Adaptive Recombination Operator Selection in Push and Pull Search for Solving Constrained Single-Objective Optimization Problems

Zhun Fan(✉), Zhaojun Wang, Yi Fang, Wenji Li, Yutong Yuan, and Xinchao Bian

Department of Electronic Engineering, Shantou University, Shantou 515063, Guangdong, China
zfan@stu.edu.cn

Abstract. This paper proposes an adaptive method to select recombination operators, including differential evolution (DE) operators and polynomial operators. Moreover, a push and pull search (PPS) method is used to handle constrained single-objective optimization problems (CSOPs). The PPS has two search stages—the push stage and the pull stage. In the push stage, a CSOP is optimized without considering constraints. In the pull stage, the CSOP is optimized with an improved epsilon constraint-handling method. In this paper, twenty-eight CSOPs are used to test the performance of the proposed adaptive GA with the PPS method (AGA-PPS). AGA-PPS is compared with three other differential evolution algorithms, including LSHADE44+IDE, LSHADE44 and UDE. The experimental results indicate that the proposed AGA-PPS is significantly better than other compared algorithms on the twenty-eight CSOPsq.

Keywords: Adaptive recombination operator selection
Constrained single-objective optimization
Constraint-handling technique · Push-pull search

1 Introduction

Many real-parameter single-objective optimization problems have constraints [1,2]. In general, a constrained single-objective optimization problem can be defined as follows:

$$\begin{aligned} \text{minimize}\ & f(\mathbf{x}),\ \mathbf{x} = (x_1, \ldots, x_D) \in S \\ \text{subject to}\quad & g_i(x) \geq 0, i = 1, \ldots, q \\ & h_j(x) = 0, j = 1, \ldots, p \end{aligned} \tag{1}$$

where $f(\mathbf{x})$ is the objective function. $\mathbf{x}$ is a decision vector. x_i is the i-th variable of $\mathbf{x}$. $S = \prod_{i=1}^{D} [L_i,\ U_i]$ is the decision space, where L_i and the U_i are the lower and the upper bounds of x_i. $g_i(\mathbf{x})$ denotes the i-th inequality constraint, and $h_j(\mathbf{x})$ denotes j-th equality constraint.

In order to evaluate the constraint violation of a solution $\mathbf{x}$, the overall constraint violation method is adopted, which summaries all constraints into a scalar value $\phi(\mathbf{x})$, as follows:

$$\phi(\mathbf{x}) = \sum_{i=1}^{q} \max(g_i(\mathbf{x}), 0) + \sum_{j=1}^{p} \max(|h_j(\mathbf{x})| - \sigma, 0) \tag{2}$$

In this paper, σ is set to 0.0001 as suggested in [3]. If $\phi(\mathbf{x}) = 0$, the $\mathbf{x}$ is a feasible solution. Otherwise it is an infeasible solution.

At present, the Constraint-handling Technique based on evolutionary algorithm can be divided into the following two categories: Penalty Function Based Methods and Multi-objective Based Methods. For example: Le Riche et al. [4] proposed a segregated genetic algorithm (SGGA), It contains two values of the penalty parameter, SGGA permits to balance the inuence of the two penalty parameters. Huang et al. [5] proposed a novel method co-evolutionary differential evolution (CDE), two kinds of populations are used in CDE, In the population denotes a set of penalty factors, and in another kind of populations denotes a decision solution. Surry and Radcliffe [6] proposed COMOGA, that is, the single-objective constrained optimization problem is considered as a constraint satisfaction problem or a single-object unconstrained optimization problem.

As a representative heuristic algorithm, generic algorithm (GA) [16] can be used to optimize real-parameter SOPs. A typical example of GA is differential evolution algorithm (DE). There are many different variants of DE. For example, FADE [7], jDE [8], JADE [9], CoDE [10], SHADE [11], L-SHADE [12], L-SHADE44 [13] and so on. FADE [7] sets the DE parameters by using the fuzzy logic control. jDE [8] updates the DE parameters with different probabilities. JADE [9] proposes a current-to-$p$best/1 with an external archive, a greedy mutation operator. It adaptively updates its parameters in each generations. CoDE [10] applies several groups of suitable parameter settings to the DE. SHADE [11] proposes an adaptive technique of parameter settings by using successful historic memories. L-SHADE [12] is an improved version of SHADE, which reduces the population size linearly during the evolutionary process. As an variant of L-SHADE, L-SHADE44 [13] proposes a strategy to select four different kinds of DE operators adaptively.

However, in some circumstance, the diversity of a population may be lost by only adopting DE operators. A polynomial operator proposed in [17] can be used to enhance the diversity of a population.

