LSHADE with S-shape Constraint-handling Technique in Push and Pull Search for Constrained Optimization Problems

Jinglei Guo[1], Tianpei Cheng[1], Zhun Fan[2†], Xinyu Zhou[3]
[1]School of Computer Science, Central China Normal University, Wuhan, China
[2]Department of Electronic Engineering, Shantou University, Shantou, China
[3]School of Computer and Information Engineering, Jiangxi Normal University, Nanchang, China
[†] Corresponding author: zfan@stu.edu.cn

***Abstract*—Constrained optimization problem is a common issue in science and engineering. The key to solve this problem is to balance the relationship between constraints and objectives. Therefore, this paper proposes an s-shape constraint-handling method in push and pull search (SLSHADE_PPS), which divides the whole evolutionary process into two phases, called push search phase and pull search phase respectively. The push search phase mainly focuses on the value of the objective function and uses LSHADE to push the population into the optimal region of the objective function. In the pull search phase, the s-shape constraint handling technique is combined with LSHADE to pull the infeasible individuals back to feasible region. The s-shape function makes the violation tolerance maintain high level at the start stage and strictly limits the solutions to reside in the feasible region at the end stage of pull phase. SLSHADE_PPS shows significant advantages over the state-of-the-art constraint algorithms on the 28 benchmark test functions from IEEE CEC2017.**

***Index Terms*—push search phase, pull search phase, s-shape function, constraint**

I. Introduction

Most scientific and engineering optimization problems have their own constraints, which are known as the constrained optimization problems (COPs). Generally, COPs can be described as follows:

$$\begin{aligned} &minimize \quad f(\vec{x}), \vec{x} = (x_1, \ldots, x_D) \in S \\ &s.t. \begin{cases} g_j(\vec{x}) \leq 0 \quad j = 1, \ldots, l \\ h_j(\vec{x}) = 0 \quad j = l+1, \ldots, m \\ L_i \leq x_i \leq U_i \quad i = 1, \ldots, D \end{cases} \end{aligned} \tag{1}$$

where $\vec{x} = (x_1, \ldots, x_D)$ is called decision vector, x_i is the ith decision variable of decision vector $\vec{x}$, L_i and U_i are the lower and upper bounds of x_i respectively. $S = \prod_{i=1}^{D} [L_i, U_i]$ is the decision space, D is the number of decision variables, $f(\vec{x})$ is objective function, $g_j(\vec{x})$ is the jth inequality constraint, $h_j(\vec{x})$ is the jth equality constraint, l and $m-l$ represent the number of inequality and equality constraints respectively.

For COPs, the degree of constraint violation of decision vector $\vec{x}$ under the jth constraint can be described as follows:

$$G_j(\vec{x}) = \begin{cases} max\{0, g_j(\vec{x})\} & j = 1, \ldots, l \\ max\{0, |h_j(\vec{x}) - \delta|\} & j = l+1, \ldots, m \end{cases} \tag{2}$$

where δ is a positive tolerance value of equality constraint. Subsequently, the degree of constraint violation under all constraints can be computed as:

$$G(\vec{x}) = \sum_{j=1}^{m} G_j(\vec{x}) \tag{3}$$

where $G(\vec{x})$ represents the degree of constraint violation of decision vector $\vec{x}$ under all constraints. The decision variable $\vec{x}$ that satisfied $G(\vec{x}) = 0$ is called the feasible solution, otherwise $\vec{x}$ is called the infeasible solution. The decision space of COP consists of feasible region and infeasible region.

Due to the complexity of objective function and feasible region in constrained optimization problems, traditional mathematical methods have limitations in solving such problems. Therefore, more and more researchers focus on using evolutionary algorithms(EAs) to solve constrained optimization problems, and a large number of constrained optimization evolution algorithms (COEAs) have been proposed. To sum up, these methods can be classified into four main categories:

1) Penalty Function Method: According to different ways of setting penalty coefficient, the penalty function can be divided into three types: (1) static penalty functions. Hernandez et al [1] proposed a static penalty function to handle constraints and used the hybrid differential evolution(DE) algorithm to search decision space. However, the fact should be faced the same penalty coefficient is not suitable to all kind problems. (2) dynamic penalty functions. An s-type dynamical penalty factor was introduced by Liu et al. [2] to balance exploration and exploitation. Although the dynamic penalty function shows good performance in some COPs, the lack of the feedback information limits its ability. (3) adaptive penalty function. Mani and Patvardhan [3] presented a hybrid constraint handling technique for a two-population adaptive coevolutionary algorithm, which uses a self determining and regulating penalty factor methods as well as feasibility rules for handling constraints.

2) Objectives and constraints separation method: The main idea of this method is considering only constraints violations or objective functions when comparing solutions. Among these

978-1-7281-6929-3/20/$31.00 ©2020 IEEE

methods, feasibility rule is a simple and efficient constraint-handling technique used by Deb [4]. Elsayed et al. [5] proposed the concept of multi-parent crossover on the basis of GA, and added random operation to real-coded genetic algorithm to solve the constraint numerical optimization problems. Wang et al. [6] proposed a hybrid multi-group PSO algorithm in which the particles are sorted by using feasibility rules. In addition, Wang et al. extended the information of the objective function to the feasibility rule, which can make those individuals with small objective function discarded by the feasible rule store into an archive [7]. In order to alleviate the constraint preference of feasibility rules, stochastic ranking was presented by Runarsson and Yao [8], which use probability P_f to determine whether selecting individuals by objective function or by degree of constraint violation. Takahama and Sakai [9] combined the ε-constraint method with the comparison of kernel regression estimation to improve the efficiency of high-quality solutions in a very small number of function evaluations.

