Hindawi Publishing Corporation
Journal of Applied Mathematics
Volume 2013, Article ID 841780, 15 pages
http://dx.doi.org/10.1155/2013/841780

Research Article

A Hybrid Multiobjective Differential Evolution Algorithm and Its Application to the Optimization of Grinding and Classification

Yalin Wang,[1] Xiaofang Chen,[1] Weihua Gui,[1] Chunhua Yang,[1] Lou Caccetta,[2] and Honglei Xu[2]

[1] *School of Information Science and Engineering, Central South University, Changsha 410083, China*
[2] *Department of Mathematics & Statistics, Curtin University, Perth, WA 6845, Australia*

Correspondence should be addressed to Xiaofang Chen; xiaofangchen@csu.edu.cn

Received 19 July 2013; Accepted 5 September 2013

Academic Editor: Dewei Li

The grinding-classification is the prerequisite process for full recovery of the nonrenewable minerals with both production quality and quantity objectives concerned. Its natural formulation is a constrained multiobjective optimization problem of complex expression since the process is composed of one grinding machine and two classification machines. In this paper, a hybrid differential evolution (DE) algorithm with multi-population is proposed. Some infeasible solutions with better performance are allowed to be saved, and they participate randomly in the evolution. In order to exploit the meaningful infeasible solutions, a functionally partitioned multi-population mechanism is designed to find an optimal solution from all possible directions. Meanwhile, a simplex method for local search is inserted into the evolution process to enhance the searching strategy in the optimization process. Simulation results from the test of some benchmark problems indicate that the proposed algorithm tends to converge quickly and effectively to the Pareto frontier with better distribution. Finally, the proposed algorithm is applied to solve a multiobjective optimization model of a grinding and classification process. Based on the technique for order performance by similarity to ideal solution (TOPSIS), the satisfactory solution is obtained by using a decision-making method for multiple attributes.

1. Introduction

Grinding-classification is an important prerequisite process for most mineral processing plants. The grinding process reduces the particle size of raw ores and is usually followed by classification to separate them into different sizes. Grinding-classification operation is required to produce pulp with suitable concentration and fineness for flotation. The pulp quality will directly influence the subsequent flotation efficiency and recovery of valuable metals from tailings. In order to improve economic efficiency and energy consumption, the process optimization objectives include product quality and output yields. Under certain mineral source conditions, the objectives are decided by a series of operation variables such as the solid flow of feed ore to ball mill, the steel ball filling rate, and the flow rates of water added to the first and the second classifier recycles. To solve the optimization model of products' output and quality in the grinding-classification process is of great significance to improve the technical and economic specifications, and it has been a continuous endeavor of the scientists and engineers [1–3].

Grinding-classification is an energy-intensive process influenced by many interacting factors with mutual restraints. The goals of grinding-classification optimization problem are decided by multiple constrained input control variables of nonlinear relationships. So, the optimization model of grinding-classification operation is a complex constrained multiobjective optimization problem (CMOP). Generally, constrained multiobjective problems are so difficult to be solved that the constraint handling techniques and multiobjective optimization methods need to be combined for optimization.

Multiobjective optimization problems (MOPs), in the case of traditional optimization methods, are often handled by aggregating multiple objectives into a single scalar objective through weighting factors. MOPs have a set of equally good (nondominating) solutions instead of a single one, called a Pareto optimum which was introduced by Edgeworth in 1881 [4] and later generalized by Pareto in 1896 [5]. The practical MOPs are often implicated in series of equations, functions, or procedures with complicated constraints. Therefore, the evolutionary algorithms are attractive approaches for low requirements on mathematical expression [6]. Since the mid 1980s, there has been a growing interest in solving MOPs using evolutionary approaches [7–10]. One of the most successful evolutionary algorithms for the optimization of continuous space functions is the differential evolution (DE) [11]. DE is simple and efficiently converges to the global optimum in most cases [12, 13]. Its efficiency has been proven [14] in many application fields such as pattern recognition [15] and mechanical engineering [16].

There have been many improvements for DE to solve MOPs. Abbass [17] firstly provided a Pareto DE (PDE) algorithm for MOPs in which DE was employed to create new solutions, and only the nondominated solutions were kept as the basis for the next generation. Madavan [18] developed a Pareto differential evolution approach (PDEA) in which new solutions were created by DE and kept in an auxiliary population. Xue et al. [19] introduced multiobjective differential evolution (MODE) and used Pareto-based ranking assignment and crowding distance metric, but in a different manner from PDEA. Robic and Filipi [20], also adopting Pareto-based ranking assignment and crowding distance metric, developed a DE for multiobjective optimization (DEMO) with a different population update strategy and achieved good results. Huang et al. [21] extended the self-adaptive DE (SADE) to solve MOPs by a so called multiobjective self-adaptive DE (MOSADE). They further extended MOSADE by using objectivewise learning strategies [22]. Adeyemo and Otieno [23] provided multiobjective differential evolution algorithm (MDEA). In MDEA, a new solution was generated by DE variant and compared with target solution. If it dominates the target solution, then it was added to the new population; otherwise, a target solution was added.

On the other hand, single-objective constrained optimization problems have been studied intensively in the past years [24–28]. Different constraint handling techniques have been proposed to solve constrained optimization problems. Michalewicz and Schoenauer [29] divided constraints handling methods used in evolutionary algorithms into four categories: preserving feasibility of solutions, penalty functions, separating the feasible and infeasible solutions, and hybrid methods. The differences among these methods are how to deal with the infeasible individuals throughout the search phases. Currently, the penalty function method is most widely used, and this algorithm strongly depends on the choice of the penalty parameter.

Although the multiobjective optimization and the constraint handling problem have received lots of contribution, respectively, the CMOPs are still difficult to be solved in practice. Coello and Christiansen [30] proposed a simple approach to solve CMOPs by ignoring any solution that violates any of the assigned constraints. Deb et al. [8] proposed a constrained multiobjective algorithm based on the concept of constrained domination, which is also known as superiority of the feasible solution. Woldesenbet et al. [31] introduced a constraint handling technique based on adaptive penalty functions and distance measures by extending the corresponding version for the single-objective constrained optimization.

In the MOP of grinding and classification process, the definitions of Pareto solutions, Pareto frontier, and Pareto dominance are in consistency with the classic definitions. Clearly, the Pareto frontier is a mapping of the Pareto-optimal solutions to the objective space. In the minimization sense, general constrained MOPs can be formulated as follows

$$\begin{aligned} \min\ & F(X) = \min\left[f_1(X), f_2(X), \ldots, f_r(X)\right], \\ \text{s.t.}\ & g_i(X) \le 0 \quad (i = 1, 2, \ldots, p), \\ & h_j(X) = 0 \quad (j = p+1, \ldots, q), \\ & x_k \in \left[x_{k\min}, x_{k\max}\right] \quad (k = 1, 2, \ldots, n), \end{aligned} \tag{1}$$

where $F(X)$ is the objective vector, $X = (x_1, \ldots, x_n) \in \mathbb{R}^n$ is a parameter vector, $g_i(X)$ is the ith inequality constraint, and $h_j(X)$ is the jth equality constraint. $x_{k\min}$ and $x_{k\max}$ are, respectively, the lower and upper bounds of the decision variable x_k.

In this paper, based on the specific industrial background of continuous bauxite grinding-classification operation, a new hybrid DE algorithm is proposed to solve complex constrained multiobjective optimization problems. Firstly, a hybrid DE algorithm for MOPs with simplex method (SM-DEMO) is designed to overcome the problems of global performance degradation and being trapped in local optimum. Then, for the MOPs with complicated constraints, the proposed algorithm is formed by combining SM-DEMO and functional partitioned multi-population. In this method, the construction of penalty functions is not required, and the meaningful infeasible solutions are fully utilized.

The remainder of the paper is structured as follows. Section 2 describes the SM-DEMO algorithm for unconstrained cases. The proposed algorithm of multipopulation for constrained MOPs is given in Section 3 with verification of performance by benchmark testing results. Section 4 describes the model of products' output and quality in the grinding-classification process in detail and the application of the proposed algorithm in the optimization model. Finally, the conclusions based on the present study are drawn in Section 5.

2. SM-DEMO Algorithm for Unconstrained MOPs

In order to efficiently solve multiobjective optimization problem and find the approximately complete and nearoptimal Pareto frontier, we proposed a hybrid DE algorithm for unconstrained multiobjective optimization with simplex method.

The differential evolution, with initialization, crossover, and selection as in usual genetic algorithms, uses a perturbation of two members as the mutation operator to produce a new individual. The mutation operator of the DE algorithm is described as follows.

Considering each target individual x_i^G, in the Gth generation of size Np, a mutant individual $\widehat{x}_i^{G+1}$ is defined by

$$\widehat{x}_i^{G+1} = x_{r3}^G + F\left(x_{r1}^G - x_{r2}^G\right), \tag{2}$$

where indexes r_1, r_2, and r_3 represent mutually different integers that are different from i and that are randomly generated over $[1, Np]$, and F is the scaling factor.

The simplex method, proposed by Spendley, Hext, and Himsworth and later refined by Nelder and Mead (NM) [32], is a derivative-free line-search method that is particularly designed for traditional unconstrained minimization scenarios. Clearly, NM method can be deemed as a direct line-search method of the steepest descent kind. The ingredients of the replacement process consist of four basic operations: reflection, expansion, contraction, and shrinkage. Through these operations, the simplex can improve itself and approximate to a local optimum point sequentially. Furthermore, the simplex can vary its shape, size, and orientation to adapt itself to the local contour of the objective function.