To solve CSOPs, an efficient constraint-handling method should be applied [14,18]. ϵ constrained method [15] is a representative constraint-handling method, which can be concluded as the following three rules:

1. When the constraint violations of two individuals are both lower than or equal to ϵ, the individual with the lower objective value is better than the other.

2. When the constraint violations of two individuals are the same, the individual with a lower objective value is better than the other.
3. When at least one constraint violation of two individuals is larger than ϵ, the individual with a lower constraint violation is better than the other.

In this paper, we propose a GA with an adaptive recombination operator selection (AGA) method and a PPS constraint-handling method, namely AGA-PPS.

The rest of this paper is organized as follows. Section 2 introduces some related work. Section 3 introduces the proposed method AGA-PPS. Section 4 shows the experimental results of AGA-PPS and other three DE algorithms (LSHADE44+IDE, LSHADE44 and UDE) on 28 test instances. Section 5 gives the conclusion.

2 Adaptive Recombination Operator Selection

2.1 Successful History Based DE Parameter Settings

In SHADE [11], a method of adaption parameter setting is proposed. For each individual $\mathbf{x}_i,\ i = 1, \ldots, N$, its matching parameters F_i and CR_i are generated according to successful historic memories M_F and M_{CR} with H cells, respectively. A pointer k is used to record the memories. k is initially set to 0. At the beginning of each generation, two sets S_F and S_{CR} are both set as $\emptyset$, which stores the successful parameter pair $\{F_i,\ CR_i\}$ for each individual $\mathbf{x}_i,\ i = 1, \ldots, N$. At the end of each generation, if S_F is not empty, the k will be increased by 1. If the value of k is larger than H, k will be reset as 1. The adaption factors m_F and m_{CR} are calculated by Eq. (3)–(8), which are stored into the k-th cell of memories M_F and M_{CR}, respectively.

$$m_F = mean_{WL}(S_F) \text{ if } S_F \neq \emptyset \tag{3}$$

$$m_{CR} = mean_{WA}(S_{CR}) \text{ if } S_{CR} \neq \emptyset \tag{4}$$

$$mean_{WL}(S_F) = \frac{\sum_{t_1=1}^{|S_F|} \omega_{t_1} F_{t_1}^2}{\sum_{t_2=1}^{|S_F|} \omega_{t_2} F_{t_2}} \tag{5}$$

$$mean_{WA}(S_{CR}) = \sum_{t=1}^{|S_{CR}|} \omega_t CR_t \tag{6}$$

$$\omega_t = \frac{\Delta func_t}{\sum_{u=1}^{|S_{CR}|} \Delta func_u} \tag{7}$$

$$\Delta func_t = |func(\mathbf{x}_t) - func(\mathbf{y}_t)| \tag{8}$$

In Eq. (8), $func(\cdot) = f(\cdot)$ when $\phi(\mathbf{x}_t) = \phi(\mathbf{y}_t)$ and $f(\mathbf{x}_t) > f(\mathbf{y}_t)$. $func(\cdot) = \phi(\cdot)$ when $\phi(\mathbf{x}) > \phi(\mathbf{y})$. Where $f(\cdot)$ and $\phi(\cdot)$ are objective function and the constraint violation according to Eqs. (1) and (2).

Before performing the DE operators for $\mathbf{x}_i$, the parameter pair $\{F_i,\ CR_i\}$ is generated by a Cauchy (Normal) distribution of mean $\mu_{F_i}(\mu_{CR_i})$ and a standard deviation σ. The means μ_{F_i} and μ_{CR_i} are generated as follows:

$$F_i = randc_i(\mu_F, 0.1) \tag{9}$$

$$CR_i = randn_i(\mu_{CR}, 0.1) \tag{10}$$

where $randc_i(\mu,\sigma)(randn_i(\mu,\sigma))$ denotes a value generated by a Cauchy (Normal) distribution. When the memories M_F and M_{CR} are both empty, the values of μ_F and μ_{CR} are set as 0.5. Otherwise, an integer r_i is uniformly selected from $[1, H]$, which means the r_i-th pair of values in M_F and M_{CR} will be selected as the means. These values should fall in $[0, 1]$. Otherwise, Eqs. (9) and (10) will be repeated until the pair $\{F_i, CR_i\}$ falls in [0,1].

2.2 Competing Strategy for Selecting DE Operators

In real world, many optimization problems can be seen as black-box problems. Without prior knowledge of problems, it is hard to select a suitable DE strategy. To enhance the robust of an algorithm, various kinds of strategies for selecting DE operators can be used [13].

