

# *A Review of Research on Differential Evolution Algorithm*

**Yufu Sun**

*College of Civil Engineering, Nanjing Forestry University, Nanjing, China  
2251924587@qq.com*

**Abstract.** Differential Evolution (DE) is a simple and widely applied swarm intelligence optimization algorithm. This paper systematically reviews the fundamental mechanism, improved variants, practical applications and future research directions of DE. Different from previous reviews that simply list algorithms in chronological order, this paper sorts existing studies into three core research branches: balancing global exploration and local exploitation, self-adaptive regulation of control parameters and mutation strategies, and algorithm extensions for complex optimization and practical engineering problems. Existing literature shows that modern DE has shifted from fixed operator design to adaptive frameworks built on search history, external archives and dynamic population sizes. Nevertheless, its performance is restricted by benchmark test settings, fair parameter comparison and population diversity maintenance. Future research can focus on interpretable adaptive mechanisms, decomposition strategies for high-dimensional optimization, and hybrid DE methods deeply integrated with domain knowledge.

**Keywords:** differential evolution algorithm, evolutionary computation, adaptive parameter control, global optimization, research review

## **1. Introduction**

A large number of scientific numerical computation and engineering structure design optimization tasks require searching for the optimal feasible solution in the solution space of continuous variables. The objective functions of such optimization problems commonly feature nonlinearity, non-convexity, multimodality, observation noise interference and simulation black-box properties. Once the objective function fails to meet preconditions like differentiability, convexity and analytical gradient solvability, traditional gradient-based optimization methods can hardly work properly. Evolutionary computation performs synchronous iteration with multiple candidate solutions in a population, and improves solution quality generation by generation through random mutation, recombination and selection operations, providing a feasible approach to the above complex optimization problems.

Differential evolution (DE) is one of the most widely applied evolutionary computation methods in the field of continuous variable optimization. Standard DE constructs vector differences using three individuals from the population to generate mutation search directions: it selects the difference vector of two candidate solutions and applies perturbation to another base candidate solution to complete the mutation operation [1]. This operational logic is clearly different from that of genetic

algorithms, which realize search through coded crossover and random mutation. The search step size and moving direction of DE are completely determined by the overall distribution characteristics of the current population. Simply put, in addition to storing all candidate solutions, the population set also implicitly contains information about the search scale and moving direction in the optimization space.

The core reason for the widespread attention paid to differential evolution algorithm lies in its simple theoretical logic and stable practical solution performance. The algorithm only sets a small number of control parameters, has low code implementation difficulty, and does not require gradient information of the objective function during the solution process [2]. Multiple review studies have confirmed that DE has been applied in many fields such as numerical benchmark optimization, engineering structure design, production scheduling, signal processing, machine learning model training, image analysis, and energy system operation optimization [3-6]. However, the solution performance of standard DE is greatly affected by four types of parameters: mutation strategy, scaling factor, crossover probability and population size. This shortcoming of parameter sensitivity has promoted the academic community to carry out research in multiple directions successively, including adaptive differential evolution, archive-assisted differential evolution, dynamic population size adjustment, and hybrid differential evolution.

This paper regards differential evolution as a complete algorithm family rather than a single fixed optimization procedure. The main body first fully explains the standard iterative process of DE, then systematically sorts out mainstream improved variants such as jDE, SaDE, JADE, SHADE and L-SHADE, and subsequently introduces the extended implementation methods of the algorithm in constrained optimization and multi-objective optimization scenarios. The core viewpoint of the whole paper is that research on differential evolution algorithm has gradually developed from the traditional idea of manually setting fixed operators to automatically adjusting parameters and mutation strategies based on historical search feedback. In the next stage, relevant research will pay more attention to experimental reproducibility and interpretability of adaptive mechanisms, and carry out algorithm improvement closely combined with the inherent structure of specific optimization problems.

