE-SA4 on 9, 9, 10, and 11 problems, respectively. The results for Friedman's test are shown in Table 5, which indicates that the self-adaptive strategy is effective.

(Table 4) Results of HEDADE-SA and HEDADE-SA variants

F	HEDADE-SA1 mean±std	HEDADE-SA2 mean±std	HEDADE-SA3 mean±std	HEDADE-SA4 mean±std	HEDADE-SA mean±std
f_1	5.85E-15±1.76E-15 −	3.33E-14±1.02E-14 −	2.49E-26±4.11E-27 −	2.49E-26±4.15E-27 −	8.91E-29±1.01E-28
f_2	2.03E+03±2.20E+02 −	2.00E+03±1.68E+02 −	6.46E-03±4.32E-03 −	1.44E-03±1.19E-03 −	1.67E-06±6.35E-06
f_3	7.30E+06±9.62E+05 −	8.16E+06±9.55E+05 −	1.10E+06±3.39E+05 ≈	1.05E+06±3.52E+05 ≈	1.11E+06±2.83E+05
f_4	4.18E+03±4.59E+02 −	4.23E+03±4.66E+02 −	4.20E+02±3.27E+02 +	4.72E+02±3.53E+02 +	6.44E+02±2.47E+02
f_5	3.20E+03±1.49E+02 +	3.11E+03±1.08E+02 +	4.44E+03±7.51E+02 −	4.27E+03±6.90E+02 −	3.70E+03±3.30E+02
f_6	8.22E+01±2.23E+01 −	6.70E+01±3.08E+01 −	5.05E+01±3.87E+01 −	3.41E+01±2.98E+01 −	3.11E-01±9.98E-01
f_7	4.70E+03±0.00E+00 ≈	4.70E+03±0.00E+00 ≈	4.70E+03±0.00E+00 ≈	4.70E+03±0.00E+00 ≈	4.70E+03±0.00E+00
f_8	2.09E+01±6.29E-02 ≈	2.09E+01±6.05E-02 ≈	2.09E+01±6.53E-02 ≈	2.09E+01±4.99E-02 ≈	2.09E+01±6.90E-02
f_9	1.17E+02±7.43E+00 −	1.32E+02±7.44E+00 −	1.07E+01±2.34E+00 +	9.57E+00±3.49E+00 +	1.28E+01±3.41E+00
f_{10}	1.71E+02±1.06E+01 −	1.74E+02±7.77E+00 −	6.36E+01±3.92E+01 −	7.16E+01±5.58E+01 −	4.08E+01±9.99E+00
f_{11}	3.95E+01±9.30E-01 −	3.93E+01±9.33E-01 −	1.36E+01±2.23E+00 ≈	1.29E+01±2.29E+00 +	1.42E+01±2.57E+00
f_{12}	1.09E+04±5.44E+03 −	8.87E+03±2.06E+03 −	4.94E+03±4.40E+03 ≈	5.36E+03±4.78E+03 ≈	5.27E+03±3.01E+03
f_{13}	1.33E+01±9.65E-01 −	1.41E+01±8.25E-01 −	1.87E+00±4.99E-01 +	4.64E+00±3.09E+00 −	2.35E+00±4.74E-01
f_{14}	1.32E+01±1.27E-01 −	1.32E+01±1.88E-01 −	1.22E+01±4.43E-01 −	1.23E+01±4.58E-01 −	1.17E+01±7.04E-01
f_{15}	3.00E+02±0.00E+00 ≈	3.03E+02±1.80E+01 ≈	3.94E+02±8.93E+01 −	3.05E+02±1.48E+02 ≈	3.08E+02±6.43E+01
f_{16}	1.91E+02±5.87E+00 −	1.87E+02±8.86E+00 −	1.97E+02±1.60E+02 −	1.35E+02±1.09E+02 −	9.15E+01±1.00E+02
f_{17}	2.08E+02±1.20E+01 −	2.10E+02±1.15E+01 −	1.81E+02±1.45E+02 −	1.66E+02±1.44E+02 −	8.26E+01±6.31E+01
f_{18}	8.43E+02±6.04E+01 ≈	8.59E+02±6.32E+01 ≈	8.85E+02±6.08E+01 −	8.79E+02±6.04E+01 ≈	8.47E+02±6.13E+01
f_{19}	8.71E+02±6.18E+01 ≈	8.67E+02±6.25E+01 ≈	8.96E+02±5.30E+01 ≈	8.94E+02±5.21E+01 ≈	8.61E+02±6.53E+01
f_{20}	8.58E+02±6.23E+01 ≈	8.79E+02±6.03E+01 ≈	8.90E+02±5.92E+01 ≈	8.79E+02±6.03E+01 ≈	8.52E+02±6.38E+01
f_{21}	5.00E+02±0.00E+00 ≈	5.00E+02±0.00E+00 ≈	5.87E+02±2.22E+02 −	5.54E+02±1.69E+02 ≈	5.00E+02±0.00E+00
f_{22}	9.52E+02±6.57E+00 +	9.50E+02±5.21E+00 +	9.76E+02±1.53E+01 −	9.78E+02±2.15E+01 −	9.61E+02±1.87E+01
f_{23}	5.34E+02±0.00E+00 ≈	5.34E+02±0.00E+00 ≈	5.97E+02±1.90E+02 ≈	5.98E+02±1.91E+02 −	5.34E+02±0.00E+00
f_{24}	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00
f_{25}	1.65E+03±1.80E+00 +	1.66E+03±0.00E+00 −	1.64E+03±8.00E+00 ≈	1.65E+03±9.07E+00 ≈	1.65E+03±6.63E+00
−	13	14	12	11	
+	3	2	3	3	
≈	9	9	10	11	