At the beginning of the evolutionary process, each DE operator has the same probability to be used. This probability is q_l, and q_l is equal to $1/K$. Where K is the number of DE operators. When a DE operator generates a successful trial vector, the probability of each DE operator will be updated as follows:

$$q_l = \frac{n_l + n_0}{\sum_{k=1}^{K}(n_k) + n_0} \tag{11}$$

where n_l is the successful number of the l-th DE operator, and $n_0 > 0$ is a constant, which is used to smooth the influence of each DE operator. When one of the probabilities is less than a threshold δ, each q_l and n_l is reset as $1/K$ and 0, respectively.

In this paper, four kinds of DE operators are used. They are DE/current-to-*p*best/1/Bin (with archive), DE/current-to-*p*best/1/Bin (without archive), DE/randr1/1/Bin and DE/current-to-randr1/1/Bin.

2.3 Local Convergence Detection

To detect the status of population, we define a convergence parameter C as follows:

$$C(G) = \frac{f(\mathbf{x}_{i,G}) - f(\mathbf{x}_{j,G-L})}{f(\mathbf{x}_{j,G-L}) - f(\mathbf{x}_{k,G-2L})} \tag{12}$$

where G is current generation and L is a positive integer defined by users, and $L < 0.5G$. $i = \arg_i \min f(\mathbf{x}_{i,G})$, $j = \arg_j \min f(\mathbf{x}_{j,G-L})$ and $k = \arg_k \min f(\mathbf{x}_{k,G-2L})$ are the indexes of the best solutions in different generations.

We initialize the flag f_{local} to zero, which means the population is not locally convergent. If $f_{local} = 1$, it means that the population is locally convergent. With a given threshold η, the f_{local} is updated at end of each generation as follows:

$$f_{local}(G) = \begin{cases} 0, \text{if} \quad C(G) > \eta \\ 1, \text{if} \quad C(G) \le \eta \end{cases} \tag{13}$$

When $f_{local} = 1$, the polynomial operator [17] will perform for each individual after the DE operator.

3 The Proposed Method

In this paper, we propose a GA with an adaptive recombination operator selection (AGA) method and a PPS constraint-handling method, namely AGA-PPS. The details are as following:

1. The GA uses the framework of SHADE [13] to perform the DE operators with the adaption parameter technique. At the beginning of each generation, the status of the current population is detected. If the population is considered to be locally convergent, the polynomial operator will be performed to each individual after the DE operators. Otherwise, each individual is only performed by the DE operators.
2. The PPS method has two stages to handle the constraints. In the first stage, the algorithm optimizes the CSOPs without considering constraints. When the status of population is considered to be locally convergent, the improved epsilon method, which controls the ϵ according to the feasible rate of the current population (fr), will be applied to the algorithm until the stopping criteria are met, which can help to accelerate the convergence of the population and escape from the local optima.

As a variant of epsilon constrained method, the PPS firstly sets the value of ϵ to be infinity, which makes the algorithm optimize a CSOP without considering constraints in the push stage. When f_{local} becomes 0 at the first time, the algorithm optimizes the CSOP in the pull stage, and then an improved ϵ constrained method will be executed.

To balance the evolutionary search of the population between feasible and infeasible regions, an improved ϵ setting approach is suggested as follows:

$$\epsilon(k) = \begin{cases} \textbf{Rule1}(if\ f_{push} = 1) : \infty \\ \textbf{Rule2}(if\ f_{push} = 0 \wedge r_k < \alpha \wedge FEs < T_c) : \epsilon(k-1)(1 - \frac{FEs}{T_c})^{cp} \\ \textbf{Rule3}(if\ f_{push} = 0 \wedge r_k \ge \alpha \wedge FEs < T_c) : (1+\tau)\phi_{max} \\ \textbf{Rule4}(otherwise) : 0 \end{cases} \tag{14}$$

where f_{push} is the pushing flag initialized as 1. r_k is the proportion of feasible solutions in the generation k. FEs is the number of objective function evaluations (FEs). T_c is a controlled objective function evaluations, which is set according to Eq. (15). ϵ will be set to 0 when FEs reaches to T_c. cp and τ are two parameters to control the decreasing and the increasing speed of ϵ. α is a threshold defined by users. When f_{push} becomes 1, the value of ϵ begins to change. When $r_k \geq \alpha$, it means that the number of feasible solutions in the population is enough, and ϵ tends to increase in order to allow more infeasible solutions to have chances to stay in the population. When $r_k < \alpha$, the ϵ tends to decrease, which tends to search for feasible solutions.

$$T_c = FEs_c + 0.8(MaxFEs - FEs_c) \tag{15}$$

where $MaxFEs$ is the maximal FEs and FEs_c is the FEs when the $f_{local} = 1$ at the first time.