## 2. Basic principles of differential evolution algorithm

### 2.1. Population representation and initialization

Differential evolution algorithm uses real-valued vectors to represent each group of candidate solutions. Let the  $i$ -th individual in the  $G$ -th generation be denoted as  $x_i^G = (x_{i,1}^G, x_{i,2}^G, \dots, x_{i,D}^G)$ , where  $D$  is the dimension of decision variables. The initial population is usually generated uniformly and randomly within the upper and lower bounds of variables. Since DE acts directly on continuous variables, it does not require binary encoding or complex decoding processes, which is also an important reason why it is suitable for numerical optimization.

The complete iterative process of differential evolution includes three core steps: mutation, crossover and selection. The industry-general naming rule DE/x/y/z is used to distinguish various mutation strategies:  $x$  represents the base vector selection rule,  $y$  represents the number of difference vectors used, and  $z$  represents the type of crossover operation. Taking DE/rand/1/bin as an example, this name corresponds to three settings: randomly selecting the base vector, using only one set of difference vectors, and binomial crossover. This naming rule is concise and intuitive, but the

selection of mutation strategy will directly change the balance between global exploration and local exploitation of the algorithm [3].

## 2.2. Mutation, crossover and selection

In the classic DE/rand/1 strategy, the algorithm randomly selects three distinct population individuals  $r_1$ ,  $r_2$  and  $r_3$  for the target vector  $x_{i,r_1}^G$ , and generates the donor vector through the following formula:

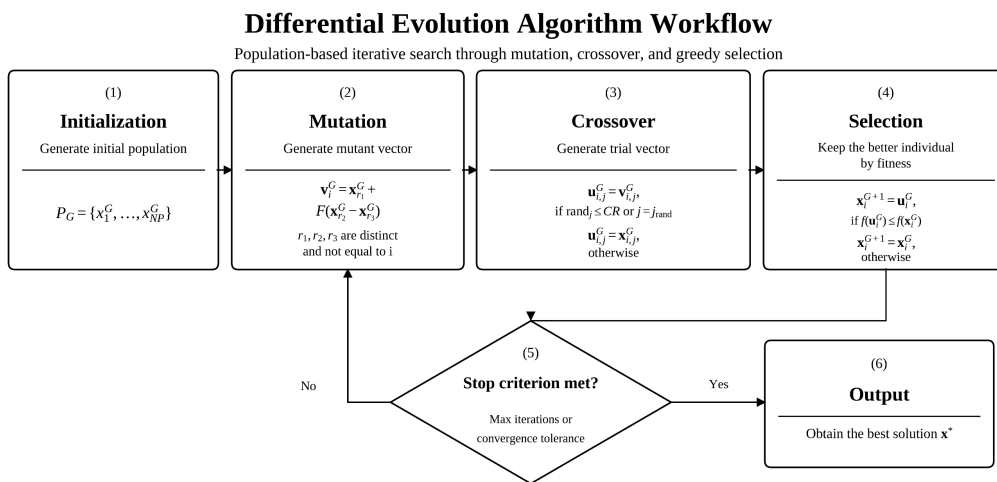
$$v_i^G = x_{i,r_1}^G + F \left( x_{i,r_2}^G - x_{i,r_3}^G \right), r_1 \neq r_2 \neq r_3 \quad (1)$$

where  $F$  is the scaling factor, which is used to control the influence intensity of the difference vector. A larger  $F$  tends to expand the search range, while a smaller  $F$  is more conducive to local exploitation. After mutation, the algorithm combines the donor vector  $v_i^G$  and the target vector  $x_{i,r_1}^G$  into the trial vector  $u_{i,j}^G$  through crossover operation. In binomial crossover, each dimension inherits the component of the donor vector with probability  $CR$ , while at least one dimension is guaranteed to come from the donor vector:

$$u_{i,j}^G = \begin{cases} v_{i,j}^G, & \text{if } rand_j \leq CR \text{ or } j = j_{rand} \\ x_{i,j}^G, & \text{otherwise} \end{cases} \quad (2)$$

In the selection stage, one-to-one greedy selection is adopted: the trial vector only competes with its corresponding target vector, and the one with better fitness enters the next generation:

$$x_i^{G+1} = \begin{cases} u_i^G, & \text{if } f(u_i^G) \leq f(x_i^G) \\ x_i^G, & \text{otherwise} \end{cases} \quad (3)$$



Typical control parameters: population size  $NP$ , mutation factor  $F$ , crossover rate  $CR$ , and generation index  $G$ .