2.1. Main Strategy of SM-DEMO.

2.1. Main Strategy of SM-DEMO. The SM-DEMO algorithm is improved by the following three points compared with DE.

2.1.1. Modified Selection Operation. After traditional DE evolution, the individual u_{ij}^{G+1} may violate the boundary constraints $x_{ij}^{\max}$ and $x_{ij}^{\min}$. u_{ij}^{G+1} is replaced by new individual w_{ij}^{G+1} being adjusted as follows:

$$w_{ij}^{G+1} = \begin{cases} x_{ij}^{\max} + \text{rand}() * \left(x_{ij}^{\max} - u_{ij}^{G+1}\right), & \text{if } \left(u_{ij}^{G+1} > x_{ij}^{\max}\right), \\ x_{ij}^{\min} + \text{rand}() * \left(x_{ij}^{\min} - u_{ij}^{G+1}\right), & \text{if } \left(u_{ij}^{G+1} < x_{ij}^{\min}\right), \\ u_{ij}^{G+1}, & \text{otherwise.} \end{cases} \tag{3}$$

The new population is combined with the existing parent population to form a new set Mg of bigger size than Np. A nondominated ranking of Mg is performed, and the Np best individuals are selected. This approach allows a global nondomination checking between both the parent and the new generation rather than only in the new generation as is done in other approaches, whereas it requires additional computational cost in sorting the combined.

2.1.2. Nondominated Ranking Based on Euclidean Distance. The solutions within each nondominated frontier that reside in the less crowded region in the frontier are assigned a higher rank, as the NSGA-II algorithm [8] developed by Deb et al. indicated. The crowding distance of the ith solution in its frontier (marked with solid circles) is the average side length of the cuboids (shown with a dashed box in Figure 1(a)). The crowding-distance computation requires sorting the population according to each objective function value in ascending order of magnitude. As shown in Figure 1, A and C are two solutions near B in the same rank, and $\sigma_0(B)$ is the crowding distance of the Bth solution, traditionally calculated as follows:

$$\sigma_0(B) = \sum_{j=1}^{n} \left| f_j(A) - f_j(C) \right|, \tag{4}$$

where $f_j(A)$, $f_j(C)$ are the objective vectors. For each objective function, the boundary solutions (solutions with the smallest and the largest function values) are assigned an infinite distance value.

A crowding-distance metric is used to estimate the density of solutions surrounding a particular solution in the population and is obtained from the average distance of the two solutions on either side of the solution along each of the objectives. As shown in Figure 1, A, B, C are the individuals of the generation on the same frontier, and we easily know that the density in Figure 1(a) is better than that in Figure 1(b). If we use (4) to calculate the crowding distance of C, we only know that in Figure 1(a) it is better than in Figure 1(b); the crowding distance of C in Figures 1(a) and 1(c) is equal, which is against the knowledge.

To distinguish the mentioned situations, we propose an improved crowding-distance metric based on Euclidean distance. M is the center point of the line AB, f_j is the jth objective vector, and the crowding distance $\sigma(B)$ is defined as follows:

$$\begin{aligned} \sigma(B) &= |AC| - |BM| \\ &= \sqrt{\sum_{j=1}^{n} \left[f_j(A) - f_j(C)\right]^2} \\ &\quad - \sqrt{\sum_{j=1}^{n} \left\{ f_j(B) - \frac{\left[f_j(A) + f_j(C)\right]}{2} \right\}^2}. \end{aligned} \tag{5}$$

The crowded-comparison operator guides the selection process at the various stages of the algorithm toward a uniformly distributed Pareto-optimal frontier. To carry out the comparison, we assume that every individual in the population has two attributes: (1) nondomination rank i_{rank} and (2) crowding distance i_{distance}. Then, a partial order is defined as $i \prec j$. If $i \prec j$, that is, between two solutions with different nondomination ranks, we prefer the solution with the lower (better) rank, namely, $i_{\text{rank}} \prec j_{\text{rank}}$. Otherwise, if both solutions belong to the same frontier, that is, $i_{\text{rank}} = j_{\text{rank}}$, then we prefer the solution that is located in a lesser crowded region, that is, $i_{\text{distance}} \succ j_{\text{distance}}$.

2.1.3. Simplex Method for Local Search. The simplex method for local search is mixed in the evolution process to enhance the searching strategy in the optimization process. The goal of integrating NM simplex method [32] and DE is to enrich the population diversity and avoid being trapped in local minimum. We apply NM simplex method operator to the present population when the number of iterations is greater than a particular value (like $G_{\max}/2$). The individuals that achieved the single extreme value in each objective function are marked as the initial vertex points of simplex method. The

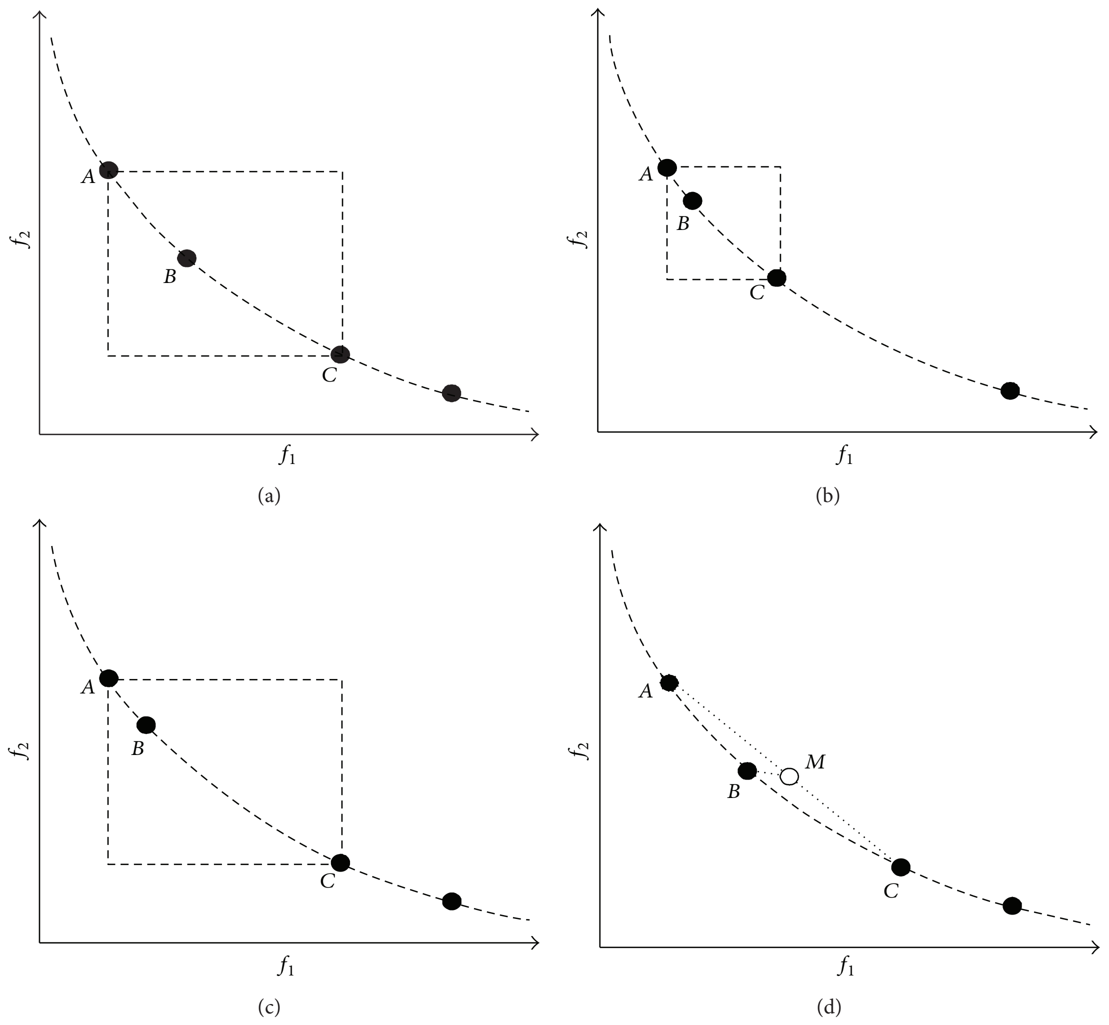


FIGURE 1: Crowding-distance diagram.

present population is updated according to simplex method until the terminal conditions are satisfied.

The computation steps of the algorithm are included in Section 3.2.

2.2. Evaluation Criteria. Unlike the single-objective optimization, it is more complicated for solution quality evaluation in the case of multiobjective optimization. Many of the suggested methods can be summarized in two types. One is to evaluate the convergence degree by computing the proximity between the solution frontier and the actual Pareto frontier. The other is to evaluate the distribution degree of the solutions in objective space by computing the distances among the individuals. Here, we choose both methods to evaluate the performance of the SM-DEMO algorithm.

(1) Convergence Evaluation. Deb et al. [8] proposed this method in 2002. It is described as follows:

$$\gamma = \frac{1}{Q}\left(\sum_{i=1}^{Q} \min \left\| P^* - P_{\mathrm{FT}} \right\|\right), \tag{6}$$

where γ is the extent of convergence to a known of Pareto-optimal set, P^* is the obtained nondomination Pareto frontier, P_{FT} is the real nondomination Pareto frontier, $\|P^* - P_{\mathrm{FT}}\|$ is the Euclidean distance of P^* with P_{FT}, and Q is the number of obtained solutions.

