Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 9957279, 11 pages
https://doi.org/10.1155/2021/9957279

Research Article

Solving Multiobjective Game in Multiconflict Situation Based on Adaptive Differential Evolution Algorithm with Simulated Annealing

Huimin Li,[1] **Shuwen Xiang**,[1,2] **Wensheng Jia**,[1] **Yanlong Yang**,[1] **and Shiguo Huang**[1,3]

[1]*College of Mathematics and Statistics, Guizhou University, Guiyang 550025, China*
[2]*College of Mathematics and Information Science, Guiyang University, Guiyang 550005, China*
[3]*Department of Mathematics and Information Science, Zhengzhou University of Light Industry, Zhengzhou 450002, China*

Correspondence should be addressed to Huimin Li; 13783513360@163.com and Shuwen Xiang; shwxiang@vip.163.com

Received 24 March 2021; Revised 18 May 2021; Accepted 4 June 2021; Published 22 July 2021

Academic Editor: Hao Gao

In this paper, we study the multiobjective game in a multiconflict situation. First, the feasible strategy set and synthetic strategy space are constructed in the multiconflict situation. Meanwhile, the value of payoff function under multiobjective is determined, and an integrated multiobjective game model is established in a multiconflict situation. Second, the multiobjective game model is transformed into the single-objective game model by the Entropy Weight Method. Then, in order to solve this multiobjective game, an adaptive differential evolution algorithm based on simulated annealing (ADESA) is proposed to solve this game, which is to improve the mutation factor and crossover operator of the differential evolution (DE) algorithm adaptively, and the Metropolis rule with probability mutation ability of the simulated annealing (SA) algorithm is used. Finally, the practicability and effectiveness of the algorithm are illustrated by a military example.

1. Introduction

As we all know, in many fields such as optimal control, engineering design, economy, and arms race, deciders usually have not merely considered one objective [1–6] but integrated many objectives in decision-making. So multiobjective decisions are more consistent with reality than a single-objective decision. Multiobjective game [7] is an effective method to solve reciprocity among many deciders in the real society. Therefore, it is significant to study the multiobjective game. For example, in the military, armed forces will fight with the enemy on multiple battlefields at the same time. Due to the military strength, equipment, and other constraints, its multiple battlefield situations are interrelated and mutually constrained. According to the game theory, a conflict situation can be described as a game environment. In this way, the multiconflict situation can be expressed as multiple interconnected games, and the constraints can be regarded as objective criteria of the game. In the game, the player's strategy choice and the payoff function are the keys to analyzing the game decision. Therefore, studying the synthesis of strategies and synthetic payoff function is of great significance for analyzing the multiobjective game in a multiconflict situation.

For the game in a multiconflict situation, Inohara et al. [8] discussed the relationship between strategy sets and studied the synthesis of finite strategies. However, they did not provide a specific strategic integrated model. Under the single objective, Yanjie et al. [9] established an integrated game model for the conflict situation described by multiple bimatrix games [10, 11] and gave an application in the military field. Yexin et al. [12] established a game-integrated model in the multiobjective and multiconflict situation, and they solved the game by using the equivalent relationship between quadratic programming and equilibrium solution of the game. When solving these game models in the above literature, the numerical calculation methods and Lingo software were generally used, and less attention has been paid to intelligent algorithms to solve these problems. The paper aims to fulfill this gap, that is, the multiobjective game

in multiconflict situation is transformed into a simpler optimization problem, and then an efficient intelligent algorithm is proposed to solve it.

In this paper, in order to solve the integrated game model better, the integrated multiobjective game model is transformed into a single-objective game model by using the Entropy Weight Method [13, 14]. Then, the game is transformed into an equivalent optimization problem with constraints [15–18]. Finally, an adaptive differential evolution algorithm based on simulated annealing (ADESA) is proposed to solve this game. The main operation steps of this hybrid algorithm can be divided as follows: first, the mutation and crossover operator of the differential evolution (DE) algorithm are adaptively improved [19, 20]. Then, the Metropolis process of the simulated annealing (SA) algorithm [21] is applied to the DE algorithm, which has the characteristic that could accept not only the good solutions but also the inferior solutions with a certain probability. It not only improves the convergence speed and accuracy of the DE but also improves the diversity of the population. Thus, the hybrid algorithm has a strong global search capability and avoids the occurrence of premature phenomena. At the end of this paper, we use a military game example to demonstrate the practicability and effectiveness of the algorithm in solving the multiobjective game in a multiconflict situation.

2. Game Synthetic Model

2.1. Multiobjective Game Model in a Multiconflict Situation

Definition 1. The N-person noncooperative game with a single objective denoted by $\Gamma = ((N, S_i, U_i, X_i, f_i)\colon i = 1, \ldots, n)$, where

(1) $N = \{1, \ldots, n\}$ is the set of players and n is the number of players.

(2) $S_i = \{s_{i1}, \ldots, s_{im_i}\}$, $\forall i \in N$ is the pure strategy set of player i, m_i represents the number of strategies available to player i, $S = \prod_{i=1}^{n} S_i$, and each pure strategy profile meets $(s_{1m_1}, s_{2m_2}, \ldots, s_{ij}, \ldots, s_{nm_n}) \in S$, $s_{ij} \in S_i$.

(3) $U_i\colon S \longrightarrow \mathbb{R}$, $\forall i \in N$ represents the payoff function of player i.

(4) $X_i = \{x_i = (x_{i1}, \ldots, x_{ik}, \ldots, x_{im_i})\colon x_{ik} \geq 0, k = 1, \ldots, m_i, \sum_{k=1}^{m_i} x_{ik} = 1\}$, $\forall i \in N$ is the set of mixed strategies of player i. $X = \prod_{i=1}^{n} X_i$ and each mixed strategy profile meets $(x_1, x_2, \ldots, x_n) \in X$.

(5) $f_i\colon X \longrightarrow \mathbb{R}$, $\forall i \in N$ represents the expected payoff function of player i:

$$f_i(x_1, \ldots, x_n) = \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} U_i\left(s_{1k_1}, \ldots, s_{nk_n}\right) \prod_{i=1}^{n} x_{ik_i}, \quad (1)$$

where $f_i(x_1, \ldots, x_n)$ represents the expected payoff value of player i when he chooses a mixed strategy $x_i = (x_{i1}, \ldots, x_{im_i}) \in X_i$. $U_i(s_{ik_i}, \ldots, s_{nk_n})$ represents the payoff value of player of i when each player chooses pure strategy $s_{ik_i} \in S_i, i = 1, \ldots, n$.

