RESEARCH ARTICLE

An improved predator-prey particle swarm optimization algorithm for Nash equilibrium solution

Yufeng Meng[1☯], **Jianhua He**[1☯*], **Shichu Luo**[1], **Siqi Tao**[2*], **Jiancheng Xu**[1]

1 School of Electronics and Information, Northwestern Polytechnical University, Xi'an, Shaanxi, China, **2** Southwest China Research Institute of Electronic Equipment, Chengdu, Sichuan, China

☯ These authors contributed equally to this work.
* hjh@nwpu.edu.cn (JH); 873692005@qq.com (ST)

Abstract

Focusing on the problem incurred during particle swarm optimization (PSO) that tends to fall into local optimization when solving Nash equilibrium solutions of games, as well as the problem of slow convergence when solving higher order game pay off matrices, this paper proposes an improved Predator-Prey particle swarm optimization (IPP-PSO) algorithm based on a Predator-Prey particle swarm optimization (PP-PSO) algorithm. First, the convergence of the algorithm is advanced by improving the distribution of the initial predator and prey. By improving the inertia weight of both predator and prey, the problem of "precocity" of the algorithm is improved. By improving the formula used to represent particle velocity, the problems of local optimizations and slowed convergence rates are solved. By increasing pathfinder weight, the diversity of the population is increased, and the global search ability of the algorithm is improved. Then, by solving the Nash equilibrium solution of both a zero-sum game and a non-zero-sum game, the convergence speed and global optimal performance of the original PSO, the PP-PSO and the IPP-PSO are compared. Simulation results demonstrated that the improved Predator-Prey algorithm is convergent and effective. The convergence speed of the IPP-PSO is significantly higher than that of the other two algorithms. In the simulation, the PSO does not converge to the global optimal solution, and PP-PSO approximately converges to the global optimal solution after about 40 iterations, while IPP-PSO approximately converges to the global optimal solution after about 20 iterations. Furthermore, the IPP-PSO is superior to the other two algorithms in terms of global optimization and accuracy.

Citation: Meng Y, He J, Luo S, Tao S, Xu J (2021) An improved predator-prey particle swarm optimization algorithm for Nash equilibrium solution. PLoS ONE 16(11): e0260231. https://doi.org/10.1371/journal.pone.0260231

Editor: Seyedali Mirjalili, Torrens University Australia, AUSTRALIA

Received: June 17, 2021

Accepted: November 4, 2021

Published: November 24, 2021

Data Availability Statement: All relevant data are within the manuscript and its Supporting information files.

Funding: The author(s) received no specific funding for this work.

Competing interests: The authors have declared that no competing interests exist.

Introduction

During the strategy selection for a swarm of Unmanned Aerial Vehicles (UAVs), not only the state of UAV itself but also the strategy of the enemy should be considered. Therefore, more scientific and reasonable strategies can be made by introducing game theory strategies into the confrontation decision of UAVs. Thus, game theory was introduced into UAV confrontation decision-making [1–5]. Due to the large number of swarming UAVs and the many available alternative countermeasures, each strategy needs to be evaluated, resulting in the amount of digital storage space needed for the evaluation values exploding exponentially as the number of countermeasures increases. Therefore, problems such as slow algorithm convergence or convergence to the local optimal solution will be encountered when solving the Nash equilibrium solution of such a game.

At present, the main methods for solving the Nash equilibrium solution include the game solution based on an evolutionary algorithm, an ant colony algorithm, application of graph theory and use of particle swarm optimization (PSO).

Evolutionary Strategies [6–9] are search algorithms proposed by Rechenberg and Schwefel from Germany. In each iteration of the algorithm, each group of individuals (parent generation individuals) in the population recombine and mutate (mutation) to produce offspring individuals. Recombination can be achieved with discrete recombination, median recombination and mixed recombination. Mutation is added by adding a random vector subject to a normal distribution, and the randomness thereof is controlled by manipulating the variance with this variance representing the mutation degree. Using an evolutionary algorithm to solve game problems [10] will use binary encoding strategies for chromosomes, and each player constitutes a chromosome. The situation faced by a player is converted into two binary chromosome groups, using different encoding, crossover, mutation and selection strategies, thus generating new situations through the calculation of fitness functions, and selection of favorable or good chromosomes. In this manner, the Nash equilibrium solution of the game is obtained.

