Mathematics

http://www.aimspress.com/journal/Math

AIMS Mathematics, 6(2): 1309–1323.
DOI:10.3934/math.2021081
Received: 05 August 2020
Accepted: 07 November 2020
Published: 17 November 2020

Research article

Differential evolution particle swarm optimization algorithm based on good point set for computing Nash equilibrium of finite noncooperative game

Huimin Li[*], **Shuwen Xiang, Yanlong Yang and Chenwei Liu**

College of mathematics and statistics, Guizhou University, Guiyang 550025, China

* **Correspondence:** Email: 13783513360@163.com; Tel: +8613783513360.

Abstract: In this paper, a hybrid differential evolution particle swarm optimization (PSO) method based on a good point set (GPDEPSO) is proposed to compute a finite noncooperative game among N people. Stochastic functional analysis is used to prove the convergence of this algorithm. First, an ergodic initial population is generated by using a good point set. Second, PSO is proposed and utilized as the variation operator to perform variation crossover selection with differential evolution (DE). Finally, the experimental results show that the proposed algorithm has a better convergence speed, accuracy, and global optimization ability than other existing algorithms in computing the Nash equilibrium of noncooperative games among N people. In particular, the efficiency of the algorithm is higher for determining the Nash equilibrium of a high-dimensional payoff matrix game.

Keywords: non-cooperative game; Nash equilibrium; good point set; differential evolution particle swarm optimization algorithm; high-dimensional payoff matrix
Mathematics Subject Classification: 68W50, 90C33, 91A06, 91A10

1. Introduction

Game theory is a mathematical theory that studies interactions between decision makers. It mainly studies how players use the information they have to make decisions. Game theory is widely used in many fields, such as economics, political science, biology and other fields [1–6]. Noncooperative game is the main component of game theory, and the Nash equilibrium is the core concept of noncooperative game. However, achieving equilibrium requires players to make predictions according to certain steps in the game. Therefore, it can be reduced to a calculation problem to a certain extent.

At present, many numerical methods have been proposed to solve the Nash equilibrium, such as the Lemke-Howson algorithm [7], global Newton algorithm [8], projection-like method [9], trust region algorithm [10] and other traditional calculation methods. In recent years, with the increased complexity of game problems, traditional numerical methods are almost impossible to use in solving due to the increasing difficulty of finding the solution and the computation time. Increasing numbers of scholars have focused on intelligent algorithms for biological simulation. Because of the rationality of imitating biological behaviors, such algorithms have the implicit rational characteristics of games. By such means, such algorithms can be considered as some sort of paths to achieve the Nash equilibrium.

Pavlidis [11] verified the effectiveness of three computational intelligence algorithms, namely, covariance matrix adaptation evolution strategies, particle swarm optimization (PSO) and differential evolution (DE), to compute the Nash equilibrium of finite strategic games. Boryczka [12] compared DE with two well-known algorithms: simplicial subdivision and the Lemke-Howson algorithm. It has also been proven that the DE method can obtain an approximate Nash equilibrium solution. Chen [13] utilized the genetic algorithm (GA) to acquire the Nash equilibrium of an N-person noncooperative game. With the examples of double matrix games, the implementation of the genetic algorithm was discussed and proven to be effective. Qiu [14] proposed an immune algorithm (IA) for solving game equilibrium. The advantages of this algorithm in solving game problems and its stable convergence were verified through examples. Franken [15] investigated the application of coevolutionary training techniques based on PSO to evolve the iterated prisoner's dilemma (IPD). Jia [16] proposed an immune particle swarm optimization (IPSO) to compute finite noncooperative games among N people. The results show that this algorithm is superior to the immune algorithm and original swarm algorithm in solving game problems. Yang [17] proposed the fireworks algorithm (FWA) to compute the Nash equilibrium of N-person noncooperative game. Computer simulation results demonstrate that the proposed algorithm is effective and superior to IPSO.

The above studies all demonstrate the advantages of intelligent algorithms in solving Nash equilibrium problems. However, several issues remain, such as the higher complexity of the algorithm, its slow convergence speed and its low accuracy; in particular, the convergence of the algorithm is not proven theoretically. The goal of this paper is to propose an efficient hybrid of GPDEPSO to solve the game problem and then prove its convergence. The paper is organized as follows: In Section 2, we summarize the concept of a noncooperative N-person game. In Section 3, we propose a hybrid algorithm, GPDEPSO, to compute the Nash equilibrium of noncooperative N-person games. First, we initialize the population with a good point set to ensure the initial particle distribution is global, which will help the algorithm avoid local convergence. Then, the position updating formula of PSO is simplified in a new form that does not have a velocity term, and it is used as a variation operator to perform variation crossover selection with DE. The convergence of GPDEPSO is proved by stochastic functional analysis in Section 4. Section 5 is devoted to computational experiments, and by comparing the algorithm proposed in this paper with other algorithms, its superiority is proven.