(2) Distribution Degree Evaluation. The nonuniformity in the distribution is measured by SP as follows:

$$\mathrm{SP} = \sqrt{\frac{1}{(Q-1)}\sum_{i=1}^{Q}\left(\overline{d} - d_i\right)^2}, \tag{7}$$

where d_i is the Euclidean distance among consecutive solutions in the obtained nondominated set of solutions and parameter $\overline{d}$ is the average distance.

2.3. Experimental Studies. Four well-known benchmark test functions [33] are used here to compare the performance of SM-DEMO with NSGA-II, DEMO/Parent. These four problems are called ZDT2, ZDT3, ZDT4, and ZDT6; each has two objective functions. We describe them in Table 1.

Table 1: Test problems.

Test problems	Objective functions $\min F(X) = \min[f_1(X), f_2(X)]$	Range of variable
ZDT2	$f_1(X) = x_1, f_2(X) = g(X)\left(1-\left(\frac{f_1}{g(X)}\right)^2\right)$, $g(X) = 1 + 9\sum_{i=2}^{n}\frac{x_i}{n-1}$	$n = 30$ $0 \le x_i \le 1$
ZDT3	$f_1(X) = x_1, f_2(X) = g\left(1-\sqrt{\left(\frac{f_1}{g}\right)} - \left(\frac{f_1}{g}\right)\sin(10\pi f_1)\right)$, $g(X) = 1 + 9\sum_{i=2}^{n}\frac{x_i}{n-1}$	$n = 30$ $0 \le x_i \le 1$
ZDT4	$f_1(X) = x_1, f_2(X) = g\left(1-\sqrt{\left(\frac{f_1}{g}\right)}\right)$, $g(X) = 1 + 10(n-1) + \sum_{i=2}^{n}\left(x_i^2 - 10\cos(4\pi x_i)\right)$	$n = 10$ $0 \le x_1 \le 1$ $-5 \le x_i \le 5,\ i = 2, \ldots, n$
ZDT6	$f_1(X) = 1 - \exp(-4x_1)\sin^6(6\pi x_1), f_2(X) = g\left(1-\left(\frac{f_1}{g}\right)^2\right)$, $g(X) = 1 + 9\sum_{i=2}^{n}\frac{x_i}{(n-1)^{0.25}}$	$n = 10$ $0 \le x_i \le 1$

Table 2: The performance results of the each algorithm on the test function.

Test function	Algorithm	γ	SP
ZDT2	DEMO/Parent	0.005120 ± 0.000312	0.000630 ± 0.000010
	NSGA-2	0.007120 ± 0.000413	0.000540 ± 0.000940
	SM-DEMO	0.004013 ± 0.000230	0.000423 ± 0.000011
ZDT3	DEMO/Parent	0.009704 ± 0.000027	0.007512 ± 0.000165
	NSGA-2	0.014067 ± 0.000059	0.006540 ± 0.000124
	SM-DEMO	0.004704 ± 0.000003	0.004450 ± 0.000153
ZDT4	DEMO/Parent	2.009704 ± 0.901164	0.011031 ± 0.001104
	NSGA-2	3.144067 ± 2.100740	0.010122 ± 0.000072
	SM-DEMO	0.874001 ± 0.014323	0.008721 ± 0.000159
ZDT6	DEMO/Parent	0.649704 ± 0.004912	0.104520 ± 0.015486
	NSGA-2	1.014067 ± 0.010421	0.007942 ± 0.000105
	SM-DEMO	0.007750 ± 0.000083	0.002014 ± 0.000117

The simulation is carried out under the environment of Intel Pentium 4, CPU 3.06 GHz, 512 MB memory, Windows XP Professional, Matlab7.1. Initialization parameters are set as follows: population size $Np = 100$, scaling factor $F = 0.8$, cross rate $C_R = 0.6$, maximum evolution generation $G_{\max} = 250$, and number of SM evolution iterations $G_{\mathrm{SM}} = 100$.

All of the three algorithms are real coded, with equal population size and equal maximum evolution generation. Each algorithm independently runs 20 times for each test function. Because we cannot get the real Pareto-optimal set, we will take 60 Pareto-optimal solutions obtained by the three algorithms as a true Pareto-optimal solution set.

We evaluated the algorithms based on the two performance indexes γ and SP. Table 2 shows the mean and variance of γ and SP using three algorithms: SM-DEMO, NSGA-II, and DEMO/Parent. We can learn from Table 2, for the ZDT2 function, that all of the three algorithms have a good performance, while the SM-DEMO is slightly better than the other two algorithms. In terms of convergences, for ZDT3, ZDT4 and ZDT6, which are more complex than ZDT2, SM-DEMO is significantly better than DEMO/Parent and NSGA-II.

Figure 2 shows a random running of SM-DEMO algorithm. It is clear that SM-DEMO algorithm can produce a good approximation and a uniform distribution.

3. Proposed Hybrid Algorithm for CMOP

The space of constrained multiobjective optimization problem can be divided into the feasible solution space and the infeasible solution space, as shown in Figure 3, where S is the search space, Ω is the feasible solution space, and Z is the infeasible solution space. x_i $(i = 1, 2, 3, 4)$ is the feasible solution, and y_i $(i = 1, 2, 3, 4)$ is the infeasible solution.

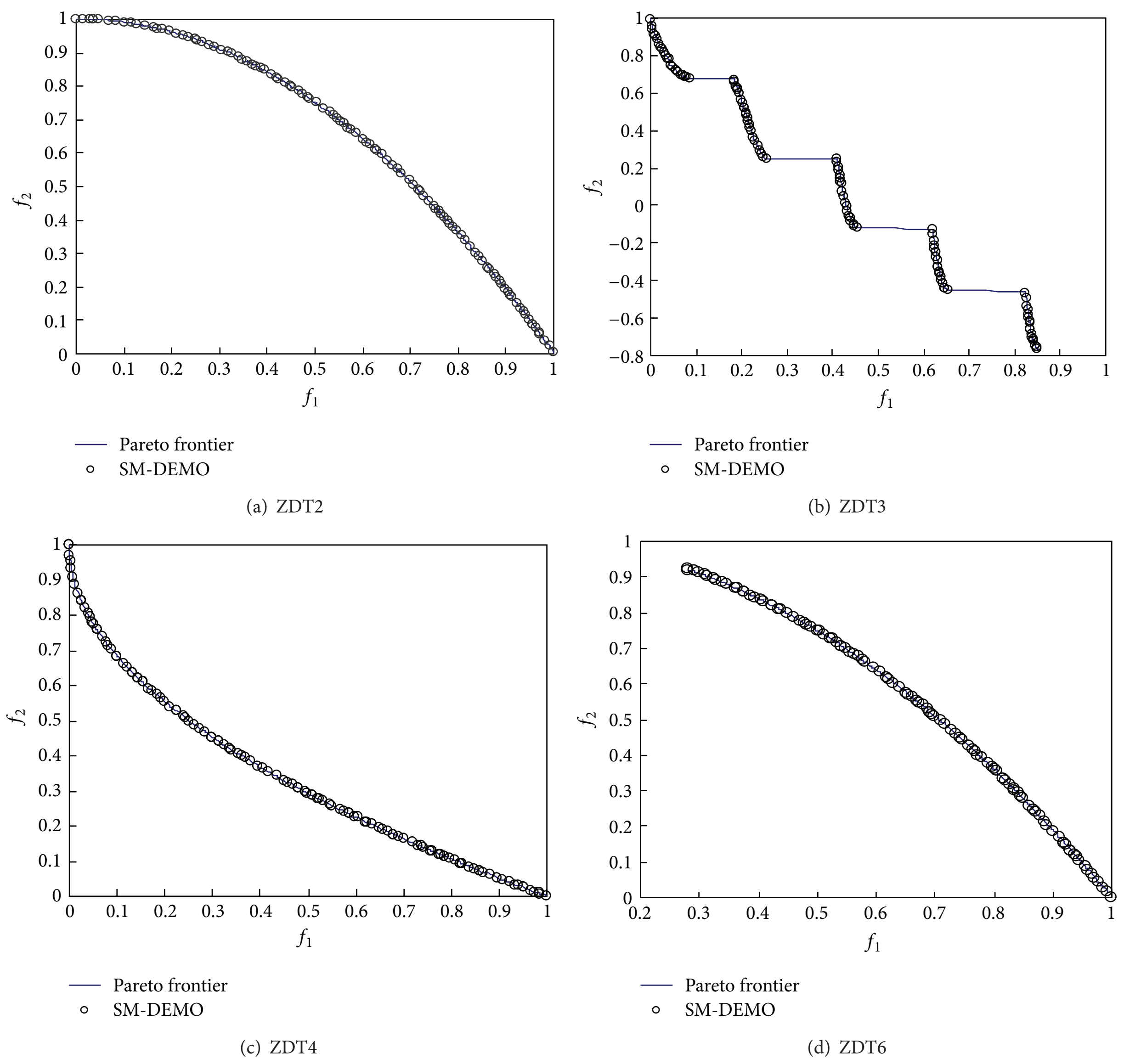


Figure 2: SM-DEMO simulation curve.

Assume that x^* is the global optimal solution and y_1 is the closest one to x^*. If the infeasible population y_1 is not excluded by the evolution algorithm, it is permitted to explore boundary regions from new directions, where the optimum is likely to be found.

3.1. General Idea of the Proposed Algorithm. Researchers have gradually realized the merit of infeasible solutions in searching for the global optimum in the feasible region. Some infeasible solutions with better performance are allowed to be saved. Farmani et al. [34] formulated a method to ensure that infeasible solutions with a slight violation become feasible in evolution. Based on the constraints processing approach of multiobjective optimization problems, the proposed hybrid DE algorithm avoids constructing penalty function and deleting meaningful infeasible solutions directly.

