Solving Nash Equilibrium for Non-cooperative Game Based on the Particle Swarm Optimization Integrating Multiply Strategies

Yuliang Zhao[1,*], Yexin Song[1] , Liwen Kang[2]

1. Department of Basic Courses, Naval University of Engineering, Wuhan, Hubei, 430000
2. Marine Map Information Center, Tianjin, 300450
E-mail: yl.zhao0811@gmail.com

Abstract: In order to solve Nash equilibrium problem of n-person finite non-cooperative game, this paper involved adaptive adjustment of inertia weight, dynamic reverse learning and local mutation search strategy into the basic particle swarm optimization (PSO), and proposed a particle swarm optimization integrating multiply strategies algorithm(IMSPSO). This algorithm introduces the state information of individual particles into the inertial weight strategy, independently adjusts the inertial weight of each particle, and reflects the difference of individual particles' weight requirements. When the algorithm is detected to be in the local optimum, a dynamic reverse learning strategy is introduced to expand the search area and enhance the algorithm's global exploration ability. At the same time, the small-scale mutation search operation of individual local neighborhood is used to guide the local learning and search of particles, so as to enhance the mining ability of particles in local space. The algorithm is used to solve two Nash equilibrium problems of non-cooperative games. The experimental results show that the algorithm can achieve good results and its performance is better than the basic particle swarm optimization algorithm.

Key Words: Non-cooperative game, Nash equilibrium, Adaptive adjustment of inertia weight, Dynamic reverse learning, Local mutation search

1 INTRODUCTION

Game theory is a mathematical theory that studies the interaction between decision makers. It mainly studies how relevant benefits agents use the information they have to make decisions. n-person finite non-cooperative game is an important part of game theory and is widely used in the selection of optimal strategies in economic, political, military computer science and other fields [1-4]. Nash equilibrium is the core and most important concept of non-cooperative games. The research on Nash equilibrium mainly involves existence, stability and solution mechanism design. Since the solution of n-person finite non-cooperative game is a NP-hard problem, the research on Nash equilibrium solution method has always been a difficult problem. At present, there are some well-known solutions, such as geometric graph algorithm[5], Lemke-Howson algorithm[6], homotopy path method[7], global Newton method[8], trust region method[9] and other traditional calculation methods. However, with the expansion of game scale, traditional calculation methods are almost impossible to solve due to the increasing complexity of solution and too long computation time. And heuristic algorithms, such as genetic algorithm[10], immune algorithm[11], fireworks algorithm[12], particle swarm optimization algorithm[13], ant colony algorithm[14], etc., are able to show better ability to solve such problems.

Particle swarm optimization algorithm, as a new swarm intelligence optimization algorithm, has the advantages of simple form, fast convergence speed and flexible parameter adjustment mechanism, and has been widely concerned and applied. Research results show that particle swarm optimization algorithm performs well in searching for Nash equilibrium[15]. However, the disadvantage of the basic particle swarm optimization algorithm is that the global search ability is not strong, easy to premature convergence and fall into the local optimum. At present, algorithms for these problems are constantly improved, such as adjusting control parameters to balance local search ability and global search ability, designing different neighborhood topologies to replace global topologies, and introducing auxiliary search strategies into basic particle swarm optimization algorithm[16-17]. The above improvements have greatly improved the optimization ability of particle swarm optimization algorithm, which is of great reference significance.

In this paper, a particle swarm optimization algorithm based on multi-strategy fusion is proposed, which provides a new way for particle swarm optimization to solve Nash equilibrium of non-cooperative game. This algorithm integrates the strategies of inertia weight adaptive adjustment, dynamic reverse learning and local variation search, so as to increase individual diversity, improve the algorithm's global exploration ability and local search ability, so as to avoid particle swarm falling into local optimum. The simulation results verify that the IMSPSO algorithm is superior to the basic particle swarm optimization algorithm in solving n-person finite non-cooperative game Nash equilibrium.