Denote by

$$f_i\left(x \| s_{ik_i}\right) \triangleq f_i\left(x_1, \ldots, x_{i-1}, s_{ik_i}, x_{i+1}, \ldots, x_n\right) \triangleq \sum_{k_1=1}^{m_1} \cdots \sum_{k_{i-1}=1}^{m_{i-1}} \sum_{k_{i+1}=1}^{m_{i+1}} \cdots \sum_{k_n=1}^{m_n} U_i\left(s_{1k_1}, \ldots, s_{nk_n}\right) x_{1k_1} \cdots x_{i-1k_{i-1}} x_{i+1k_{i+1}} \cdots x_{nk_n}, \quad (2)$$

where $s_{ik_i}\,(1 \leq k_i \leq m_i)$ is a pure strategy of player i, $(x \| s_{ik_i})$ represents s_{ik_i} of player i instead of x_i, and the other players do not change their own mixed strategy.

Definition 2. If there is $x^* = (x_1^*, \ldots, x_n^*) \in X$, $f_i(x_i^*, x_{i^\wedge}^*) = \max_{u_i \in X_i} f_i(u_i, x_{i^\wedge}^*)$, $\forall i \in N$, then x^* is the Nash equilibrium of n-person finite noncooperative game, where $i^\wedge = N \backslash i$, $\forall i \in N$.

Conclusion 1. A mixed strategy x^* is the Nash equilibrium point of a game if and only if every pure strategy $s_{ik_i}\,(1 \leq k_i \leq m_i)$ of each player satisfies $f_i(x^*) \geq f_i(x^* \| s_{ik_i})$.

Theorem 1 (see [18]). *A mixed strategy $x^* \in X$ is the Nash equilibrium point of the game Γ if and only if x^* is the optimal solution to the following optimization problem, and the optimal value is 0:*

$$\begin{cases} \min f(x) = \sum_{i=1}^{n} \max_{1 \leq k_i \leq m_i} \left\{ f_i\left(x \| s_{ik_i}\right) - f_i(x), 0 \right\}, \\ \sum_{k_i=1}^{m_i} x_{ik_i} = 1, \quad 0 \leq x_{ik_i} \leq 1, i = 1, \ldots, n; k_i = 1, \ldots, m_i. \end{cases} \quad (3)$$

Especially, for the two-player matrix game, it can be seen from Theorem 1 that finding the Nash equilibrium (x_1^*, x_2^*) of the game is equivalent:

$$\begin{cases} \min f(x) = \max\left\{\max\limits_{1\le i\le m_1}\left\{A_i x_2^T - x_1 A x_2^T, 0\right\}\right\} + \max\left\{\max\limits_{1\le j\le m_2}\left\{x_1 B_j - x_1 B x_2^T, 0\right\}\right\}, \\ \sum\limits_{k_1=1}^{m_1} x_{1k_1} = 1, \sum\limits_{k_2=1}^{m_2} x_{2k_2} = 1, \quad 0 \le x_{ik_i} \le 1; i = 1, 2, \end{cases} \tag{4}$$

where A and B are payoff matrices of players, A_i is the ith row of matrix A, and B_j is the jth column of matrix B.

Definition 3. The multiconflict situation of the two persons with multiobjective refers to the situation in which player 1 and player 2 have conflict under the $L(L \ge 2)$ objectives in the $K(K \ge 2)$ situations. Each conflict situation can be described by a multiobjective bimatrix game model. Suppose that in the kth $(k = 1, \ldots, K)$ conflict situation, the payoff matrices of player 1 and player 2 under the lth $(l = 1, \ldots, L)$ objective are A^{lk} and B^{lk}, respectively:

$$A^{lk} = \begin{bmatrix} a_{11}^{lk} & a_{12}^{lk} & \cdots & a_{1n_k}^{lk} \\ a_{21}^{lk} & a_{22}^{lk} & \cdots & a_{1n_k}^{lk} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m_k1}^{lk} & a_{m_k2}^{lk} & \cdots & a_{m_kn_k}^{lk} \end{bmatrix},$$
$$B^{lk} = \begin{bmatrix} b_{11}^{lk} & b_{12}^{lk} & \cdots & b_{1n_k}^{lk} \\ b_{21}^{lk} & b_{22}^{lk} & \cdots & b_{1n_k}^{lk} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m_k1}^{lk} & b_{m_k2}^{lk} & \cdots & b_{m_kn_k}^{lk} \end{bmatrix}, \tag{5}$$

where m_k and n_k are the number of strategies of player 1 and player 2, respectively.

Definition 4. If player 1 chooses a pure strategy $\alpha_{i_k}^k$ in the kth $(k = 1, \ldots, K)$ game G^k and combines pure strategies in all conflict situations, the feasible strategy string of player 1 in the multiconflict situation is obtained, which is recorded as $\alpha_i \triangleq (\alpha_{i_1}^1; \alpha_{i_2}^2; \ldots; \alpha_{i_k}^k; \ldots; \alpha_{i_K}^K)$, $i_k \in \{1, 2, \ldots, m_k\}$. Similarly, the feasible strategy string of player 2 in a multiconflict situation is $\beta_j \triangleq (\beta_{j_1}^1; \beta_{j_2}^2; \ldots; \beta_{j_k}^k; \ldots; \beta_{j_K}^K)$, $j_k \in \{1, 2, \ldots, n_k\}$.

Definition 5. The feasible strategy sets of player 1 and player 2 in a multiconflict situation are recorded as $S_1 = \{\alpha_1, \alpha_2, \ldots, \alpha_t\}$ and $S_2 = \{\beta_1, \beta_2, \ldots, \beta_r\}$, where t and r represent the number of feasible strategy strings of player 1 and player 2, respectively.

Definition 6. A synthetic strategy in a multiconflict situation is (α_i, β_j), where α_i is a feasible strategy string selected by player 1 from S_1, and β_j is a feasible strategy string selected by player 2 from S_2. The synthetic strategy space consists of a set of all synthetic strategies of player 1 and player 2, which is recorded as $S = S_1 \times S_2$.

Definition 7. In a synthetic strategy, a strategic combination of player 1 and player 2 in each game is called the substrategy of the synthetic strategy. For example, the substrategies of (α_i, β_j) are $(\alpha_{i_1}^1, \beta_{j_1}^1)$, $(\alpha_{i_2}^2, \beta_{j_2}^2), \ldots, (\alpha_{i_K}^K, \beta_{j_K}^K)$.