3) Multi-objective optimization method: In this method, the constraint conditions are usually transformed into objective functions to form multi-objective optimization problems. In Cai and Wang's method (CW) [10], important information of some infeasible individuals is utilized to guide the population to converge quickly. But the algorithm must use a trial-and-error scheme to determine some problem-dependent parameters. To overcome shortcomings of CW algorithm, Wang and Cai proposed an improved version CMODE [11], which combines multi-objective optimization with DE to solve constrained optimization problems. Jiao et al. [12] presented a new selection strategy, which first chooses the non-dominated individuals within an allowed level of constraint violation, then selects other individuals based on a specially defined fitness function.

4) Ensemble of constraint-handling techniques: Based on the no free lunch theory [13], Mallipeddi and Suganthan [14] proposed a hybrid of four constraint processing techniques: feasibility rules, stochastic ranking, self-adaptive penalty functions and ε-constraint, each of which is applied on a specific subpopulation. Elsayed et al. [15] used a hybrid of feasibility rules with ε-constraint method. Li et al. [16] presented an adaptive constrained artificial bee colony algorithm combining feasibility rules with multi-objective optimization method. Wang et al. [17], [18] introduced a multi-objective optimization method to select individuals in the infeasible stage.

In this paper, LSHADE is used as the search engine and "push-and-pull" is adopted as the search model. In push phase, the offspring are selected according to the value of the objective function, without considering the influence of constraints. In pull stage, the infeasible individuals obtained in push search phase are pulled back to the feasible region by s-shape constraint-handling technique. In summary, the main contributions of this paper are as follows:

- An s-shape function is suggested for the adapting of violation tolerance ratio. In the proposed strategy, the violation tolerance keeps at a high level to make the candidates with better objective value have more opportunity to survive in the early pull phase. After that, the violation tolerance decreases gradually to enforce the constraints limitation. In the late of pull phase, the violation tolerance is close to 0 to ensure more feasible solutions stay in population.
- The performance of the proposed SLSHADE_PPS is verified by comparing with three existing algorithms on 28 benchmark COPs in CEC2017 [19], including AGA_PPS [20], LSHADE44_IDE [21] and LSHADE44 [22]. The experiment result shows that the proposed SLSHADE_PPS demonstrates significantly better performance than the compared algorithms.

The remainder of this paper is organized as follows. In Section II, LSHADE and "push-and-pull" model are reviewed. Section III presents the proposed algorithm in detail. The results on 28 test instances of CEC 2017 are shown and discussed in section IV. Section V concludes this paper.

II. LSHADE AND "PUSH-AND-PULL" MODEL

A. LSHADE

LSHADE was proposed by R. Tanabe and A. Fukunaga [23] in 2014. The following is a brief introduction of LSHADE.

1) DE operator in LSHADE: In LSHADE, DE/current-to-pbest/1 is used as the search algorithm, which is modified from DE/current-to-best/1 [24]. In DE/current-to-pbest/1, the individual $\vec{x}_{pbest,t}$ is randomly selected out one of the individuals whose objective function values rank the first $100p\%$ (p∈(0,1]) in the current population. Meanwhile, an external archive A is used to store inferior solutions which are eliminated in the process of selection, so that some individuals in A participate in the later evolution process to guide the direction of evolution. The DE/current-to-pbest/1 with external archive A is described as:

$$\vec{v}_{i,t} = \vec{x}_{i,t} + F \cdot (\vec{x}_{pbest,t} - \vec{x}_{i,t}) + F \cdot (\vec{x}_{r1,t} - \vec{x}^{'}_{r2,t}) \quad (4)$$

where $\vec{x}_{r1,t}$ is a vector randomly selected from the current population P, and $\vec{x}^{'}_{r2,t}$ represents an individual randomly selected from the union, $P \cup A$.

2) Scale control factor F : In each generation, the scale control factor F of each individual $\vec{x}_{i,t}$ is generated by Cauchy distribution function based on the historical success information. F is limited in [0,1], and its formula is as follows:

$$F = randc(M_F, 0.1) \quad (5)$$

M_F is described as follows:

$$M_F = \begin{cases} mean_{WL}(S_F), & S_F \neq \emptyset \\ M_F, & otherwise \end{cases} \quad (6)$$

where S_F represents the archive of the F which has successfully generated an individual that survives into next generation. The calculation formula of $mean_{WL}(S_F)$ is as follows:

$$mean_{WL}(S_F) = \frac{\sum_{k=1}^{|S_F|} w_k \cdot S_{F_k}^2}{\sum_{k=1}^{|S_F|} w_k \cdot S_{F_k}} \quad (7)$$

The description of w_k is as follows:

$$w_k = \frac{\Delta f_k}{\sum_{k=1}^{|S_F|} \Delta f_k} \tag{8}$$

where Δf_k is the difference in the value of the objective function between the parent and the offspring.

3) Crossover control factor CR : Similar to the way of generating F, the crossover control factor is generated by the standard normal distribution based on the successful historical information. The range of CR is in [0,1], and the mathematical description is as follows:

$$CR = randn(M_{CR}, 0.1) \tag{9}$$

M_{CR} is described as follows:

$$M_{CR} = \begin{cases} mean_{WA}(S_{CR}), & S_{CR} \neq \emptyset \\ M_{CR}, & otherwise \end{cases} \tag{10}$$

where S_{CR} represents the archive of CR which has successfully generated an individual that survives into next generation, and the calculation formula of $mean_{WA}(S_{CR})$ is as follows:

$$mean_{WA}(S_{CR}) = \frac{\sum_{k=1}^{|S_CR|} w_k \cdot S_{CR_k}^2}{\sum_{k=1}^{|S_CR|} w_k \cdot S_{CR_k}} \tag{11}$$

4) Linear population size reduction: A dynamically decreasing function is used in LSHADE to improve the performance of SHADE. At generation t, the population size N_{t+1} is computed according to the formula:

$$N_{t+1} = round[(\frac{N_{min} - N_{init}}{MaxFEs}) \cdot FEs + N_{init}] \tag{12}$$

where N_{min} and N_{init} represent the minimum population size and the initial population size in the evolution process respectively.

