Differential Evolution Algorithm Based on Ensemble of Constraint Handling Techniques and Multi-Population Framework

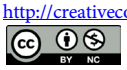
Yanting Wei[1,2], Quanxi Feng[1,2*], Sainan Yuan[1]

[1]Center for Data Analysis and Algorithm Technology, Guilin University of Technology, Guilin, China
[2]College of Science, Guilin University of Technology, Guilin, China
Email: *fqx9904@163.com

How to cite this paper: Wei, Y.T., Feng, Q.X. and Yuan, S.N. (2020) Differential Evolution Algorithm Based on Ensemble of Constraint Handling Techniques and Multi-Population Framework. *International Journal of Intelligence Science*, **10**, 22-40.
https://doi.org/10.4236/ijis.2020.102003

Received: March 20, 2020
Accepted: April 19, 2020
Published: April 22, 2020

Copyright © 2020 by author(s) and Scientific Research Publishing Inc.
This work is licensed under the Creative Commons Attribution-NonCommercial International License (CC BY-NC 4.0).
http://creativecommons.org/licenses/by-nc/4.0/



Open Access

Abstract

Aimed at improving the insufficient search ability of constraint differential evolution with single constraint handling technique when solving complex optimization problem, this paper proposes a constraint differential evolution algorithm based on ensemble of constraint handling techniques and multi-population framework, called ECMPDE. First, handling three improved variants of differential evolution algorithms are dynamically matched with two constraint handling techniques through the constraint allocation mechanism. Each combination includes three variants with corresponding constraint handling technique and these combinations are in the set. Second, the population is divided into three smaller subpopulations and one larger reward subpopulation. Then a combination with three constraint algorithms is randomly selected from the set, and the three constraint algorithms are run in three sub-populations respectively. According to the improvement of fitness value, the optimal constraint algorithm is selected to run on the reward sub-population, which can share information and close cooperation among populations. In order to verify the effectiveness of the proposed algorithm, 12 standard constraint optimization problems and 10 engineering constraint optimization problems are tested. The experimental results show that ECMPDE is an effective algorithm for solving constraint optimization problems.

Keywords

Constraint Optimization, Differential Evolution Algorithm, Multi-Population, ε Constraint Handling Technique

1. Introduction

Most of the decision-making, engineering application, and control problems can

be summed up as optimization problems, which are widely used in real life or production, such as engineering design [1], intelligent control [2], job scheduling [3], traffic optimization [4], network communication [5], financial investment [6], and so on. These problems can be clarified into two categories: unconstraint optimization problem and constraint optimization problem. In comparison, the constraint of the constraint optimization problem leads to the decrease of the feasible domain in the search space, which leads that constraint optimization problem is more complex than the unconstraint problem. Moreover, the optimal solution often locates at the boundary of the feasible domain, so it is necessary to balance the constraints and optimization objectives, which increases the difficulty of finding the optimal solution. Therefore, finding solution to constraint optimization problems, the handling of constraints is always a difficult problem.

The commonly used constraint handling techniques include penalty function method [7], feasibility rule method [8], random sorting method [9], ε-constraint method [10] and ensemble constraint handling techniques [11]. In 2010, Mallipeddi [11] proposed an (the ensemble of constraint handling techniques using the DE, ECHT-DE) algorithm based on multi-population strategy by using ensemble of constraint handling techniques, such as feasibility rules, penalty function method, constraint handling technique, and random sorting method. ECHT-DE can deal with complex optimization problems through mutual learning among populations. In 2011, Elsayed [12] proposed a constraint differential evolution algorithm by randomly combining four differential evolution operations with different mutation operations, two different recombination operations, and two constraint handling techniques (the feasibility rule and ε-constraint handling method), and then formulated 16 different strategies. In the process of evolution, each individual can choose a strategy randomly. The probability of choosing the strategy is determined by its ability of the individual improvement. In 2017, Wang Yong [13] proposed a method to integrate the objective function information into the feasibility rule (the feasible rule with the incorporation of objective function information, FROFI) to achieve an effective balance between the objective function and the constraints. In this method, all the experimental individuals with good objective function are archived and saved, and then the offspring in the archive are replaced by some of the individuals in the new population into the next generation population according to the objective function.

Differential Evolution (DE) [14] is a stochastic global search algorithm first proposed by R. Storn and K. Price. DE can effective solve nonlinear, high-dimensional and complex optimization problems. In 2006, Brest J. [15] proposed a jDE algorithm by adjusting control parameters F and Cr, in which F and Cr mutate randomly with a small probability ζ in each generation. In 2009, J. Zhang [16] proposed a JADE algorithm, which uses a learning scheme to adjust the values of F and Cr. In 2011, Wang Yong [17] proposed a composite differential evolution algorithm (CoDE), for unconstraint optimization problem, which uses three mutation strategies "DE/rand/1", "DE/rand/1", and "DE/current to rand/1" and three

groups of F and Cr values randomly to produce three experimental individuals, from which the best individual and the target individual are selected for one-to-one greedy selection. In 2011, Mallipeddi [18] proposed a combined experimental vector generation strategy and control parameter (EPSDE), EPSDE uses different characteristics of variation strategy pool and related parameter value pool, which can compete to produce offspring on the basis of the success of previous generations. In 2018, Guohua Wu [19] proposed a differential evolution algorithm (ensemble of differential evolution variants, EDEV) based on multi-population framework to solve unconstraint optimization problems. The basic idea of this algorithm is to use a framework to integrate three different DE variants with their own characteristics.

Many scholars introduce differential evolution algorithm into constraint handling technique to solve constraint optimization problems. In 2012, Wang and Cai [20] proposed a hybrid framework (DyHF), which is composed of global model and local model dynamically. According to the proportion of feasible solutions in the current population, DyHF spontaneously implements the global search model or the local search model to solve the constraint problem. In 2018, based on CoDE, Wang [21] proposed a constraint differential evolution algorithm (CCoDE), which can deal with constraint optimization problems.

According to the principle of free lunch [22], different constraint optimization problems have different characteristics. Thus, a single differential evolution strategy or a single constraint handlinghandling technique is difficult to solve all constraint optimization problems at the same time. Therefore, in order to improve the performance of constraint optimization algorithm, a constraint differential evolution algorithm based on ensemble of constraint handling techniques and multi-population framework is proposed in this paper. The algorithm adopts three complementary differential evolution algorithms, JADE, jDE and EPSDE, and two effective constraint handlinghandling techniques. Each DE variant has a subpopulation, and two constraint handling mechanisms are assigned to each DE variant through the constraint allocation mechanism. The algorithm is combined in three smaller subpopulations, and then the one with better performance is selected to run in the larger reward subpopulation according to the fitness value. Finally, the population is updated by the combined constraint method.

The structure of this paper is organized as follows. Section 2 introduces the preliminary knowledge. Section 3 introduces the proposed method in detail, including its constraint allocation mechanism, adaptive value calculation and multi-population framework. Extensive experiments and discussions are carried out in Section 4. Section 5 concludes this paper.