Figure 1. Workflow of the differential evolution algorithm

This selection rule is very simple, but it determines the basic convergence characteristics of DE. In single-objective minimization problems, it ensures that the fitness of each individual will not be replaced by inferior solutions, while the population still continuously generates new search directions through vector differences. Figure 1 shows the basic workflow of DE.

### 3. Research progress of differential evolution algorithm variants

#### 3.1. From fixed parameters to adaptive control

The standard differential evolution algorithm requires researchers to manually set the population size NP, scaling factor F, crossover probability CR and mutation strategy. The above parameters have complex coupling relationships with the dimension of the optimization problem, the shape of the fitness landscape, multimodal characteristics, and the structure of constraint conditions. A set of parameters that performs excellently on one type of optimization problem is very likely to cause problems such as slow convergence speed and premature search stagnation when applied to other problems. Therefore, adaptive parameter adjustment has become the most popular research direction in the field of differential evolution.

As shown in Table 1, the jDE algorithm embeds two types of control parameters, F and CR, into each individual in the population, and completes the adaptive update of parameters synchronously during the iteration process [7]. The SaDE algorithm further relies on the statistical information of successful historical search samples to simultaneously learn the optimal mutation strategy and suitable parameter ranges [8]. The JADE algorithm proposes a new mutation strategy called current-to-pbest, matched with an optional external archive set. It uses historically successful search directions to guide subsequent mutation operations, and the archive stores eliminated individuals to expand the optional range of difference vectors and improve population diversity [9]. These improvement schemes have completely changed the traditional mode of manually fixing parameters and built an adaptive control framework based on search feedback.

The SHADE and L-SHADE algorithms continue the research and development idea of adaptive control. SHADE constructs a success-history memory bank to store the F and CR parameters that perform well during the iteration process, and generates new control parameters by random sampling from the memory bank in each generation [10]. On the basis of SHADE, L-SHADE adds a linear population size reduction mechanism: it maintains a large population in the early iteration stage to strengthen global exploration, and gradually reduces the population size in the late stage to focus on local fine exploitation [11]. Multiple existing review studies have confirmed that success-history memory banks, dynamic population size adjustment, and adaptive strategy selection are still the core research directions for improving DE algorithms at present [12, 13, 14].

Table 1. Representative differential evolution variants

| Variant       | Main Mechanism                                                | Typical Advantages                               | Main Limitations                              |
|---------------|---------------------------------------------------------------|--------------------------------------------------|-----------------------------------------------|
| DE/rand/1/bin | Random base vector, one difference vector, binomial crossover | Simple structure, can serve as a robust baseline | Sensitive to F, CR, NP and strategy selection |
| jDE           | Adaptive adjustment of F and CR at the individual level       | Reduces the burden of manual parameter tuning    | May adapt slowly on complex landscapes        |

Table 1. (continued)

|         |                                                                  |                                                        |                                                              |
|---------|------------------------------------------------------------------|--------------------------------------------------------|--------------------------------------------------------------|
| SaDE    | Learns strategy and parameter preferences from successful trials | Suitable for problems with no single dominant strategy | Requires additional learning windows and statistical records |
| JADE    | current-to-pbest mutation with external archive                  | Balances exploitation ability and diversity            | Performance is affected by archive and p-best settings       |
| SHADE   | Success-history based memory for F and CR                        | Strong parameter control capability                    | Memory behavior may depend on problem types                  |
| L-SHADE | Linear population size reduction based on SHADE                  | Outstanding performance in benchmark tests             | May lose diversity prematurely on some problems              |

### 3.2. Diversity, archive mechanism and population size

The core difficulty of differential evolution algorithm is not only to generate new candidate solutions, but also to maintain effective diversity within the population in the middle and late stages of iteration when selection pressure continues to rise. If the distribution of population individuals converges rapidly, the vector differences will continue to shrink, and the algorithm will fall into local optimum and search stagnation. If high population diversity is maintained throughout the process, the overall convergence speed will be greatly slowed down. Archive sets, neighborhood individual screening, population restart strategies, and dynamic population size scheduling are common methods used to balance the above contradictions.