(Table 5) The average rankings of HEDADE-SA with HEDADE-SA variants

Algorithm	HEDADE-SA1	HEDADE-SA2	HEDADE-SA3	HEDADE-SA4	HEDADE-SA
Ranking	3.30	3.48	3.12	2.90	**2.20**

4.2.3 Comparison of HEDADE-SA and other four algorithms

In this experiment, HEDADE-SA is compared with the other four algorithms, such as UMDAc [23], DE [11], DE/EDA [12] and JDE [24]. UMDAc is a classical algorithm of continuous EDA. DE is a representative algorithm of EA. DE/EDA is a combination algorithm of DE and EDA. JDE is a famous algorithm of DE. For UMDAc, *NP* is set to *NP* = 1000, and the Sampling Rate *(SR)* is set to *SR* = 30%. For DE, *NP* = 100, *F* is set to *F* = 0.5, *CR* is set to *CR* = 0.9 and DE/rand/1 is selected. For DE/EDA, *NP* = 1000, *F* = 0.5, *CR* = 0.9, *SR* = 50%. For JDE, the settings of parameters follow its original literature.

Experiment results are listed in Table 6. HEDADE-SA significantly outperforms UMDAc except for *f*8 because UMDAc converges slowly and the solutions are also inferior. In addition, compared with HEDADE-SA, DE is worse on 11 test problems, JDE is worse on 13, and DE/EDA is worse on 18 tests. Furthermore, Table 7 displays the results of the Friedman test, and a graphical representation of the Friedman test is shown in Figure 2. The superior performance of HEDADE-SA is evidently according to Table 7 and Figure

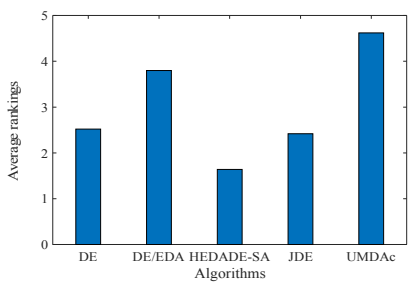
(Table 6) Experimental results of HEDADE-SA and others for benchmark functions of CEC2005