2. Preliminary Knowledge

A) Constraint optimization problems

Without loss of generality, a constraint optimization problems (minimization problem) can be defined as:

$$\begin{aligned} &\min \quad f(\boldsymbol{x}),\ \boldsymbol{x}=(x_1,\cdots,x_D)\in S \\ &\text{s.t.} \quad g_j(\boldsymbol{x})\le 0,\ j=1,2,\cdots,m; \\ &\qquad h_j(\boldsymbol{x})=0,\ j=m+1,m+2,\cdots,n; \end{aligned} \tag{1}$$

where $f(\boldsymbol{x})$ is the objective function, $\boldsymbol{x}=(x_1,\cdots,x_D)\in\Omega\subseteq S$ represents the D-dimension decision vector . The feasible region Ω is restricted by n linear or nonlinear constraints:

$$\Omega=\left\{\boldsymbol{x}\in S\middle| g_j(\boldsymbol{x})\le 0,\ j=1,\cdots,m; h_j(\boldsymbol{x})=0,\ j=m+1,\cdots,n\right\} \tag{2}$$

where S is decision space with upper boundary L_i and lower boundary U_i:

$$L_i\le x_i\le U_i,\quad 1\le i\le D \tag{3}$$

$g_j(X)(j=1,2,\cdots,m)$ and $h_j(X)(j=m+1,\cdots,n)$ are the jth inequality constraint and the n-jth equality constraint, respectively.

For COPs, the total constraint violation degree of the solution $\boldsymbol{x}$ is defined as:

$$G(\boldsymbol{x})=\sum_{j=1}^{n}G_j(\boldsymbol{x}) \tag{4}$$

where $G_j(\boldsymbol{x})$ is the degree of constraint violation on the jth constraint, it can be expressed as follows:

$$G_j(X)=\begin{cases}\max\left\{g_j(X),0\right\}, & 1\le j\le m\\ \max\left\{\left|h_j(X)-\delta\right|,0\right\}, & m+1\le j\le n\end{cases} \tag{5}$$

in (5), δ is a positive tolerance value in equality constraints and usually set to 0.0001. The solution $\boldsymbol{x}$ is a feasible solution when $G(\boldsymbol{x})=0$.

B) Constraint handling techniques

1) Feasibility rule method

Feasibility Rule [8], first proposed by Deb, is the most common used constraint handling technique. For each comparison of paired individuals, it first compares the merits and weaknesses of individuals based on whether the individuals are feasible or not. If all of them are feasible, the individuals with small objective function value are better; if none of them are feasible, the individuals with small constraint violation degree are better. For the two individuals, the feasibility rule is used to compare as follows:

a) If $G(X_1)<G(X_2)$, X_1 is better than X_2.

b) When $G(X_1)=G(X_2)=0$, if $f(X_1)<f(X_2)$, X_1 is better than X_2.

c) When $G(X_1)>0$ and $G(X_2)>0$, if $G(X_1)<G(X_2)$, X_1 is better than X_2.

In the feasibility rule, in the first stage, the infeasible solution with low violation of the overall constraint is selected. In the second stage, we first select all the feasible solutions, and then select the infeasible solutions with low degree of overall constraint default. In the third stage, only the feasible scheme with the best target value is selected. The above criteria (1) and criterion (3) are for constraints, the two infeasible solutions push the infeasible solutions to the feasible region

through the comparison of the total constraint violation degree, while the criterion (2) is aimed at the objective. The comparison of the two feasible solutions on the target value improves the global solution.

Feasible rules were first used in genetic algorithms by Ded. Mezura-Montes [23] used feasible rules in an improved differential evolution algorithm. In the algorithm, multiple experimental vectors can be generated from one objective vector at the same time, which improves the probability of generating excellent solutions. At the same time, when using feasible rules, certain probability is given to allow defaulting individuals with better objective function values to enter the population, which increases the diversity of the population. Huang [24] regards SaDE as the search algorithm and a feasible rule as the individual update condition, and uses the sequential quadratic programming method to enhance the local search ability of the algorithm.

2) ε-constraint handling method

The ε-constraint handling method [10] was first proposed by Takahama and Sakai, which can be regarded as an improvement of the feasible rule method. Because the feasible individual in the feasible rule is always better than the infeasible individual, it is easy to lose some excellent infeasible individuals, which makes the population fall into the local optimal solution. The infeasible individuals are relaxed by the constraint treatment method, so that the infeasible individuals with smaller violation of constraints and better objective function value also have a chance to stay in the population. The ε-constraint handling method is used in this paper to deal with constraints. For the solution x_i and x_j, x_i is preference to x_j if both of them meet the following conditions:

$$\begin{cases} f(x_i) < f(x_j), & \text{if } G(x_j) \le \varepsilon \wedge G(x_j) \le \varepsilon; \\ f(x_i) < f(x_j), & \text{if } G(x_i) = G(x_j); \\ G(x_i) < G(x_j), & \text{otherwise.} \end{cases} \tag{6}$$

where ε is decreaing as its iterative generation grows, and its calculation formula is as follows:

$$\varepsilon = \begin{cases} \varepsilon_0 \left(1 - \dfrac{t}{T}\right)^{cp}, & \text{if } \dfrac{t}{T} \le p; \\ 0, & \text{otherwise.} \end{cases} \tag{7}$$

$$cp = -\frac{\log \varepsilon_0 + \lambda}{\log(1-p)} \tag{8}$$

where ε represents intialized as $\varepsilon_0 = V(X_{0.2*PS})$, if $\varepsilon = 0$, ε-constraint handling method is feasible rule method. $X_{0.2*PS}$ is the top 0.2 * PS th individual according to vilation degree, and λ is 6. The control parameter ranges are $T \in [0.1 * T_{\max}, 0.8 * T_{\max}]$ and $cp \in [2,10]$ representive.

Takahama [25] used the "DE/rand1/exp" strategy as the basic algorithm and the mutation operator based on gradient information as the local search algorithm. When the newly generated experimental individual is an infeasible indi-

vidual, the mutation based on gradient information is used to make it a feasible solution. The algorithm has achieved good results in the test function of CEC2006 [26] and won the first place in the competition of CEC2006 constraint evolutionary algorithm. However, the algorithm increases the computational complexity because of the need to calculate the gradient information. Subsequently, Takahama [27] proposed an improved parameter setting method, which can effectively improve the performance of the algorithm when the default value decreases fast enough in evolution, which can effectively improve the performance of the algorithm when dealing with equality constraints. Takahama [28] proposed a method of decreasing probability to use the mutation operator based on gradient information, and cut off and map individuals outside the search space to deal with the case that the optimal solution is at the constraint boundary. After that, Takahama [29] proposed a mutation strategy based on archiving and gradient information. The algorithm uses archiving mechanism to enhance population diversity and proposes a new mechanism to control parameters. The algorithm was tested on the CEC2010 [30] test function and won the first place in the competition of CEC2010 constraint evolutionary algorithm.