The external archive set matched with JADE is a typical solution: old individuals eliminated and replaced by high-quality trial vectors are not directly discarded, but stored in the archive set to participate in subsequent mutation operations, effectively expanding the optional range of difference vectors [9]. Relevant studies on population size NP show that NP is not only a parameter that controls computational cost, but also changes the statistical distribution characteristics of the overall search process [12]. The core reason why the SHADE and L-SHADE series of algorithms have received extensive research attention is that they deeply bind population diversity management with iterative search stages.

### 3.3. Constrained optimization and multi-objective extension

Practical engineering optimization problems are rarely unconstrained single-objective models. Tasks such as structural design, resource allocation, controller parameter tuning, and production scheduling generally include equality and inequality constraints. The current mainstream treatment method is to convert the amount of constraint violation into a penalty term and superimpose it on the objective function, but it is difficult to unify the standard for selecting the penalty coefficient. Therefore, most research on DE constraint handling adopts three types of ideas: feasible solution priority comparison rules, adaptive penalty functions, and direct hierarchical comparison of feasible/infeasible solutions [15].

Multi-objective differential evolution algorithms need to reconstruct the original single-objective greedy selection rule, as there is no absolute total order relationship of superiority or inferiority among multiple groups of candidate solutions. Researchers usually combine Pareto dominance

relationship, solution set diversity maintenance, objective decomposition strategies, external archive management and DE mutation operators. Early research on Pareto differential evolution confirmed that DE mutation operators can be embedded into the overall framework of multi-objective evolutionary computation [16]. In multi-objective optimization scenarios, the algorithm needs to continuously approach the real Pareto optimal front while ensuring uniform distribution of the front solution set. Therefore, the design difficulty of the selection mechanism and archive management is much higher than that of single-objective DE.

#### 4. Application fields and comparative advantages

When the objective function can only be solved by simulation programs, gradient information is distorted and unavailable, or decision variables are continuous bounded values, the adaptability advantage of the differential evolution algorithm is significant. The algorithm has been widely used in many fields such as engineering structure optimization, control system parameter tuning, neural network model training, feature subset selection, model parameter identification, energy system scheduling, image and signal processing, and logistics path optimization. The practical value of DE does not rely on strict theoretical convergence proofs, but on low code implementation thresholds, stable black-box optimization solution capabilities, and flexible expansion space adapted to various problem structures. Typical application scenarios are shown in Table 2.

Compared with particle swarm optimization algorithm, DE directly mines the geometric distribution information of the population through differential mutation, and the one-to-one greedy selection mechanism can stably retain high-quality solutions optimized in each generation. Compared with genetic algorithms, DE does not need binary encoding conversion for continuous variables, which simplifies the program development process. Compared with the covariance matrix adaptation evolution strategy, DE has lower computational cost per generation iteration and simpler code implementation, but when dealing with ill-conditioned functions and ultra-high-dimensional optimization problems, it needs to be matched with more refined parameter control and mutation strategy design. The above comparisons do not mean that DE has the best performance in all optimization scenarios, but explain the core reason why this algorithm is often used as a benchmark algorithm for black-box continuous optimization.