2. Game and Nash equilibrium Definitions [18]

Definition 1. We consider an N-**person finite strategic game** defined by $\Gamma = ((N, S_i, U_i, X_i, f_i), i = 1, \dots, n)$,

where

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

(2). $S_i = \{s_{i1}, \dots, s_{im_i}\}$ $\forall i \in N$ is the pure strategy set of player i, m_i represents the number of strategies available to player i, $S = \prod_{i=1}^{n} S_i$ is the Cartesian product of pure strategy sets of all players, and each pure strategy profile meets $(S_1, S_2, \dots, S_n) \in S$;

(3). $U_i : S \to \mathbb{R}\ \forall i \in N$ represents the payoff function;

(4). $X_i = \{x_i = (x_{i1}, \ldots, x_{im_i}) : x_{ik} \geq 0, k = 1, \ldots, m_i, \sum_{k_i=1}^{m_i} x_{ik_i} = 1\}\ \forall i \in N$ is the set of mixed strategies, where x_{ij} is the probability that player i adopts s_{ij} for $j = 1, \cdots, m_i$. $X = \prod_{i=1}^{n} X_i$ is the Cartesian product of mixed strategy sets of all players, and each mixed strategy profile meets $(x_1, x_2, \ldots, x_n) \in X$;

(5). $f_i : X \to \mathbb{R}\ \forall i \in N$ represents the expected payoff function.

$f_i(x_1, \ldots, x_n) = \sum_{k_1=1}^{m_1} \cdots \sum_{k_n=1}^{m_n} U_i(s_{1k_1}, \ldots, s_{nk_n}) \prod_{i=1}^{n} x_{ik_i}$ represents player i getting the expected payoff value when he chooses a mixed strategy $x_i = (x_{i1}, \ldots, x_{im_i}) \in X_i$. where $U_i(s_{ik_i}, \ldots, s_{nk_n})$ represents player i getting the payoff value when each player i chooses the pure strategy $s_{ik_i} \in S_i, i = 1, \ldots, n$.

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

Conclusion 1. Mixed strategy x^* is the Nash equilibrium point if and only if every pure strategy $s_{ik_i} (1 \leq k_i \leq m_i)$ of each player i has $f_i(x^*) \geq f_i(x^* \parallel s_{ik_i})$, where $(x^* \parallel s_{ik_i})$ is only the player i replacing their own strategy with s_{ik_i}, and the other players do not change their own strategy under the condition of equilibrium solution x^*.

In particular, the Γ is a bimatrix game when $N = 2$, (x^*, y^*) is the Nash equilibrium solution if and only if $\begin{cases} x^*Ay^* \geq xAy^{*T}, & \forall x, \\ x^*By^* \geq x^*By^T, & \forall y, \end{cases}$ where A and B are payoff matrices for each player.

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

$$\begin{cases} min\ f(x) = \sum_{i=1}^{n} \max_{1 \leq k_i \leq m_i} \{f_i(x \parallel s_{ik_i}) - f_i(x), 0\} \\ \sum_{k_i=1}^{m_i} x_{ik_i} = 1 \\ 0 \leq x_{ik_i} \leq 1 \\ i = 1, \cdots, n; k_i = 1, \cdots, m_i \end{cases} \tag{2.1}$$

Proof Necessity: Suppose that x^* is the Nash equilibrium point. According to Conclusion 1, we have $f_i(x^*) \geq f_i(x^* \parallel s_{ik_i}), \forall i \in N, \forall s_{ik_i} \in S_i$, then

$$\sum_{i=1}^{n} \max_{1 \leq k_i \leq m_i} \{f_i(x^* \parallel s_{ik_i}) - f_i(x^*), 0\} = 0.$$

Because of $x^* \in X = \prod_{i=1}^{n} X_i$, the constraint condition of (2.1) hold. In formula (2.1), $f(x)$ is nonnegative, so $f(x)$ gets the minimum value 0 at x^*.

Sufficiency: Assume that x^* is the solution of problem (2.1). According to x^* satisfies conditions of (2.1), we can know that $x^* \in X = \prod_{i=1}^{n} X_i$. And because of $f(x^*) = 0$, so $f_i(x^* \parallel s_{ik_i}) - f_i(x^*) \leq 0$, that is $f_i(x^*) \geq f_i(x^* \parallel s_{ik_i}), \forall i \in N, \forall s_{ik_i} \in S_i$. So x^* is the Nash equilibrium point of the game Γ.

For the two-person double matrix game, the above optimization problem can be simplified as:

$$f(x, y) = \max\{\max_{1<i<m}\{A_i y^T - xAy^T, 0\}\} + \max\{\max_{1<j<n}\{xB_j - xBy^T, 0\}\}. \tag{2.2}$$

where A_i is the ith row of matrix A and B_j is the jth column of matrix B. Then, (x^*, y^*) is a Nash equilibrium solution of a two-person noncooperative game, which is also $f(x^*, y^*) = 0$.

3. Differential evolution particle swarm optimization algorithm based on good point set (GPDEPSO)

3.1. Good point set

Good point set was originally proposed by Hua Luogeng et al. [19] and defined as follows:

(1). Let G_s be a unit cube in s-dimensional Euclidean space. Let $x \in G_s$

$$x = (x_1, x_2, \ldots, x_s) \in G_s, 0 \leq x_j \leq 1, j = 1, \ldots, s. \tag{3.1}$$

(2). Let G_s have a set of points $P_n(i)$ with n points:

$$\begin{aligned}
&P_n(i) = \{x^{(1)}, \cdots, x^{(i)}, \cdots, x^{(n)}\} \\
&x^{(i)} = \{\{x_1^{(i)}\}, \cdots, \{x_j^{(i)}\}, \cdots, \{x_s^{(i)}\}\} \\
&0 \leq x_j^{(i)} \leq 1;\ i = 1, \cdots, n;\ j = 1, 2, \cdots, s.
\end{aligned} \tag{3.2}$$

where $\{\cdot\}$ represents the decimal part of the value.