C) Differential evolution

Differential evolution algorithm is a real-coded parallel search algorithm. Its initial iteration mainly includes four operation steps: initialization, mutation, crossover and selection. After initializing the population, the algorithm generates a new generation of population through mutation, crossover and selection to guide the search process to approach the global optimal solution.

Firstly, an initial population containing NP objective vectors (NP individuals) is randomly generated in the decision space. After G generation, $X_{i,G} = \left[x_{i,1}^G, x_{i,2}^G, \cdots, x_{i,D}^G \right], i = 1, 2, \cdots, NP$ is expressed as the i vector, where D represents the dimension of the optimization problem to be solved. Let V denote the mutant vector, and the six commonly used mutation operators are enumerated as follows:

$$\text{DE}|rand|1 \text{ [14]}: \ V = X_{r1} + F * \left(X_{r2} - X_{r3} \right) \tag{9}$$

$$DE|rand|2 \text{ [29]}: \ V = X_{r1} + F * \left(X_{r2} - X_{r3} \right) + F * \left(X_{r4} - X_{r5} \right) \tag{10}$$

$$DE|best|1 \text{ [14]}: \ V = X_{best} + F * \left(X_{r1} - X_{r2} \right) \tag{11}$$

$$DE|best|2 \text{ [30]}: \ V = X_{best} + F * \left(X_{r1} - X_{r2} \right) + F * \left(X_{r3} - X_{r4} \right) \tag{12}$$

$$DE|current\text{-}to\text{-}best|1 \text{ [16]}: \ V = X_{current} + F * \left(X_{best} - X_{r1} \right) + F * \left(X_{r2} - X_{r3} \right) \tag{13}$$

$$DE|current\text{-}to\text{-}pbest|1 \text{ [31]}: \ V = X_{current} + F * \left(X_{pbest} - X_{current} \right) + F * \left(X_{r1} - X_{r2} \right) \tag{14}$$

where F is a scaling factor with values between 0 and 1. where r_1, r_2, r_3, r_4 and r_5 are mutually different integers randomly chosen from [1, NP], "*best*" is the individual with the best fitness in the population, "*current*" represents the best individual in the current population, and "*pbest*" represents p individuals with the best fitness in the population.

In the crossover stage, a crossover operator is conducted on each pair of $X_{i,d}$ and $V_{i,d}$ to produce a trial vector. This operation can further enhance the population diversity. The commonly used binomial crossover operators are defined as:

$$U_{i,d} = \begin{cases} V_{i,d}, & \text{if } rand(0,1) < Cr \text{ or } j = j_{rand}; \\ X_{i,d}, & \text{otherwise.} \end{cases} \tag{15}$$

where $j_{rand} \in \{1,2,3,\cdots,D\}$ represents a random integer uniformly generated between 1 and D, The cross strength is controlled by a parameter $Cr \in [0,1]$.

When the trial vector is generated, the selection operator will select the best individual as $t+1$ generation in the pairwise comparison between $U_{i,d}$ and $X_{i,d}$:

$$X_{i,d}^{t+1} = \begin{cases} U_{i,d}^{t}, & \text{if } f\left(U_{i,d}^{t}\right) \le f\left(X_{i,d}^{t}\right); \\ X_{i,d}^{t}, & \text{otherwise.} \end{cases} \tag{16}$$

JADE is a simple and efficient variant of DE. In JADE, a new "current-to-pbest/1" mutation strategy is adopted as follows:

$$V = X_{current} + F * \left(X_{pbest} - X_{r1}\right) + F * \left(X_{r2} - \tilde{X}_{r3}\right) \tag{17}$$

JADE's mutation strategy "current-to-pbest/1" is incorporated with an archive A which contains some recently declared inferior solutions. Let P denote the current population, and $\tilde{X}_{r3}$ is from the union of A and P.

The algorithm refers to a complex parameter adaptation mechanism. The parameters F and Cr are generated by the normal distribution and Cauchy distribution of each objective vector, respectively. Better control parameter values are passed to subsequent generation by recording promising F and Cr values and using them for new parameter generation.

CoDE proposed by Wang *et al.* is a DE variant with compound trial vector generation strategy and control parameters. These strategies and parameters have obvious advantages, and it can be regarded as a low-level integrated DE of variation strategy and control parameters. Three kinds of trial vector generation strategies namely " rand/1/bin", "rand/2/bin" and "current-rand/1" and three combinations of control parameters are adopted in CoDE, i.e. [F = 1.0, Cr = 0.9], [F = 1.0, Cr = 0.1] and [F = 0.8, Cr = 0.2]. In each generation, the experimental vector generation strategy and control parameter settings are randomly selected from the policy candidate pool and the parameter candidate pool, respectively.

EPSDE, originally designed by Mallipeddi *et al.*, is a popular DE variant that integrates mutation strategies and parameters. In EPSDE, the variation strategy pool and parameter pool are constructed respectively. "DE/best/2/bin", "DE/rand/1/bin" and "DE/current-to-rand/1/bin" are included in the policy pool. The pool of CR values is taken in the range 0.1 - 0.9 in steps of 0.1, and the pool of F values is taken in the range 0.4 - 0.9 in steps of 0.1. Each individual in the initial population is a signed variation strategy and parameter values randomly selected from

each pool. Variation strategies and parameter values that produce better offspring survive, while those that do not produce better offspring are reinitialized.

Experimental research shows that JADE can effectively solve unimodal optimization problems, CoDE shows good performance in solving various optimization problems, especially in dealing with some multimodal optimization problems, while EPSDE is particularly effective in solving some highly complex problems.

3. The Proposed Method

In the process of global searching of the constraint optimization problem, the population is generally clarified into three cases: 1) there are infeasible solutions in the population, and the main purpose is to find the feasible region and increase the proportion of feasible solutions; 2) there are partial feasible solutions in the population. When solving the problem, we not only make full use of the information of the feasible solution, but also consider the information of the infeasible solution to enhance the diversity of the solution; 3) when all the solutions are feasible, the optimal solution is found in the feasible region.

Different constraint handling methods can be used handling in different situations of the search process. How to select different algorithms and handling constraint handling techniques according to different solving stages of constraint optimization problems plays an important role in the search process. Based on the above consideration, this paper proposes a constraint differential evolution algorithm based on ensemble of constraint handling techniques and multi-population framework, which is denoted as ECMPDE. The algorithm uses three complementary DE variants, namely JADE, jDE and EPSDE, and combines these DE variants with two constraint handlinghandling techniques one-to-one. In the three combinations, the computing resources are allocated to the optimal collocation by improving the proportion of their respective fitness values. First, the population is divided into three small-scale equal subpopulations and larger reward subpopulations, and a subpopulation is randomly assigned to the three DE variants. Second, the constraint handling technique is combined with DE variant by the constraint allocation mechanism, and the combined constraint handling mechanism is used to compare the target vector with the experimental vector to generate a new population and calculate the improved value of the fitness value. Third, the best combination is determined according to the cumulative improved fitness value and the proportion of cumulative consumption, and then the reward subpopulation is assigned to the combination; finally, when running to the next generation, if the optimal collocation of the previous generation is smaller than the solution obtained by the current optimal collocation, it will be added to the constraint mechanism pool to increase its selection probability.