In this paper, the PPS method is embedded in the adaptive GA (AGA-PPS), which is devoted to solve CSOPs, and the pseudo-code of AGA-PPS is shown in Algorithm 1. The best solution $\mathbf{x}_{best}$ updated by each evaluation is the optimal result.

4 Experimental Study

All 28 test instances defined in the report [3] are optimized by the proposed AGA-PPS in this paper. Each instance is a single-objective optimization problem with some inequality or equality constraints. Twenty-five independent runs are carried out for each problem at each kind of dimension levels ($D = 30,\ 50,$). The maximal FEs is set as $20000D$. The experimental results are shown in Tables 1, 2 and 3. As defined in [3], 'Mean' and 'std' in tables respectively denote the mean value and the standard deviation of the objective during the 25 runs.

4.1 Experimental Settings

The parameter settings are listed as follows:

(1) Population size: $N = 5D$.
(2) The length of historic memory: $H = 10$.
(3) Parameters of strategy for selecting DE operators: $K = 4$, $n_0 = 2$, $\delta = 0.05$.
(4) DE/current-to-$p$best/1 parameter: $p = 0.2$.
(5) The size of external archive: $N_A = 2.5N$
(6) Parameters of IEpsilon: $cp = 2$, $\eta = 0.01$, $\alpha = 0.5$, $\tau = 0.1$, $L = 5$.

4.2 Comparison Among AGA-PPS and Three DE Algorithms

We compare AGA-PPS with three algorithms in 30 and 50 dimensions. Three algorithms are LSHADE44+IDE [19], LSHADE44 [13] and UDE [20]. All the experimental results of these three algorithms come from the official website of CEC2018.

```
Algorithm 1: AGA-PPS
Input:
    A CSOP and a stopping criterion.
    $N$, $N_A$: the sizes of population and external archive.
    $cp$, $L$, $\eta$, $T_c$, $\alpha$, $\tau$: parameters of PPS.
    $H$: the length of the historic memory.
    $n_0$, $\delta$: parameters of strategy competition.
Output: The best solution $\mathbf{x}_{best}$.
Step 1: Initialization:
    a) Set probabilities $q_l = 1/4\ for\ l = 1,2,3,4$.
    b) Set counts $n_l = 0\ for\ l = 1,2,3,4$.
    c) Generate a population $P = \{\mathbf{x}_1, ..., \mathbf{x}_N\}$.
    d) Evaluate $f(\mathbf{x}_i)$ and $\phi(\mathbf{x}_i)$, $i = 1, ..., N$. Get the maximal constraint violations $\phi_{max}$.
    e) Set $\epsilon = \infty$. Set $f_{local} = 0$, $f_{push} = 1$.
Step 2: Population update
For $i = 1, \ldots, N$, do
    a) Choose the $l$th strategy according to the $q_l, l = 1,2,3,4$.
    b) Generate DE parameters $F_i$ and $CR_i$ according to the memories $M_F$ and $M_{CR}$.
    c) If $f_{local} = 0$ then perform the selected DE strategy to generate a trial vector $\mathbf{y}_i$.
    d) Else perform the selected DE strategy and polynomial operator to generate a trial
       vector $\mathbf{y}_i$.
    e) Evaluate $f(\mathbf{y}_i)$ and $\phi(\mathbf{y}_i)$,update $\mathbf{x}_{best}$ and $\phi_{max}$.
    f) If $max(\phi(\mathbf{y}_i) - \epsilon, 0) < max(\phi(\mathbf{x}_i) - \epsilon, 0)$ then
        1) replace $\mathbf{x}_i$ with $\mathbf{y}_i$,
        2) $n_l = n_l + 1$, update $q_s, s = 1,2,3,4$ according to Eq (11),
        3) store $|\phi(\mathbf{x}_i) - \phi(\mathbf{y}_i)|$, store $F_i$ and $CR_i$.
    g) ElseIf $max(\phi(\mathbf{y}_i) - \epsilon, 0) = max(\phi(\mathbf{x}_i) - \epsilon, 0) \wedge f(\mathbf{y}_i) < f(\mathbf{x}_i)$ then
        1) insert $\mathbf{x}_i$ to the archive $A$, replace $\mathbf{x}_i$ with $\mathbf{y}_i$.
        2) $n_l = n_l + 1$, update $q_s, s = 1,2,3,4$ according to Eq (11),
        3) store $|f(\mathbf{x}_i) - f(\mathbf{y}_i)|$, store $F_i$ and $CR_i$.
Step 3: Memories update
Update $M_F$ and $M_{CR}$ for each strategy.
Step 4: Convergence status update
Update $f_{local}$, $T_c$, and $f_{push}$ according to Eq (13), (15).
Step 5: Epsilon update
If $f_{push} = 0$, then
    a) Get the proportion of feasible solutions $r_k$ in the current generation $k$.
    b) If $r_k < \alpha\ \wedge\ FEs < T_c$ then
       $\epsilon = \epsilon(1 - FEs/T_c)^{cp}$,
    c) ElseIf $r_k \geq \alpha\ \wedge\ FEs < T_c$ then
       $\epsilon = (1 + \tau)\phi_{max}$,
    d) ElseIf $FEs \geq T_c$ then
       $\epsilon = 0$.
Step 6: Termination
If stopping criteria are satisfied, output the best solution $\mathbf{x}_{best}$. Otherwise, go to Step 2.
```