Here, the proposed algorithm will produce multiple groups of functional partitions, which include an evolutionary population Pg of size Np, an intermediate population Mg to save feasible individuals, an intermediate population Sg to save infeasible individuals, a population Pf to save the optimal feasible solution found in the search process and a population Pc to save the optimal infeasible solution. The relationship of multi-population is shown in Figure 4.

With the description of (1), equality constraints are always transformed into inequality constraints as $|h_j(X)| - \delta \leq 0$, where $j = p+1, \ldots, q$ and δ is a positive tolerance value. To evaluate the infeasible solution, the degree of constraint violation of individual X on the jth constraint is calculated as follows:

$$V_i(X) = \begin{cases} \max\{0, g_i(X)\} & (i = 1, 2, \ldots, p), \\ \max\{0, |h_j(X)| - \delta\} & (j = p+1, \ldots, q). \end{cases} \quad (8)$$

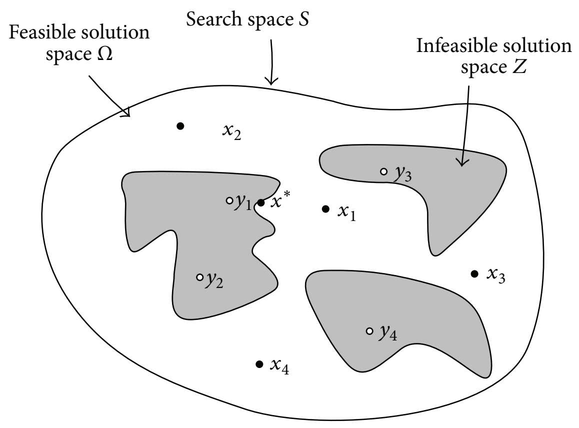


FIGURE 3: Distribution diagram of search space.

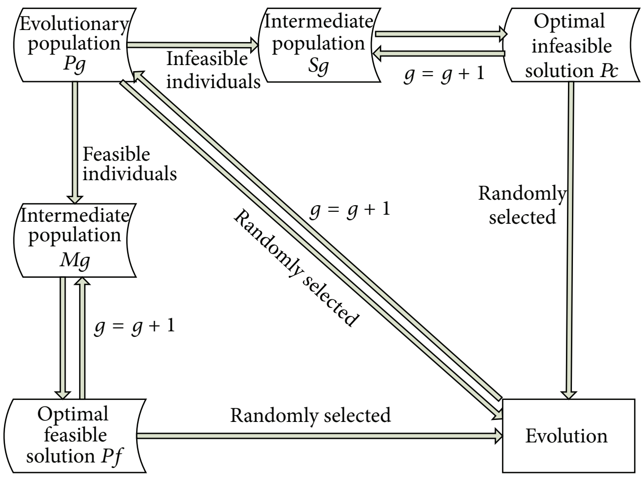

Evolutionary population Pg
Infeasible individuals
Intermediate population Sg
g = g + 1
Optimal infeasible solution Pc
Feasible individuals
g = g + 1
Randomly selected
Randomly selected
Intermediate population Mg
g = g + 1
Optimal feasible solution Pf
Randomly selected
Evolution

FIGURE 4: The relationship diagram of multipopulation demonstrating.

The final constraint violation of each individual in the population can be obtained by calculating the mean of the normalized constraint violations.

In order to take advantage of the infeasible solutions with better performance, we proposed the following adaptive differential mutation operator to generate individual variation learning from the mutation operators in DE/rand-to-best/1/bin, according to rules defined by Price et al. [11]. Considering each individual vector x_i^G, a mutant individual $\widehat{x}_i^{G+1}$ is defined by

$$\widehat{x}_i^{G+1} = \begin{cases} x_i^G + F_1 \cdot \left(x_{f1}^G - x_{r1}^G\right) + F_2 \cdot \left(x_{f2}^G - x_{r2}^G\right), & R_C \geq \text{rand}\,(\,), \\ x_i^G + F_1 \cdot \left(y_i^G - x_{r1}^G\right) + F_2 \cdot \left(x_{f1}^G - x_{r2}^G\right), & R_C < \text{rand}\,(\,), \end{cases} \tag{9}$$

where r_1 and r_2 represent different integers and also different from i, randomly generated over $[1, Np]$; F is the scaling factor; x_{fi}^G $(i = 1, 2, \ldots, n)$ is randomly generated from Pf, y_i^G $(i = 1, 2, \ldots, n)$ is randomly generated from Pc; and R_C is the mutation factor as follows:

$$R_C = Rc_0 \cdot \frac{\gamma^G + \text{const}}{\gamma^{G-1} + \text{const}}, \tag{10}$$

where Rc_0 is the initial value of the variability factor, const is a small constant, to ensure that the fractional is meaningful, and γ^G is defined as follows:

$$\gamma^G = \frac{\text{the number of infeasible solutions in } Pg}{Np}. \tag{11}$$

3.2. Framework of the Proposed Algorithm. The proposed algorithm is described as follows.

Step 1 (initialization). Generate the population Pg, Pf, and Pc of size Np, NP_1, and NP_2. Set the value of C_R (crossover probability), $G_{\max}$ (the number of function evaluations), G_{SM} (the iterative number of evolution by NM simplex method), $g = 1$ (the current generation number), and positive control parameter for scaling the difference vectors F_1, F_2. Randomly generate the parent population Pg from the decision space. Set the Pf, and Pc, and let the intermediate populations Sg and Mg be empty.

Step 2 (DE reproduction). By (3) and (9) for mutation, crossover, and selection, an offspring Sg is created. Judge the constraints of all individuals in Pg. In accordance with (8), we first calculate constraint violation degree $V_i(X)$ of all of the individuals. If $V_i(X) = 0$, the solution is feasible and preserved to the intermediate set Mg; if $V_i(X) > 0$, the solution is infeasible and preserved to the intermediate set Sg.

Step 3 (simplex method). Apply NM simplex method operator to the present population if $g \geq G_{\max}/2$. Update the present population Mg when the number of iterations exceeds maximum iterations.

Step 4 (Pf construction). Rank chromosomes in Mg based on (5), and generate the elitist population Pf (the size is Np) from the ranked population Mg.

Step 5 (Pc construction). Add the chromosomes in Sg with slight constraint violations to the Pc.

Step 6 (mixing the population). Combine Sg with the existing parent population to form a new set Mg. Remove the duplicate individuals in Mg and the existing parent population.

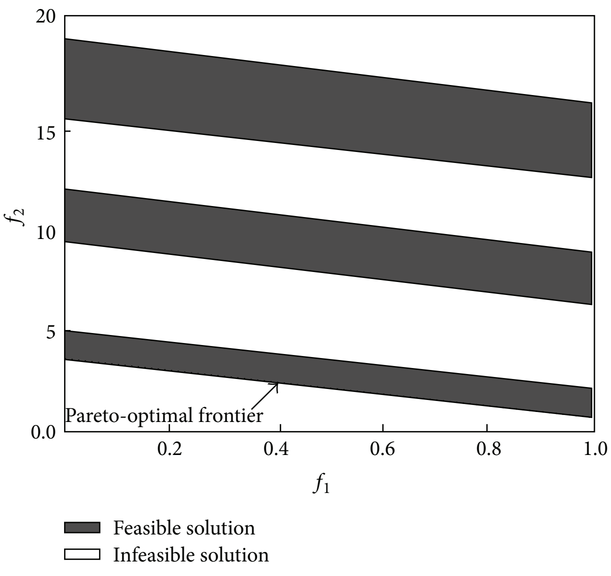
Step 7 (evolution). Randomly choose chromosomes from Pc, Pg, and Pf. Use the adaptive differential mutation and uniform discrete crossover to obtain the offspring population $Pg + 1$.

Step 8 (termination). If the stopping criterion is met, stop and output the best solution; else, go to Step 2.

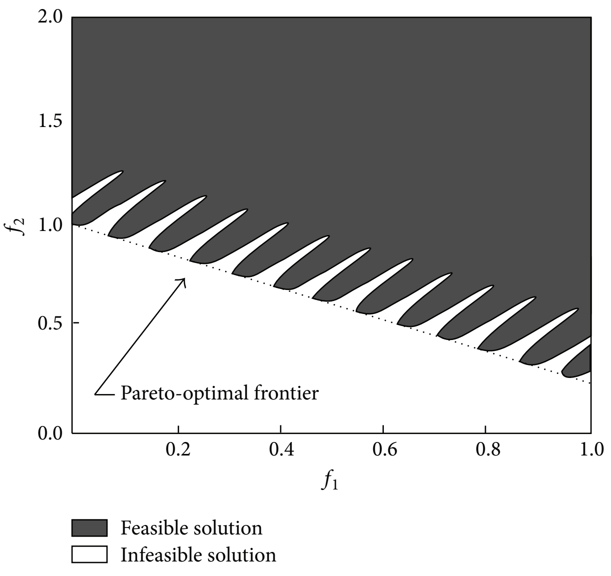
3.3. Experimental Study. In this section, we choose three problems CTP, TNK, and BNH, as shown in Table 3, to test the proposed method, and compare the method with the current CNSGA-II [35].

Table 3: Test functions.