A) Constraint allocation mechanism

Different problems have different characteristics. According to the theory of free lunch, single constraint handling technique can easily lead to weak performance in solving some problems. In this paper, feasibility rules and constraint

handling methods are put into the constraint mechanism pool, and a constraint optimization algorithm allocation mechanism is designed. In the constraint handling mechanism, there are 8 combinations according to the combination principle.

Suppose the feasibility rule is A, the constraint handling method is B, and the combined pool P is as follows:

$$\text{the combined pool } P=\begin{cases} P_1=\left\{pop_1\left(A_{JADE}\right),pop_2\left(A_{jDE}\right),pop_3\left(A_{EPSDE}\right)\right\} \\ P_2=\left\{pop_1\left(A_{JADE}\right),pop_2\left(B_{jDE}\right),pop_3\left(A_{EPSDE}\right)\right\} \\ P_3=\left\{pop_1\left(A_{JADE}\right),pop_2\left(A_{jDE}\right),pop_3\left(B_{EPSDE}\right)\right\} \\ P_4=\left\{pop_1\left(A_{JADE}\right),pop_2\left(B_{jDE}\right),pop_3\left(B_{EPSDE}\right)\right\} \\ P_5=\left\{pop_1\left(B_{JADE}\right),pop_2\left(A_{jDE}\right),pop_3\left(A_{EPSDE}\right)\right\} \\ P_6=\left\{pop_1\left(B_{JADE}\right),pop_2\left(A_{jDE}\right),pop_3\left(B_{EPSDE}\right)\right\} \\ P_7=\left\{pop_1\left(B_{JADE}\right),pop_2\left(B_{jDE}\right),pop_3\left(A_{EPSDE}\right)\right\} \\ P_8=\left\{pop_1\left(B_{JADE}\right),pop_2\left(B_{jDE}\right),pop_3\left(B_{EPSDE}\right)\right\} \end{cases} \tag{18}$$

When each generation triggers the constraint allocation mechanism, a collocation is randomly selected from the constraint mechanism pool and assigned to the corresponding three DE variants, each with a population, for example, Suppose P_2 is randomly selected, JADE algorithm and feasibility rules are run in pop_1, jDE algorithm and constraint handling methods are run in pop_2, and EPSDE algorithm and feasibility rules are run in pop_3.

B) Fitness calculation

In 2009 [32], Biruk Tessema and Gary G. Yen proposed a fitness value calculation method based on standardized constraint violations and standardized objective function values. This method uses the Feasible Ratio (FR) as a penalty. Here, the feasibility rate is the ratio of the number of feasible individuals to the population size in the t-generation operation of the algorithm. The calculation method of its fitness value is as follows:

$$FF\left(\boldsymbol{x}_i\right)=\begin{cases} f_{norm}\left(\boldsymbol{x}_i\right), & \text{if } RF=1; \\ G_{norm}\left(\boldsymbol{x}_i\right), & \text{if } RF=0; \\ \sqrt{f_{norm}\left(\boldsymbol{x}_i\right)^2+G_{norm}\left(\boldsymbol{x}_i\right)^2}, & \text{otherwise.} \end{cases} \tag{19}$$

In the semi-feasible case, the population contains both unfeasible individuals and feasible individuals. For constraint optimization, how to deal with the unfeasible individuals in the population is a very important problem. Since some information carried by some infeasible individuals may be very important to find the optimal solution, it is unreasonable to exclude all infeasible individuals in the semi-feasible case.

C) Multi-population based ensemble framework

In this paper, similar to EDEV [19], the entire population is divided into three

subpopulations and one reward subpopulation. Population division is triggered at every generation. The three subpopulations are represented by pop_1, pop_2 and pop_3, respectively, while the reward subpopulations are represented by pop_4. The three subpopulations have the same size, and the size of the subpopulation is much smaller than that of the reward subpopulation. Let pop be the overall population. We have

$$pop = \bigcup_{i=1,2,3,4} pop_i \tag{20}$$

Suppose that NP is the size of the population and NP_i is the size of pop_i. λ_i denotes the proportion between subpopulations and population. Thus we have

$$NP_i = \lambda_i \cdot NP \tag{21}$$

$$\sum_{i=1,2,3,4} \lambda_i = 1 \tag{22}$$

Here we just let $\lambda_1 = \lambda_2 = \lambda_3$, Each subpopulation is randomly assigned to a combination of DE variants and constraint handling techniques, while the rewarded subpopulation is assigned to the combination with the best results of the three populations. A population division operation is performed in each generation. With the progress of the algorithm, we determine the most effective DE variant (i_{best}) in the last period of time according to the ratio between the cumulative fitness improvement and the function evaluation of consumption after each generation.

$$\Delta f_i = \sum \left(FF_t - FF_{t+1} \right) \tag{23}$$

$$i_{best} = \arg\max_{i=1,2,3} \left(\frac{\Delta f_i}{\Delta fes_i} \right) \tag{24}$$

where Δf_i represents the accumulated function fitness improvements during the last ng generations attributed by the ith constituent DE variant and Δfes_i is the consumed number of function evaluations. $\arg\max_{i=1,2,3}$ represents the maximum value selected in $\Delta f_i / \Delta fes_i, (i = 1,2,3)$.

The population division used in this article is different from ECHDE and EDEV. In ECHDE, the population is divided into four populations of the same size. The four constraint handling mechanisms each have a population. In EDEV, the population is divided into three equal subpopulations and a larger reward subpopulation, which is given to the best DE variants after accumulating a certain algebra. The best combination is selected directly from the three combinations and then given to the reward subpopulation to run.