2 NASH EQUILIBRIUM FOR NONCOOPERATIVE GAME

Let the n-person finite non-cooperative game be represented by triple $\Gamma(n,\{S^i\}_{i\in n},\{u^i\}_{i\in n})$, where n represents the number of players, $S^i=\{s_1^i,s_2^i,\cdots,s_{m_i}^i\}_{i\in n}$ represents the strategy set for a limited number of strategies of player i , and $u^i:S\to R$ represents profit function of player i .

The mixed strategy set of player i in the game is:

This work is supported by National Nature Science Foundation under Grant NO. 71171198, NO. 41631072, and NO. 41771487.

$$S^* = \{(x_1^i, x_2^i, \cdots, x_{m_i}^i) \mid x_k^i \geq 0 (k = 1, 2, \cdots, m_i), \sum_{k=1}^{m_i} x_k^i = 1\}$$

Where, $x_k^i (k = 1, 2, \cdots, m_i)$ represents the probability that player i take the pure strategy s_k^i. And the expected payoff for player i in mixed situation $x = (x_1, x_2, \cdots, x_n)$ $(x_i \in S_i^*, i = 1, 2, \cdots, n)$ is as follows:

$$E_i(x) = \sum_{s_{j_1}^1 \in S^1} \sum_{s_{j_2}^2 \in S^2} \cdots \sum_{s_{j_n}^n \in S^n} u^i(s_{j_1}^1, s_{j_2}^2, \cdots, s_{j_n}^n) x_{j_1}^1 x_{j_2}^2 \cdots x_{j_n}^n, i = 1, 2, \cdots n$$

Definition 1. If mixed strategy x^* meets $E_i(x^*) \geq E_i(x^* \| x_i)$, $(i = 1, 2, \cdots, n)$, then x^* is one of a Nash equilibrium solution of n-person finite non-cooperative game. Where, $x^* \| x_i$ represents that only player i replaces its strategy in the equilibrium solution x^* with x_i, and none of other players change their strategies in the equilibrium solution.

3 PARTICLE SWARM OPTIMZATION ALGORITHM INTEGRATING MULTIPLY STRATEGIES

Particle swarm optimization is a swarm intelligence optimization algorithm which simulates the foraging process of birds. For an n-dimensional minimization problem, let the population size be N, the position and velocity of the ith $(i = 1, 2, \cdots, N)$ particle are expressed as $z_i = (z_i^1, z_i^2, \cdots, z_i^n)$ and $v_i = (v_i^1, v_i^2, \cdots, v_i^n)$ respectively. And the updating formulas of z_i and v_i are as follows:

$$v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (p_i(t) - z_i(t)) + c_2 \times r_2 \times (p_g(t) - z_i(t)) \quad (1)$$

$$z_i(t+1) = z_i(t) + v_i(t+1) \quad (2)$$

Where, w represents inertia weight, c_1 and c_1 represent the acceleration factor, r_1 and r_2 are random number in $(0,1)$, $v_i(t+1)$ is the update velocity of particle i at iteration t, $z_i(t+1)$ is the new position of particle i at iteration t, $p_i(t)$ is best position of particle i at iteration t; $p_g(t)$ is the global best position of all particle at iteration t.

Aiming at the shortcomings of basic particle swarm optimization algorithm, such as premature convergence and low convergence precision, this paper proposes an optimization integrating multiply strategies algorithm. The specific improvement strategy is as follows:

3.1 Adaptive Adjustment of Inertia Weight

The ratio of individual particle fitness to population mean fitness can be used to describe the relative position of individual particle in the population. When the ratio is large, it indicates that the individual particles are far away from the optimal position of the group, and the inertial weight should be appropriately increased to improve the exploration ability. On the contrary, the inertia weight can be appropriately reduced to improve the mining capacity. Based on this, this paper uses the ratio of individual fitness to population average fitness to construct the adaptive inertia weight, which is described as follows:

The adaptive inertia weight of ith particle is as follows:

$$w_i = w_{start} - (w_{start} - w_{end}) \times \left(\frac{t}{T_{\max}} \right)^{\lambda_i} \quad (3)$$

Where, w_{start} is the start weigh, w_{end} is the end weigh, and usually $w_{start} = 0.9$, $w_{end} = 0.4$. $T_{\max}$ is the maximum iterations. λ_i is the adaptive exponential factor, and its calculation method is as follow:

$$\lambda_i = 1.8 + 3.6 \times \frac{\arctan\left(\log \frac{fitness(z_i)}{fitness_{avg}} \right)}{\pi} \quad (4)$$

Where, $fitness(z_i)$ represents the fitness of the ith particle and $fitness_{avg}$ represents the average fitness of all particles.

3.2 Dynamic Reverse Learning Strategy

When the population optimal solution remains unchanged in the last iteration (update stagnation times) and the fitness is not zero, the algorithm is considered to have fallen into local optimum. Dynamic reverse learning can significantly change the position structure of individuals by reversing the search variables of each dimension of particles, so as to get away from the solution point of local extreme value and guide the algorithm to escape from the local extreme value. Therefore, in this paper, when the algorithm falls into the local optimum, the dynamic reverse learning method is adopted to increase the population diversity so as to enhance the ability of particles to explore the global space. The specific description is as follows:

First, the algorithm determines whether the algorithm is trapped in local optimum according to formula (5). If it is true, the inverse learning of each dimension of particles can be carried out according to formula (6). If the reverse solution of some dimensions exceeds the boundary [0,1] and becomes an infeasible solution, the algorithm can be reset according to formula (7).Then, each dimension is normalized according to equation (8). Then, we determine whether $f(z_i''(t)) < f(z_i(t))$ is satisfied, if so, then set $z_i(t) = z_i''(t)$, otherwise, keep the original value of $z_i(t)$.

$$if\ U(\{f(p_g(t)), f(p_g(t-1)), ..., f(p_g(t-C_t+1))\}) \&\& f(p_g(t)) > 0 \&\& r_i \le C_g \quad (5)$$

$$z'_{i,j} = r(da_j + db_j) - z_{i,j} \quad (6)$$

Where, $da_j = \min(z_{i,j}), db_j = \max(z_{i,j})$;

$$z'_{i,j} = rand(0,1)\ \ if\ z'_{i,j} < 0\ or\ z'_{i,j} > 1 \quad (7)$$

$$z''_{i,j}(t) = \left(1 \Big/ \sum_{j=1}^{N_R+N_B} z'_{i,j}\right) z'_{i,j} \quad (8)$$

Where, $U(\cdot)$ determines whether all the elements inside are the same, $f(\cdot)$ is the particle fitness, $p_g(t)$ is the global best position of all particle at iteration t, r_1 and r_2 are random number in $(0,1)$. C_g is a constant in $(0,1)$, which controls the frequency of variation. C_t is the threshold of update stagnation algebra, and generally takes C_t smaller value, so that the algorithm can learn more times in a limited number of iterations, which is conducive to finding the optimal solution with A greater probability. da_j and db_j are respectively the minimum and maximum values on the jth dimension of particle swarm.

3.3 Local Variation Search Strategy

Dynamic reverse learning improves the search ability of particle swarms for new solutions and the escape ability of local optimal solutions. However, when the mutated particle moves to the current optimal solution, the feasible solution on its single motion trajectory is probably not the optimal position in its local neighborhood. However, each particle position is only a local optimal solution, and then it may be a global optimal solution. To this end, a small-scale mutation search operation in the local neighborhood is proposed, and local neighborhood optimization is performed on the particle iteration position to enhance the mining ability of particles in local space. The specific description is as follows:

First, k particles are randomly generated in the local neighborhood of each particle z_i. k particles are represented as $z_i^l(t+1), l = 1,2,\cdots,k$. The specific method is as follows: Firstly, k particles $z_i^{l'}(t+1)$ are randomly generated in the feasible domain, and then $z_i^l(t+1)$ is generated according to formula (9). Then, the position with the optimal fitness value in $k+1$ particles (including $z_i(t+1)$) was selected as the position after $t+1$ iterations: $z_i(t+1) = \min[f(\{z_i(t+1), z_i^l(t+1), l = 1,2,\cdots,k\})]$.

$$z_i^l(t+1) = \begin{cases} z_i(t+1) + \dfrac{\alpha}{\beta}\left(z_i^{l'}(t+1) - z_i(t+1)\right) rd_t & \alpha < \beta \\ z_i(t+1) + \left(z_i^{l'}(t+1) - z_i(t+1)\right) rd_t & \alpha \ge \beta \end{cases} \quad (9)$$

Where, $\alpha = \left\| z_i(t+1) - z_i(t) \right\|$, $\beta = \left\| z_i(t+1) - z_i^{l'}(t) \right\|$, and α determines the neighborhood radius of random sample. r is a random number in $[0,1]$. $d_t = 1 - \dfrac{t}{T_{\max}}$, $r \in U(0,1)$, and d_t is the local scaling factor in generation t, which decreases linearly with the increase of evolutionary algebra and gradually reduces the search radius. So that, in the early stage, local mining is carried out with a larger radius to accelerate the convergence speed, and in the later stage, mining is refined with a smaller radius.

3.4 Algorithm Procedure

Each particle in the IMSPSO algorithm is represented by the mixed strategy of all players, that is, $x = (x_1, x_2, \cdots, x_n)$. In this paper, the fitness function for solving Nash equilibrium of n-person finite non-cooperative game is adopted, which is given in literature [19]:

$$f(x) = \sum_{i=1}^{n} \max\{\max_{1 \le k \le m_i} E_i(x \,||\, s_j^i) - E_i(x)), 0\}$$
$$1 \le j \le m_i \quad (10)$$

It is easy to know that if and only if the mixed strategy is the Nash equilibrium solution, the fitness function obtains the minimum value 0, that is, $f(x) = 0$. Therefore, the mixed strategy that makes the fitness 0 is the Nash equilibrium solution of n-person finite non-cooperative game.

In particular, the fitness function for the solution of Nash equilibrium in a bimatrix game is [20]:

$$f(x,y) = \max_{1 \le i \le m}(A_{i.}y^T - xAy^T, 0\} + \max_{1 \le j \le n}(xB_{.j} - xBy^T, 0\} \quad (11)$$

Where, $A_{i.}$ is the ith row vector of A, and $B_{.j}$ is the jth column vector of B.

The overall process of IMSPSO algorithm is described as follows:

1) Determine particle swarm parameters: population size N, maximum iterations $T_{\max}$ and c_1, c_2, C_t, C_g ,etc.

2) Initialization: Randomly generate an initial population of size N, where the position vector of the particle i is $z_i(1) = (z_i^1(1), z_i^2(1), \cdots z_i^n(1))$,and $z_i^k(1) = (z_{i1}^k(1), z_{i2}^k(1), \cdots, z_{im_i}^k(1))$ $(k = 1,2,\cdots,n)$ is a mixed strategy for the player k ,and $z_i^k(1) = (z_{i1}^k(1), z_{i2}^k(1), \cdots, z_{im_i}^k(1))$ meets $\sum_{j=1}^{m_i} z_{ij}^k(1) = 1$. The initial velocity of particle i $(i = 1,2,\cdots,N)$ is $v_i(1) = (v_i^1(1), v_i^2(1), \cdots v_i^n(1))$ $(i = 1,2,\cdots,N)$,where

$v_i^k(1)=(v_{i1}^k(1), v_{i2}^k(1),\cdots,v_{im_i}^k(1))$ meets $\sum_{j=1}^{m_i} v_{ij}^k(1)=0$.