Tables 1 and 3 show the results of mean values of the four single-objective constrained optimization algorithms applying to 28 constraint problems after 25 independent runs implemented in each problem at each kind of dimension levels (D = 30, 50). According to the Friedman aligned test, AGA-PPS achieves the highest ranking among the four single-objective constrained optimization algorithms. The p values calculated by the statistics of the Friedman aligned test are 3.61923E−05, 1.02665E−05, and 1.49E−05 for D = 30, 50, which reveals the difference among the four algorithms. To compare the statistical difference between the AGA-PPS and other three algorithms, we perform a series of post-hoc tests. Since each adjusted p value in Tables 2 and 4, is less than the preset significant level 0.05, To control the Family-Wise Error Rate (FWER), a set of post-hoc procedures are used as suggested in [21]. we can conclude that AGA-PPS is significantly better than the other three algorithms in the performance of mean value of the objective.

The experimental results in Tables 1 and 3 indicate that AGA-PPS significantly outperforms other three algorithms on C01, C02, C16, C18, C21 and C27 test problems. We analyze the possible reason for C01 test problem. In Fig. 1, the blue area is the objective function and the red area is the constraint function. AGA-PPS can find unconstrained optimal solutions in the push stage, and the unconstrained optimal solutions are the very feasible solutions.

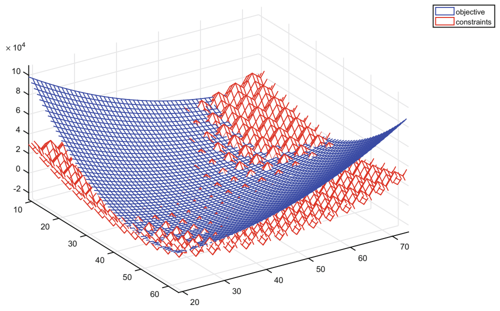

Fig. 1. The C01 test problem. (Color figure online)

Table 1. The mean value and the standard deviation of the objective during the 25 runs on the test instances C01 - C28 with D = 30.

Test instances		AGA-PPS	LSHADE44+IDE	LSHADE44	UDE
C01	Mean	7.10E−29	3.37E−11	1.02E−21	2.21E−15
	Std	4.23E−29	4.11E−11	4.87E−21	7.08E−15
C02	Mean	6.27E−29	1.77E−11	2.86E−21	1.17E−14
	Std	4.24E−29	2.52E−11	9.27E−21	3.65E−14
C03	Mean	1.08E+03	1.13E+07	1.12E+06	8.59E+01
	Std	4.16E+02	4.60E+06	1.95E+06	22.89161
C04	Mean	2.19E+01	1.39E+01	1.97E+01	8.45E+01
	Std	3.87E+00	0.778232	0.540477	23.64712

(continued)

Table 1. (*continued*)