D) Algorithm framework

In this paper, a new constraint differential evolution algorithm is proposed through the methods of multi-population, multi-DE variants and ensemble of constraint handling techniques. The ECMPDE framework is as follows (Algorithm 1):

The ECMPDE framework is as follows (Algorithm 1):

STEP 1: Initial parameters of ECMPDE including *ng*, *NP*, λ_i, *MaxG* and *MaxFes*; Initialize *pop*; $X_{i,d} = L_d + rand(0,1) * (U_d - L_d)$ (25);

STPE 2: Evaluate the objective function value and overall constraint violation value of each individual;

STPE 3: Initial the parameters for JADE, jDE and EPSDE; Set $\Delta f_i = 0$ and $\Delta fes_i = 0 (i = 1,2,3)$;

STPE 4: Set $NP_i = \lambda_i \cdot NP$, Randomly assigned based on population size pop_1, pop_2, pop_3 and pop_4;

STPE 5: A combination P_i is randomly selected from the combination pool;

STPE 6: Execute JADE on pop_1, pop_1 is selected and updated by $pop_1(S), S \in \{A, B\}$ and calculated in (23) Δf_1;

STPE 7: Execute jDE on pop_2, pop_2 is selected and updated by $pop_2(S), S \in \{A, B\}$ and calculated in (23) Δf_2;

STPE 8: Execute EPSDE on pop_3, pop_3 is selected and updated by $pop_3(S), S \in \{A, B\}$ and calculated in (23) Δf_3;

STPE 9: The best combination of the three populations is selected by (24), it will run to the reward population pop_4, and calculated (23) and saved Δf_4;

STPE 10: By comparing the optimal solution of the optimal combination of the previous generation with that of the current generation, the successful combination will be preserved;

STPE 11: Combine updated pop_1, pop_2, pop_3 and pop_4, i.e., $pop = \bigcup_{i=1,2,3,4} pop_i$;

STPE 12: Randomly assigned based on population size pop_1, pop_2, pop_3 and pop_4;

STPE 13: Stop if the termination condition is met. If not, $k = k + 1$, proceed to step 5.

4. Experimental Study

In this paper, ECMPDE is compared with CCoDE [20] ECHTEP [12], FROFI [13], DyHF [19], jDE [15], JADE [16] and other constraint differential evolution algorithms on standard constraint optimization problems and engineering constraint optimization problems to verify the effectiveness of the algorithm. Twelve standard constraint optimization problems (g01 - g012) of CEC2006 [26] are selected as the test problems of the algorithm. 10 engineering constraint optimization problems [29] included tension-compression spring design, welding beam design, pressure vessel design, reducer design, three-bar truss design, hydrodynamic thrust bearing design, conical wheel design, rolling bearing design, butterfly spring design, gear train design.

In this paper, the population size of the ECMPDE algorithm is set as 100 for the experiment, and the parameters in the variant are set as shown in **Table 1** below. For each problem, the algorithm runs 30 times independently. According to the results of 30 runs, the best value (MinBest), average (MinMean), middle value (MinMedian), worst difference (MinWorst) and standard deviation (Std) are used to evaluate the performance indicators. According to the results obtained for statistical analysis, this paper uses three statistical analysis methods, including the Mann-Whitney rank-sum test, Iman-Davenport test, and Wilcoxon signed-rank test.

Table 1. Parameter configuration of ECMPDE.

Algorithm	parameter settings
ECMPDE	$\lambda_1 = \lambda_2 = \lambda_3 = 0.1, \lambda_4 = 0.7, ng = 20, NP = 100$
JADE	$p = 0.05, c = 0.1, NP = 100$
jDE	$\tau_1 = \tau_2 = 0.1, F_l = 0.1, F_u = 0.9, NP = 100$
EPSDE	$CR \in [0.1, 0.9], F \in [0.4, 0.9], NP = 100$

A) Result Analysis of Standard COPs

The MaxFEs of 12 standard test problems is set as 500,000. The population size of CCoDE, ECHTEP, FROFI, DyHF, jDE and JADE was set as 100. The test results of ECMPDE algorithm and six comparison algorithms are shown in **Table 2**. **Table 2** uses the mann-whitney rank sum test to compare the degree of difference between different algorithms in the same problem. The significance level of the mann-whitney rank sum test in the algorithm is set as 0.05, and NAN means that the corresponding optimal value is not found. The fonts with skewed and bold numerical results, conventional, bold marks, bold and gray shading marks in the cell represent the solution results of ECMPDE algorithm, the solution results of ECMPDE algorithm are good, similar to the results of ECMPDE, and the solution results of ECMPDE algorithm are poor. The comparison results are shown in **Table 3**, where "+", "≈" and "−" respectively represent the number of problems whose numerical results of ECMPDE are better than those of the comparison algorithm, similar to those of the comparison algorithm, and worse than those of the comparison algorithm.

As can be seen from **Table 2**, except for JADE,ECMPDE, CCoDE, ECHTDE, FROFI, DyHF and jDE algorithms, all the feasible solutions of all problems can be obtained at 100%. The numerical results of G01, G02, G04, G05, G6, G8, G9, G10, G11 and G12 by ECMPDE are all better or better than those of CCoDE, ECHTDE, FROFI, DyHF, jDE and JADE. It is worth noting that the function G02 mainly examines the search ability of evolutionary algorithms, and the numerical results of ECMPDE are better than CCoDE, ECHTEP, FROFI, jDE and DyHF, indicating that the search ability of the algorithm is stronger in the framework of DE variants and multi-constraint mechanisms. For function g03, the numerical results of this algorithm are better than those of jDE and JADE, and slightly worse than those of CCoDE, ECHTDE, FROFI and DyHF. For the function g07, the numerical results of this algorithm are better than those of ECHTDE, jDE and JADE, but slightly worse than those of CCoDE, FROFI and DyHF. From the comparison of the results of Mann-Whitney rank sum test in **Table 3**, we can see that compared with CCoDE, ECHTDE, FROFI, DyHF and jDE algorithms, the number of numerical results won by ECMPDE is 3, 3, 6, 3, 6 and 9 respectively, the number of similar numerical results is 7, 7, 5, 7, 6 and 3, and the number of differences in numerical results is 3, 2, 1, 3, 0 and 0, respectively. Then the average value and the best value are tested by Iman-Davenport test to further compare the differences of all algorithms in different test problems. The significance level of the test algorithm is set to 0.05, and the test results are shown in **Table 3**.

Table 2. Experimental results of standard COPs.

Algorithm	statistics	G01	G02	G03	G04	G05	G06
ECMPDE	**Minbest**	**−15**	**−0.80362**	**−0.95559**	**−30665.5**	**5126.497**	**−6961.81**
	Minmean	**−15**	**−0.80325**	**−0.77313**	**−30665.5**	**5126.497**	**−6961.81**
	Std	**0**	**0.000976**	**0.00179**	**1.09E−11**	**9.09E−13**	**1.82E−12**
CCoDE	Minbest	**−15**	−0.80362	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Minmean	**−15**	−0.80181	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Std	**0**	0.005241	**2.47E−16**	**1.09E−11**	**9.09E−13**	**1.82E−12**
DyHF	Minbest	**−15**	−0.80362	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Minmean	**−15**	−0.80314	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Std	**0**	0.001852	**5.32E−16**	**1.09E−11**	**9.09E−13**	**1.82E−12**
ECHTDE	Minbest	−15	−0.80362	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Minmean	−15	−0.80185	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Std	2.47E−07	0.003963	**5.75E−16**	**1.09E−11**	**9.09E−13**	**1.82E−12**
FROFI	Minbest	**−15**	−0.80362	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Minmean	**−15**	−0.8031	**−1.0005**	**−30665.5**	**5126.497**	**−6961.81**
	Std	**0**	0.002783	**4.78E−16**	**1.09E−11**	**9.09E−13**	**1.82E−12**
jDE	Minbest	**−15**	−0.80362	−0.7137	**−30665.5**	5126.497	**−6961.81**
	Minmean	**−15**	−0.80252	−0.51277	**−30665.5**	5162.889	**−6961.81**
	Std	**0**	0.003303	0.096372	**1.09E−11**	61.37129	**1.82E−12**
JADE	Minbest	−15	−0.78594	−0.78462	−30665.5	65535	**−6961.81**
	Minmean	−15	−0.76934	−0.49245	−30665.5	65535	**−6961.81**
	Std	2.59E−09	0.006689	0.106397	0.012892	NAN	**1.82E−12**