The synthetic payoff value of the player in a synthetic strategy is the sum of the payoff values of all substrategies in the same objective. Let the synthetic payoff function values of the players 1 and 2 be c_{ij}^l and d_{ij}^l under the lth $(l = 1, 2, \ldots, L)$ objective when players choose the synthetic strategy (α_i, β_j), and

$$c_{ij}^l = \sum_{k=1}^{K} a_{i_k j_k}^{lk},$$
$$d_{ij}^l = \sum_{k=1}^{K} b_{i_k j_k}^{lk}. \tag{6}$$

Definition 8. The integrated model of the multiobjective bimatrix game in a multiconflict situation is $G = \{S_1, S_2, C^l, D^l\}$, where

$$C^l = \begin{bmatrix} c_{11}^l & c_{12}^l & \cdots & c_{1r}^l \\ c_{21}^l & c_{22}^l & \cdots & c_{1r}^l \\ \vdots & \vdots & \ddots & \vdots \\ c_{t1}^l & c_{r2}^l & \cdots & c_{tr}^l \end{bmatrix},$$
$$D^l = \begin{bmatrix} d_{11}^l & d_{12}^l & \cdots & d_{1r}^l \\ d_{21}^l & d_{22}^l & \cdots & d_{1r}^l \\ \vdots & \vdots & \ddots & \vdots \\ d_{t1}^l & d_{t2}^l & \cdots & d_{tr}^l \end{bmatrix}, \quad l = 1, 2, \ldots, L. \tag{7}$$

In order to solve the integrated game model in a multiconflict situation, the Entropy Weight Method is introduced below.

2.2. Integrated Game Model Based on the Entropy Weight Method

(1) The payoff matrix of each objective in the integrated model is transformed into a standardized matrix by

using the extreme difference method. For payoff matrices C^l and D^l of the lth $(l = 1, 2, \ldots, L)$ objective, take $\overline{\theta}^l = \max_{1\le i\le t, 1\le j\le r}\{c_{ij}^l, d_{ij}^l\}$, $\underline{\theta}^l = \min_{1\le i\le t, 1\le j\le r}\{c_{ij}^l, d_{ij}^l\}$.

For the negative objective,

$$e_{ij}^l = \frac{\left(\overline{\theta}^l - c_{ij}^l\right)}{\left(\overline{\theta}^l - \underline{\theta}^l\right)}, \quad f_{ij}^l = \frac{\left(\overline{\theta}^l - d_{ij}^l\right)}{\left(\overline{\theta}^l - \underline{\theta}^l\right)}, \quad 1 \le i \le t, 1 \le j \le r. \tag{8}$$

For the positive objective,

$$e_{ij}^l = \frac{\left(c_{ij}^l - \underline{\theta}^l\right)}{\left(\overline{\theta}^l - \underline{\theta}^l\right)}, \quad f_{ij}^l = \frac{\left(d_{ij}^l - \underline{\theta}^l\right)}{\left(\overline{\theta}^l - \underline{\theta}^l\right)}, \quad 1 \le i \le t, 1 \le j \le r. \tag{9}$$

Let E^l and F^l be the normalized matrices corresponding to C^l and D^l, respectively, and $E^l = (e_{ij}^l)_{t\times r}$, $F^l = (f_{ij}^l)_{t\times r}$.

(2) For E^l and F^l, the weight of each objective is calculated by the Entropy Weight Method [13, 14].

(i) Calculate the entropy value of the lth $(l = 1, 2 \ldots, L)$ objective:

$$h^l = -\lambda \sum_{i=1}^{t} \sum_{j=1}^{r} \left[\left(\frac{e_{ij}^l}{R^l}\right)\ln\left(\frac{e_{ij}^l}{R^l}\right) + \left(\frac{f_{ij}^l}{R^l}\right)\ln\left(\frac{f_{ij}^l}{R^l}\right)\right], \tag{10}$$

where $\lambda = (1/\ln(2tr))$, $R^l = \sum_{i=1}^{t}\sum_{j=1}^{r}(e_{ij}^l + f_{ij}^l)$. If $e_{ij}^l = 0$ or $f_{ij}^l = 0$, then $(e_{ij}^l/R^l)\ln(e_{ij}^l/R^l) = 0$ or $(f_{ij}^l/R^l)\ln(f_{ij}^l/R^l) = 0$.

(ii) Calculate the difference index of the lth objective:

$$g^l = 1 - h^l, \quad l = 1, 2 \ldots, L. \tag{11}$$

(iii) Calculate the weight of the lth objective:

$$\omega^l = \frac{g^l}{\sum_{l=1}^{L} g^l}, \quad l = 1, 2, \ldots, L. \tag{12}$$

(3) Weighting and summing E^l and F^l of all objectives in the synthetic model:

$$\begin{cases} E = \left(e_{ij}\right)_{t\times r} = \sum_{l=1}^{L} w^l E^l, \\ F = \left(f_{ij}\right)_{t\times r} = \sum_{l=1}^{L} w^l F^l, \end{cases} \tag{13}$$

where $e_{ij} = \sum_{l=1}^{L} w^l e_{ij}^l$ and $f_{ij} = \sum_{l=1}^{L} w^l f_{ij}^l$.

By weighted summation, the above-integrated model of a multiobjective bimatrix game in a multiconflict situation is transformed into a single-objective matrix game model $G^* = \{S_1, S_2, E, F\}$. In order to solve the game model more conveniently, the ADESA algorithm is proposed.

3. Adaptive Differential Evolution Algorithm Based on Simulated Annealing (ADESA)

In this section, we outline a novel DE algorithm, ADESA algorithm, and explain the steps of the algorithm in detail.

3.1. Differential Evolution (DE) Algorithm. The DE algorithm is first introduced by Storm and Price [22]. Due to its outstanding characteristics, such as simple structure, robustness and speediness, being easy to understand and implement, and fewer control parameters, it has become more and more popular and has been extended to handle a variety of optimization problems. The DE algorithm mainly consists of four operations: initialization population, mutation operation, crossover operation, and selection operation.

3.1.1. Initialization. In the study of the DE algorithm, it is generally assumed that the initial population conform to a uniform probability distribution. Each individual $X = (x_{i1}, \ldots, x_{ij}, \ldots, x_{iD})$, $i = 1, \ldots, N$, $j = 1, \ldots, D$ can be expressed as

$$x_{i,j} = \text{rand}[0, 1] \cdot \left(x_{i,j}^U - x_{i,j}^L\right) + x_{i,j}^L, \tag{14}$$

where N and D denote population size and space dimension, respectively, and $x_{i,j}^L$ and $x_{i,j}^U$, respectively, are the lower and upper bound of the search space.