Test instances		AGA-PPS	LSHADE44+IDE	LSHADE44	UDE
C05	Mean	6.47E−28	1.30E−16	4.25E−03	7.22E+00
	Std	9.26E−28	7.82E−17	0.004395	1.065887
C06	Mean	4.09E+02	5.67E+03	3.96E+03	3.28E+02
	Std	5.59E+01	1031.592	722.409	105.1588
C07	Mean	−2.21E+02	−1.02E+01	−5.55E+01	−4.11E+02
	Std	6.65E+01	96.7726	108.028	225.5643
C08	Mean	−2.84E−04	−2.40E−04	−2.80E−04	−2.40E−04
	Std	3.56E−09	4.05E−05	5.77E−10	4.94E−05
C09	Mean	−2.67E−03	−2.67E−03	−2.67E−03	−2.67E−03
	Std	8.85E−19	5.44E−09	1.33E−18	3.32E−16
C10	Mean	−1.03E−04	−9.00E−05	−1.00E−04	−9.12E−05
	Std	4.25E−09	8.64E−06	4.76E−10	1.79E−05
C11	Mean	−3.04E+02	−8.55E−01	−8.75E−01	−2.70E+01
	Std	3.06E+02	0.096998	0.109523	4.755493
C12	Mean	3.98E+00	6.07E+00	4.00E+00	1.57E+01
	Std	4.26E−04	2.839335	0.013465	8.832025
C13	Mean	1.29E+01	3.27E+01	5.03E+01	9.64E+01
	Std	3.02E+01	39.16682	13.6338	129.0651
C14	Mean	1.45E+00	1.93E+00	1.86E+00	1.59E+00
	Std	6.09E−02	0.046647	0.044671	0.193245
C15	Mean	2.73E+00	1.29E+01	1.92E+01	9.27E+00
	Std	1.38E+00	1.539043	3.61396	2.22144
C16	Mean	0	1.56E+02	1.54E+02	8.92E+00
	Std	0	13.61105	15.3015	3.066072
C17	Mean	1.21E+00	1.03E+00	1.00E+00	1.03E+00
	Std	3.17E−01	0.005842	0.018234	0.002783
C18	Mean	3.66E+01	7.54E+03	9.13E+03	9.84E+03
	Std	1.39E−01	5261.00831	6634.57	3779.395
C19	Mean	0	1.28E−03	1.08E−03	1.97E+00
	Std	0	0.000399	0.00094	3.515707
C20	Mean	4.38E+00	2.92E+00	3.55E+00	4.00E+00
	Std	6.63E−01	0.31498	0.221087	1.064174
C21	Mean	9.37E+00	2.77E+01	2.28E+01	1.25E+01
	Std	6.49E+00	9.187089	8.98664	8.474046
C22	Mean	1.84E+02	1.18E+03	3.24E+03	2.21E+02
	Std	2.09E+02	2023.388	3173.81	181.7431
C23	Mean	1.43E+00	1.91E+00	1.86E+00	1.50E+00
	Std	4.48E−02	0.05499	0.060907	0.117495
C24	Mean	3.36E+00	1.42E+01	1.22E+01	9.27E+00
	Std	1.50E+00	1.369376	1.04194	1.28255
C25	Mean	1.83E+01	1.48E+02	1.47E+02	1.59E+01
	Std	7.25E+00	13.87661	12.7452	3.636656
C26	Mean	9.05E−01	1.03E+00	1.00E+00	1.03E+00
	Std	1.92E−01	0.0018	0.021067	0.005126
C27	Mean	3.71E+01	4.16E+04	3.19E+04	3.07E+04
	Std	1.83E+00	19984.77	11319.9	13370.99
C28	Mean	4.94E+01	1.55E+02	1.51E+02	6.50E+01
	Std	2.17E+01	19.07424	20.4233	19.27773
Friedman aligned test		3.13E+01	7.03E+01	7.01E+01	5.43E+01

Table 2. Adjusted p-values for the Friedman Aligned test in terms of mean metric (AGA-PPS is the control method and $D = 30$).

Friedman aligned	Unadjusted	Bonferroni	Holm	Hochberg	Hommel	Holland	Rom	Finner	Li
LSHADE44 +IDE	0.000007	0.000022	0.000022	0.000016	0.000014	0.000022	0.000016	0.000022	0.000007
LSHADE44	0.000008	0.000023	0.000022	0.000016	0.000016	0.000022	0.000016	0.000022	0.000008
UDE	0.008149	0.024448	0.008149	0.008149	0.008149	0.008149	0.008149	0.008149	0.008149

Table 3. Results obtained for $D = 50$, all results for C01–C28.