Test function	Objective function $\min F(X) = \min[f_1(X), f_2(X)]$	Constraints	Range of variable
CTP	$f_1(X) = x_1$, $f_2(X) = \exp\left(-\frac{f_1(X)}{c(X)}\right) \times \left\{41 + \sum_{i=2}^{5}\left[x_i^2 - 10\cos(2\pi x_i)\right]\right\}$	$g_1(X) = \cos(\theta)(f_2(X) - e) - \sin(\theta) f_1(X)$, $g_2(X) = a\left\|\sin\{b\pi[\sin(\theta)(f_2(X) - e) + \cos(\theta) f_1(X)]^c\}\right\|^d$, $g_1(X) \geq g_2(X)$	$0 \leq x_1 \leq 1$, $-5 \leq x_i \leq 5$, $i = 2, 3, 4, 5$
BNH	$f_1(X) = 4x_1^2 + 4x_2^2$, $f_2(X) = (x_1 - 5)^2 + (x_2 - 5)^2$	$g_1(X) = (x_1 - 5)^2 + x_2^2 - 25$, $g_2(X) = -(x_1 - 8)^2 + (x_2 + 3) + 7.7$, $g_1(X) \leq 0$, $g_2(X) \leq 0$	$0 \leq x_1 \leq 5$, $0 \leq x_2 \leq 3$
TNK	$f_1(X) = x_1$, $f_2(X) = x_2$	$g_1(X) = -x_1^2 - x_2^2 + 1 + 0.1\cos\left(16\arctan\left(\frac{x_1}{x_2}\right)\right)$, $g_2(X) = (x_1 - 0.5)^2 + (x_2 - 0.5)$, $g_1(X) \leq 0$, $g_2(X) - 0.5 \leq 0$	$0 \leq x_i \leq \pi$, $i = 1, 2$



(a) CTP1



(b) CTP2

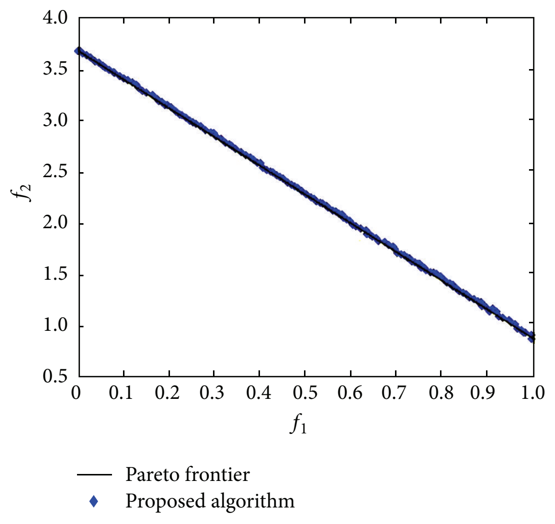
Figure 5: CTP solution space.

For CTP problem, there are the six parameters θ, a, b, c, d, and e that must be chosen in a way so that a portion of the unconstrained Pareto-optimal region is infeasible. Each constraint is an implicit non-linear function of decision variables. Thus, it may be difficult to find a number of solutions on a non-linear constraint boundary. We take two sets of values for six parameters in CTP problem, which are determined as (1) CTP1: $\theta = 0.1\pi$, $a = 40$, $b = 0.5$, $c = 1$, $d = 2$, and $e = -2$; (2) CTP2: $\theta = -0.2\pi$, $a = 0.2$, $b = 10$, $c = 1$, $d = 6$, and $e = 1$. The Pareto frontiers, the feasible solution spaces, and the infeasible solution spaces are shown in Figure 5.

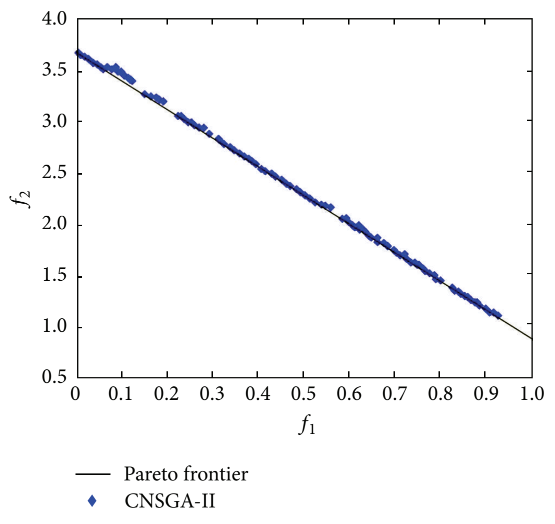
The parameters are initialized as follows. The size of population Pg is $Np = 200$, size of Pf is $NP_1 = 150$, size of Pc is $NP_2 = 10$, scaling factors F_1 and F_2 are randomly generated within $[0.5, 1]$, cross rate is $C_R = 0.6$, maximum evolution generation is $G_{\max} = 300$, and number of SM evolution iterations is $G_{\text{SM}} = 100$. All of the proposed algorithms and CNSGA-II are real coded with equal population size and equal maximum evolution generation. Each algorithm runs 20 times independently for each test function.

Figure 6 shows the result of a random running of the proposed algorithm and NSGA-II, the smooth curve "—" represents the Pareto frontier, and "◆" stands for the solution achieved by the proposed algorithm or NSGA-II.

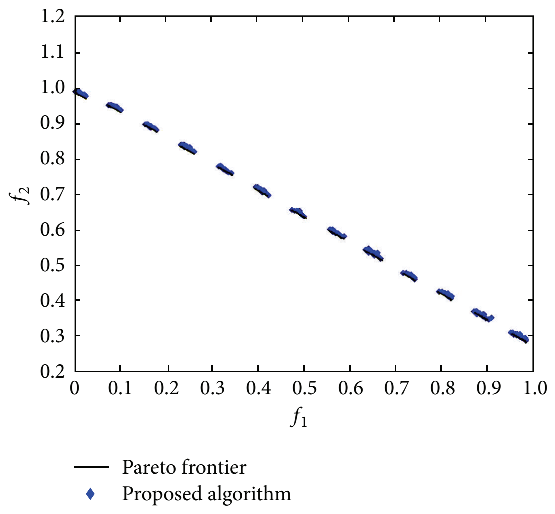
It is obvious that the proposed algorithm returns a better approximation to the true Pareto-optimal frontie and a distribution of higher uniformity. We also evaluated the



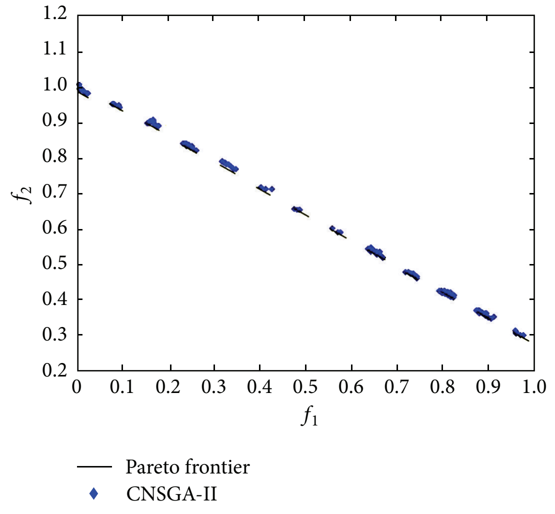
(a) CTP1 calculated by the proposed algorithm



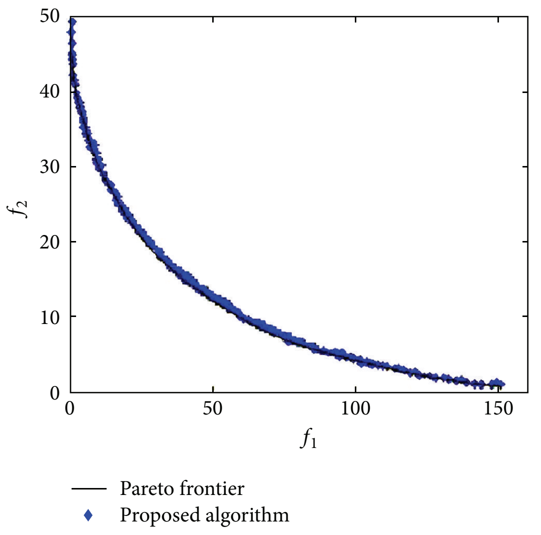
(b) CTP1 calculated by CNSGA-II



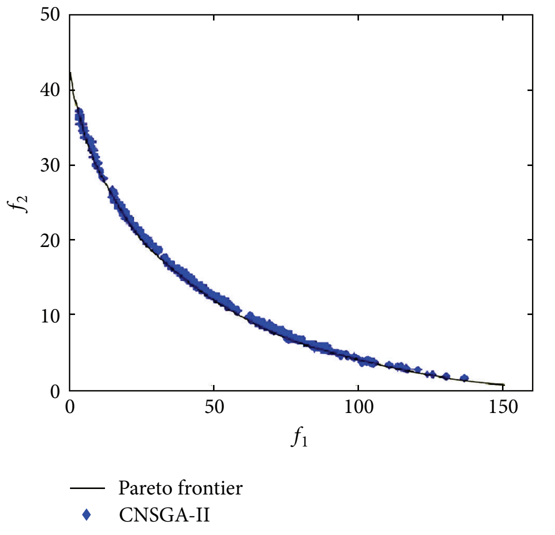
(c) CTP2 calculated by the proposed algorithm