B. Push And Pull Framework

Push and pull framework is a two-phase search process proposed by Fan et al [25]. The framework is mainly composed of push search operator and pull search operator.

1) Push search operator: The main purpose of push search operator is to clear the obstacle of infeasible regions and push the objective function value to the minima. The constraint conditions are ignored in the push search operation and the COPs are simplified into a problem only considering an objective function. In [20], the search model shifts from the push phase to the pull phase, when the value of the objective function does not change significantly, The indicator $C(t)$ in [20] is as follows:

$$C(t) = \frac{f(\vec{x}_{i,t}) - f(\vec{x}_{j,t-L})}{f(\vec{x}_{j,t-L}) - f(\vec{x}_{k,t-2L})} < eta \tag{13}$$

where t represents the generation of the current population, eta is the given threshold, $\vec{x}_{i,t}$, $\vec{x}_{j,t-L}$ and $\vec{x}_{k,t-2L}$ are the best individuals in the period (0, t), (0, $t-L$) and (0, $t-2L$), respectively.

2) Pull search operator: Because constraint conditions are not considered in the push phase, the best solution which obtained in push search operator may fall in infeasible region. In pull phase, the infeasible solutions are pulled back into the feasible region by some constraint handling means.

III. Proposed Approach

A. Motivation

How to balance the relationship between the objective function and the constraints is the primary problem to solve the COPs. ε-constraint method [26] is popular and widely adopted by researchers because of its simple and stable selection mechanism. In ε-level comparison, an order relation on a pair of objective function value and constraint violation $(f(\vec{x}), G(\vec{x}))$ is defined. The ε-constraint method converts a constrained optimization problem into an unconstrained one by replacing the order relation with the ε level comparison. However, it is difficult to find an appropriate ε value to balance the relationship between constraints and objectives.

In view of the above discussion, we propose an s-shape curve as the violation tolerance valve, which dynamically changes the value of ε in the evolution process. This method not only pushes the population into the feasible region efficiently, but also alleviates the sensitivity of specific ε in advance.

B. S-shape constraint handling technique

In the SLSHADE_PPS, we introduce an s-shape curve to adaptively adjust the violation tolerance ratio $s(t)$ in each generation t. The function of s-shape curve $s(t)$ is as follows:

$$s(t) = \frac{1}{1 + e^{\alpha(\frac{t}{MaxFEs} - \beta)}} \tag{14}$$

The shape of $s(t)$ is determined by two parameters α and β. The parameter α impacts slope of the curve. Fig.1 shows the curves of different α values when β stays 0.5. It can be seen from Fig.1 that the larger the α is, the greater the slope is in the middle of the curve and the slower the change is at both start and end stage. Fig.2 shows the curves of different

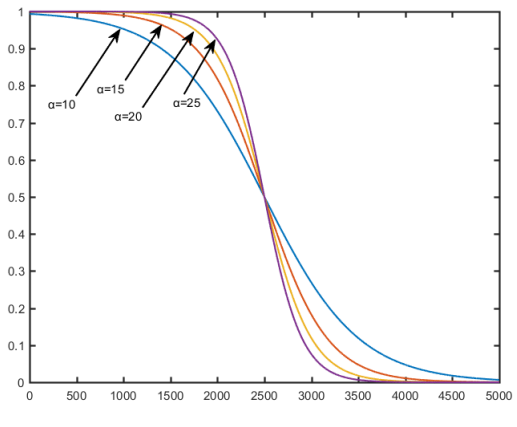

Fig. 1. $\beta = 0.5$, α changes.

β values when α=15. It can be seen from Fig.2 that β value influences the length of head and tail. The curve has a long even tail with a small β, whereas the curve has a short tail and a long flat head with a large β.

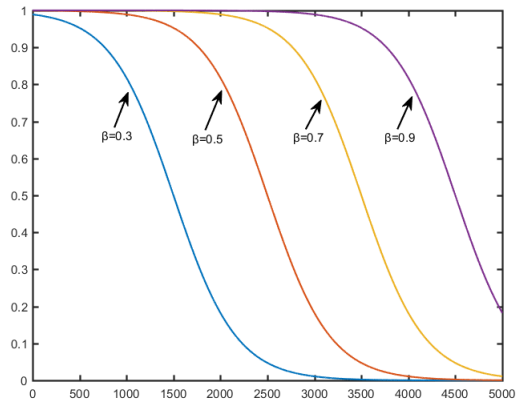

Fig. 2. α=20, β changes.

Using s-shape constraint handling method to deal with constraint conditions, the value of violation tolerance ε decreases very slowly in the early stage to improve the diversity of the population, whereas ε should be maintained near or at 0 for a certain period to enforce the limitation of violation in the later stage. Therefore, we use s-shape curve to adjust ε in the pull phase, which is described as follows:

$$\varepsilon(t) = \begin{cases} Max_{cv} \cdot s(t), & t < 0.8 * MaxFES \\ 0\ , & otherwise \end{cases} \tag{15}$$

where Max_{cv} is the maximum of constraint violation in the current population. Because of the small variation of $s(t)$ in the early stage, $\varepsilon(t)$ decreases very slowly which is beneficial to the diversity of the population. In the later stage, when its value approaches 0, ε-constraint method will degenerate into feasibility rules, which is beneficial to accelerating the convergence rate of the population.

C. Framework

SLSHADE_PPS includes two main phases: the push phase and the pull phase. LSHADE is employed as a search engine in the above two phases. Because constraints are not considered in the push phase, the constraint violation tolerance ε=∞. In the pull phase, ε is updated by the s-shape constraint handling technique. The framework of SLSHADE_PPS is presented in Algorithm 1.