Table 2. Typical application settings for differential evolution

| Application Scenario         | Optimization Object                                       | Reasons for DE Applicability                                                      |
|------------------------------|-----------------------------------------------------------|-----------------------------------------------------------------------------------|
| Engineering design           | Continuous design variables under nonlinear constraints   | No derivatives required, easy to combine with constraint handling                 |
| Control and parameter tuning | Controller gains or model parameters                      | Suitable for simulation objectives and noisy evaluations                          |
| Machine learning             | Hyperparameters, feature subsets or model weights         | Suitable for scenarios where gradients are unavailable or evaluation is black-box |
| Energy systems               | Scheduling, capacity configuration or operation variables | Can handle multiple constraints and nonlinear costs                               |

Table 2. (continued)

| Image and signal processing | Thresholds, filters or registration parameters | Suitable for multimodal and irregular objective landscapes |
|-----------------------------|------------------------------------------------|------------------------------------------------------------|
|-----------------------------|------------------------------------------------|------------------------------------------------------------|

## 5. Challenges and future directions

### 5.1. Benchmark testing and reproducibility

The field of differential evolution has long suffered from non-standard experimental comparisons: benchmark test functions, total function evaluation times, initial parameter settings, and statistical test procedures adopted by various new improved algorithms are not unified. Most improved variants adjust multiple internal mechanisms at the same time, and relying solely on the final optimal objective value cannot locate the core modules that truly improve solution performance. A standardized comparative experimental design should unify the total function evaluation budget, cover multiple types of test functions, introduce non-parametric statistical significance tests, add ablation control experiments, and open source the complete code program. Without the above standardized procedures, the effect of algorithm improvement often only holds under specific narrow test conditions and is not universal.

### 5.2. High-dimensional optimization and scalability

In ultra-high-dimensional solution spaces, the search efficiency of difference vectors between population individuals drops significantly, while the number of function evaluations required for solution increases exponentially. Improvement research for large-scale optimization problems mainly focuses on several ideas: cooperative coevolution, variable grouping and splitting, dimensionality reduction preprocessing, surrogate model approximation, and problem decomposition. The core difficulty of the research lies in completely retaining the natural search advantages of differential mutation operators while reducing the effective search dimension. For hybrid complex optimization problems where some variables are smooth and some can only be solved by simulation, hybrid differential evolution algorithms integrated with local search have higher engineering application value.

### 5.3. Adaptive strategy selection and algorithm configuration

Multiple recent review papers point out that adaptive control mechanisms remain the core line of innovation in differential evolution algorithms [13, 14]. However, an adaptive framework does not mean that the algorithm naturally has strong robustness: some adaptive rules may overfit specific benchmark functions, respond laggingly to changes in search status during iteration, or rapidly lose population diversity in the middle and late stages of iteration. Future research needs to clarify the applicable boundaries of adaptive mechanisms, the response update speed of control parameters, and the interpretability of the internal memory bank's operating logic. Meta-learning and automatic algorithm configuration tools can match suitable DE variants for different types of optimization problems, but relevant improvement schemes still need to be verified under a standardized and reproducible experimental framework.

## 5.4. Integration with domain knowledge

Well-implemented engineering applications of DE rarely directly call general standard optimization programs; instead, they additionally embed feasible solution repair operators, inherent problem structure information, surrogate approximation models, and local fine search modules. In practical engineering optimization tasks, in addition to pursuing the optimal objective function value, attention is also paid to solution reliability, model interpretability, and the degree of constraint satisfaction. A research-worthy direction in the future is interpretable hybrid differential evolution algorithms: the internal adaptive control logic and industry-specific expansion modules of the algorithm can be decomposed and analyzed, rather than merely serving as black-box improvement methods to enhance solution performance.

## 6. Conclusion

Relying on concise vector differential mutation logic and a flexible population iterative search framework, the differential evolution algorithm remains an algorithm family with great research value in the field of continuous global optimization to this day. Standard DE/rand/1/bin can be used as a benchmark for performance comparison, but the mode with fixed parameters and a single mutation strategy is difficult to adapt to optimization problems with various forms. Classic improved algorithms such as jDE, SaDE, JADE, SHADE and L-SHADE have fully verified the optimization gains of adaptive parameter control, success-history memory banks, external archive sets, and dynamic population size adjustment.