(d) CTP2 calculated by CNSGA-II

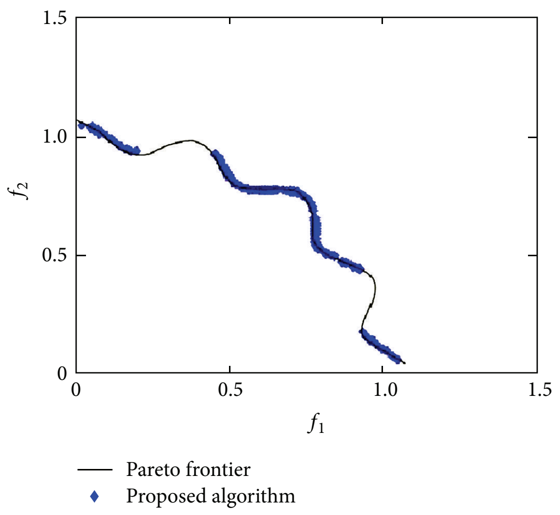


(e) BNH calculated by the proposed algorithm

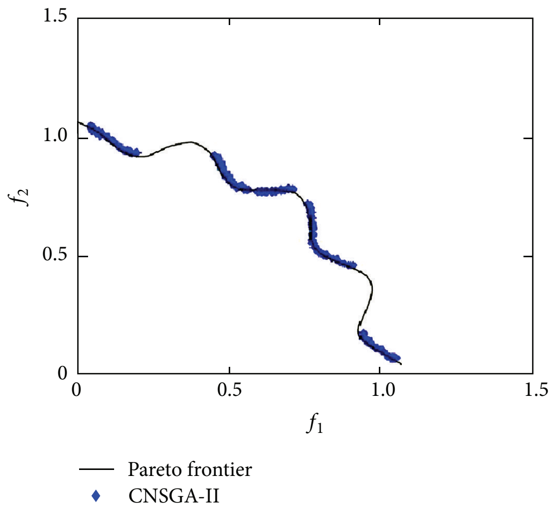


(f) BNH calculated by CNSGA-II

FIGURE 6: Continued.



(g) TNK calculated by the proposed algorithm



(h) TNK calculated by CNSGA-II

FIGURE 6: The comparison chart of Pareto frontier.

TABLE 4: The comparison of performance.

Test function	Algorithm	γ	SP
CTP1	CNSGA-Π	0.021317 ± 0.000323	0.873321 ± 0.08725
	The proposed	0.009836 ± 0.000410	0.567933 ± 0.01845
CTP2	CNSGA-Π	0.011120 ± 0.000753	0.78314 ± 0.02843
	The proposed	0.007013 ± 0.000554	0.296543 ± 0.00453
BNH	CNSGA-Π	0.014947 ± 0.000632	0.336941 ± 0.00917
	The proposed	0.013766 ± 0.000043	0.209321 ± 0.00561
TNK	CNSGA-Π	0.013235 ± 0.000740	0.464542 ± 0.00730
	The proposed	0.006435 ± 0.000017	0.224560 ± 0.00159

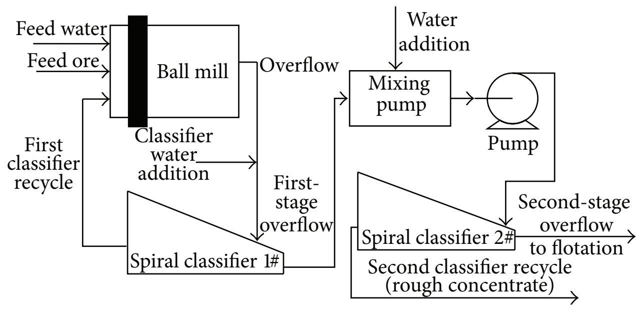


FIGURE 7: Flow diagram of the grinding and classification process.

algorithms based on the two criterions γ and SP, as shown in Table 4. It can be observed from the data in Table 4 that the proposed algorithm performs significantly better than the classical CNSGA-II algorithm in convergence and distribution uniformity. The simulation results show that this algorithm can accurately converge to global Pareto solutions and can maintain diversity of population.

4. Optimization of Grinding and Classification Process

4.1. Bauxite Grinding and Classification Process. The grinding and classification process is the key preparation for the bauxite mineral processing. Here, we consider a bauxite grinding process in a certain mineral company with single grinding and two-stage classification, as shown in Figure 7.

The process consists in a grinding ball mill and two spiral classifiers. First classifier recycle will be put back to the ball mill for regrinding, and the first-stage overflow will be put into second spiral classifier after being mixed with water; the second classifier recycle will be prepared for Bayer production as the rough concentrate, and the second-stage overflow will be sent to the next flotation process. The production objectives are composed of the production yields, technically represented by the solid flow of feed ore since the process is nonstorable, and the mineral processing quality, represented by percentage of the small-size fractions of mineral particles in the second-stage overflows.

Table 5: Notations for the model of the grinding and classification process.

Notation	Description
p_i	Particle percentage of the ith size fraction in the ball mill overflow
f_j	Particle percentage of the ith size fraction in the Feed ore
C	Rate of the first classifier recycle
Ea_i	The efficiency of the first spiral classifier
τ	The mean residence time
P_1	The internal concentration in ball mill
M_{MF}	The solid flow of feed ore
W_1	The water addition of the first classifier recycle
W_2	The classifier water addition
b_{ij}	The breakage distribution function
S_i	The breakage rate function
d_i	The particle with the ith size
αc_i	The particle percentage of the ith size in the first classifier overflow
d_o	The unit size of the particle
P_2	First-stage overflow
ϕ_B	Ball filling rate
A_c, B_c	Parameters of the first-stage classifier overflow size fraction distribution
Ea_i	The efficiency of the first spiral classifier
d_{50c}	The particle size fraction after correction separation
m	The separation accuracy
k	The intermix index
$Ea'_{\min}, d'_{50c}, m', k'$	The corresponding key parameters to the efficiency of the second spiral classifier
a, α, μ, Λ	Four key parameters to control the breakage rate function
$d_{\min}, d_{\max}$	The minimum and maximum particle sizes
$A_{\mathrm{MF}}, B_{\mathrm{MF}}$	Parameters of feed ore size fraction distribution
$F(i)$	The cumulative particle percentage less than the ith size fraction in feed ore
$\alpha c'_i$	The particle percentage of the ith size fraction in the second classifier overflow
Ea'_i	The efficiency of the second spiral classifier

4.2. Predictive Model of the Grinding and Classification Process. Here, we establish the mathematical predictive model of each unit process in the bauxite grinding and classification process. The notations of the indexes, decision variables, and parameters are listed in Table 5. These notations will be used for the model of the grinding and classification process.

4.2.1. Ball Mill Circuit Model. Here, p_i is the particle percentage of ith size fraction in the ball mill overflow, f_j is the particle percentage of ith size fraction in feed ore, rate of the first classifier recycle C is known, and Ea_i is the efficiency of the first spiral classifier. According to a technical report of field investigation and study, we have that

$$p_i(1+C) = \frac{d_{ii}f_i + \sum_{j=1,i>1}^{i-1} d_{ij}\left[Ea_i p_j(1+C) + f_j\right]}{1 - d_{ii}Ea_i},$$

$$d_{ij} = \begin{cases} e_j, & i = j, \\ \displaystyle\sum_{k=j}^{i-1} c_{ik}c_{jk}(e_k - e_i), & i > j, \end{cases}$$

$$c_{ij} = \begin{cases} -\displaystyle\sum_{k=i}^{j-1} c_{ik}c_{jk}, & i < j, \\ 1, & i = j, \\ \displaystyle\frac{1}{S_i - S_j}\sum_{k=j}^{i-1} S_k b_{ik} c_{kj}, & i > j, \end{cases}$$

$$e_j = \frac{1}{\left(1 + 0.5\cdot\tau S_j\right)\left(1 + 0.25\cdot\tau S_j\right)^2}, \qquad \tau = \frac{8P_1}{M_{\mathrm{MF}}}, \tag{12}$$

where τ is the mean residence time of minerals, P_1 is the internal concentration in ball mill, and

$$P_1 = \frac{M_{\mathrm{MF}}(1+C)}{M_{\mathrm{MF}}(1+C) + W_1 + 0.3(W_1 + W_2)}, \tag{13}$$

where M_{MF} (t/h) is the solid flow of feed ore, W_1 is the water addition of the first classifier recycle, and W_2 is the classifier water addition. b_{ij} is the breakage distribution function; S_i is the breakage rate function, and it satisfied the following equation:

$$S_i = \frac{\left(a(d_i/d_o)^{\alpha}\right)}{\left(1 + (d_i/\mu)^{\Lambda}\right)}, \tag{14}$$

where d_i is the particle with the ith size, d_o it is a unit, when per millimeter is a unit, $d_o = 1$, $d_i = i$ (mm), and a, α, μ, and Λ are four key parameters to control the breakage rate function.