(3). For any given point $r = (r_1, r_2, \cdots, r_s) \in G_s$, let $N_n(r) = N_n(r_1, r_2, \cdots, r_s)$ represents the number of points in $P_n(i)$ that meet the following inequality:

$$0 \leq x_j^{(i)} \leq r_j,\ j = 1, 2, \cdots, s.$$

$\varphi(n) = sup|(N_n(r)/n) - |r||$, where $|r| = r_1 r_2 \cdot \cdots \cdot r_s$, is called a deviation of the point set $P_n(i)$. If $\forall n, \varphi(n) = O(1)$, then $P_n(i)$ is said to be uniformly distributed on G_s and the deviation is $\varphi(n)$.

(4). Let $r \in G_s$, and $P_n(i) = \{(r_1 * i, r_2 * i, \cdots, r_s * i,), i = 1, \cdots, n\}$, the deviation $\varphi(n)$ meets $\varphi(n) = C(r, \varepsilon) \cdot n^{-1+\varepsilon}$, where $C(r, \varepsilon)$ is a constant related only to r, ε(ε is an arbitrarily small positive number), then, the $P_n(i)$ is called the good point set, r is called the good point.

In this paper, we take

$$r_i = \{e^i, 1 \leq i \leq s\}. \tag{3.3}$$

In the following, we generate two distribution maps (Figures 1 and 2) of 500 populations with the random point method and the exponential good point set method, respectively.

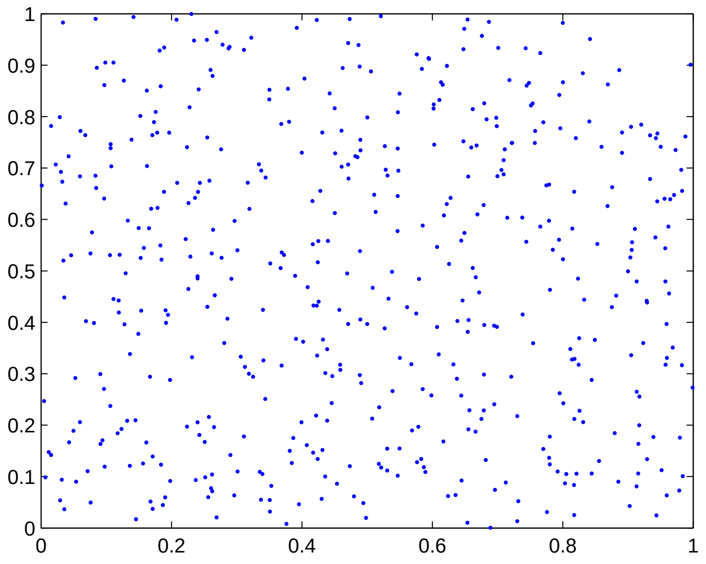

Figure 1. Two-dimensional initial population generated by random method.

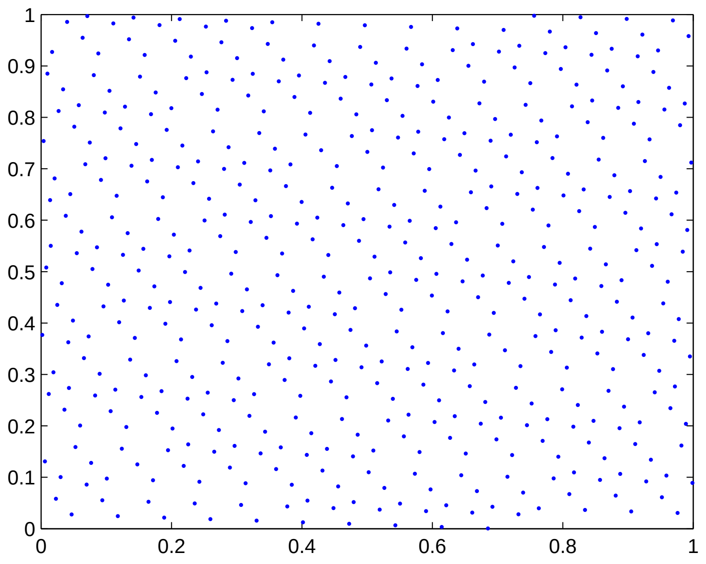

Figure 2. Two-dimensional initial population generated by exponential sequence.

As shown, the good point sequence is more uniform and global than the random point method. In addition, the good point method is independent with spatial dimensions, thus it can be well adapted to the high-dimensional problems. It is also stable that the distribution results are the same every time when the number of points is the same.

3.2. Differential evolution algorithm (DE)

DE is a new simple and robust evolutionary algorithm that was first introduced by Storm and Price [20]. There are four operations of DE: initialization, variation, crossover, and selection operation.

(1). Initialization

Let each individual in a population be

$$X = (x_{i1}, \dots, x_{ij}, \dots, x_{iD}) \;\; i = 1, \dots, N; j = 1, \dots, D. \tag{3.4}$$

where N and D represent, respectively, the population size and space dimension. In the study of DE, it is generally assumed that the initial population conforms to uniform probability distribution, and its form is as follows:

$$x_{ij} = rand[0,1] \cdot (x_j^U - x_j^L) + x_j^L \;\; i = 1, \dots, N; j = 1, \dots, D. \tag{3.5}$$

where $rand[0,1]$ represents random values in the range [0,1], x_j^U and x_j^L represent, respectively, the upper and lower bounds of parameter variables.