IV. Experimental Results

A. Benchmark Test Functions and Parameter Settings

To validate the performance of the SLSHADE_PPS, we employ a set of 28 benchmark functions with 30-D developed in IEEE CEC2017 [19] . The details of these functions can be found in [19] and the objective function of all test functions should be minimized. In the experimental study

Algorithm 1 SLSHADE_PPS

Input:
- $MaxFEs$: maximum number of fitness evaluation.
- H: the length of the historic memory.
- α, β: the parameters of s-shaped curve.
- N_{min}, N_{init}: the minimum size of population and the initial size of population.
- A: external archive.
- S_{CR}: successful CR storage.
- S_F: successful F storage.

Output:
- the feasible individual with the smallest objective function value in the population.

Initialization:
- t=1.
- Generate an initial population $P_t = \{x_{1,t}, \ldots, x_{N,t}\}$ by uniformly randomly sampling from the decision space .
- set $f_{push} = 1$ //set the population in push phase
- set $CR = 0.5$, $F = 0.5$, ε=∞.
- Evaluate $f(\vec{x}_{i,t+1})$ and $G(\vec{x}_{i,t+1})$ for each individual in P_t.
- $FEs = N$.

while $FEs \leq MaxFES$ **do**
  for $i := 1$ to N **do**
    For $\vec{x}_{i,t}$, an offspring $\vec{x}_{i,t+1}$ is generated by Eq.(4)
    Evaluate $f(\vec{x}_{i,t+1})$ and $G(\vec{x}_{i,t+1})$, update $\vec{x}_{best}$.
    if $\max(G(\vec{x}_{i,t+1})-\varepsilon, 0) < \max(G(\vec{x}_{i,t})-\varepsilon, 0)$ **then**
      replace $\vec{x}_{i,t}$ with $\vec{x}_{i,t+1}$, store $\vec{x}_{i,t}$ to A.
      store $|G(\vec{x}_{i,t}) - G(\vec{x}_{i,t+1})|$.
      store F to S_F.
      store CR to S_{CR}.
    end if
    if $\max(G(\vec{x}_{i,t+1}) - \varepsilon, 0) = \max(G(\vec{x}_{i,t}) - \varepsilon, 0) \wedge f(\vec{x}_{i,t+1}) < f(\vec{x}_{i,t})$ **then**
      replace $\vec{x}_{i,t}$ with $\vec{x}_{i,t+1}$,store $\vec{x}_{i,t}$ to A.
      store $|f(\vec{x}_{i,t}) - f(\vec{x}_{i,t+1})|$.
      store F to S_F.
      store CR to S_{CR}.
    end if
  end for
  Calculate the value of $C(t)$ by Eq.(13)
  if $C(t) < eta$ **then**
    $f_{push} = 0$ //change the push phase into pull phase
  end if
  Update F by Eq.(5)
  Update CR by Eq.(9)
  Update population size N_t by Eq.(12)
  Resize archive size N_A of $|A|$ according to N_t
  if $f_{push} = 0$ **then**
    Update ε by Eq.(15)
  end if
  t=t+1
end while

of SLSHADE_PPS, the maximum number of evaluations MaxFEs and the tolerance value δ for equality constraints are respectively set to 1.0×10^6 and 0.0001 as suggested in [19].

LSHADE is served as a search engine in our algorithm, the population size N is decreased linearly in the whole process of evolution. The control parameters of LSHADE in the proposed algorithm are: (1) the initial population size N_{init} is 200, (2) the minimum population size N_{min} is 50, (3) the size of the external archive N_A is $2.6 \cdot N_t$, (4) p value in DE/current-to-pbest/1 is 0.11.

Next, we investigate different α and β values in formula (14) to select the best ones for optimizing the performance of SLSHADE_PPS.

1) Adjusting the parameter β: Under the condition of $\alpha = 15$, we take the values of β as 0.3, 0.5, 0.7 and 0.9, respectively, and run them independently on the 28 benchmark test functions for 25 times. Table I presents the mean objective function value and standard deviation (denoted as "mean" and "std") with the different β, and the relatively good values are shown in **boldface**. To show the performance clearly, the Friedman's ranking is used to get their rankings. As seen in Table II, β=0.5 achieves the best ranking. Therefore, the parameter β is set to 0.5 in the following experiments.

2) Adjusting the parameter α: Again, β is fixed at 0.5, α = 10, 15, 20 and 25 respectively, and run independently on the test functions for 25 times. The mean objective function value and standard deviation with different α value are demonstrated in Table III. The Friedman's ranking is also used to get their rankings. As seen in Table IV, α=15 achieves the best ranking. Therefore, the parameter β=0.5 and α=15 are selected in the following experiments.

B. Comparison SLSHADE_PPS with other state-of-the-art Algorithms

Finally, the performance of SLSHADE_PPS is compared with three popular COEAs on the 28 test functions from IEEE 2017 [15].