F	UMDAc mean±std	DE mean±std	JDE mean±std	DE/EDA mean±std	HEDADE-SA mean±std
f_1	1.99E+04±9.35E+02 −	1.52E-29±5.08E-29 +	0.00E+00±0.00E+00 +	9.69E-03±2.14E-03 −	8.91E-29±1.01E-28
f_2	2.94E+04±8.54E+02 −	3.13E-05±2.53E-05 −	7.04E+02±2.39E+02 −	2.65E+03±4.21E+02 −	1.67E-06±6.35E-06
f_3	2.93E+08±2.88E+07 −	3.96E+05±2.21E+05 +	9.43E+06±3.16E+06 −	2.14E+07±3.58E+06 −	1.11E+06±2.83E+05
f_4	3.16E+04±9.90E+02 −	1.63E-02±1.75E-02 +	2.52E+03±8.49E+02 −	5.29E+03±9.03E+02 −	6.44E+02±2.47E+02
f_5	2.49E+04±5.42E+02 −	4.61E+01±5.30E+01 +	1.84E+03±2.77E+02 +	3.79E+03±3.11E+02 ≈	3.70E+03±3.30E+02
f_6	3.65E+09±3.01E+08 −	6.33E-02±1.13E-01 +	1.65E+01±1.63E+01 −	5.36E+01±8.92E+00 −	3.11E-01±9.98E-01
f_7	9.78E+03±9.31E+01 −	4.70E+03±0.00E+00 ≈	4.70E+03±0.00E+00 ≈	5.02E+03±2.04E+01 −	4.70E+03±0.00E+00
f_8	2.10E+01±4.96E-02 ≈	2.10E+01±5.62E-02 ≈	2.09E+01±7.27E-02 ≈	2.09E+01±5.59E-02 ≈	2.09E+01±6.90E-02
f_9	1.49E+02±1.08E+01 −	1.38E+02±2.65E+01 −	5.08E-01±1.55E+00 +	1.96E+02±7.12E+00 −	1.28E+01±3.41E+00
f_{10}	2.61E+02±1.40E+01 −	1.80E+02±1.26E+01 −	1.71E+02±1.07E+01 −	2.00E+02±7.76E+00 −	4.08E+01±9.99E+00
f_{11}	2.71E+01±1.76E+00 −	3.94E+01±1.04E+00 −	3.66E+01±1.34E+00 −	3.90E+01±1.39E+00 −	1.42E+01±2.57E+00
f_{12}	5.63E+05±4.70E+04 −	1.50E+03±1.72E+03 +	7.79E+04±3.04E+04 −	2.65E+05±4.88E+04 −	5.27E+03±3.01E+03
f_{13}	8.97E+00±1.27E+00 −	1.50E+01±1.00E+00 −	7.08E+00±5.63E-01 −	1.70E+01±8.37E-01 −	2.35E+00±4.74E-01
f_{14}	1.24E+01±2.04E-01 −	1.33E+01±1.29E-01 −	1.33E+01±2.00E-01 −	1.33E+01±1.28E-01 −	1.17E+01±7.04E-01
f_{15}	8.75E+02±1.84E+01 −	4.03E+02±1.80E+01 −	3.70E+02±4.58E+01 −	4.03E+02±1.80E+01 −	3.08E+02±6.43E+01
f_{16}	8.34E+02±4.91E+01 −	2.05E+02±7.89E+00 −	1.91E+02±9.34E+00 −	2.24E+02±5.89E+00 −	9.15E+01±1.00E+02
f_{17}	9.08E+02±6.08E+01 −	2.28E+02±9.22E+00 −	2.17E+02±1.27E+01 −	2.51E+02±1.15E+01 −	8.26E+01±6.31E+01
f_{18}	1.20E+03±6.37E+00 −	9.05E+02±1.14E+00 ≈	9.07E+02±1.41E+00 ≈	9.18E+02±1.05E+00 −	8.47E+02±6.13E+01
f_{19}	1.20E+03±7.52E+00 −	9.01E+02±1.88E+01 ≈	9.07E+02±1.48E+00 ≈	9.19E+02±9.12E-01 ≈	8.61E+02±6.53E+01
f_{20}	1.20E+03±9.55E+00 −	9.01E+02±1.88E+01 ≈	9.07E+02±1.22E+00 ≈	9.18E+02±1.09E+00 ≈	8.52E+02±6.38E+01
f_{21}	1.26E+03±4.42E+00 −	5.10E+02±5.39E+01 ≈	5.00E+02±0.00E+00 ≈	5.00E+02±0.00E+00 ≈	5.00E+02±0.00E+00
f_{22}	1.24E+03±3.25E+01 −	9.09E+02±9.38E+00 +	9.27E+02±4.97E+00 +	9.58E+02±6.40E+00 ≈	9.61E+02±1.87E+01
f_{23}	1.26E+03±6.51E+00 −	5.70E+02±1.64E+01 −	5.34E+02±0.00E+00 ≈	5.83E+02±1.19E+01 −	5.34E+02±0.00E+00
f_{24}	1.30E+03±5.73E+00 −	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00 ≈	2.00E+02±0.00E+00
f_{25}	1.86E+03±5.82E+00 −	1.66E+03±4.82E+00 −	1.66E+03±4.23E+00 −	1.68E+03±4.82E+00 −	1.65E+03±6.63E+00
−	24	11	13	18	
+	0	7	4	0	
≈	1	7	8	7	