Test Instances		AGA-PPS	LSHADE44+IDE	LSHADE44	UDE
C01	Mean	6.76E−25	1.21E−03	9.80E−19	6.77E−04
	Std	8.43E−25	0.000758	1.88E−18	0.000977
C02	Mean	1.01E−24	8.25E−04	2.70E−17	2.89E−04
	Std	3.71E−24	0.0007	7.75E−17	0.00033
C03	Mean	5.44E+03	4.14E+07	3.54E+06	3.41E+02
	Std	1.40E+03	1.36E+07	5.08E+06	115.2714
C04	Mean	1.40E+02	1.40E+01	1.48E+02	1.61E+02
	Std	2.67E+01	0.986834	7.4323	27.97972
C05	Mean	1.29E−19	4.31E−09	2.11E+01	3.19E+01
	Std	3.98E−19	1.05E−08	0.379687	3.211975
C06	Mean	8.16E+02	8.99E+03	7.41E+03	6.56E+02
	Std	8.44E+01	1064.459	1203.89	224.7076
C07	Mean	−1.89E+02	−3.65E+01	−3.94E+01	−6.73E+02
	Std	9.48E+01	121.0128	160.717	244.2575
C08	Mean	−1.17E−04	2.96E−04	−1.30E−04	1.62E−03
	Std	3.21E−05	7.59E−05	2.33E−07	0.00079
C09	Mean	−2.04E−03	−1.56E−03	−2.04E−03	−2.04E−03
	Std	1.94E−09	0.000235	1.33E−18	5.84E−11
C10	Mean	−4.75E−05	9.36E−05	−4.82E−05	6.06E−05
	Std	1.33E−06	3.77E−05	8.10E−08	4.70E−05
C11	Mean	−2.59E+03	−7.30E−01	−1.19E+00	−9.48E+01
	Std	3.64E+02	3.298387	2.44299	46.61275
C12	Mean	6.63E+00	7.36E+00	5.20E+01	1.25E+01
	Std	4.07E+00	2.860924	20.8675	5.861807
C13	Mean	6.34E+01	9.14E+01	6.50E+02	1.37E+03
	Std	5.42E+01	24.87958	101.664	416.8811

(*continued*)

Table 3. (*continued*)

Test Instances		AGA-PPS	LSHADE44+IDE	LSHADE44	UDE
C14	Mean	1.17E+00	1.49E+00	1.41E+00	1.29E+00
	Std	8.75E−02	0.029674	0.029579	0.09744
C15	Mean	5.25E+00	1.45E+01	1.78E+01	1.17E+01
	Std	1.26E+00	1.652444	2.99689	1.428187
C16	Mean	6.28E−02	2.72E+02	2.72E+02	1.26E+01
	Std	3.14E−01	17.73679	18.4244	7.25E−15
C17	Mean	1.01E+00	1.05E+00	1.04E+00	1.05E+00
	Std	2.75E−01	0.000586	0.005574	0.001564
C18	Mean	3.66E+01	2.00E+04	2.05E+04	3.40E+04
	Std	3.74E−01	6831.056	7214.01	9621.109
C19	Mean	0	3.54E−02	6.66E−02	6.42E+00
	Std	0	0.019309991	0.038894	7.264131
C20	Mean	1.03E+01	5.63E+00	8.12E+00	7.85E+00
	Std	6.01E−01	0.293081	0.299347	1.640922
C21	Mean	6.62E+00	6.28E+01	6.53E+01	7.64E+00
	Std	3.77E+00	1.433125	2.04016	4.221377
C22	Mean	4.12E+03	1.13E+04	1.45E+04	4.09E+03
	Std	6.42E+03	6028.40364	7731.5	3048.246
C23	Mean	1.15E+00	1.44E+00	1.42E+00	1.26E+00
	Std	3.99E−02	0.029772	0.031323	0.077029
C24	Mean	5.50E+00	1.56E+01	1.43E+01	1.14E+01
	Std	9.07E−01	1.570784	1.28254	1.381348
C25	Mean	5.30E+01	2.65E+02	2.53E+02	2.34E+01
	Std	1.67E+01	20.04611	16.8844	7.592
C26	Mean	9.91E−01	1.05E+00	1.04E+00	1.05E+00
	Std	1.50E−01	0.00346	0.003285	0.003759
C27	Mean	4.07E+01	7.60E+04	8.40E+04	1.09E+05
	Std	1.93E+01	2.03E+04	28825.7	18819.77
C28	Mean	1.39E+02	2.74E+02	2.67E+02	1.33E+02
	Std	3.65E+01	18.77917	17.8361	21.89726
Friedman aligned test		3.21E+01	6.94E+01	7.47E+01	4.98E+01

Table 4. Adjusted p-values for the Friedman Aligned test in terms of mean value of the objective (AGA-PPS is the control method and $D = 50$).