(2). Variation

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

$$\begin{aligned} &V_i^{t+1} = X_{r1}^t + F \cdot (X_{r2}^t - X_{r3}^t) \\ &X_{r1}^t, X_{r2}^t, X_{r3}^t \in X; \;\; i = 1, \dots, N. \end{aligned} \tag{3.6}$$

where $r1$, $r2$, and $r3$ are different integers between 1 and N, and they are also different from i; F is a constriction factor to control the size of difference of two individuals, and t is the current iterate point.

(3). Crossover

We use the crossover between the parent and offspring with the given probability for generating new individual $U = (u_{i1}, \dots, u_{iD})$:

$$u_{ij}^{t+1} = \begin{cases} v_{ij}^{t+1}, & if(rand(j) \le CR)or(j = rnbr(i)), \\ x_{ij}^{t+1}, & otherwise. \end{cases} \tag{3.7}$$

where $rand(j)$ is random value in the range [0,1], CR is crossover operator in the range [0,1], and $rnbr(i) \in \{1, \dots, D\}$ is a randomly selected sequence, which ensures that a new individual gets at least one component value from the variation vector.

(4). Selection

The offspring X_i^{t+1} is generated by selecting the individual and parent according to the following equation:

$$X_{ij}^{t+1} = \begin{cases} U_i^{t+1}, & if(f(U_i^{t+1}) < f(X_i^t)), \\ X_i^t, & otherwise. \end{cases} \tag{3.8}$$

where $f(\cdot)$ is the fitness function value.

3.3. *Particle swarm optimization algorithm (PSO)*

PSO was proposed by Eberhart and Kennedy [21], and is a random search algorithm, which is inspired by the activities of flocking birds. PSO uses a population of individuals called particles, and there are two main operations in PSO, speed updating and position updating:

$$\begin{aligned} v_{ij}^{t+1} &= \omega v_{ij}^{t} + c_1 r_1 (x_{pbest}^{t} - x_{ij}^{t}) + c_2 r_2 (x_{gbest}^{t} - x_{ij}^{t}) \\ x_{ij}^{t+1} &= x_{ij}^{t} + v_{ij}^{t} \\ i &= 1, \dots, N; \ j = 1, \dots, D. \end{aligned} \tag{3.9}$$

The ω is the inertia weight, c_1 and c_2 are acceleration constants, r_1 and r_2 are random values in the interval (0, 1), v_{ij}^{t} is the ith particle's velocity in generation t, x_{ij}^{t} is the ith particle's position in generation t, x_{pbest}^{t} is the personal best position of particle i before generation t, and x_{gbest}^{t} is the global best position in the searching history [22].

Speed updating can be further explained as follows: ωv_{ij}^{t} is called the current state of the particle with the ability to balance global and local search. $c_1 r_1 (x_{pbest}^{t} - x_{ij}^{t})$ is the cognitive modal of the particle, which represents the ability of learning from itself and endows particles with strong local search capabilities. $c_2 r_2 (x_{gbest}^{t} - x_{ij}^{t})$ represents the social cognition modal of the particle and information sharing among particles, that is, the ability to learn from the entire population and endow particles with strong global search ability. Then, the position updating makes the particles reach the new position.

A simplified position transformation formula without velocity term is expressed as follows [23]:

$$x_{ij}^{t+1} = \omega x_{ij}^{t} + c_1 r_1 (x_{pbest}^{t} - x_{ij}^{t}) + c_2 r_2 (x_{gbest}^{t} - x_{ij}^{t}). \tag{3.10}$$

3.4. *GPDEPSO experimental steps and its implementation*

The steps of GPDEPSO is described as follows :

Step 1: Set the parameters of GPDEPSO, such as N, D, CR, F_0, ω_{min}, ω_{max}, x^L, x^U, c_1, c_2, and set the maximum number of iteratios T, accuracy ε, where

$$\begin{aligned} \omega &= \omega_{max} - (\omega_{max} - \omega_{min}) \cdot t/T \\ F &= F_0 \cdot 2^{\lambda} \\ \lambda &= e^{1-T/(T+1-t)} \end{aligned}$$

Step 2: Randomly generate N initial populations $P(0)$ by using a good point set, and $\sum_{k_i=1}^{m_i} x_{ik_i} = 1, x_{ik_i} \geq 0, x_{ik_i} \in X_i, \ \ i = 1, \dots, N; k_i = 1, \dots, m_i$.

Step 3: Calculate the fitness function value $f(x)$ of each individual in population $P(t)$ and determine the x_{pbest}^{t} and x_{gbest}^{t}.

Step 4: The next generation population $P_1(t)$ is generated by variation of formula (3.10), and population $P_2(t)$ is generated by variation of formula (3.6).

Step 5: The population $P'(t)$ is generated by crossover of formula (3.7).

Step 6: According to formula (3.8), populations $P(t)$ and $P'(t)$ are selected to generate offspring population $P(t+1)$ and the fitness function value of population $P(t+1)$ is calculated.

Step 7: Determine whether to end according to the accuracy and the maximum number of iterations, and output the optimal value; otherwise, turn to step 3.