3.1.2. Mutation. The mutation operation is mainly executed to distinguish DE from other evolutionary algorithms. The mutation individual $V = (v_{i1}, \ldots, v_{ij}, \ldots, v_{iD})$ is generated by the following equation:

$$v_i^{t+1} = x_{r_1}^t + F \cdot \left(x_{r_2}^t - x_{r_3}^t\right), \tag{15}$$

where $r_1 r_2$ and r_3 are randomly generated integers within $[1, N]$, $r_1 \ne r_2 \ne r_3 \ne i$, F is a constriction factor to control the size of difference of two individuals, and t is the current generation.

3.1.3. Crossover. We use the crossover between the parent and offspring with the given probability to generate a new individual $U = (u_{i1}, \ldots, u_{ij}, \ldots, u_{iD})$:

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, & \text{if } (\text{rand}(j) \le \text{CR}) \text{or} (j = rnbr(i)), \\ x_{ij}, & \text{otherwise}, \end{cases} \tag{16}$$

where $\text{rand}(j) \in [0,1]$ is random values, $\text{CR} \in [0,1]$ is crossover operator, $rnbr(i)$ is a randomly selected integer on $[1, D]$, which ensures at least one component of new individual is inherited from the mutant vector.

3.1.4. Selection. In the problem with boundary constraints, it is necessary to ensure that the parameter values of the new individuals are in the feasible region. There is a simple method is boundary treatment, in which the new individuals beyond the bounds are replaced by the parameter vectors randomly generated in the feasible region. Then, the offspring X_i^{t+1} is generated by selecting the individual and parent according to the following formula (for a minimization problem):

$$X_{ij}^{t+1} = \begin{cases} U_i^{t+1}, & \text{if}\left(f\left(U_i^{t+1}\right) < f\left(U_i^t\right)\right), \\ X_i^t, & \text{otherwise}, \end{cases} \tag{17}$$

where $f(\cdot)$ is the fitness function. The pseudocode of the standard DE algorithm is shown in Algorithm 1.

3.2. Adaptive Differential Evolution Algorithm (ADE). Although the DE algorithm is widely used in optimization problems, with the increase of the complexity of solving problems, the DE algorithm also has some disadvantages, such as slow convergence, low accuracy, and weak stability. Therefore, in order to solve the multiobjective game better, the DE algorithm is improved.

Since the DE algorithm mainly performs the genetic operation through differential mutation operator, the performance of the algorithm mainly depends on the selection of mutation and crossover operations and related parameters. Many scholars have verified that the mutation factor F and crossover operator CR directly affect the searching capability and solving efficiency of the DE [23–25]. In order to make the algorithm have better global search capability and convergence speed, adaptive mutation and crossover operator are adopted:

$$\begin{cases} \lambda = e^{1-(T/T+1-t)}, \\ F = F_0 \cdot 2^{\lambda}, \\ \text{CR} = \text{CR}_0 \cdot 2^{\lambda}, \end{cases} \tag{18}$$

where t is the current iterate time, T is the maximum number of iteration, and F_0 and CR_0 are the initial mutation factor and crossover operator, respectively.

There are two main ways of traditional mutation operations:

$$\begin{aligned} &\text{(i) DE/rand/1/bin: } v_i^{t+1} = x_{r1}^t + F \cdot \left(x_{r2}^t - x_{r3}^t\right), \\ &\text{(ii) DE/best/1/bin: } v_i^{t+1} = x_{\text{best}}^t + F \cdot \left(x_{r2}^t - x_{r3}^t\right), \end{aligned} \tag{19}$$

where x_{best}^t represents the best individual in the current generation, that is, the optimal position searched by this individual so far [26]. The first mutation method has a strong global search capability but a slow convergence speed. The second method has a fast convergence speed, but it is easy to fall into local optimum values [27]. In order to overcome these shortcomings, many researchers have improved the mutation strategy [28, 29], and a new mutation operation is proposed in this paper:

$$v_i^{t+1} = \lambda x_{r1}^t + (1-\lambda) x_{\text{best}}^t + F \cdot \left(x_{r2}^t - x_{r3}^t\right). \tag{20}$$

The new mutation strategy has strong global search ability and fast convergence speed and can find better solutions.

3.3. Simulated Annealing Algorithm (SA). SA is not only a statistical method, but also a global optimization algorithm. It is first proposed by Metropolis in 1953 [21], and Kirkpatrick first used SA to solve combinatorial optimization problems in 1983 [30]. SA is derived from the simulation of the solid annealing cooling process. The main feature is to accept inferior solutions with a certain probability according to the Metropolis rule, which can avoid the algorithm falling into the local optimum and "premature" phenomenon.

The Metropolis rule defines the internal energy probability P_{ij}^M of an object state i transferring to state j at a certain temperature M, it can be expressed as follows:

$$P_{ij}^M = \begin{cases} 1, & E(j) \le E(i), \\ e^{-(E(j)-E(i)/KM)}, & \text{otherwise}, \end{cases} \tag{21}$$

where $E(i)$ and $E(j)$ represent internal energy of solid in states i and j, respectively. K is attenuation parameter, and $\Delta E = E(j) - E(i)$ represents increment of internal energy.

When the combined optimization problem is simulated by solid annealing, the internal energy E is simulated as the objective function value, and the temperature M becomes a parameter. That is,

$$P_{i+1}^M = e^{-\left(f\left(X_{i+1}\right) - f\left(X_i\right)/KM\right)}, \tag{22}$$

where $f(X_i)$ is the value of the function and M is the temperature at the ith iteration. When $f(X_{i+1}) \le f(X_i)$, we select $f(X_{i+1})$ with probability 1; otherwise, we select the inferior solution $f(X_i)$ with probability P_{i+1}^M. In this paper, the SA is applied to the DE to enhance its global optimization ability.

3.4. The ADESA Algorithm Experimental Steps. The pseudocode of the ADESA algorithm is shown in Algorithm 2, and the specific steps are described in detail as follows:

Step 1: set the parameters of the ADESA, such as N, D, F_0, CR_0, K, M, T, ε.