Based on the existing literature, it can be found that the final solution performance of the differential evolution algorithm is jointly determined by the dynamic coupling relationship of four factors: mutation operators, selection rules, population diversity maintenance, and the inherent structure of the optimization problem. The breakthrough points of future research are likely to focus on four directions: efficient adaptive control frameworks with complete interpretability, standardized and fair benchmark test experimental systems, and reproducible hybrid improved algorithms combined with industry prior knowledge. Therefore, differential evolution should not only be regarded as a single optimization tool, but also as a core research carrier for studying the laws of population geometric features guiding search in complex continuous spaces.

## References

- [1] Storn, R., & Price, K. (1997). Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341-359. <https://doi.org/10.1023/A:1008202821328>
- [2] Price, K., Storn, R. M., & Lampinen, J. A. (2005). *Differential evolution: A practical approach to global optimization*. Springer. <https://doi.org/10.1007/3-540-31306-0>
- [3] Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1), 4-31. <https://doi.org/10.1109/TEVC.2010.2059031>
- [4] Das, S., Mullick, S. S., & Suganthan, P. N. (2016). Recent advances in differential evolution - An updated survey. *Swarm and Evolutionary Computation*, 27, 1-30. <https://doi.org/10.1016/j.swevo.2016.01.004>
- [5] Bilal, Pant, M., Zaheer, H., Garcia-Hernandez, L., & Abraham, A. (2020). Differential evolution: A review of more than two decades of research. *Engineering Applications of Artificial Intelligence*, 90, 103479. <https://doi.org/10.1016/j.engappai.2020.103479>
- [6] Ahmad, M. F., Isa, N. A. M., Lim, W. H., & Ang, K. M. (2022). Differential evolution: A recent review based on state-of-the-art works. *Alexandria Engineering Journal*, 61(5), 3831-3872. <https://doi.org/10.1016/j.aej.2021.09.013>
- [7] Brest, J., Greiner, S., Boskovic, B., Mernik, M., & Zumer, V. (2006). Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems. *IEEE Transactions on Evolutionary*

- Computation*, 10(6), 646-657. <https://doi.org/10.1109/TEVC.2006.872133>
- [8] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2009). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2), 398-417. <https://doi.org/10.1109/TEVC.2008.927706>
  - [9] Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5), 945-958. <https://doi.org/10.1109/TEVC.2009.2014613>
  - [10] Tanabe, R., & Fukunaga, A. (2013). Success-history based parameter adaptation for differential evolution. *2013 IEEE Congress on Evolutionary Computation*, 71-78. <https://doi.org/10.1109/CEC.2013.6557555>
  - [11] Tanabe, R., & Fukunaga, A. (2014). Improving the search performance of SHADE using linear population size reduction. *2014 IEEE Congress on Evolutionary Computation (CEC)*, 1658-1665. <https://doi.org/10.1109/CEC.2014.6900380>
  - [12] Piotrowski, A. P. (2017). Review of differential evolution population size. *Swarm and Evolutionary Computation*, 32, 1-24. <https://doi.org/10.1016/j.swevo.2016.05.003>
  - [13] Ye, X., Li, J., Wang, P., & Suganthan, P. N. (2025). A comprehensive survey of adaptive strategies in differential evolutionary algorithms. *Swarm and Evolutionary Computation*, 98, 102081. <https://doi.org/10.1016/j.swevo.2025.102081>
  - [14] Piotrowski, A. P., Piotrowska, A. E., & Napiorkowski, J. J. (2026). Experimental survey of L-SHADE and SHADE-based adaptive differential evolution algorithms. *Swarm and Evolutionary Computation*, 101, 102286. <https://doi.org/10.1016/j.swevo.2026.102286>
  - [15] Lampinen, J. (2002). A constraint handling approach for the differential evolution algorithm. *Proceedings of the 2002 Congress on Evolutionary Computation*, 1468-1473. <https://doi.org/10.1109/CEC.2002.1004459>
  - [16] Madavan, N. K. (2002). Multiobjective optimization using a Pareto differential evolution approach. *Proceedings of the 2002 Congress on Evolutionary Computation*, 1145-1150. <https://doi.org/10.1109/CEC.2002.1004404>