The pseudo code of GPDEPSO is as follows:

Algorithm 1 GPDEPSO

```
Input: Parameters $N$, $D$, $CR$, $F_0$, $T$, $\omega_{min}$, $\omega_{max}$, $x^L$, $x^U$, $c_1$, $c_2$, $\varepsilon$
Output: The best vector (Solution) $\cdots$ $\Delta$
  $t \leftarrow 1$    (Initialization with good point set)
  for $i = 0$ to $N$ do
    for $j = 0$ to $D$ do
      $x^t_{i,j} = rand(0,1) \cdot (x^U_{i,j} - x^L_{i,j}) + x^L_{i,j}$
    end for
  end for
  while $|f(\Delta)| \geq \varepsilon$ or $t \leq T$ do
    for $i = 0$ to $N$ do
      (Update the $x^t_p, x^t_g$)
      for $j = 0$ to $D$ do
        $x^t_p = x^t_{pbest}$
        $x^t_g = x^t_{gbest}$
      end for
      (Variation and Crossover)
      for $j = 0$ to $D$ do
        $v^t_{i,j} = Variation\ (x^t_p, x^t_g, x^t_{i,j})$
        $u^t_{i,j} = Crossover\ (v^t_{i,j}, x^t_{i,j})$
      end for
      (Selection)
      if $f(u^t_{i,j}) < f(x^t_{i,j})$ then
        $x^t_{i,j} \leftarrow u^t_{i,j}$
      else $\{f(x^t_{i,j}) < f(\Delta)\}$
        $\Delta \leftarrow x^t_{i,j}$
      end if
    end for
    $t = t + 1$
  end while

  return  the best vector $\Delta$
```

4. Convergence analysis of GPDEPSO

The main operations of GPDEPSO are a variation operation based on PSO, a crossover operation, and a selection operation of DE. Considering variation crossover operations as a difference operator based on PSO (DOPSO), which is essentially a variation operation, the individual X_i generates an intermediate individual V_i with at least $1 - (1 - CR - 1/D)^D$ probability. The selection operation is regarded as a selection operator (SO), which is strictly based on the strategy of survival of the fittest. It produces a better new generation of individuals by eliminating the inferior individuals in X_i and V_i.

For the minimal optimization problem, GPDEPSO evaluation function $\{f(X_i^t) : 1 < t < T\}$ is a monotonous nonincremental sequence. The convergence of GPDEPSO is illustrated by He's stochastic functional analysis method in the literature [24]. An iteration process of GPDEPSO is abstracted as a composite random mapping of a DOPSO and a SO.

Definition 3. The process of generating test vector U by DOPSO is to recombine and transform everyone dimensional component of the target vector according to probability $\theta = CR + 1/D$. In addition, the process can be described as **a random mapping** of the solution space $\Psi_1 : \Omega \times S \to S^2$, which is defined as:

$$\mu\{\omega|\Psi_1(\omega, X_i) = \langle X_i, V_i\rangle\} = \mu\{V_i = X_{r1} + F \cdot (X_{r2} - X_{r3})\} = 1 - (1-\theta)^D. \tag{4.1}$$

where (Ω, A, μ) is the complete probability measure space and Ω is the nonempty abstract set, and its element ω is a basic event. A is the $\sigma - algebra$ composed of some subsets of Ω, μ is the probability measure on A, and S is the solution space: $S = \{X|X = (x_1, x_2, \cdots, x_D), x_j^L \le x_j \le x_j^U, j = 1, 2, \cdots, D\}$.

Definition 4. The selection operator SO is the process of selecting the optimal individual from the test vector U_i and the target vector X_i according to the greedy selection method. It is **a mapping** on the solution space $\Psi_2 : S^2 \to S$:

$$\Psi_2(\langle X_i, U_i\rangle) = \min\{f(X_i), f(U_i)\}. \tag{4.2}$$

Combining the above two definitions, it can be seen that one iteration of GPDEPSO is equivalent to mapping $\Psi = (\Psi_2 \circ \Psi_1) : \Omega \times S \to S(\Omega \times S \to S^2 \to S)$ to the current population $P(t)$, where Ψ is a reverse-order synthesis mapping corresponding to DOPSO and SO mapping. Then, for the new population $P(t+1)$ it can be expressed as $P(t+1) = \Psi(\omega, P(t)) = \Psi_2(\Psi_1(\omega, P(t))), 0 \le t \le T-1$.

Let $f(X_{best}^t)$ be the fitness function value of the best individual X_{best}^t in $P(t)$. Under the effect of Ψ, the new generation population generated by GPDEPSO will be superior to the previous one. Therefore, the sequence $\{f(X_{best}^t)\}_{1 \le t \le T}$ is necessarily a monotonous nonincremental sequence (assuming that $f(X)$ is the minimum optimization function in this paper). In addition, the evolutionary process of GPDEPSO can also be characterized by the optimal individual, and the mapping Ψ can be redefined as a mapping corresponding to the process of generating the optimal individual, that is, $X_{best}^{t+1} = \Psi(\omega, X_{best}^t) = \Psi_2(\Psi_1(\omega, X_{best}^t))$.

Lemma 1. Let $\lambda : S \times S \to R$ be the distance defined on S and satisfy $\lambda(X_i, X_j) = |f(X_i) - f(X_j)|, \forall X_i, X_j \in S$; then, (S, λ) is a complete separable metric space.

Similar to the method presented in Ref. [25], Lemma 1 is established.

Theorem 2. The random mapping $\Psi = (\Psi_2 \circ \Psi_1) : \Omega \times S \to S$ formed by one iteration of GPDEPSO is a random contraction operator.