Step 2: initialize the population and each individual satisfies

$$\sum_{k_i=1}^{m_i} x_{ik_i} = 1, \quad x_{ik_i} \ge 0, i = 1, \ldots, N; k_i = 1, \ldots, m_i. \tag{23}$$

Input: Parameters N, D, T, F, CR, ε
Output: The best vector (Solution) $\cdots \Delta$
(1) $t \leftarrow 0$ (Initialization)
(2) **for** $i = 0$ to N **do**
(3) **for** $j = 0$ to D **do**
(4) $x_{i,j}^t = \text{rand}[0,1] \cdot (x_{i,j}^U - x_{i,j}^L) + x_{i,j}^L$
(5) **end for**
(6) **end for**
(7) **while** $|f(\Delta)| \geq \varepsilon$ or $t \leq T$ **do**
(8) **for** $i = 1$ to N **do**
(9) (Mutation and Crossover)
(10) **for** $j = 1$ to D **do**
(11) $v_{i,j}^t = \text{Mutation}(x_{i,j}^t)$ (formula (15))
(12) $u_{i,j}^t = \text{Crossover}(v_{i,j}^t, x_{i,j}^t)$
(13) **end for**
(14) (Selection)
(15) **if** $f(u_{i,j}^t) < f(x_{i,j}^t)$ **then**
(16) $x_{i,j}^t \leftarrow u_{i,j}^t$
(17) **else**
(18) $\Delta \leftarrow x_{i,j}^t$
(19) **end if**
(20) **end for**
(21) $t = t + 1$
(22) **end while**
(23) **return** the best vector Δ

Algorithm 1: DE.

Step 3: calculate the fitness function value $f(x)$ of each individual in population $P(t)$ and determine x_{best}^t and $f(x_{\text{best}}^t)$.

Step 4: the next generation population $P_1(t)$ is generated by mutation of formula (20), and population $P_2(t)$ is generated by crossover of formula (16).

Step 5: the offspring population $P(t+1)$ is selected from the $P(t)$ and $P_2(t)$ populations according to formulas (21) and (22), and calculate the fitness function value of population $P(t+1)$.

Step 6: determine whether to end this procedure according to the accuracy and the maximum number of iterations and output the optimal value; otherwise, turn to step 3.

According to Algorithm 1, it can be judged that the experimental steps of the ADE algorithm are the same as the DE algorithm, so the time complexity of the ADE algorithm does not change. Comparing the implementation process of Algorithms 1 and 2, we can see that the ADESA algorithm has only one more judgment in the selection operation than the DE algorithm. Therefore, the time complexity remains unchanged. In conclusion, the computational complexity provides a guarantee for the performance of the ADESA algorithm proposed in this paper. In the next section, the superiority of the proposed algorithm is verified by calculating an example of the multiobjective military game.

4. Experimental Design and Results

4.1. Military Example. Suppose that there are conflicts between Red and Blue armies on islands A and B. Under the dual objectives of attack time and damage effectiveness, Red's strategies on the two islands are no-attack and attack, and Blue's strategies are retreat and defend. Two parties have different strategic purposes on the two islands; therefore, the degree of preference is different under the same objective. In addition, due to the limitation of military equipment, it is assumed that the Red army cannot choose to attack on both islands at the same time, and the Blue army cannot choose to defend on both islands at the same time. In this way, the available strategies for both Red and Blue on A island are $\{\alpha_1^A, \alpha_2^A\} = \{\text{no} - \text{attack}, \text{attack}\}$, $\{\beta_1^A, \beta_2^A\} = \{\text{retreat}, \text{defend}\}$. On island B, the available strategies for both Red and Blue are $\{\alpha_1^B, \alpha_2^B\} = \{\text{no} - \text{attack}, \text{attack}\}$, $\{\beta_1^B, \beta_2^B\} = \{\text{retreat}, \text{defend}\}$.

Aiming at the attack time ($l = 1$), the payoff matrices of the Red and Blue on the islands A and B are, respectively (the payoff values are expressed in the preference order of −2, −1, 0, 1, 2), as follows:

$$A_{11} = \begin{bmatrix} 1 & 0 \\ 2 & -2 \end{bmatrix}, \quad B_{11} = \begin{bmatrix} 1 & 2 \\ 0 & -1 \end{bmatrix}, \quad A_{12} = \begin{bmatrix} 2 & -1 \\ 1 & 0 \end{bmatrix}, \quad B_{12} = \begin{bmatrix} 1 & 2 \\ -1 & 0 \end{bmatrix}. \tag{24}$$

Aiming at the damage effectiveness ($l = 2$), the payoff matrices of the Red and Blue on the two islands A and B are, respectively, as follows:

Input: Parameters N, D, T, K, M, F_0, CR_0, ε
Output: The best vector (Solution) $\cdots \Delta$
(1) $t \leftarrow 0$ (Initialization)
(2) **for** $i = 0$ to N do
(3) **for** $j = 0$ to D do
(4) $x_{i,j}^t = \mathrm{rand}[0,1] \cdot (x_{i,j}^U - x_{i,j}^L) + x_{i,j}^L$
(5) **end for**
(6) **end for**
(7) **while** $|f(\Delta)| \geq \varepsilon$ or $t \leq T$ **do**
(8) **for** $i = 1$ to N **do**
(9) (Mutation and Crossover)
(10) **for** $j = 1$ to D **do**
(11) $v_{i,j}^t = \mathrm{Mutation}(x_{i,j}^t)$ (formula (20))
(12) $u_{i,j}^t = \mathrm{Crossover}(v_{i,j}^t, x_{i,j}^t)$
(13) **end for**
(14) (Metropolis Selection)
(15) if $f(u_{i,j}^t) < f(x_{i,j}^t)$ **then**
(16) $x_{i,j}^t \leftarrow u_{i,j}^t$ (accept the new solutions)
Let $P_1 = e^{-(f(U_{i,j}^t) - f(X_{i,j}^t))/KM}$
(17) else $\{P_1 > \mathrm{rand}\}$
(18) $x_{i,j}^t \leftarrow u_{i,j}^t$ (accept the inferior solutions)
(19) **else** $\{f(x_{i,j}^t) < f(\Delta)\}$
(20) $\Delta \leftarrow x_{i,j}^t$
(21) **end if**
(22) **end for**
(23) $M = K \cdot M$; $t = t + 1$
(24) **end while**
(25) **return** the best vector Δ

Algorithm 2: ADESA.

$$A_{21} = \begin{bmatrix} 0 & -1 \\ 2 & 1 \end{bmatrix},$$
$$B_{21} = \begin{bmatrix} 2 & 1 \\ -1 & 0 \end{bmatrix},$$
$$A_{22} = \begin{bmatrix} 1 & -1 \\ 2 & 0 \end{bmatrix},$$
$$B_{22} = \begin{bmatrix} 2 & 0 \\ 1 & -1 \end{bmatrix}. \tag{25}$$