Algorithm	statistics	G07	G08	G09	G10	G11	G12
ECMPDE	**Minbest**	**24.30621**	**−0.09583**	**680.6301**	**7049.248**	**0.7499**	**−1**
	Minmean	**24.30621**	**−0.09583**	**680.6301**	**7049.248**	**0.7499**	**−1**
	Std	**1.23E−14**	**2.09E−17**	**3.09E−13**	**2.12E−14**	**1.11E−16**	**0**
CCoDE	Minbest	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Minmean	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Std	**6.83E−15**	**2.78E−17**	4.06E−13	4.55E−12	**1.11E−16**	**0**
DyHF	Minbest	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Minmean	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Std	**5.84E−15**	**2.78E−17**	3.41E−13	4.49E−12	**1.11E−16**	**0**
ECHTDE	Minbest	24.30621	−0.09583	680.6301	7049.248	**0.7499**	**−1**
	Minmean	24.30621	−0.09583	680.6301	7049.248	**0.7499**	**−1**
	Std	8.45E−12	2.83E−17	3.98E−13	1.17E−11	**1.11E−16**	**0**
FROFI	Minbest	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Minmean	**24.30621**	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Std	**7.81E−15**	**2.78E−17**	4.53E−13	4.12E−12	**1.11E−16**	**0**
jDE	Minbest	24.30621	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Minmean	24.30621	**−0.09583**	680.6301	7049.248	**0.7499**	**−1**
	Std	6.14E−07	**2.78E−17**	4.24E−13	6.64E−06	**1.11E−16**	**0**
JADE	Minbest	24.44074	**−0.09583**	680.6731	7095.689	0.749901	**−1**
	Minmean	24.8193	**−0.09583**	680.8112	7160.225	0.749929	**−1**
	Std	0.256666	**2.78E−17**	0.082211	29.72393	3.01E−05	**0**

Table 3. Statistical test results of standard COPs.

Statistic Test	Symbol	ECMPDE	CCoDE	DyHF	ECHTDE	FROFI	jDE	JADE
Mann-Whitney	Win	+	3	3	6	3	6	9
	Tied	≈	7	7	5	7	6	3
	Lose	−	2	2	1	2	0	0
Iman-Davenport	MinMean's Rank	**2.956**	3.47	3.01	3.89	3.41	4.32	4.52
	MinBest's Rank	**2.81**	3.02	2.98	3.57	3.33	4.12	4.25

From the Iman-Davenport test in Table 3, it can be seen that for the average value, ECMPDE ranks first, which is similar to that of DyHF, while DyHF ranks second, while for the best value, ECMPDE has the smallest rank, which is 2.81. In this part of the comparison, the performance of ECMPDE algorithm is relatively equal to that of DyHF algorithm, and both are better than the other four algorithms.

B) Comparisons of Engineering COPs

In this paper, ECMPDE algorithm and CCoDE, ECHTEP, FROFI, DyHF, jDE and other five algorithms are compared and tested on 10 engineering constraint optimization problems to further verify the effectiveness of ECMPDE algorithm in practical problems. The group size setting of CCoDE, ECHTEP, FROFI, DyHF, jDE algorithm is the same as that of ECMPDE algorithm, which is set to 100. In order to facilitate the comparison of the results, the maximum evaluation times of all algorithms for solving engineering optimization problems MaxFEs is set to 500,000 times, the number of runs is 30 times, and the parameter setting is the same as that of IV. The test results are shown in Table 4, and the marks of the data in the table are the same as those described in IV-A. This time, the statistical test is the same as that of IV-A, and the Wilcoxon symbolic rank test is used to test the difference between ECMPDE and other algorithms. Table 5 shows the test results of engineering constraint optimization design problems.

Similar to the previous section, the test results of the three statistical analysis methods are compared in Table 5 below. It can be seen from Table 4 that ECMPDE, CCoDE, ECHTEP, FROFI, DyHF and jDE can find 100% feasible solutions for all engineering problems. The numerical results of ECMPDE solving welded beam design, pressure vessel design, reducer design, three-bar truss design, hydrodynamic thrust bearing design, conical wheel design, rolling bearing design, butterfly spring design, 10-bar plane truss size, multi-disc clutch, robot grip, disconnected beam domain uniform design are all up to or better than the algorithms CCoDE, ECHTEP, FROFI, etc. The solution results of DyHF and jDE.

From the comparison of the results of Mann-Whitney rank sum test in Table 5, it can be seen that compared with CCoDE, ECHTEP, FROFI, DyHF and jDE, the number of problems won by ECMPDE algorithm is 4, 6, 2, 5, 4 respectively, the number of similar results is 10, 9, 11, 9, 10, and the number of differences is 1, 0, 2, 1, 1, respectively. From the Friedman test, we can know that on MinMean, the rank of ECMPDE algorithm is the first, and its value is 3; on MinBest,

Table 4. Experimental results of engineering COPs.

Algorithm	Statistics	Tension pressure spring design	Welded beam design	Pressure vessel design	Reducer design	Design of three-bar truss
ECMPDE	**Minbest**	**0.012665**	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	**0.012665**	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	**3.94E−18**	**1.11E−15**	**9.09E−13**	**1.82E−12**	**0**
CCoDE	Minbest	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	6.43E−18	**1.11E−15**	**9.09E−13**	**1.82E−12**	**0**
DyHF	Minbest	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	2.01E−16	**1.11E−15**	**9.09E−13**	**1.82E−12**	**4.3E−12**
ECHTDE	Minbest	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	7.2E−18	**1.11E−15**	**9.09E−13**	**1.82E−12**	**0**
FROFI	Minbest	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	5.09E−18	**1.11E−15**	**9.09E−13**	**1.82E−12**	**0**
jDE	Minbest	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Minmean	0.012665	**1.724852**	**5885.333**	**2994.471**	**263.8958**
	Std	1.38E−12	**1.11E−15**	**9.09E−13**	**1.82E−12**	**0**