Proof. According to the definition of DOPSO and SO operation, it can be seen that the new population generated by GPDEPSO in each iteration is better than the previous one. Therefore, for random mapping $\Psi = (\Psi_2 \circ \Psi_1) : \Omega \times S \to S$, there exists a random variable with nonnegative real value $0 \le K(\omega) < 1$, which makes the following formula hold:

$$\begin{aligned} \lambda(\Psi(\omega, X_{best}^{t-1}), \Psi(\omega, X_{best}^t)) &= \lambda(X_{best}^t, X_{best}^{t+1}) \\ &= |f(X_{best}^t) - f(X_{best}^{t+1})| \\ &\le K(\omega) \cdot |f(X_{best}^{t-1}) - f(X_{best}^t)| \\ &= K(\omega) \cdot \lambda(X_{best}^{t-1}, X_{best}^t). \end{aligned} \tag{4.3}$$

letting

$$\Omega_0 = \{\omega|\lambda(\Psi(\omega, X_{best}^{t-1}), \Psi(\omega, X_{best}^{t})) \leq K(\omega) \cdot \lambda(X_{best}^{t-1}, X_{best}^{t})\} \subseteq \Omega$$

then

$$\mu(\Omega_0) = 1.$$

Therefore, the mapping formed by GPDEPSO, $\Psi : \Omega \times S \to S$ is a random contraction operator. □

Lemma 2. (Random Contraction Mapping Theorem) [26]

Letting $\Psi : \Omega \times S \to S$ be a random operator, satisfying that for almost all $\omega \in \Omega$ and $\Psi(\omega)$ is contractive operator, then for $\Psi(\omega)$ there exists a unique random fixed point $\xi(\omega)$, and $\Psi(\omega, \xi(\omega)) = \xi(\omega)$.

According to Theorem 2, the random mapping Ψ is a random contraction operator. By using the conclusion of Lemma 2, for $\Psi(\omega)$ there must exist a unique random fixed point. Then, the convergence criterion of GPDEPSO can be derived, which means that GPDEPSO is asymptotically convergent.

5. Experimental design and results

5.1. Numerical examples

We take the classical games given in [16] and [17] as our numerical examples, and solve them separately by using the GPDEPSO given in this paper and comparing with other algorithms. (For Examples 1-3, the parameters of the algorithm are set as $N = 50$, $T = 100$, $\varepsilon = 10^{-4}$, etc. For examples 4 and 5 of high dimension, the parameters setting are, respectively, $N = 50$, $T = 100$, $\varepsilon = 10^{-5}$, etc., and $N = 100$, $T = 300$, $\varepsilon = 10^{-5}$, etc.).

Example 1 Prisoner's Dilemma Game $\Gamma(X_1, Y_1, A_1, B_1)$:

$$A_1 = \begin{bmatrix} -8 & 0 \\ -15 & -1 \end{bmatrix}, \quad B_1 = \begin{bmatrix} -8 & -15 \\ 0 & -1 \end{bmatrix}.$$

Example 2 Guessing Game $\Gamma(X_2, Y_2, A_2, B_2)$:

$$A_2 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, \quad B_2 = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}.$$

Example 3 Supervisory Game $\Gamma(X_3, Y_3, A_3, B_3)$:

$$A_3 = \begin{bmatrix} 0 & 50 \\ 30 & 30 \end{bmatrix}, \quad B_3 = \begin{bmatrix} -10 & -50 \\ 60 & 70 \end{bmatrix}.$$

Example 4 Consider a Three-Dimensional Payoff Matrix Game $\Gamma(X_4, Y_4, A_4, B_4)$:

$$A_4 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \quad B_4 = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}.$$

Example 5 Consider a 10-Dimensional Payoff Matrix Game $\Gamma(X_5, Y_5, A_5, B_5)$:

$$A_5 = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}, \quad B_5 = \begin{bmatrix} 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ 1 & 0 & \cdots & 0 \end{bmatrix}.$$

5.2. Results and discussion

The following Tables 1–4 present computational results of the above examples by GPDEPSO.

Table 1. Results of Prisoner's Dilemma Game $\Gamma(X_1, Y_1, A_1, B_1)$.

Test times	Iteration times	Mixed strategy of player 1	Mixed strategy of player 2	Fitness value
Firstly	1	(1,0)	(1,0)	0
Secondly	2	(1,0)	(1,0)	0
Thirdly	1	(1,0)	(1,0)	0
Fourthly	1	(1,0)	(1,0)	0

Table 2. Results of Guessing Game $\Gamma(X_2, Y_2, A_2, B_2)$.

Test times	Iteration times	Mixed strategy of player 1	Mixed strategy of player 2	Fitness value
Firstly	4	(0.5000,0.5000)	(0.5000,0.5000)	3.8124e-05
Secondly	4	(0.5000,0.5000)	(0.5000,0.5000)	2.2619e-05
Thirdly	7	(0.5000,0.5000)	(0.5000,0.5000)	8.2926e-05
Fourthly	6	(0.5000,0.5000)	(0.5000,0.5000)	8.5655e-05

Table 3. Results of Supervisory Game $\Gamma(X_3, Y_3, A_3, B_3)$.

Test times	Iteration times	Mixed strategy of player 1	Mixed strategy of player 2	Fitness value
Firstly	10	(0.2000,0.8000)	(0.4000,0.6000)	6.7765e-05
Secondly	12	(0.2000,0.8000)	(0.4000,0.6000)	9.8433e-05
Thirdly	11	(0.2000,0.8000)	(0.4000,0.6000)	2.9027e-05
Fourthly	12	(0.2000,0.8000)	(0.4000,0.6000)	3.3894e-05

Table 4. Results of Three-Dimensional Payoff Matrix Game $\Gamma(X_4, Y_4, A_4, B_4)$.