- AGA_PPS [20]: adaptive GA with push and pull search method
- LSHADE44_IDE [21]: Framework of L-SHADE44 and IDE
- LSHADE44 [22]: An enhanced version of L-SHADE algorithm

The parameters setting of above there compared algorithms is the same as their original articles. Table VI summarizes the mean objective function value and standard deviation derived from the three compared algorithms over 25 independent runs. As shown in Table VI, SLSHADE_PPS obtains 22 best results among the 28 test functions, except C01, C05, C11, C16, C20, C28. The compared AGA_PPS, LSHADE44_IDE and LSHADE44 get 8, 2 and 8 best results respectively. In contrast, AGA_PPS surpass SLSHADE_PPS on 3 functions (C11, C16 and C28), LSHADE44_IDE can not surpass SLSHADE_PPS on any functions, LSHADE44 outperforms SLSHADE_PPS on 3 functions(C01, C05 and C20). The Friedman's ranking is still conducted to obtain the rankings for D = 30. and results

TABLE I
THE RESULTS OF SLSHADE_PPS WITH DIFFERENT β

		β=0.3	β=0.5	β=0.7	β=0.9
C01	mean	**2.97E-30**	5.94E-30	6.12E-30	4.36E-30
	std	4.99E-30	8.61E-30	8.21E-30	6.79E-30
C02	mean	8.56E-30	6.24E-30	**2.39E-30**	4.61E-30
	std	9.90E-30	7.54E-30	4.69E-30	6.95E-30
C03	mean	1.01E+02	8.88E+01	7.85E+01	**7.25E+01**
	std	5.02E+01	3.24E+01	1.44E+01	1.56E+01
C04	mean	**1.36E+01**	**1.36E+01**	**1.36E+01**	**1.36E+01**
	std	3.63E-15	3.63E-15	3.78E-03	3.63E-15
C05	mean	6.11E-30	5.25E-30	5.44E-30	**3.82E-30**
	std	1.36E-29	1.09E-29	1.38E-29	8.12E-30
C06	mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C07	mean	-1.16E+03	**-1.28E+03**	-1.27E+03	-1.27E+03
	std	4.48E+01	3.17E+01	2.98E+01	3.74E+01
C08	mean	**-2.84E-04**	**-2.84E-04**	**-2.84E-04**	5.98E-05
	std	1.11E-19	1.11E-19	4.68E-07	3.66E-04
C09	mean	**-2.67E-03**	**-2.67E-03**	**-2.67E-03**	-2.67E-03
	std	8.85E-19	8.85E-19	8.85E-19	2.22E-07
C10	mean	**-1.03E-04**	**-1.03E-04**	**-1.03E-04**	-7.46E-05
	std	1.38E-20	1.38E-20	6.26E-08	3.60E-05
C11	mean	-1.85E+01	-2.65E+02	-1.10E+03	**-1.36E+03**
	std	3.40E+01	2.83E+02	4.76E+02	4.55E+02
C12	mean	**3.98E+00**	**3.98E+00**	**3.98E+00**	**3.98E+00**
	std	4.78E-05	5.85E-05	4.78E-05	4.88E-05
C13	mean	1.45E-27	**7.93E-28**	1.59E-01	3.75E-27
	std	2.63E-27	1.65E-27	7.97E-01	1.14E-26
C14	mean	**1.41E+00**	**1.41E+00**	**1.41E+00**	1.43E+00
	std	6.80E-16	6.80E-16	1.06E-02	3.92E-02
C15	mean	5.50E+00	**2.36E+00**	**2.36E+00**	**2.36E+00**
	std	9.06E-16	9.06E-16	9.06E-16	9.06E-16
C16	mean	6.85E+00	**6.28E+00**	**6.28E+00**	**6.28E+00**
	std	1.35E+00	6.45E-08	1.30E-06	1.57E-06
C17	mean	4.69E-01	3.09E-01	3.52E-01	**2.07E-01**
	std	1.80E-01	4.40E-01	5.33E-01	3.28E-01
C18	mean	**3.65E+01**	**3.65E+01**	**3.65E+01**	**3.65E+01**
	std	1.35E-03	2.25E-03	1.44E-03	6.03E-02
C19	mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C20	mean	1.65E+00	**1.64E+00**	1.66E+00	**1.64E+00**
	std	1.05E-01	1.46E-01	1.20E-01	1.49E-01
C21	mean	2.12E+01	**4.92E+00**	8.35E+00	7.09E+00
	std	9.84E+00	2.17E+00	9.04E+00	6.81E+00
C22	mean	3.23E+00	3.25E+00	**2.11E-01**	3.30E+00
	std	1.61E+01	1.63E+01	7.48E-01	1.65E+01
C23	mean	1.42E+00	**1.41E+00**	1.42E+00	1.42E+00
	std	2.88E-02	1.74E-02	2.88E-02	2.88E-02
C24	mean	5.37E+00	2.73E+00	**2.36E+00**	2.48E+00
	std	6.28E-01	1.04E+00	9.06E-16	6.28E-01
C25	mean	1.04E+01	**6.35E+00**	**6.35E+00**	6.41E+00
	std	3.33E+00	3.14E-01	3.14E-01	4.35E-01
C26	mean	7.34E-01	3.63E-01	**2.55E-01**	3.13E-01
	std	1.37E-01	2.40E-01	2.65E-01	3.52E-01
C27	mean	3.65E+01	3.65E+01	3.65E+01	**3.64E+01**
	std	1.87E-03	1.33E-03	2.00E-03	2.13E-01
C28	mean	6.38E+01	5.62E+01	3.79E+00	**3.05E+00**
	std	2.37E+01	3.25E+01	4.19E+00	4.45E+00