3) According to formula (10), the fitness value of particles is calculated, and the smaller the fitness function is, the better the fitness function will be, so as to obtain the historical optimal solution of particles p_i and the global optimal solution p_g.

4) Determine whether the condition is satisfied in the formula (5) for each particle, if it is established, turn to step 5), otherwise turn to step 6).

5) Dynamic reverse learning for each particle, then turn to step 8).

6) Update each particle's search speed and position according to formula (12), (13).

$$v_i^z(t+1)=w_i(t)v_i^z(t)+c_1r_1(p_i(t)-z_i(t)) +c_2r_2(p_g(t)-z_i(t)) \quad (12)$$

$$z_i(t+1)=z_i(t)+v_i^z(t+1) \quad (13)$$

7) Check whether each dimension of the particle i satisfies $z_{i,j}(t+1)\geq 0$. If not, calculate the control step size α_i according to formula (14) and make $z_i(t+1)=z_i(t)+\alpha_i v_i^z(t+1)\geq 0$.

$$\alpha_i=\min_j\left\{\alpha_i^j\geq 0 \mid \alpha_i^j=-\frac{z_{i,j}(t)}{v_{i,j}^z(t+1)}\right\} \quad (14)$$

8) Local mutation search for each particle.

9) Calculate the fitness value of each particle, update the particle group individual optimal, global optimal position information.

10) Determine whether the maximum number of iterations is reached. If it is satisfied, the iteration is terminated and the optimal solution is output; otherwise, return to step 3).

4 SIMULATION EXAMPLE AND RESULTS ANALYSIS

Next, the Nash equilibrium solutions of two bimatrix game problems are solved to compare the performance of IMSPSO and basic particle swarm optimization.

The following parameters for two algorithms are set equally: population size N=10, maximum iterations $T_{\max}=50$, learning factor $c_1=c_2=1.5$, and computational accuracy $\varepsilon=10^{-4}$. In addition, the other parameters in IMSPSO are set as follows: the evolutionary stagnation algebraic threshold $Ct=7$, mutation coefficient $C_g=0.5$, and k=4. Where, the mutation coefficient $C_g=0.5$, which can ensure that about half of the particles can always participate in the reverse learning when the population falls into the local optimum.

Experiment 1. For three-dimensional bimatrix game $\Gamma(x_1,y_1;A_1,B_1)$, its payment matrix is as follows [21]:

$$A_1=\begin{bmatrix}3&1&6\\0&0&4\\1&2&5\end{bmatrix},\ B_1=\begin{bmatrix}3&0&1\\1&0&2\\6&4&5\end{bmatrix}$$

The only Nash equilibrium solution to this problem is $(1,0,0;1,0,0)$.

Experiment 1 was solved by using IMSPSO and basic particle swarm optimization for 5 times respectively, among which IMPSO iterated for 4 times on average and the fitness value accuracy was up to 10^{-4}, while the basic particle swarm optimization algorithm in literature [19] iterated for 9 times on average and the fitness value accuracy was lower than 10^{-4}. The results are shown in table 1.

Table 1. Calculation results of experiment 1

Calculation times	Iteration times	Optimal fitness value
1	6	2.913e-04
2	5	4.214e-04
3	3	2.365e-04
4	4	5.423e-04
5	2	1.293e-04

Fig. 1 shows the evolution curve of IMSPSO and basic particle swarm optimization algorithm when experiment 1 was solved for the fifth time.