Test times	Iteration times	Mixed strategy of player 1	Mixed strategy of player 2	Fitness value
Firstly	19	(0.3333,0.3333,0.3333)	(0.3333,0.3333,0.3333)	8.6491e-06
Secondly	20	(0.3333,0.3333,0.3333)	(0.3333,0.3333,0.3333)	9.3384e-06
Thirdly	17	(0.3333,0.3333,0.3333)	(0.3333,0.3333,0.3333)	9.3761e-06
Fourthly	19	(0.3333,0.3333,0.3333)	(0.3333,0.3333,0.3333)	6.3213e-06

From above four tables, we can see that the Nash equilibrium can be obtained under the given parameters by using GPDEPSO proposed in this paper. In addition, the accuracy of fitness function values is higher approximately 10^4 times and 10^2 times than that of Refs. [16] and [17]. It can be seen that the speed of this algorithm does not be affected under the condition of increasing the population size (N=50). On the contrary, according to the results of Examples 1, 3, and 4, it is show that the number of iterations is reduced significantly, and the Example 4 is the most obvious. Compared with Refs. [16], the number of iteration of Example 4 is reduced by approximately 16 times, which is approximately 8 times compared with Ref. [17]. Through the above analysis, GPDEPSO is superior to the algorithms presented in the existing literature in terms of iteration times and accuracy of results. Furthermore, although the number of populations is lager than that in the previous literature, it does not affect the speed and accuracy of the algorithm, but plays a powerful role in finding the global optimal solution.

The following are two comparison figures for solving Example 5, a high-dimensional payoff matrix game. Figure 3 is a comparison between GPDEPSO and hybrid differential evolution particle swarm optimization algorithm (DEPSO). Figure 4 is a comparison of GPDEPSO, differential evolution algorithm based on good point set (GPDE), and particle swarm optimization algorithm based on good point set (GPPSO). The Nash equilibrium solution calculated by all methods is $x^* = y^* = (0.1, \ldots, 0.1)_{1\times10}$.

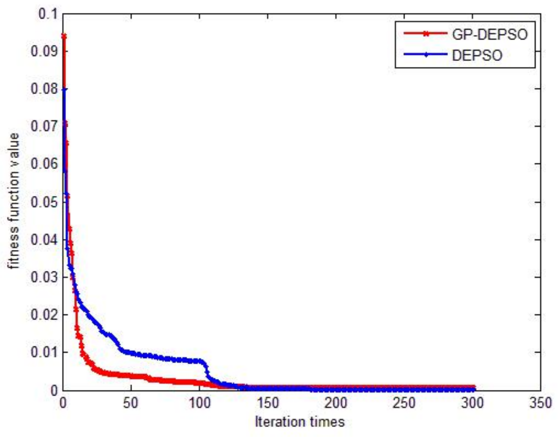

Figure 3. Comparison of GPDEPSO and DEPSO.

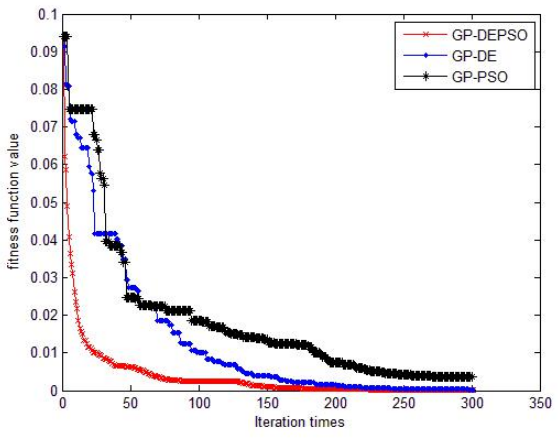

Figure 4. Comparison of GPDEPSO, GPDE and GPPSO.

As shown in Figures 3 and 4, the high-dimensional payoff matrix game can be solved better by GPDEPSO. Compared with Ref. [17], it can be seen that not only the calculation speed is greatly improved, but the accuracy of the fitness function value is also better. It can be seen from Figure 3 that GPDEPSO converges faster and more smoothly than DEPSO, which indicates that the good point set can avoid falling into a local optimum during the calculation process. From Figure 4, comparing GPDEPSO, GPPSO, with GPDE, and find that GPDEPSO combines the ability of PSO to find the global optimum quickly with the fast convergence ability of DE. In Example 5, GPDEPSO can quickly find the global optimal solution within the first 50 iterations, and the approximate exact solution can be obtained after 150 iterations. Therefore, GPDEPSO has a good advantage in solving Nash equilibrium problems with a high-dimensional payoff matrix.

6. Conclusions

We propose GPDEPSO from the point of view of DE, considering the different characteristics of DE and PSO, and we use a good point set to make the initial data more uniform. This algorithm combines the advantages of DE and PSO, which not only ensures the simple operation, easy implementation, and fast convergence of DE, but also enhances its global optimization ability. By solving the Nash equilibrium of noncooperative games, we find that the proposed algorithm is superior to the comparative algorithms in terms of the calculation accuracy and convergence. In particular, for a high-dimensional payoff matrix game, the efficiency of the algorithm is remarkable. In the future, since a Nash equilibrium problem is a complex, NP-hard problem, it would be interesting to consider the influence of different variation operations and selection strategies on the solution of the Nash equilibrium, as well as to consider the Nash equilibrium problems for more complex multiobjective games and multiple games.