TABLE II
FRIEDMAN'S RANKINGS OF SLSHADE_PPS WITH DIFFERENT β

β values	Ranking
β=0.3	2.82
β=0.5	**2.11**
β=0.7	2.55
β=0.9	2.51

TABLE III
THE RESULTS OF SLSHADE_PPS WITH DIFFERENT α

		α=10	α=15	α=20	α=25
C01	mean	5.71E-30	5.94E-30	5.68E-30	**4.48E-30**
	std	6.88E-30	8.61E-30	7.55E-30	6.94E-30
C02	mean	**4.03E-30**	6.24E-30	5.78E-30	6.45E-30
	std	6.04E-30	7.54E-30	7.78E-30	7.74E-30
C03	mean	8.32E+01	8.88E+01	**7.17E+01**	9.01E+01
	std	2.20E+01	3.24E+01	1.49E+01	3.01E+01
C04	mean	**1.36E+01**	**1.36E+01**	**1.36E+01**	**1.36E+01**
	std	3.63E-15	3.63E-15	3.63E-15	3.63E-15
C05	mean	4.39E-30	5.25E-30	5.88E-30	**4.07E-30**
	std	1.43E-29	1.09E-29	1.52E-29	7.46E-30
C06	mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C07	mean	-1.25E+03	**-1.28E+03**	-1.26E+03	-1.27E+03
	std	2.45E+01	3.17E+01	4.16E+01	3.82E+01
C08	mean	**-2.84E-04**	**-2.84E-04**	**-2.84E-04**	**-2.84E-04**
	std	1.11E-19	1.11E-19	1.11E-19	5.77E-12
C09	mean	**-2.67E-03**	**-2.67E-03**	**-2.67E-03**	**-2.67E-03**
	std	8.85E-19	8.85E-19	8.85E-19	8.85E-19
C10	mean	**-1.03E-04**	**-1.03E-04**	**-1.03E-04**	**-1.03E-04**
	std	1.38E-20	1.38E-20	1.38E-20	1.08E-11
C11	mean	-9.62E+01	-2.65E+02	-4.73E+02	**-7.95E+02**
	std	2.17E+02	2.83E+02	4.06E+02	6.18E+02
C12	mean	**3.98E+00**	**3.98E+00**	**3.98E+00**	**3.98E+00**
	std	4.54E-05	5.85E-05	3.44E-05	5.18E-05
C13	mean	4.99E-28	7.93E-28	**4.49E-28**	1.90E-27
	std	1.14E-27	1.65E-27	1.70E-27	4.88E-27
C14	mean	**1.41E+00**	**1.41E+00**	**1.41E+00**	1.43E+00
	std	6.80E-16	6.80E-16	1.74E-02	3.32E-02
C15	mean	4.62E+00	**2.36E+00**	**2.36E+00**	2.36E+00
	std	1.44E+00	9.06E-16	9.06E-16	2.00E-08
C16	mean	6.35E+00	**6.28E+00**	6.28E+00	6.28E+00
	std	3.14E-01	6.45E-08	2.00E-07	8.21E-08
C17	mean	4.19E-01	3.09E-01	**1.64E-01**	3.67E-01
	std	4.39E-01	4.40E-01	2.82E-01	5.02E-01
C18	mean	**3.65E+01**	3.65E+01	3.65E+01	3.65E+01
	std	2.24E-03	2.25E-03	2.61E-03	2.50E-03
C19	mean	**0.00E+00**	**0.00E+00**	**0.00E+00**	**0.00E+00**
	std	0.00E+00	0.00E+00	0.00E+00	0.00E+00
C20	mean	1.66E+00	1.64E+00	1.63E+00	**1.62E+00**
	std	1.34E-01	1.46E-01	1.40E-01	1.67E-01
C21	mean	1.29E+01	**4.92E+00**	7.60E+00	6.40E+00
	std	1.10E+01	2.17E+00	8.04E+00	6.79E+00
C22	mean	**1.60E-01**	3.25E+00	3.27E+00	3.58E+00
	std	7.97E-01	1.63E+01	1.63E+01	1.62E+01
C23	mean	1.42E+00	**1.41E+00**	1.42E+00	1.42E+00
	std	2.88E-02	1.74E-02	3.25E-02	2.41E-02
C24	mean	5.00E+00	2.73E+00	**2.36E+00**	**2.36E+00**
	std	1.18E+00	1.04E+00	2.00E-08	9.06E-16
C25	mean	8.61E+00	**6.35E+00**	**6.35E+00**	6.41E+00
	std	2.84E+00	3.14E-01	3.14E-01	4.35E-01
C26	mean	6.78E-01	3.63E-01	3.05E-01	**2.50E-01**
	std	1.58E-01	2.40E-01	2.51E-01	1.83E-01
C27	mean	**3.65E+01**	**3.65E+01**	**3.65E+01**	**3.65E+01**
	std	1.35E-03	1.33E-03	2.31E-03	2.01E-03
C28	mean	6.22E+01	5.62E+01	5.89E+00	**2.47E+00**
	std	2.76E+01	3.25E+01	5.78E+00	4.11E+00

reported in Table V. From Table V, SLSHADE_PPS achieves the first average ranking among the four algorithms, followed by AGA_PPS. LSHADE44 and LSHADE44_IDE obtain the thrid ranking and the fourth ranking ,respectively.

Furthermore, the Friedman's test with the Bonferroni-Dunn method is carried out via KEEL software to compare the performance of multiple methods concurrently. In Table VII, SLSHADE_PPS obtains higher R^+ values than R^- values in all cases, the p-values of all cases are less than 0.05 which indicates that SLSHADE_PPS significantly better than the three competitors.

The above comparison verifies that SLSHADE_PPS has better performance than the three competitors on the 28 benchmark test functions from IEEE CEC2017.

TABLE IV
FRIEDMAN'S RANKINGS OF SLSHADE_PPS WITH DIFFERENT α

α values	Ranking
α=10	2.95
α=15	**2.21**
α=20	2.38
α=25	2.46

TABLE V
AVERAGE RANKINGS OF FOUR ALGORITHMS

Algorithms	Ranking
AGA_PPS	2.23
LSHADE44_IDE	3.48
LSHADE	2.86
SLSHADE_PPS	**1.43**

TABLE VII
STASTICAL TEST RESULTS OF PPS_LSHADE AND OTHER THREE COMPARED ALGORITHMS BY THE MULTIPLE-PROBLEM WILCOXON'S SOGMED RANK TEST

SLSHADE_PPS VS	R^+	R^-	p-value	α=0.1	α=0.05
AGA_PPS	339.5	66.5	1.23E-3	Yes	Yes
LSHADE44_IDE	378.0	0.0	1.49E-8	Yes	Yes
LSHADE44	378.0	28.0	1.11E-5	Yes	Yes

V. CONCLUSION

Based on the discussion of the balancing between the constraints and objective functions, we divide the entire evolution process into two phases under the push and pull search model. In the push phase, constraint violation is without consideration and the change of objective values is monitored. In the pull search phase, we use s-shape curve to dynamically change the violation tolerance of ε. The SLSHADE_PPS algorithm balances well the relationship between the objective function and the constraint violation. Experimental results show that:

1) SLSHADE_PPS shows better or at least competitive performance over the compared constrained optimization evolutionary algorithms.
2) The use of s-shape constraint technique improves the diversity of the population in the pull search phase and has a significant impact on balancing the relationship between constraints and objective functions.