In a concrete grinding and classification process, the ball mill size is fixed, and the speed of ball mill is constant. Through data acquisition and testing of grinding and classification steady-state loop, the regression model between a, α, μ, Λ and condition variables, size fraction distribution can be established. The input variables are ball filling rate ϕ_B, solid flow of feed ore M_{MF}, water addition of the first classifier recycle W_1, and parameters of feed ore size fraction distribution A_{MF}, B_{MF}. The regression model is

$$\begin{bmatrix} a & \alpha & \mu & \Lambda \end{bmatrix}^T = \begin{bmatrix} x_{11} & \cdots & x_{1j} \\ \vdots & \ddots & \vdots \\ x_{i1} & \cdots & x_{ij} \end{bmatrix} \cdot \begin{bmatrix} W_1 & \phi_B & A_{\mathrm{MF}} & B_{\mathrm{MF}} & M_{\mathrm{MF}} \end{bmatrix}^T. \tag{15}$$

The value of x_{ij} $(i = 1,2,3,4;\ j = 1,2,\ldots,5)$ can be obtained by the experimental data regression, A_{MF}, B_{MF} can be obtained from feed ore size fraction distribution, and $F(i)$ is the cumulative particle percentage less than the ith size fraction in feed ore, and it is represented as follows:

$$F(i) = 1 - \exp\left(-A_{\mathrm{MF}} d_i^{B_{\mathrm{MF}}}\right). \quad (16)$$

4.2.2. Spiral Classifier Model. αc_i is the particle percentage of the ith size fraction in the first classifier overflow, and p_i is the particle percentage of the ith size fraction in ball mill overflow. The spiral classifier model is as follows:

$$\alpha c_i = \frac{p_i \times (1 - Ea_i)}{\sum_{i=1}(p_i \times (1 - Ea_i))} \times 100\%, \quad (17)$$

where Ea_i is the efficiency of the first spiral classifier and the mechanism formula of Ea_i is shown as follows:

$$\begin{aligned} Ea_i = {} & 1 - \exp\left[-0.693\left(\frac{d_i - d_{\min}}{d_{50c} - d_{\min}}\right)^m\right] \\ & + Ea_{\min} \cdot \left[1 - \left(\frac{d_i - d_{\min}}{d_{\max}}\right)\right]^k, \end{aligned} \quad (18)$$

where d_i is the particle with the ith size, $d_{\min}$ and $d_{\max}$ represent maximum and minimum particle sizes, d_{50c} is the particle size fraction after correction separation, m is separation accuracy, and k is intermix index.

Through data acquisition and testing of grinding and classification steady-state loop, the regression model between classification parameters and condition variables, size fraction distribution can be established. The input variables include the solid flow of feed ore M_{MF}, the classifier water addition W_2, and the parameters of ball mill overflow size fraction distribution A_{MF}, B_{MF}. The regression model is shown as follows:

$$\begin{aligned} \left[Ea_{\min} \;\; d_{50c} \;\; m \;\; k\right]^T = {} & \begin{bmatrix} y_{11} & \cdots & y_{1j} \\ \vdots & \ddots & \vdots \\ y_{i1} & \cdots & y_{ij} \end{bmatrix} \\ & \cdot \left[M_{\mathrm{MF}} \;\; W_2 \;\; A_{\mathrm{MP}} \;\; B_{\mathrm{MP}}\right]^T, \end{aligned} \quad (19)$$

where the value of y_{ij} $(i = 1, 2, 3, 4; j = 1, 2, 3, 4)$ can be obtained by data regression.

The first-stage overflow P_2 calculation formula is as follows:

$$P_2 = \frac{M_{\mathrm{MF}}}{(M_{\mathrm{MF}} + W_1 + W_2)}. \quad (20)$$

Similarly, we can get the second spiral classifier model as follows:

$$\alpha c_i' = \frac{\alpha c_i \times \left(1 - Ea_i'\right)}{\sum_i (\alpha c_i \times (1 - Ea'))} \times 100\%, \quad (21)$$

where $\alpha c_i'$ is the particle percentage of the ith size fraction in the second classifier overflow, αc_i is the particle percentage of the ith size fraction in the first classifier overflow, and Ea_i' is the efficiency of the second spiral classifier. The spiral classifier model is as follows:

$$\begin{aligned} Ea_i' = {} & 1 - \exp\left[-0.693\left(\frac{d_i' - d_{\min}'}{d_{50c}' - d_{\min}'}\right)^{m'}\right] \\ & + Ea_{\min}' \cdot \left[1 - \left(\frac{d_i' - d_{\min}'}{d_{\max}'}\right)\right]^{k'}, \end{aligned} \quad (22)$$

where, $Ea_{\min}'$, d_{50c}', m', and k' are key parameters to the efficiency of the second spiral classifier. Through data acquisition and testing of grinding and classification steady-state loop, the regression model between classification parameters and condition variables, size fraction distribution can be established. The input variables include solid flow of feed ore M_{MF} and parameters of the first-stage classifier overflow size fraction distribution A_c, B_c, which are solved by similar equation to (20). The regression model is shown as follows:

$$\begin{aligned} \left[Ea_{\min}' \;\; d_{50c}' \;\; m' \;\; k'\right]^T = {} & \begin{bmatrix} y_{11}' & \cdots & y_{1j}' \\ \vdots & \ddots & \vdots \\ y_{i1}' & \cdots & y_{ij}' \end{bmatrix} \\ & \cdot \left[M_{\mathrm{MF}} \;\; A_c \;\; B_c\right]^T, \end{aligned} \quad (23)$$

where the value of y_{ij}' $(i = 1, 2, 3, 4; j = 1, 2, 3)$ can be obtained by experimental data regression.

4.3. Optimization Model of Grinding and Classification Process. Two objective functions in the process are identified: one is to maximize output $f_1(X)$, and the other is to maximize the small-size fractions (less than 0.075 mm fractions) in the second-stage overflow $f_2(X)$. It is also necessary to ensure that the grinding product meets all of other technical requirements and the least disturbance in the following flotation circuit. As the constraints, the feed load of the grinding circuit M_{MF}, the steel ball filling rate ϕ_B, the first and the second overflows P_1 and P_2, and the particle percentage of fine size fraction in the first and the second classifier overflows $\alpha c_{-0.075}$ and $\alpha c_{-0.075}'$ should be within the user specified bounds.

The operation variables are the solid flow of feed ore M_{MF}, water addition of the first classifier recycle W_1, ball filling rate ϕ_B, and water addition of the second classifier W_2. Based on all of the above, grinding and classification process multiobjective optimization model is as follows:

$$\begin{aligned} \max \quad & F = \max\left[f_1(X), f_2(X)\right], \\ & f_1(X) = M_{\mathrm{MF}}, \\ & f_2(X) = \alpha c_{-0.075}' \\ & \qquad = f\left(M_{\mathrm{MF}}, W_1, W_2, \phi_B\right), \\ \text{s.t.} \quad & M_{\mathrm{MFmin}} \le M_{\mathrm{MF}} \le M_{\mathrm{MFmax}}, \\ & \phi_{B\min} \le \phi_B \le \phi_{B\max}, \\ & P_{1\min} \le P_1 \le P_{1\max}, \\ & P_{2\min} \le P_2 \le P_{2\max}, \\ & \alpha c_{-0.075} \ge \alpha c_{\min}, \\ & \alpha c_{-0.075}' \ge \alpha c_{\min}'. \end{aligned} \quad (24)$$

Table 6: The optimization results calculated by the proposed algorithm.

Number	$f_1(X)$ (t/h)	$f_2(X)$ (%)
1	92.972	90.814
2	91.810	91.900
3	92.360	90.923
4	90.620	93.460
5	91.310	92.494
6	90.170	93.800
7	89.390	95.200
8	89.480	94.932
9	91.530	92.400
10	89.298	95.763
11	89.824	94.549
12	89.800	94.395
13	89.710	94.618
14	91.170	92.800
15	91.880	91.840
16	92.190	91.281
17	89.870	94.090
18	91.000	93.285
19	91.960	91.520
20	91.124	93.221

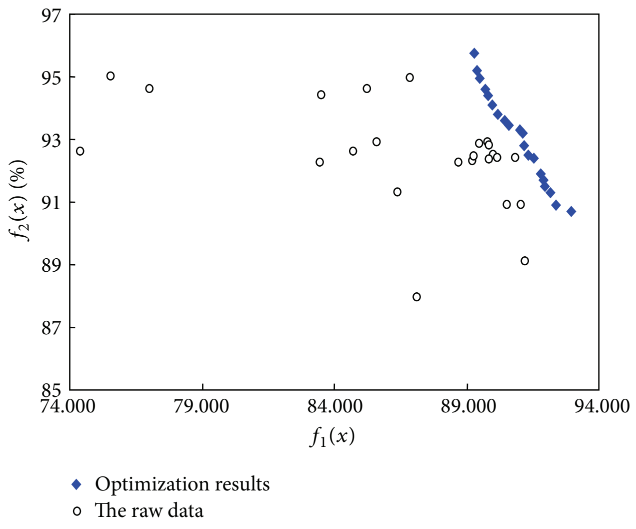


Figure 8: The comparison chart between optimization results and industrial data.

In (24), $f(M_{\mathrm{MF}}, W_1, W_2, \phi_B)$ implicates the model of grinding and classification represented by (12)–(23). The proposed algorithm is applied to solve the problem, and the optimization results are shown in Table 6.

With the practical process data from a grinding circuit of a mineral plant, the simulation of this hybrid intelligent method adopted the same parameters on the variation in fresh slurry feed velocity, density, particle size distribution, and cyclone feed operating configurations.

The comparison of production data and optimization results in Table 6 is shown in Figure 8, where "◆" represents the proposed algorithm optimization results and "○" represents the original data collected from the field without optimization of raw data. According to the objectives, the data point closer to the upper right edge is more beneficial. Obviously, the proposed optimization result is far better than the original data, indicating the effectiveness of the optimization approach.

4.4. TOPSIS Method for Solution Selection. The resolution of a multiobjective optimization problem does not end when the Pareto-optimal set is found. In practical operational problems, a single solution must be selected. TOPSIS [36] is a useful technique in dealing with multiattribute or multicriteria decision-making (MADM/MCDM) problems in the real world. The standard TOPSIS method attempts to choose alternatives that simultaneously have the shortest distance from the positive-ideal solution and the farthest distance from the negative-ideal solution. According to the TOPSIS method, the relative closeness coefficient is calculated, and the best solution in Table 6 is the solution number 10 as $f_1(X) = 89.298$ (t/h) and $f_2(X) = 95.763$ (%). The corresponding decision variables are $M_{\mathrm{MF}} = 89.298$ (t/h), $\phi_B = 32\%$, $W_1 = 75.127$ (t/h), and $W_2 = 16.296$ (t/h).