2. Figure 3 indicates the convergence process of these algorithms on representative test functions.

(Table 7) The average rankings of HEDADE-SA and other four algorithms

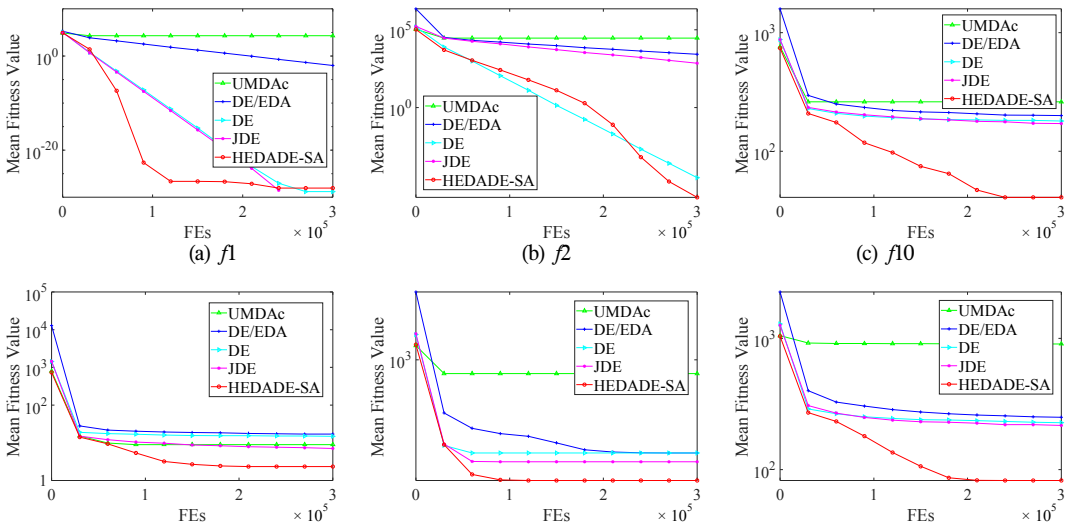
Algorithm	UMDAc	DE	JDE	DE/EDA	HEDADE-SA
Ranking	4.62	2.52	2.42	3.8	**1.64**



(Figure 2) The graphical representation of the Friedman test of the algorithms

5. Conclusion

In this research, a hybrid estimation of distribution algorithm with differential evolution based on self-adaptive strategy (namely HEDADE-SA) is proposed. An alternative probability model for sampling is utilized for the proposed algorithm. Moreover, a self-adaptive strategy is adopted to make full use of EDA and DE operators. Through these methods, the population diversity and the convergence speed of the algorithm are improved. Experiment results show that HEDADE-SA exhibits superior performance than the other comparison algorithms. In the future, the proposed HEDADE-SA algorithm can be used to deal with large-scale global problems.