Friedman Aligned	Unadjusted	Bonferroni	Holm	Hochberg	Hommel	Holland	Rom	Finner	Li
LSHADE44 +IDE	0.000001	0.000003	0.000003	0.000003	0.000003	0.000003	0.000003	0.000003	0.000001
LSHADE44	0.000017	0.000052	0.000034	0.000034	0.000034	0.000034	0.000034	0.000026	0.000018
UDE	0.0425	0.127499	0.0425	0.0425	0.0425	0.0425	0.0425	0.0425	0.0425

5 Conclusion

The paper proposes a method to adaptively select the operators according to current convergence status of population, which can prevent the population from being trapped into local optimal. Moreover, the paper also proposes a novel PPS method for solving CSOPs. It divides the search process in two stages, which can help to accelerate the convergence of population and maintain a good balance of searching between feasible and infeasible regions. The proposed AGA-PPS and other three DEs (LSHADE44+IDE, LSHADE44 and UDE) are tested on the CEC2017 benchmarks with 30 and 50 dimensions. The experimental results show that AGA-PPS is significantly better than other three DEs, which manifests that AGA-PPS is a quite competitive algorithm for solving these CSOPs.

References

1. Floudas, C.A., Pardalos, P.M.: A Collection of Test Problems for Constrained Global Optimization Algorithms. LNCS, vol. 455. Springer, Heidelberg (1990). https://doi.org/10.1007/3-540-53032-0
2. Gen, M., Cheng, R.: Genetic Algorithms and Engineering Optimization. Wiley, Hoboken (2000)
3. Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the CEC 2017 Competition on Constrained Real-Parameter Optimization. In: Nation University of Defense Technology, Changsha, Hunan, PA China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Techical report (2016)
4. Le, R.R.G., Knopf, L.C., Haftka, R.T.: A segregated genetic algorithm for constrained structural optimization. In: International Conference on Genetic Algorithms, pp. 558–565. Morgan Kaufmann Publishers Inc. (1995)
5. Huang, F., Wang, L., He, Q.: An effective co-evolutionary differential evolution for constrained optimization. Appl. Math. Comput. **186**(1), 340–356 (2007)
6. Surry, P.D., Radcliffe, N.J.: The COMOGA method: constrained optimisation by multi-objective genetic algorithms. Control Cybern. **26**(3), 391–412 (1997)
7. Lampinen, J.: A fuzzy adaptive differential evolution algorithm. Soft Comput. **9**(6), 448–462 (2005)
8. Brest, J., Greiner, S., Boskovic, B., Mernik, M., Zumer, V.: Self-adapting control parameters in differential evolution: a comparative study on numerical benchmark problems. IEEE Trans. Evol. Comput. **10**(6), 646–657 (2006)
9. Zhang, J., Sanderson, A.C.: JADE: adaptive differential evolution with optional external archive. IEEE Trans. Evol. Comput. **13**(5), 945–958 (2009)
10. Wang, Y., Cai, Z., Zhang, Q.: Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans. Evol. Comput. **15**(1), 55–66 (2011)
11. Tanabe, R., Fukunaga, A.: Success-history based parameter adaptation for Differential Evolution. In: 2013 IEEE Congress on Evolutionary Computation, pp. 71–78. IEEE (2013)
12. Tanabe, R., Fukunaga, A.S.: Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE Congress on Evolutionary Computation, pp. 1658–1665. IEEE (2014)

13. Polkov, R.: L-SHADE with competing strategies applied to constrained optimization. In: 2014 IEEE Congress on Evolutionary Computation, pp. 1683–1689. IEEE (2017)
14. Jordehi, A.R.: A review on constraint handling strategies in particle swarm optimisation. Neural Comput. **26**(6), 1265–1275 (2015)
15. Takahama, T., Sakai, S., Iwane, N.: Solving nonlinear constrained optimization problems by the ε constrained differential evolution. In: IEEE International Conference on Systems, Man and Cybernetics, pp. 2322–2327. IEEE (2006)
16. Holland, J.H.: Genetic algorithms. Sci. Am. **267**(1), 66–73 (1992)
17. Deb, K., Goyal, M.: A combined genetic adaptive search (GeneAS) for engineering design. Comput. Sci. Inform. **26**, 30–45 (1996)
18. Coello, C.A.C.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. Comput. Methods Appl. Mech. Eng. **191**(11–12), 1245–1287 (2002)
19. Tvrdik, J., Polakova, R.: A simple framework for constrained problems with application of L-SHADE44 and IDE. In: 2017 IEEE Congress on Evolutionary Computation, pp. 1436–1443. IEEE (2017)
20. Trivedi, A.: A unified differential evolution algorithm for constrained optimization problems. In: 2017 IEEE Congress on Evolutionary Computation, pp. 1231–1238. IEEE (2017)
21. Derrac, J., Garcia, S., Molina, D., Herrera, F.: A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. Swarm Evol. Comput. **1**(1), 3–18 (2011)