ACKNOWLEDGMENT

This work is part funded by the National Natural Science Foundation of China (Nos. 61966019, 61603163) the open fund from Key Lab of Digital Signal and Image Processing of Guangdong Province (No.2019GDDSIPL-04) and the Fundamental Research Funds for the Central Universities (No.CCNU20TS026).

TABLE VI
EXPERIMENTAL RESULTS OF AGA_PPS, LSHADE44_IDE, LSHADE44 AND SLSHADE_PPS

		AGA_PPS	LSHADE44_IDE	LSHADE44	SLSAHDE_PPS
C01	Mean	6.27E-29	8.66E-17	**4.92E-30**	5.94E-30
	Std	3.11E-29	3.03E-17	9.49E-30	8.61E-30
C02	Mean	9.31E-29	9.95E-17	6.79E-30	**6.24E-30**
	Std	7.32E-29	3.43E-17	8.35E-30	7.54E-30
C03	Mean	9.76E+02	6.43E+06	3.93E+05	**8.88E+01**
	Std	3.22E+02	2.72E+06	4.76E+05	3.24E+01
C04	Mean	2.13E+01	1.46E+01	**1.36E+01**	**1.36E+01**
	Std	6.43E+00	1.24E+00	3.63E-15	3.63E-15
C05	Mean	6.40E-28	1.33E-16	**0.00E+00**	5.25E-30
	Std	9.10E-28	7.41E-17	0.00E+00	1.09E-29
C06	Mean	4.09E+02	5.39E+03	4.23E+03	**0.00E+00**
	Std	4.79E+01	8.04E+02	9.25E+02	0.00E+00
C07	Mean	-1.74E+02	-9.93E+01	-9.91E+01	**-1.28E+03**
	Std	5.66E+01	6.43E+01	7.75E+01	3.17E+01
C08	Mean	**-2.84E-04**	-2.52E-04	**-2.84E-04**	**-2.84E-04**
	Std	3.45E-09	3.43E-05	1.11E-19	1.11E-19
C09	Mean	**-2.67E-03**	**-2.67E-03**	**-2.67E-03**	**-2.67E-03**
	Std	8.85E-19	1.62E-09	8.85E-19	8.85E-19
C10	Mean	**-1.03E-04**	-9.78E-05	**-1.03E-04**	**-1.03E-04**
	Std	3.47E-09	4.87E-06	1.38E-20	1.38E-20
C11	Mean	**-3.81E+02**	-8.39E-01	-9.00E-01	-2.65E+02
	Std	3.44E+02	1.24E-01	7.39E-02	2.83E+02
C12	Mean	**3.98E+00**	5.37E+00	**3.98E+00**	**3.98E+00**
	Std	2.86E-04	2.52E+00	1.23E-03	5.85E-05
C13	Mean	1.59E-01	2.27E+01	4.69E+00	**7.93E-28**
	Std	7.97E-01	3.71E+01	4.23E+00	1.65E-27
C14	Mean	1.45E+00	1.92E+00	1.82E+00	**1.41E+00**
	Std	7.09E-02	5.51E-02	7.92E-02	6.80E-16
C15	Mean	2.98E+00	1.27E+01	1.79E+01	**2.36E+00**
	Std	1.81E+00	1.44E+00	3.20E+00	9.06E-16
C16	Mean	**0.00E+00**	1.49E+02	1.50E+02	6.28E+00
	Std	0.00E+00	1.04E+01	9.37E+00	6.45E-08
C17	Mean	1.23E+00	1.01E+00	9.99E-01	**3.09E-01**
	Std	2.46E-01	1.80E-02	1.50E-02	4.40E-01
C18	Mean	3.66E+01	6.06E+03	2.90E+03	**3.65E+01**
	Std	3.39E-01	1.13E+04	4.05E+03	2.25E-03
C19	Mean	**0.00E+00**	**0.00E+00**	6.35E-06	**0.00E+00**
	Std	0.00E+00	0.00E+00	2.88E-07	0.00E+00
C20	Mean	4.44E+00	2.23E+00	**1.45E+00**	1.64E+00
	Std	7.48E-01	2.14E-01	1.13E-01	1.46E-01
C21	Mean	7.65E+00	2.93E+01	2.10E+01	**4.92E+00**
	Std	3.20E+00	1.03E+01	9.33E+00	2.17E+00
C22	Mean	1.43E+02	8.29E+02	1.42E+03	**3.25E+00**
	Std	1.80E+02	1.17E+03	2.10E+03	1.63E+01
C23	Mean	1.43E+00	1.82E+00	1.71E+00	**1.41E+00**
	Std	2.96E-02	6.05E-02	9.28E-02	1.74E-02
C24	Mean	2.98E+00	1.34E+01	1.29E+01	**2.73E+00**
	Std	1.28E+00	1.60E+00	1.54E+00	1.04E+00
C25	Mean	1.90E+01	1.44E+02	1.42E+02	**6.35E+00**
	Std	9.54E+00	1.16E+01	7.55E+00	3.14E-01
C26	Mean	9.24E-01	1.01E+00	1.00E+00	**3.63E-01**
	Std	2.01E-01	8.63E-03	1.49E-02	2.40E-01
C27	Mean	3.79E+01	4.49E+04	1.29E+04	**3.65E+01**
	Std	5.37E+00	4.60E+04	2.21E+04	1.33E-03
C28	Mean	**5.22E+01**	1.33E+02	1.44E+02	5.62E+01
	Std	2.21E+01	3.85E+01	1.44E+02	3.25E+01

REFERENCES

[1] S. Hernndez, G. Leguizamn, E. Mezura-Montes, "Hybridization of differential evolution using hill climbing to solve constrained optimization problems," *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial*, 2013, 16(52): 3-15.