(Figure 3) Convergence curve of the algorithms on representative test functions

References

[1] S.U. Khan, S. Yang, L. Wang, and L. Liu, "A Modified Particle Swarm Optimization Algorithm for Global Optimizations of Inverse Problems," IEEE Transactions on Magnetics, Vol. 52, No. 3, pp. 1-4, 2016.
https://doi.org/10.1109/TMAG.2015.2487678

[2] W. Deng, H. Zhao, L. Zou, G. Li, X. Yang et al., "A novel collaborative optimization algorithm in solving complex optimization problems," Soft Computing, Vol. 21, No. 15, pp.4387-4398,2017.
https://doi.org/10.1007/s00500-016-2071-8

[3] H. Peng, C. Deng, and Z. Wu, "Best neighbor-guided artificial bee colony algorithm for continuous optimization problems," Soft Computing, vol. 23, No.18, pp. 8723-8740, 2019.
https://doi.org/10.1007/s00500-018-3473-6

[4] H. Mühlenbein and G. Paass, "From recombination of genes to the estimation of distributions I. Binary parameters," in Proc. Of international conference on parallel problem solving from nature, pp. 178-187, Springer, Berlin, Heidelberg, 2005.
https://doi.org/10.1007/3-540-61723-X_982

[5] C. Liu, H. Chen, R. Xu, and Y. Wang, "Minimizing the resource consumption of heterogeneous batch-processing machines using a copula-based estimation of distribution algorithm," Applied soft computing, Vol. 73, pp. 283-305, 2018.
https://doi.org/10.1016/j.asoc.2018.08.036

[6] W. Dong, Y. Wang, and M. Zhou, "A latent space-based estimation of distribution algorithm for large-scale global optimization," Soft Computing, Vol.23, No.13, pp. 4593-4615, 2019.
https://doi.org/10.1007/s00500-018-3390-8

[7] Q. Yang, W.N. Chen, Y. Li, C.P. Chen, X.M. Xu, and J. Zhang, "Multimodal Estimation of Distribution Algorithms," IEEE Transactions on cybernetics, Vol. 47, No. 3, pp. 636-650, 2017.
https://doi.org/10.1109/TCYB.2016.2523000

[8] S. Gao, and C.W. de Silva, "Estimation distribution algorithms on constrained optimization problems," Applied Mathematics and Computation, Vol. 339, pp. 323-345, 2018.
https://doi.org/10.1016/j.amc.2018.07.037

[9] M. Hauschild, and M. Pelikan, "An introduction and survey of estimation of distribution algorithms," Swarm and evolutionary computation, Vol. 1, No. 3, pp. 111-128,

2011.
https:/doi.org/10.1016/j.swevo.2011.08.003

[10] R. Storn and K. Price, "Differential evolution: a simple and efficient heuristic for global optimization over continuous spaces," Journal of Global Optimization, Vol. 11, No. 4, pp. 341-359, 1997.
https://doi.org/10.1023/A:1008202821328

[11] S. Das, S.S. Mullick, and P.N. Suganthan, "Recent advances in differential evolution – an updated survey," Swarm and Evolutionary Computation, Vol. 27, pp. 1-30, 2016.
https://doi.org/10.1016/j.swevo.2016.01.004

[12] J.Y. Sun, Q. Zhang, and E.P. Tsang, "DE/EDA: A new evolutionary algorithm for global optimization," Information sciences, Vol. 169, No. 3-4, pp. 249-262, 2005.
https://doi.org/10.1016/j.ins.2004.06.009

[13] W. Shao, L. Shang, L. Ma, Z. Shao, and X. Ying, "Hybrid differential evolution/estimation of distribution algorithm based on adaptive incremental learning," Journal of Computational Information Systems, Vol. 10, No. 12, pp. 5355-5364, 2014.
https://www.semanticscholar.org/paper/Hybrid-differential-evolution%2Festimation-of-based-Shao-Shang/8c12307eb372622ee30d34e7af3abb2bb205e4bf

[14] B. Dong, A. Zhou, and G. Zhang, "A hybrid estimation of distribution algorithm with differential evolution for global optimization," In Proc. Of 2016 IEEE Symposium Series on Computational Intelligence, pp. 1-7, 2016.
https://doi.org/10.1109/SSCI.2016.7850201