Obviously, considering real situations of the two islands A and B, the available strategy sets of Red and Blue are

$$S_1 = \{\alpha_1, \alpha_2, \alpha_3\} = \left\{\left(\alpha_1^A; \alpha_1^B\right), \left(\alpha_1^A; \alpha_2^B\right), \left(\alpha_2^A; \alpha_1^B\right)\right\},$$
$$S_2 = \{\beta_1, \beta_2, \beta_3\} = \left\{\left(\beta_1^A; \beta_1^B\right), \left(\beta_1^A; \beta_2^B\right), \left(\beta_2^A; \beta_1^B\right)\right\}. \tag{26}$$

Using formula (6), the synthetic payoff matrices of the Red and Blue under the two objectives are, respectively, as follows:

$$R_1 = \begin{bmatrix} 3 & 0 & 2 \\ 2 & 1 & 1 \\ 4 & 1 & 0 \end{bmatrix},$$
$$B_1 = \begin{bmatrix} 2 & 3 & 1 \\ 0 & 1 & 1 \\ 1 & 2 & 0 \end{bmatrix},$$
$$R_2 = \begin{bmatrix} 1 & -1 & 0 \\ 2 & 0 & 1 \\ 3 & 1 & 2 \end{bmatrix},$$
$$B_2 = \begin{bmatrix} 4 & 2 & 3 \\ 3 & 1 & 2 \\ 1 & -1 & 2 \end{bmatrix}. \tag{27}$$

Using formulas (8) and (9), the above synthetic payoff matrices can be transformed into the standardized matrices as follows:

$$E_1 = \begin{bmatrix} 0.25 & 1.00 & 0.50 \\ 0.50 & 0.75 & 0.75 \\ 0 & 0.75 & 1.00 \end{bmatrix},$$
$$F_1 = \begin{bmatrix} 0.50 & 0.25 & 0.75 \\ 1.00 & 0.75 & 0.75 \\ 0.75 & 0.50 & 1.00 \end{bmatrix},$$
$$E_2 = \begin{bmatrix} 0.40 & 0 & 0.20 \\ 0.60 & 0.20 & 0.40 \\ 0.80 & 0.40 & 0.60 \end{bmatrix},$$
$$F_2 = \begin{bmatrix} 1.00 & 0.60 & 0.80 \\ 0.80 & 0.40 & 0.60 \\ 0.40 & 0 & 0.60 \end{bmatrix}. \quad (28)$$

By formulas (10)–(12), the entropy values of the lth ($l = 1,2$) objective are $h^1 = 0.9581$, $h^2 = 0.9311$. The difference indices are $g^1 = 0.0419$, $g^2 = 0.0689$. The weights are $\omega^1 = 0.3779$, $\omega^2 = 0.6221$. Finally, by formula (13), we can get

$$E = \begin{bmatrix} 0.3433 & 0.3779 & 0.3134 \\ 0.5622 & 0.4078 & 0.5323 \\ 0.4977 & 0.5323 & 0.7512 \end{bmatrix},$$
$$F = \begin{bmatrix} 0.8111 & 0.4677 & 0.7811 \\ 0.8756 & 0.5323 & 0.6567 \\ 0.5323 & 0.1889 & 0.7512 \end{bmatrix}. \quad (29)$$

Use formula (4) to solve the following optimization problem:

$$\min f(x) = \max(f_{11} - f_1, f_{12} - f_1, f_{13} - f_1, 0) + \max(f_{21} - f_2, f_{22} - f_2, f_{23} - f_2, 0),$$
$$\begin{cases} x_{11} + x_{12} + x_{13} = 1, \\ x_{21} + x_{22} + x_{23} = 1, \\ 0 \le x_{11}, x_{12}, x_{13}, x_{21}, x_{22}, x_{23} \le 1, \end{cases} \quad (30)$$

where

$$\begin{aligned} f_1 &= (0.3433x_{11} + 0.5622x_{12} + 0.4977x_{13})x_{21} + (0.3779x_{11} + 0.4078x_{12} + 0.5323x_{13})x_{22} \\ &\quad + (0.3134x_{11} + 0.5323x_{12} + 0.7512x_{13})x_{23}, \\ f_{11} &= 0.3433x_{21} + 0.3779x_{22} + 0.3134x_{23}, \\ f_{12} &= 0.5622x_{21} + 0.4078x_{22} + 0.5323x_{23}, \\ f_{13} &= 0.4977x_{21} + 0.5323x_{22} + 0.7512x_{23}, \\ f_2 &= (0.8111x_{11} + 0.8756x_{12} + 0.5323x_{13})x_{21} + (0.4677x_{11} + 0.5323x_{12} + 0.1889x_{13})x_{22} \\ &\quad + (0.7811x_{11} + 0.6567x_{12} + 0.7512x_{13})x_{23}, \\ f_{21} &= 0.8111x_{11} + 0.8756x_{12} + 0.5323x_{13}, \\ f_{22} &= 0.4677x_{11} + 0.5323x_{12} + 0.1889x_{13}, \\ f_{23} &= 0.7811x_{11} + 0.6567x_{12} + 0.7512x_{13}. \end{aligned} \quad (31)$$

4.2. Results and Discussion. In order to solve the game, the DE, ADE, and ADESA algorithms are used to calculate the above optimization problem. According to [24, 31, 32] and experimental experiences, the parameters are set to $N = 20$, $D = 6$, $\mathrm{CR} = 0.8$, $F = 0.6$, $\mathrm{CR}_0 = 1$, $F_0 = 0.4$, $K = 0.998$, $M = 100$, $T = 60$, $\varepsilon = 10^{-8}$. The calculation results and the corresponding Nash equilibria are shown in Table 1. Figure 1 is a comparison of solving this problem with DE, ADE, and ADESA algorithms, and Figure 2 is the result of the first ten iterations of Figure 1.

It can be intuitively seen from Table 1 that when we solve this military game by using these three algorithms, two Nash equilibria are obtained. One is (α_3, β_3), the other is (α_2, β_1), and their corresponding feasible strategy choices are $(\alpha_2^A; \alpha_1^B)$, $(\beta_2^A; \beta_1^B)$ and $(\alpha_1^A; \alpha_2^B)$, $(\beta_1^A; \beta_1^B)$. The feasible strategy $(\alpha_1^A; \alpha_2^B)$, $(\beta_1^A; \beta_1^B)$ means the Red attacks island B and the Blue retreats on both islands. Given the fact that the Blue is unlikely to retreat on both islands, this solution will be abandoned. The other solution $(\alpha_2^A; \alpha_1^B)$, $(\beta_2^A; \beta_1^B)$ represents that the Red attacks island A and the Blue defend on it, which is consistent with the real battle situation, so the final result of this game is (α_2, β_1).