Acknowledgments

This study was supported by the National Natural Science Foundation of China (71961003), the Qian Jiaohe YJSCXJH ((2019)029), the Qian Kehe Platform Talents ((2017)5788), and the Qian Kehe LH Zi ((2017)7223).

Conflict of interest

The authors declare no conflict of interest.

References

1. T. Kunieda, K. Nishimura, Finance and Economic Growth in a Dynamic Game, *Dyn. Games Appl.,* **8** (2018), 588–600.
2. G. M. Korres, A. Kokkinou, Political Decision in a Game Theory Approach, *European Socio-Economic Integration,* **28** (2013), 51–61.
3. A. Traulsen, Biological Models in Game Theory, *Journal of Statistical Theory and Practice,* **10** (2016), 472–474.

4. M. Qingfeng, T. Shaohong, L. Zhen, Ch. Bingyao, Sh. Weixiang, A Review of Game Theory Application Research in Safety Management, *IEEE Access,* **8** (2020), 107301–107313.
5. Q. Wang, L. Zhao, L. Guo, J. Ran, Z. Lijun, X. Yujing, et al., A Generalized Nash Equilibrium Game Model for Removing Regional Air Pollutant, *J. Clean. Prod.,* **227** (2019), 522–531.
6. J. Sheng, W. Zhou, B. Zhu, The Coordination of Stakeholder Interests in Environmental Regulation: Lessons From China's Environmental Regulation Policies From The Perspective of The Evolutionary Game Theory, *J. Clean. Prod.,* **249** (2020), 119385.
7. C. E. Lemke, J. T. Howson, Jr, Equilibrium Points of Bimatrix Games, *Journal of the Society for Industrial and Applied Mathematics,* **12** (1964), 413–423.
8. S. Govindan, R. Wilson, A Global Newton Method to Compute Nash Equilibria, *Journal of Economic Theory,* **110** (2003), 65–86.
9. J. Zhang, B. Qu, N. Xiu, Some Projection-like Methods for The Generalized Nash Equilibria, *Comput. Optim. Appl.,* **45** (2010), 89–109.
10. Y. Ya xiang, A Trust Region Algorithm for Nash Equilibrium Problems, *Pac. J. Optim.,* **7** (2011), 125–138.
11. N. G. Pavlidis, K. E. Parsopoulos, M. N. Vrahatis, Computing Nash Equilibria Through Computational Intelligence Methods, *J. Comput. Appl. Math.,* **175** (2005), 113–136.
12. U. Boryczka, P. Juszczuk, Differential Evolution as a New Method of Computing Nash Equilibria, *Transactions on Computational Collective Intelligence IX,* **7770** (2013), 192–216.
13. C. ShiJun, S. YongGuang, W. ZongXin, A Genetic Algorithm to Acquire the Nash Equilibrium, *System Engineering,* **19** (2001), 67–70.
14. Q. ZhongHua, G. Jie, Z. YueXing, Applying Immune Algorithm to Solving Game Problem, *Journal of Systems Engineering,* **21** (2006), 398–404.
15. N. Franken, A. P. Engelbrecht, Particle Swarm Optimization Approaches to Coevolve Strategies for The Iterated Prisoner's Dilemma, *IEEE T. Evolut. Comput.,* **9** (2005), 562–579.
16. J. WenSheng, X. ShuWen, Y. JianFeng, W. S. Hu, Solving Nash Equilibrium for N-persons' Non-cooperative Game Based on Immune Particle Swarm Algorithm, *Application Research of Computers,* **29** (2012), 28–31.
17. Y. Yanlong, X. Shuwen, X. Shunyou, J. Wensheng, Solving Nash Equilibrium of Non-Cooperative Game Based on Fireworks Algorithm, *Computer Applications and Software,* (in Chinese) **35** (2018), 215–218.
18. Y. Jian, *Selection of Game Theory*, Chinese: Science Press, 2014.
19. H. Luogeng, W. Yuan, *Application of Number Theory in Modern Analysis*, Beijing: Science Press, 1978.
20. R. Storn, K. Price, Minimizing The Real Functions of The ICEC'96 Contest by Differential Evolution, *Proceedings of IEEE International Conference on Evolutionary Computation IEEE,* (1996), 824–844.
21. R. C. Eberhart, J. Kennedy, A New Optimizer Using Particle Swarm Theory. In: *6th Symposium on Micro Machine and Human Science*, Nagoya, Japan, (1995), 39–43.

22. Y. Wu, Y. Wu, X. Liu, Couple-based Particle Swarm Optimization for Short-term Hydrothermal Scheduling, *Appl. Soft Comput.,* **74** (2019), 440–450.

23. J. Shen, F. Liang, M. Zheng, New Hybrid Differential Evolution and Particle Swarm Optimization Algorithm and Its Application, *Sichuan Daxue Xuebao (Gongcheng Kexue Ban) Journal of Sichuan University (Engineering Science Edition),* **46** (2014), 38–43.

24. Y. C. He, X. Z. Wang, K. Q. Liu, Y. Q. Wang, Convergent Analysis and Algorithmic Improvement of Differential Evolution, *Journal of Software,* **21** (2010), 875–885.

25. M. Q. Li, J. S. Kou, D. Lin, S. Q. Li, *Basic Theory and Application of Genetic Algorithm*, Beijing: Science Press, (in Chinese) (2002), 115–119.

26. T. S. Lu, *Random Functional Analysis and Its Application*, Qingdao: Qingdao Ocean University Press, 1990.