Algorithm	Statistics	Design of hydrodynamic thrust bearing	Conical wheel design	Rolling bearing design	Butterfly spring design	Design of gear train
ECMPDE	**Minbest**	**1625.443**	**14.67253**	**−81845.3**	**1.979675**	**2.7E−12**
	Minmean	**1625.443**	**14.67253**	**−79216.2**	**1.979675**	**5.42E−12**
	Std	**4.55E−13**	**9.4E−15**	**0.00045**	**1.33E−15**	**6.93E−12**
CCoDE	Minbest	1625.443	**14.67253**	**−81845.3**	**1.979675**	**2.7E−12**
	Minmean	1667.956	**14.67253**	**−81845.3**	**1.979675**	**4.74E−12**
	Std	228.9411	**3.55E−15**	**0**	**1.33E−15**	**6.11E−12**
DyHF	Minbest	1625.443	14.67253	−81845.3	1.979675	**2.7E−12**
	Minmean	1888.634	14.67253	−81678	1.983829	**2.7E−12**
	Std	181.67	1.04E−14	900.7875	0.02237	**1.21E−27**
ECHTDE	Minbest	**1625.443**	**14.67253**	−81845.3	1.979675	2.7E−12
	Minmean	**1625.443**	**14.67253**	−81844.9	1.992137	7.86E−12
	Std	**4.55E−13**	**3.55E−15**	1.357644	0.037387	2.08E−11
FROFI	Minbest	1654.128	**14.67253**	**−81845.3**	**1.979675**	**2.7E−12**
	Minmean	1725.325	**14.67253**	**−81845.3**	**1.979675**	**2.7E−12**
	Std	33.08269	**3.55E−15**	**0**	**1.33E−15**	**1.21E−27**
jDE	Minbest	**1625.443**	14.67253	−81845.3	1.979675	**2.7E−12**
	Minmean	**1625.443**	14.67253	−79216.2	1.979675	**4.06E−12**
	Std	**4.55E−13**	1.04E−14	9837.045	1.42E−15	**5.08E−11**

Table 5. Statistical test results of engineering COPs.

Statistic Test	Symbol	ECMPDE vs	CCoDE	ECHTEP	FROFI	DyHF	jDE
Mann-Whitney	Win	+	4	6	2	5	4
	Tied	≈	10	9	11	9	10
	Lose	−	1	0	2	1	1
Iman-Davenport	MinMean's Rank	**3**	3.36	3.76	3.23	4.03	3.6
	MinBest's Rank	**3.06**	3.56	3.56	3.6	3.53	3.3
Wilcoxon	Signed Rank	R^-	59.5	59.5	58.5	59.5	56
		R^+	60.5	60.5	61.5	60.5	64

the rank of ECMPDE algorithm is the smallest which is 3.06. In these comparisons, the performance of ECMPDE algorithm is better than that of other comparison algorithms. In the comparison results of Wilcoxon symbolic rank test, the symbolic rank sum is less than that, indicating that the performance of ECMPDE algorithm is the best.

Based on the analysis of the above results, we can know that the overall solution effect of ECMPDE algorithm is better than that of CCoDE, ECHTEP, FROFI, DyHF and jDE in 15 engineering constraint optimization problems, and the performance of the algorithm is the best.

5. Conclusion

In this paper, we use two constraint handling mechanisms to combine multiple DE variants and combine multiple group frameworks to overcome the shortcomings of weak constraint solving ability and low search ability. In this paper, ECMPDE combines DE variants such as JADE, jDE and EPSDE. The whole population of the algorithm is divided into two categories, namely, three subpopulations and one reward subpopulation. Each constituent DE variant has a subpopulation, and each variant is assigned a constraint handling mechanism through the constraint allocation mechanism, while the reward subpopulation is dynamically assigned to the combination of the best-performing DE variants and constraint handling mechanisms. The improved population division occurs in each generation, and the redistribution of reward subpopulations is triggered by each generation. Through the comparison of the best collocation of the previous generation and the current best collocation, the excellent collocation is saved in the constraint mechanism pool to improve the probability of selection. According to the analysis of the simulation results, ECMPDE is better than DyHF, ECHTEP, FROFI, jDE and JADE in solving test problems, and its performance is similar to that of CCoDE. This algorithm can improve the searching ability of the algorithm, improve the ability to solve constraints, and successfully inherit the advantages of its DE variant, showing strong robustness and high accuracy. In the future research, it is hoped that there will be excellent constraint handling mechanisms to further improve ECMPDE.

Acknowledgements

The author would like to thank the area editor and anonymous reviewers for their valuable comments and suggestions for this paper. This work is proudly supported in part by National Natural Science Foundation of China (No. 61763008, 11661030, 11401357, 11861027), Natural Science Foundation of Guangxi (No. 2018GXNSFAA138095), Fangchenggang Scientific research and technology development plan 2014 No. 42, and Project supported by Program to Sponsor Teams for Innovation in the Construction of Talent Highlands in Guangxi Institutions of Higher Learning ([2011] 47) and Doctoral Research Fund of Guilin University of Technology.

Conflicts of Interest

The authors declare that there is no conflict of interests regarding the publication of this article.

References

[1] Janga Reddy, M. and Nagesh Kumar, D. (2007) An Efficient Multi-Objective Optimization Algorithm Based on Swarm Intelligence for Engineering Design. *Engineering Optimization*, **39**, 49-68. https://doi.org/10.1080/03052150600930493

[2] Jamshidi, M. (2003) Tools for Intelligent Control: Fuzzy Controllers, Neural Networks and Genetic Algorithms. *Philosophical Transactions of the Royal Society of London. Series A*: *Mathematical*, *Physical and Engineering Sciences*, **361**, 1781-1808. https://doi.org/10.1098/rsta.2003.1225

[3] Onwubolu, G. and Davendra, D. (2006) Scheduling Flow Shops Using Differential Evolution Algorithm. *European Journal of Operational Research*, **171**, 674-692. https://doi.org/10.1016/j.ejor.2004.08.043

[4] Putha, R., Quadrifoglio, L. and Zechman, E. (2012) Comparing Ant Colony Optimization and Genetic Algorithm Approaches for Solving Traffic Signal Coordination under Oversaturation Conditions. *Computer-Aided Civil and Infrastructure Engineering*, **27**, 14-28. https://doi.org/10.1111/j.1467-8667.2010.00715.x

[5] Zhou, G. and Gen, M. (2003) A Genetic Algorithm Approach on Tree-Like Telecommunication Network Design Problem. *Journal of the Operational Research Society*, **4**, 248-254. https://doi.org/10.1057/palgrave.jors.2601510

[6] Durucasu, H. and Acar, E. (2013) Use of Evolutionary Algorithm in the Investment Project Evaluation. *Frontiers in Finance and Economics*, **12**, 32-50.