Since finding the Nash equilibria of the integrated game is equivalent to solving the optimal solution of the

Table 1: Results of the game $G^*(S_1, S_2, E, F)$.

Number	Red	Blue	Synthetic strategy	Strategy string
Nash equilibrium 1	(0, 0, 1)	(0, 0, 1)	(α_3, β_3)	$(\alpha_2^A; \alpha_1^B)$, $(\beta_2^A; \beta_1^B)$
Nash equilibrium 2	(0, 1, 0)	(1, 0, 0)	(α_2, β_1)	$(\alpha_1^A; \alpha_2^B)$, $(\beta_1^A; \beta_1^B)$

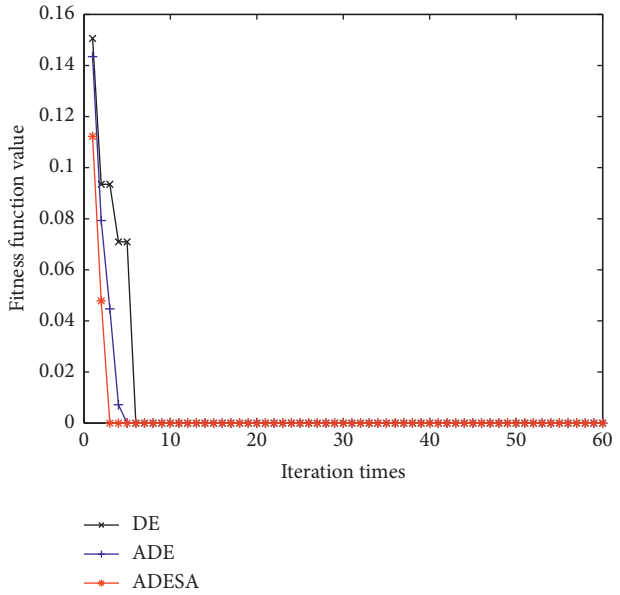

Figure 1: Comparison of DE, ADE, and ADESA

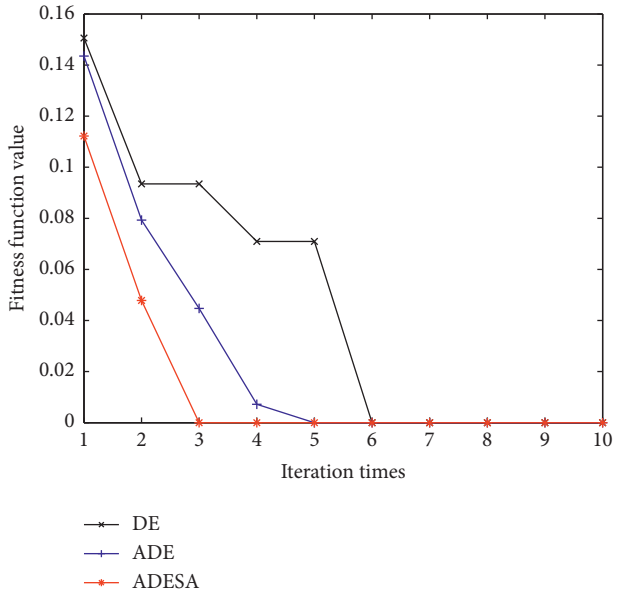

Figure 2: The first ten iterations of Figure 1.

optimization problem. It can be clearly seen from Figures 1 and 2 that the ADESA algorithm has the best performance in the calculation process, followed by the ADE algorithm, and DE algorithm at the end. On the one hand, we found that the ADE algorithm and the ADESA algorithm generally have similar performance. It seems rational because the ADESA algorithm and ADE algorithm both have self-adaptively improved the parameters compared with the DE algorithm. As the number of generations increases, it is difficult to find the optimal solution, so the DE algorithm has pauses in the local optimal solution. At this time, the self-adaptive operations of the ADESA and ADE algorithms play a role in avoiding the local optimal solution and accelerating the convergence speed. On the other hand, with the iteration process, there are more and more noninferior solutions. The ADE algorithm only selects the optimal solution according to the greedy criterion, so it will ignore some excellent solutions, while the ADESA algorithm retains some noninferior solutions because it adds the Metropolis rule in the selection operation. Therefore, the speed of searching for the global optimal solution is accelerated. It can be clearly observed in Figure 2.

By analyzing the results obtained above, it can be seen that the algorithm ADESA proposed in this paper can well find the equilibrium solutions under the two objectives. And comparing this algorithm with DE and ADE algorithms, we find that this algorithm has a faster convergence speed and avoids falling into local optimum. It provides a reference for the study of solving more complex multiobjective games in the future.

5. Conclusions

In this paper, we study the multiobjective game in a multiconflict situation. Firstly, an integrated multiobjective game model is established in a multiconflict situation; then the multiobjective game is transformed into a single-objective game according to the Entropy Weight Method. Finally, using the equivalence theorem of Nash equilibrium, finding the Nash equilibria of the game is equivalent to solving the optimal solution of an optimization problem. According to the excellent performances of the DE algorithm solving the optimization problems, we choose the DE algorithm to solve this problem. Because the game itself is a complex decision-making problem, and this paper studies the multiobjective game in a multiconflict situation, therefore, the DE algorithm is improved, namely, the ADESA algorithm, which applies the Metropolis rule of the SA algorithm to the selection operation of the ADE algorithm. Moreover, the ADE algorithm is a self-adaptive improvement to the control parameters of the DE algorithm. At the end of the paper, the computational results of a military example show that our proposed ADESA algorithm solves the multiobjective game more quickly and effectively, which provides a reference for future research on related issues. In the future, people may be in the situation of multiobjective mutual transfer in reality, so we will further consider the more complex dynamic multiobjective game in a multiconflict situation. In addition, it could be interesting

to develop other algorithms and compare them with the ADESA algorithm.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflicts of interest.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (Grant nos. 71961003 and 12061020), the Qian Jiaohe YJSCXJH ([2019]029), and the Scientific Research Foundation of Guizhou University ([2019]49).