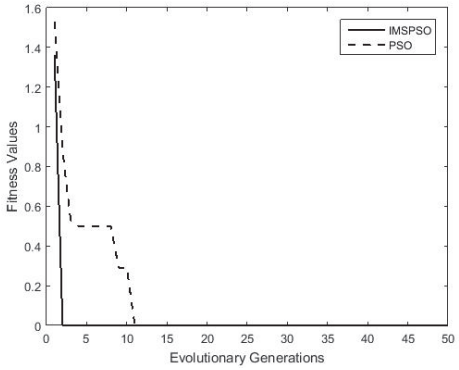

Fig 1. Fitness function value change along with iterations

Experiment 2. For Dekel Scotchmer game $\Gamma(x_2,y_2;A_2,B_2)$, its payment matrix is as follows [19]:

$$A_2=\begin{bmatrix}1&235&0&0.1\\0&1&235&0.1\\235&0&1&0.1\\1.1&1.1&1.1&0\end{bmatrix}$$

$$B_2=\begin{bmatrix}1&0&235&1.1\\235&1&0&1.1\\0&235&1&1.1\\0.1&0.1&0.1&0\end{bmatrix}$$

The only Nash equilibrium solution to this problem is $(1/3,1/3,1/3,0;1/3,1/3,1/3,0)$.

Experiment 2 was solved by using IMSPSO and basic particle swarm optimization for 5 times respectively, among which IMPSO iterated for 7 times on average and the fitness value accuracy was up to 10^{-4}, while the basic particle swarm optimization algorithm in literature [19] iterated for 11.6 times on average and the fitness value accuracy was lower than 10^{-4}. The results are shown in table 2.

Table 2. Calculation results of experiment 2

Calculation times	Iteration times	Optimal fitness value
1	4	3.101e-04
2	10	1.457e-04
3	6	4.294e-04
4	8	5.307e-04
5	10	1.4253e-04

Fig. 2 shows the evolution curve of IMSPSO and basic particle swarm optimization algorithm when experiment 2 was solved for the fifth time.

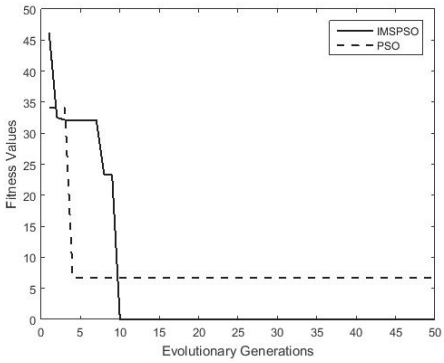

Fig 2. Fitness function value change along with iterations

Simulation results show that IMSPSO can effectively solve Nash equilibrium of non-cooperative games, and its performance is better than basic particle swarm optimization algorithm.

5 CONCLUSION

In this paper, a particle swarm optimization integrating multiply strategies algorithm is proposed to solve the Nash equilibrium of non-cooperative games. Different from the basic particle swarm optimization algorithm, IMSPSO algorithm constructs the adaptive inertia weight by using the ratio of individual fitness to the average fitness of the population, so that the weight change conforms to the evolutionary demand of individual particles. When the algorithm appears premature phenomenon, it will start the reverse learning process to improve the diversity of the population to escape from the local optimal. At the same time, the local neighborhood small-scale mutation search is adopted to improve the ability of PSO's local accurate search. The three mechanisms work together to effectively balance the contradiction between local development and global exploration, so that the convergence speed of the algorithm is significantly improved. The simulation experiment shows that the IMSPSO algorithm is an effective algorithm for solving the Nash equilibrium problem of n-person finite non-cooperative game. Compared with the basic particle swarm optimization algorithm, IMSPSO algorithm can effectively avoid the local convergence problem and has obvious convergence accuracy and speed.

REFERENCES

[1] Zhang Y X, He J S, Zhao B. Security Mechanism in Access Control Based on Non-cooperative Game, Journal on Communications, 2017, 35(Z2): 246-250.

[2] Liu L G, Pan M M, Tian S M, et al. A Non-cooperative Game Analysis of an Competitive Electricity Retail Considering Multiple Subjects of Source-grid-load[J]. Proceedings of the CSEE, 2017, 37(6): 1618-1626.

[3] Kong X Y, Li D H. Region Air-Defense Deployment Decision Based on Particle Swarm Optimization[J]. Computer & Digital Engineering, 2016, 44(12): 2320-2324.