[7] Kohli, M. and Arora, S. (2018) Chaotic Grey Wolf Optimization Algorithm for Constraint Optimization Problems. *Journal of Computational Design and Engineering*, **5**, 458-472. https://doi.org/10.1016/j.jcde.2017.02.005

[8] Deb, K. (2000) An Efficient Constraint Handling Method for Genetic Algorithms. *Computer Methods in Applied Mechanics & Engineering*, **186**, 311-338. https://doi.org/10.1016/S0045-7825(99)00389-8

[9] Ying, W.Q., Peng, D.X., Xie, Y.H., *et al*. (2017) An Annealing Stochastic Ranking Mechanism for Constraint Evolutionary Optimization. 2017 *International Conference on Information System and Artificial Intelligence*, Jakarta, Indonesia, 5-7 December 2017, 576-580.

[10] Zamuda, A. (2017) Adaptive Constraint Handling and Success History Differential

Evolution for CEC 2017 Constraint Real-Parameter Optimization. 2017 *IEEE Congress on Evolutionary Computation* (*CEC*), San Sebastián, Spain, 5-8 June 2017, 2443-2450. https://doi.org/10.1109/CEC.2017.7969601

[11] Mallipeddi, R. and Suganthan, P.N. (2010) Ensemble of Constraint Handling Techniques. *IEEE Transactions on Evolutionary Computation*, **14**, 561-579. https://doi.org/10.1109/TEVC.2009.2033582

[12] Elsayed, S.M., Sarker, R.A. and Essam, D.L. (2011) Integrated Strategies Differential Evolution Algorithm with a Local Search for Constraint Optimization. 2011 *IEEE Congress of Evolutionary Computation* (*CEC*), New Orleans, LA, 5-8 June 2011, 2618-2625. https://doi.org/10.1109/CEC.2011.5949945

[13] Wang, Y., Wang, B.C., Li, H.X., *et al*. (2017) Incorporating Objective Function Information Into the Feasibility Rule for Constraint Evolutionary Optimization. *IEEE Transactions on Cybernetics*, **46**, 2938-2952. https://doi.org/10.1109/TCYB.2015.2493239

[14] Storn, R. and Price, K. (1997) Differential Evolution: A Simple and Efficient Heuristic for global Optimization over Continuous Spaces. *Journal of Global Optimization*, **11**, 341-359. https://doi.org/10.1023/A:1008202821328

[15] Brest, J., Greiner, S., Boskovic, B., *et al*. (2006) Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. *IEEE Transactions on Evolutionary Computation*, **10**, 646-657. https://doi.org/10.1109/TEVC.2006.872133

[16] Zhang, J. and Sanderson, A.C. (2009) JADE: Adaptive Differential Evolution with Optional External Archive. *IEEE Transactions on Evolutionary Computation*, **13**, 945-958. https://doi.org/10.1109/TEVC.2009.2014613

[17] Wang, Y., Cai, Z. and Zhang, Q. (2011) Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters. *IEEE Transactions on Evolutionary Computation*, **15**, 55-66. https://doi.org/10.1109/TEVC.2010.2087271

[18] Mallipeddi, R., Suganthan, P.N., Pan, Q.K., *et al*. (2011) Differential Evolution Algorithm with Ensemble of Parameters and Mutation Trategies. *Applied Soft Computing*, **11**, 1679-1696. https://doi.org/10.1016/j.asoc.2010.04.024

[19] Wu, G., Shen, X., Li, H., *et al*. (2018) Ensemble of Differential Evolution Variants. *Information Sciences*, **423**, 172-186. https://doi.org/10.1016/j.ins.2017.09.053

[20] Wang, Y. and Cai, Z. (2012) A Dynamic Hybrid Framework for Constraint Evolutionary Optimization. *IEEE Transactions on Systems*, *Man and Cybernetics*, *Part B* (*Cybernetics*), **42**, 203-217. https://doi.org/10.1109/TSMCB.2011.2161467

[21] Wang, B.C., Li, H.X., Li, J.P., *et al*. (2018) Composite Differential Evolution for Constraint Evolutionary Optimization. *IEEE Transactions on Systems*, *Man*, *and Cybernetics: Systems*, **49**, 1482-1495. https://doi.org/10.1109/TSMC.2018.2807785

[22] Wolpert, D.H. and Macready, W.G. (1997) No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation*, **1**, 67-82. https://doi.org/10.1109/4235.585893

[23] Mezura-Montes, E., Velázquez-Reyes, J. and Coello, C.A.C. (2006) Modified Differential Evolution for Constraint Optimization. 2006 *IEEE International Conference on Evolutionary Computation*, Vancouver, Canada, 16-21 July 2006, 25-32. https://doi.org/10.1109/CEC.2006.1688286

[24] Brest, J., Zumer, V. and Maucec, M.S. (2006) Self-Adaptive Differential Evolution Algorithm in Constraint Real-Parameter Optimization. 2006 *IEEE International Conference on Evolutionary Computation*, Vancouver, Canada, 16-21 July 2006, 215-222. https://doi.org/10.1109/CEC.2006.1688311

[25] Takahama, T. and Sakai, S. (2006) Constraint Optimization by the ε Constraint Differential Evolution with Gradient-Based Mutation and Feasible Elites. 2006 *IEEE International Conference on Evolutionary Computation*, Vancouver, Canada, 16-21 July 2006, 1-8.

[26] Peng, F., Tang, K., Chen, G. and Yao, X. (2010) Population-Based Algorithm Portfolios for Numerical Optimization. *IEEE Transactions on Evolutionary Computation*, **14**, 782-800. https://doi.org/10.1109/TEVC.2010.2040183

[27] Takahama, T. and Sakai, S. (2008) Constraint Optimization by ε Constraint Differential Evolution with Dynamic ε-Level Control. In: *Advances in Differential Evolution*, Springer, Berlin, Heidelberg, 139-154. https://doi.org/10.1007/978-3-540-68830-3_5

[28] Takahama, T. and Sakai, S. (2009) Solving Difficult Constraint Optimization Problems by the ε Constraint Differential Evolution with Gradient-Based Mutation. In: *Constraint-Handling in Evolutionary Optimization*, Springer, Berlin, Heidelberg, 51-72. https://doi.org/10.1007/978-3-642-00619-7_3

[29] Takahama, T. and Sakai, S. (2010) Constraint Optimization by the ε Constraint Differential Evolution with an Archive and Gradient-Based Mutation. 2010 *IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 18-23 July 2010, 1-9.

[30] Wu, G., Mallipeddi, R., Suganthan, P.N., *et al*. (2016) Differential Evolution with Multi-Population Based Ensemble of Mutation Strategies. *Information Sciences*, **329**, 329-345. https://doi.org/10.1016/j.ins.2015.09.009

[31] Qin, A.K., Huang, V.L. and Suganthan, P.N. (2009) Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization. *IEEE Transactions on Evolutionary Computation*, **13**, 398-417. https://doi.org/10.1109/TEVC.2008.927706

[32] Wang, H., Hu, Z., Sun, Y., *et al*. (2018) Modified Backtracking Search Optimization Algorithm Inspired by Simulated Annealing for Constraint Engineering Optimization Problems. *Computational Intelligence and Neuroscience*, **2018**, Article ID: 9167414. https://doi.org/10.1155/2018/9167414