References

[1] M. Baumann, M. Weil, J. F. Peters, N. Chibeles-Martins, and A. B. Moniz, "A review of multi-criteria decision making approaches for evaluating energy storage systems for grid applications," *Renewable and Sustainable Energy Reviews*, vol. 107, pp. 516–534, 2019.

[2] Y. Peng and Y. Shi, "Editorial: multiple criteria decision making and operations research," *Annals of Operations Research*, vol. 197, no. 1, pp. 1–4, 2012.

[3] E. Rashidi, M. Jahandar, and M. Zandieh, "An improved hybrid multi-objective parallel genetic algorithm for hybrid flow shop scheduling with unrelated parallel machines," *International Journal of Advanced Manufacturing Technology*, vol. 49, no. 9–12, pp. 1129–1139, 2010.

[4] H. Shi, Y. Dong, L. Yi et al., "Study on the route optimization of military logistics distribution in wartime based on the ant colony algorithm," *Computer and Information Science*, vol. 3, no. 1, pp. 139–143, 2010.

[5] V. Fragnelli and L. Pusillo, "Multiobjective games for detecting abnormally expressed genes," *Mathematics*, vol. 8, no. 3, p. 350, 2020.

[6] X. Cai, H. Zhao, S. Shang et al., "An improved quantum-inspired cooperative co-evolution algorithm with muli-strategy and its application," *Expert Systems with Applications*, vol. 171, Article ID 114629, 2021.

[7] C. S. Lee, "Multi-objective game-theory models for conflict analysis in reservoir watershed management," *Chemosphere*, vol. 87, no. 6, pp. 608–613, 2012.

[8] T. Inohara, S. Takahashi, and B. Nakano, "Integration of games and hypergames generated from a class of games," *Journal of the Operational Research Society*, vol. 48, no. 4, pp. 423–432, 1997.

[9] W. Yanjie, S. Yexin, and Z. Xianhai, "Integration model of bimatrix games in two-person multi-conflict situations," *Journal of Naval University of Engineering*, vol. 21, no. 1, pp. 22–25, 2009.

[10] I. Nishizaki and M. Sakawa, "Equilibrium solutions in multiobjective bimatrix games with fuzzy payoffs and fuzzy goals," *Fuzzy Sets and Systems*, vol. 111, no. 1, pp. 99–116, 2000.

[11] V. Vidyottama, S. Chandra, and C. R. Bector, "Bi-matrix games with fuzzy goals and fuzzy," *Fuzzy Optimization and Decision Making*, vol. 3, no. 4, pp. 327–344, 2004.

[12] S. Yexin, "Integration model of multi-objective bimatrix games in multiconflict situations," *Journal of Huazhong University of Science and Technology (Nature Science Edition)*, vol. 37, no. 6, pp. 32–35, 2009.

[13] J. Tian, L. Tao, and H. Jiao, "Entropy weight coefficient method for evaluating intrusion detection systems," in *Proceedings of the International Symposium on Electronic Commerce and Security*, pp. 592–598, Guangzhou, China, August 2008.

[14] S. Ding and Z. Shi, "Studies on incidence pattern recognition based on information entropy," *Journal of Information Science*, vol. 31, no. 6, pp. 497–502, 2005.

[15] N. G. Pavlidis, K. E. Parsopoulos, and M. N. Vrahatis, "Computing Nash equilibria through computational intelligence methods," *Journal of Computational and Applied Mathematics*, vol. 175, no. 1, pp. 113–136, 2005.

[16] A. S. Strekalovskii and R. Enkhbat, "Polymatrix games and optimization problems," *Automation and Remote Control*, vol. 75, no. 4, pp. 632–645, 2014.

[17] R. Enkhbat, N. Tungalag, A. Gornov, and A. Anikin, "The curvilinear search algorithm for solving three-person game," in *Proceedings of DOOR 2016 (CEUR-WS)*, A. Kononov, Ed., Vladivostok, Russia, 2016.

[18] L. Huimin, X. Shuwen, Y. Yanlong et al., "Differential evolution particle swarm optimization algorithm based on good point set for computing Nash equilibrium of finite noncooperative game," *AIMS Mathematics*, vol. 6, no. 2, pp. 1309–1323, 2021.

[19] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.

[20] F. Zhao, F. Xue, Y. Zhang, W. Ma, C. Zhang, and H. Song, "A hybrid algorithm based on self-adaptive gravitational search algorithm and differential evolution," *Expert Systems with Applications*, vol. 113, pp. 515–530, 2018.

[21] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[22] R. Storn and K. Price, "Minimizing the real functions of the ICEC'96 contest by differential evolution," in *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 824–844, Nagoya, Japan, May 1996.

[23] D. Wu, X. Junjie, Y. Song, and H. Zhao, "Differential evolution algorithm with wavelet basis function and optimal mutation strategy for complex optimization problem," *Applied Soft Computing*, vol. 100, Article ID 106724, 2021.

[24] A. W. Mohamed and A. K. Mohamed, "Adaptive guided differential evolution algorithm with novel mutation for numerical optimization," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 2, pp. 253–277, 2019.

[25] Z. Yanping, "An adaptive differential evolution algorithm and its application," *Computer Technology and Development*, vol. 7, pp. 119–123, 2019.

[26] Y. Wu, Y. Wu, and X. Liu, "Couple-based particle swarm optimization for short-term hydrothermal scheduling," *Applied Soft Computing*, vol. 74, pp. 440–450, 2019.

[27] H. Yi Chao, X. Z. Wang, K. Q. Liu, and Y. Q. Wang, "Convergent analysis and algorithmic improvement of differential evolution," *Journal of Software*, vol. 21, no. 5, pp. 875–885, 2010.

[28] A. W. Mohamed and P. N. Suganthan, "Real-parameter unconstrained optimization based on enhanced fitness-

adaptive differential evolution algorithm with novel mutation," *Soft Computing*, vol. 22, no. 10, pp. 3215–3235, 2018.

[29] W. Deng, S. Shang, X. Cai, H. Zhao, Y. Song, and J. Xu, "An improved differential evolution algorithm and its application in optimization problem," *Soft Computing*, vol. 25, no. 7, pp. 5277–5298, 2021.

[30] S. Kirkpatrick, D. Celatt, and M. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 1983, pp. 671–680, 1986.

[31] R. Gamperle, S. D. Muller, and P. Koumoutsakos, "A parameter study for differential evolution," *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, vol. 10, pp. 293–298, 2002.

[32] L. Ingber, "Simulated annealing: practice versus theory," *Mathematical and Computer Modelling*, no. 18, pp. 29–57, 1993.