[2] J. Liu, K. L. Teo, X. Wang, et al, "An exact penalty function-based differential search algorithm for constrained global optimization," *Soft Computing*, 2016, 20(4): 1305-1313.

[3] A. Mani, C. Patvardhan, "A novel hybrid constraint handling technique for evolutionary optimization," *IEEE Congress on Evolutionary Computation. IEEE*, 2009: 2577-2583.

[4] K. Deb, "An efficient constraint handling method for genetic algorithms," *Computer methods in applied mechanics and engineering*, 2000, 186(2-4): 311-338.

[5] S. M. Elsayed, R. A. Sarker, D. L. Essam, "GA with a new multi-parent crossover for constrained optimization," *IEEE Congress of Evolutionary Computation (CEC). IEEE*, 2011: 857-864.

[6] Y. Wang, Z. Cai, "A hybrid multi-swarm particle swarm optimization to solve constrained optimization problems," *Frontiers of Computer Science in China*, 2009, 3(1): 38-52.

[7] Y. Wang, B. C. Wang, H. X. Li, et al, "Incorporating objective function information into the feasibility rule for constrained evolutionary optimization," *IEEE Transactions on Cybernetics*, 2015, 46(12): 2938-2952.

[8] T. P. Runarsson, X. Yao, "Stochastic ranking for constrained evolutionary optimization," *IEEE Transactions on evolutionary computation*, 2000, 4(3): 284-294.

[9] T. Takahama, S. Sakai, "Efficient constrained optimization by the ε constrained differential evolution with rough approximation using kernel regression," *IEEE Congress on Evolutionary Computation. IEEE*, 2013: 1334-1341.

[10] Z. Cai, Y. Wang, "A multiobjective optimization-based evolutionary algorithm for constrained optimization," *IEEE Transactions on evolutionary computation*, 2006, 10(6): 658-675.

[11] Y. Wang, Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, 2012, 16(1): 117-134.

[12] L. Jiao, L. Li, R. Shang, et al, "A novel selection evolutionary strategy for constrained optimization," *Information Sciences*, 2013, 239: 122-141.

[13] D. H. Wolpert, W. G. Macready, "No free lunch theorems for optimization," *IEEE transactions on evolutionary computation*, 1997, 1(1): 67-82.

[14] R. Mallipeddi, P. N. Suganthan, "Ensemble of constraint handling techniques," *IEEE Transactions on Evolutionary Computation*, 2010, 14(4): 561-579.

[15] S. M. Elsayed, R. A. Sarker, D. L. Essam, "Integrated strategies differential evolution algorithm with a local search for constrained optimization," *IEEE Congress of Evolutionary Computation (CEC). IEEE*, 2011: 2618-2625.

[16] X. Li, M. Yin, "Self-adaptive constrained artificial bee colony for constrained numerical optimization," *Neural Computing and Applications*, 2014, 24(3-4): 723-734.

[17] Y. Wang, Z. Cai, Y. Zhou, et al, "An adaptive tradeoff model for constrained evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, 2008, 12(1): 80-92.

[18] Y. Wang, Z. Cai, "Constrained evolutionary optimization by means of (μ+λ)-differential evolution and improved adaptive trade-off model," *Evolutionary Computation*, 2011, 19(2): 249-285.

[19] G. Wu, R. Mallipeddi, P. N. Suganthan, "Problem definitions and evaluation criteria for the CEC 2017 competition on constrained real-parameter optimization," *National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report*, 2017.

[20] Z. Fan, Z. Wang, Y. Fang, W. Li, Y. Yuan, X. Bian, "Adaptive Recombination Operator Selection in Push and Pull Search for Solving Constrained Single-Objective Optimization Problems," *International Conference on Bio-Inspired Computing: Theories and Applications*, 2018: 355-367.

[21] J. Tvrdk, R. Polkov, "A simple framework for constrained problems with application of L-SHADE44 and IDE," *IEEE Congress on Evolutionary Computation (CEC). IEEE*, 2017: 1436-1443.

[22] R. Polkov, "L-SHADE with competing strategies applied to constrained optimization," *IEEE congress on evolutionary computation (CEC). IEEE*, 2017: 1683-1689.

[23] R. Tanabe, A. S. Fukunaga, "Improving the search performance of SHADE using linear population size reduction," *IEEE congress on evolutionary computation (CEC). IEEE*, 2014: 1658-1665.

[24] R. Storn, K. Price, "Differential evolution – A simple and efficient adaptive scheme for global optimization over continuous spaces," *Int. Comput. Sci. Inst., Berkeley, CA, USA, Tech. Rep*. TR-95-012, 1995.

[25] Z. Fan, W. Li, X. Cai, et al, "Push and pull search for solving constrained multi-objective optimization problems," *Swarm and evolutionary computation*, 2019, 44: 665-679.

[26] Takahama. T, Sakai. S, "Constrained optimization by the ε constrained differential evolution with an archive and gradient-based mutation," *IEEE Congress of Evolutionary Computation (CEC). IEEE*, 2010: 1-9.