[15] H. Fang, A. Zhou, and H. Zhang, "Information fusion in offspring generation: A case study in DE and EDA," Swarm and evolutionary computation, Vol. 42, pp. 99-108, 2018.
https://doi.org/10.1016/j.swevo.2018.02.014

[16] S.L. Wang, T.F. Ng, and F. Morsidi, "Self-adaptive Ensemble Based Differential Evolution," International Journal of Machine Learning and Computing, Vol. 8, No. 3, pp. 286-293, 2018.
https://doi.org/10.18178/ijmlc.2018.8.3.701

[17] H. Liu, J. Han, L. Yuan, and B. Yu, "Self-adaptive bare-bones differential evolution based on bi-mutation strategy," Journal on Communications, Vol.38, No.8, pp.201-212, 2017.
https://doi.org/10.11959/j.issn.1000-436x.2017051

[18] S. Wang, Y. Li, and H. Yang, "Self-adaptive mutation differential evolution algorithm based on particle swarm optimization," Applied Soft Computing, Vol. 81, pp. 1-22, 2019. https://doi.org/10.1016/j.asoc.2019.105496

[19] M.Z. Ali, N.H. Awad, P.N. Suganthan, and R.G. Reynolds, "An adaptive multipopulation differential evolution with dynamic population reduction," IEEE transactions on cybernetics, Vol. 47, No. 9, pp. 2768-2779, 2017.
https://doi.org/10.1109/TCYB.2016.2617301

[20] L.B. Deng, L.L. Zhang, N. Fu, H.L. Sun, and L.Y. Qiao, "ERG-DE: An Elites Regeneration Framework for Differential Evolution," Information Sciences, Vol.539, pp. 81-103, 2020.
https://doi.org/10.1016/j.ins.2020.05.108

[21] J. Liu, H. Peng, Z. Wu, J. Chen, and C. Deng, "Multi-strategy brain storm optimization algorithm with dynamic parameters adjustment," Applied Intelligence, Vol. 50, No. 4, pp. 1289-1315, 2020.
https://doi.org/10.1007/s10489-019-01600-7

[22] P. N. Suganthan, N. Hansen, J.J. Liang, K. Deb, Y. Chen et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," Technical Report, Singapore: Nanyang Technological University, 2005.
https://www3.ntu.edu.sg/home/epnsugan/index_files/CEC-05/CEC05.htm

[23] P. Larrañaga, R. Etxeberria, J.A. Lozano, and J.M. Peña, "Optimization in continuous domains by learning and simulation of Gaussian networks," In Proc. Of the 2000 Genetic and Evolutionary Computation Conference Workshop Program, pp. 201-204, 2000.
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.31.3105

[24] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," IEEE transactions on evolutionary computation, Vol.10, No. 6, pp. 646-657, 2006.
https://doi.org/10.1109/TEVC.2006.872133

◐ 저 자 소 개 ◑

Debin Fan

2003 B.S. in Information and Computing Science, Jiangxi University of Science and Technology, Ganzhou, China
2010 M.S. in Computer Technology, Huazhong University of Science and Technology, Wuhan, China
2021 Ph.D. in Information and Communication Engineering, Kunsan National University, Gunsan, Korea
2013.12~Present, Associate Professor at School of Information Science and Technology, Jiujiang University, Jiujiang, China
Research Interests: evolutionary computation, cloud computing
E-mail : dbfan@kunsan.ac.kr

Jaewan Lee

1984 B.S. in Computer Engineering, Chung-Ang University, Seoul, Korea
1987 M.S. in Computer Engineering, Chung-Ang University, Seoul, Korea
1992 Ph.D. in Computer Engineering, Chung-Ang University, Seoul, Korea
1995.03~Present, Professor at the Department of Information and Communication Engineering, Kunsan National University
Research Interests: distributed systems, cloud computing, data mining, and computer networks
E-mail : jwlee@kunsan.ac.kr