5. Conclusions

Promoted by the requirements of engineering optimization in complex practical processes of grinding and classification, we proposed a hybrid multiobjective differential evolution algorithm with a few beneficial features integrated. Firstly, an archiving mechanism for infeasible solutions is established with functional partitioned multi-population, which aims to direct the population to approach or land in the feasible region from different directions during evolution. Secondly, we propose an infeasible constraint violations function to select infeasible population with better performance, so that they are allowed to be saved and to participate in the subsequent evolution. Thirdly, a nondominating ranking strategy is designed to improve the crowding-distance sorting and return uniform distribution of Pareto solutions. Finally, the simplex method is inserted in the differential evolution process to purposefully enrich the diversity without excessive computation cost. The advantage of the proposed algorithm is the exemption from constructing penalty function and the preservation of meaningful infeasible solutions directly. Simulation results on benchmarks indicate that the proposed algorithm can converge quickly and effectively to the true Pareto frontier with better distribution.

Based on the investigated information about grinding circuit process, we established a multiobjective optimal model with equations from mechanism knowledge, parameters recognized by data regression, and constraints of technical requirements. The nonlinear multiobjective optimization model is too complicated to be solved by traditional gradient-based algorithms. The proposed hybrid differential evolution algorithm is applied and tested to achieve a Pareto solution set. It is proven to be valuable for operation decision making in the industrial process and showed superiority to the operation carried out in the production. In fact, many operating parameters in complex processes are highly coupled and conflicting with each other. The optimal operation of the entire production process is very difficult to obtain by manual calculation; let alone the fluctuation situation of process conditions. The application case indicates that the proposed method has good performance and is helpful to inspire further research on evolutionary methods for engineering optimization.

Conflict of Interests

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

Acknowledgments

This work is supported by the National Natural Science Foundation of China (61374156, 61273187, and 61134006), the National Key Technology Research and Development Program of the Ministry of Science and Technology of China (2012BAK09B04), the Science Fund for Creative Research Groups of the National Natural Science Foundation of China (61321003), the Fund for Doctor Station of the Ministry of Education (20110162130011 and 20100162120019) and the Hunan Province Science and Technology Plan Project (2012CK4018).

References

[1] K. Mitra and R. Gopinath, "Multiobjective optimization of an industrial grinding operation using elitist nondominated sorting genetic algorithm," *Chemical Engineering Science*, vol. 59, no. 2, pp. 385–396, 2004.

[2] G. Yu, T. Y. Chai, and X. C. Luo, "Multiobjective production planning optimization using hybrid evolutionary algorithms for mineral processing," *IEEE Transactions on Evolutionary Computation*, vol. 15, no. 4, pp. 487–514, 2011.

[3] T. Y. Ma, W. H. Gui, Y. L. Wang, and C. H. Yang, "Dynamic optimization control for grinding and classification process," *Control and Decision*, vol. 27, no. 2, pp. 286–290, 2012 (Chinese).

[4] F. Y. Edgeworth, *Mathematical Physics: An Essay on the Application of Mathematics to the Moral Sciences*, C. Kegan Paul & Company, London, UK, 1881.

[5] V. Pareto, *Cours d'Economie Politique*, F. Rouge, Lausanne, Switzerland, 1896.

[6] X. F. Chen, W. H. Gui, Y. L. Wang, and L. Cen, "Multi-step optimal control of complex process: a genetic programming strategy and its application," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 5, pp. 491–500, 2004.

[7] C. A. C. Coello, "Evolutionary multi-objective optimization: a historical view of the field," *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 28–36, 2006.

[8] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[9] C. M. Fonesca and P. J. Fleming, "Genetic algorithms for multi objective optimization: formulation, discussion and generalization," in *Proceedings of the 5th International Conference on Genetic Algorithms*, pp. 415–423, Morgan Kaufmann, San Mateo, Calif, USA, 1993.

[10] S. Sengupta, S. Das, M. Nasir, A. V. Vasilakos, and W. Pedrycz, "Energy-efficient differentiated coverage of dynamic objects using an improved evolutionary multi-objective optimization algorithm with fuzzy-dominance," in *Proceedings of IEEE Congress on Evolutionary Computation*, pp. 1–8, IEEE Press, Brisbane, Australia, 2012.

[11] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution a Practical Approach to Global Optimization*, Springer, Berlin, Germany, 2005.

[12] W. Y. Gong and Z. H. Cai, "An improved multiobjective differential evolution based on Pareto-adaptive ε-dominance and orthogonal design," *European Journal of Operational Research*, vol. 198, no. 2, pp. 576–601, 2009.

[13] X. H. Zeng, W. K. Wong, and S. Y. Leung, "An operator allocation optimization model for balancing control of the hybrid assembly lines using Pareto utility discrete differential evolution algorithm," *Computers and Operations Research*, vol. 39, no. 5, pp. 1145–1159, 2012.

[14] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[15] A. Al-Ani, A. Alsukker, and R. N. Khushaba, "Feature subset selection using differential evolution and a wheel based search strategy," *Swarm and Evolutionary Computation*, vol. 9, pp. 15–26, 2013.

[16] G. Y. Li and M. G. Liu, "The summary of differential evolution algorithm and its improvements," in *Proceedings of the 3rd International Conference on Advanced Computer Theory and Engineering (ICACTE '10)*, pp. V3153–V3156, Chengdu, China, August 2010.

[17] H. A. Abbass, "The self-adaptive Pareto differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 831–836, IEEE Press, Honolulu, Hawaii, USA, May 2002.

[18] N. K. Madavan, "Multiobjective optimization using a Pareto differential evolution approach," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 1145–1150, IEEE Press, Honolulu, Hawaii, USA, May 2002.

[19] F. Xue, A. C. Sanderson, and R. J. Graves, "Pareto-based multi-objective differential evolution," in *Proceedings of the IEEE Congress on Evolutionary Computation,*, pp. 862–869, IEEE Press, Piscataway, NJ, USA, December 2003.

[20] T. Robic and B. Filipi, "DEMO: differential evolution for multiobjective optimization," in *Evolutionary Multi-Criterion Optimization*, pp. 520–533, Springer, Berlin, Germany, 2005.

[21] V. L. Huang, A. K. Qin, P. N. Suganthan, and M. F. Tasgetiren, "Multi-objective optimization based on self-adaptive differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '07)*, pp. 3601–3608, IEEE Press, Singapore, September 2007.

[22] V. L. Huang, S. Z. Zhao, R. Mallipeddi, and P. N. Suganthan, "Multi-objective optimization using self-adaptive differential evolution algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '09)*, pp. 190–194, IEEE Press, Trondheim, Norway, May 2009.

[23] J. A. Adeyemo and F. A. O. Otieno, "Multi-objective differential evolution algorithm for solving engineering problems," *Journal of Applied Sciences*, vol. 9, no. 20, pp. 3652–3661, 2009.

[24] Y. Wang and Z. Cai, "Combining multiobjective optimization with differential evolution to solve constrained optimization problems," *IEEE Transactions on Evolutionary Computation*, vol. 16, no. 1, pp. 117–134, 2012.

[25] A. Homaifar, C. X. Qi, and S. H. Lai, "Constrained optimization via genetic algorithms," *Simulation*, vol. 62, no. 4, pp. 242–253, 1994.

[26] J. A. Joines and C. R. Houck, "On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with GA's," in *Proceedings of the 1st IEEE Conference on Evolutionary Computation*, pp. 579–584, Orlando, Fla, USA, June 1994.

[27] F. Jiménez and J. L. Verdegay, "Evolutionary techniques for constrained optimization problems," in *Proceedings of the 7th European Congress on Intelligent Techniques and Soft Computing (EUFIT '99)*, Aachen, Germany, 1999.

[28] C. Sun, J. Zeng, and J. Pan, "An improved vector particle swarm optimization for constrained optimization problems," *Information Sciences*, vol. 181, no. 6, pp. 1153–1163, 2011.

[29] Z. Michalewicz and M. Schoenauer, "Evolutionary algorithms for constrained parameter optimization problems," *Evolutionary Computation*, vol. 4, no. 1, pp. 1–32, 1996.

[30] C. A. C. Coello and A. D. Christiansen, "Moses: a multiobjective optimization tool for engineering design," *Engineering Optimization*, vol. 31, no. 1–3, pp. 337–368, 1999.

[31] Y. G. Woldesenbet, G. G. Yen, and B. G. Tessema, "Constraint handling in multiobjective evolutionary optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 3, pp. 514–525, 2009.

[32] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308–313, 1965.

[33] D. A. van Veldhuizen and G. B. Lamont, "Multiobjective evolutionary algorithms: analyzing the state-of-the-art," *Evolutionary Computation*, vol. 8, no. 2, pp. 125–147, 2000.

[34] R. Farmani, D. A. Savic, and G. A. Walters, "Evolutionary multiobjective optimization in water distribution network design," *Engineering Optimization*, vol. 37, no. 2, pp. 167–183, 2005.

[35] K. Deb and T. Goel, *Controlled Elitist Non-Dominated Sorting Genetic Algorithms for Better Convergence*, Springer, Berlin, Germany, 2001.

[36] Y. J. Lai, T. Y. Liu, and C. L. Hwang, "Topsis for MODM," *European Journal of Operational Research*, vol. 76, no. 3, pp. 486–500, 1994.