[4] Zaman F, Elsayed S M, Ray T, et al. Evolutionary Algorithms for Finding Nash Equilibria in Electricity Markets[J]. IEEE Transactions on Evolutionary Computation, 2018, 22(4): 536-549.

[5] Nair K G K, Ranjith G. Solution of 3 x 3 Games Using Graphical Method[J]. European Journal of Operational Research, 1999, 112(2):472-478.

[6] Lemke C E, Howson J T. Equilibrium Points of Bimatrix Games[J]. Journal of the Society for Industrial & Applied Mathematics, 1964, 12(2):413-423.

[7] Herings J J, Peeters R. Homotopy Methods to Compute Equilibria in Game Theory[J]. Economic Theory, 2010, 42(1):119-156.

[8] Dreves A, von Heusinger A, Kanzow C, et al. A Globalized Newton Method for the Computation of Normalized Nash Equilibria[J]. Journal of Global Optimization, 2013, 56(2): 327-340.

[9] Yuan Y X . A Trust Region Algorithm for Nash Equilibrium Problems[J]. Pacific Journal of Optimization, 2011, 7(1): 125-138.

[10] Shi S H, Luo M K, Li S L. Applications of Genetic Differentiation Algorithm in Solving Many Person Non- Cooperative Games Nash Equilibrium[J].Journal of Sichuan University(Nature Science Edition),2010,47(3):415-419.

[11] Jia W S, Xiang S W, Yang J F, et al. Solving Nash Equilibrium for N-persons Non-cooperative Game Based on Immune Particle Swarm Algorithm[J]. Application Research of Computers, 2012, 29(1): 28-31.

[12] Yang Y L, Xiang S W, Xia S Y. Solving Nash Equilibrium of Non-cooperative Game Based on Fireworks Algorithm[J]. Computer Applications and Software, 2018, 35(3):215-218.

[13] Wang Z Y, Han X, Xu W S, et al. Nash Equilibrium Solution Based on Improved Ant Colony Algorithm[J]. Computer Engineering, 2010, 36(14):166-171.

[14] Wang Z Y, Han X, Xu W S. Nash Equilibrium Solution Based on Improved Ant Colony Algorithm[J]. Computer Engineering, 2010, 36(14):166-168..

[15] Pavlidis N G, Parsopoulos K E, Vrahatis M N. Computing Nash Equilibria through Computational Intelligence Methods [J]. Journal of Computational & Applied Mathematics, 2005, 175(1):113-136.

[16] Kang L L , Dong W Y , Song W J , et al. Non-inertial Opposition-based Particle Swarm Optimization with Adaptive Elite Mutation[J]. Journal on Communications, 2017, 38(8): 1-13.

[17] Liu Z, Zhou X. Particle Swarm Optimization Algorithm Based on Independent Weight and Classification Mutation Strategy[J]. Journal of Jilin University (Science Edition), 2017, 55(2): 333-339.

[18] Xia X , Liu J , Li Y . Particle Swarm Optimization Algorithm with Reverse-Learning and Local-Learning Behavior[J]. Journal of Software, 2015, 38(7): 1397-1407.

[19] Yu Q, Wang X. Evolutionary Algorithm for Solving Nash Equilibrium Based on Particle Swarm Optimization[J]. Journal of Wuhan University(Natural Science Edition), 2006, 52(1): 25-29.

[20] Qu Y, Zhang J J, Song Y X. Particle Swarm Optimization Algorithm for Solving Multiple Nash Equilibrium Solutions [J]. Operations Research & Management Science, 2010, 19(2):52-55.

[21] Zhai S A, Fan Q Q, Hu Z H. Self-adaptive Particle Swarm Optimization Algorithm Based on Mixed Knowledge and Its Application for Game Theory[J]. Journal of East China University of Science and Technology(Natural Science Edition), 2018, 44(4):595-608.