An Ant Colony System is an intelligent biological algorithm proposed by Italian scholars Dorigo, Maniezzo et al. [11], through studying the foraging behavior of ants. An improved ant colony algorithm was proposed for solving Nash equilibrium solutions of finite N-player non-cooperative games [12]. In this class of analysis, ants are first randomly distributed in the range of the feasible solution and initial parameters. Thereafter, based on an individual ant's variable pheromone strength and fitness function calculation of transition probability, and global search, an ant deploys a dynamic random search technique for local searches, and then updates their pheromone strength. When the optimal solution meeting the precision requirements of the algorithm reaches the maximum allowed number of iterations, the output is considered the optimal solution. Compared with the evolutionary algorithm, the improved ant colony algorithm has better computational performance and can more quickly converge to a more precise solution.

Graph Theory has its roots in the Konigsberg problem. A solution method of Nash equilibrium based on graph theory was proposed to solve two-person non-cooperative pure strategy games [13]. The problem of solving the Nash equilibrium solution of the game model is transformed into the problem of solving the confluence point of the directed graph. If there is a confluence point in the graph, then that point is a Nash equilibrium solution. If there is no confluence point in the graph, then the game has no Nash equilibrium solution. Compared with other methods, the graph theory method finds Nash equilibrium by finding the confluence point, which is more intuitive and simple, and has lower time complexity that other options while improving the speed of solving Nash equilibrium solution of two-person non-cooperative game.

Finally, a PSO method [14, 15] can be used to solve the Nash equilibrium of the attack and defense confrontation game of clustered UAVs.

Aiming at the dynamic multi-strategy of UAV cooperative attack, Wang Y et al. [1] combined the dynamic target assignment problem with the Nash equilibrium concept of game theory, and used the improved PSO algorithm based on an elite reselection mechanism to obtain the optimal strategy combination of both sides in the sense of Nash equilibrium.

A game theory method based on Predator-Prey Particle Swarms Optimization (PP-PSO) was proposed [2], which de-composed the dynamic task assignment problem of multiple UAVs in military operations and modeled it as a two-person game at each decision-making stage. In each decision stage, both parties seek the best solution for the purpose of maximizing their own objective function, and use the PP-PSO algorithm to solve the mixed Nash equilibrium as the optimal allocation scheme at each stage.

The original PSO algorithm easily falls into local optimum. Inspired by the swarm behavior of sardines and herring, this algorithm was optimized according to the predation behavior, and this improved the subject algorithm's shortcoming regarding its prevalence of local optimization to a certain extent [16–18]. However, due to the use of two populations, the convergence speed of the algorithm is slow when approaching the global optimal value, and the whole population cannot entirely converge to the global optimal value (that is, the average fitness value of the population cannot converge). This paper mainly studies a generally improved Predator-Prey Particle Swarm Optimization(IPP-PSO) algorithm to solve the Nash equilibrium of the game. This algorithm can also be used to solve the Nash equilibrium of the game problem of the attack and defense of UAV swarms.

The main contributions of this paper are summarized as follows:

- Improved initial predator and prey distribution. During initialization, some particles with better fitness values are deemed as prey, and the remaining particles with poorer fitness values are set as predators. The predators can improve the convergence of the algorithm by simultaneously chasing the target and prey;
- Improved inertia weights in predators and prey. In the inertial weight value, the average fitness value of the population of particles is considered, so that the nearer the particle is to the global optimal, the smaller the inertial weight value is, so as to improve the "precocity" problem of the algorithm;
- Improved particle velocity calculation formula. When the fitness value of a particle is less than a certain threshold value, the particle no longer considers its own population optimal, but approaches the global optimal, so as to solve the problems of global optimal and algorithm convergence speed;
- Add pathfinder to increase the diversity of the population. Pathfinder can be a particle randomly distributed in the solution space, or it can be a particle evolved from the current global optimal solution. Pathfinder can improve the global search ability of the algorithm, the "precocity" of the better algorithm and the problem of the global optimal solution.

This paper is organized as follows: Section I introduces the reason why we establish the IPP-PSO. Section II introduces the related word of the PSO. Section III introduces the PSO and PP-PSO. In section IV the IPP-PSO algorithm is established. Section V shows the simulation results and analysis. Finally, conclusions are set forth in Section VI.

Related work

The implementation process of the PSO algorithm has a great relationship with the value of its parameters. How to select these parameters is an urgent problem to be solved. When a PSO algorithm is applied to the optimization of complex high-dimensional problems, premature convergence and other problems are often encountered [19]. In the past two decades, a large number of scholars have conducted in-depth research on the improvement of the PSO algorithm.

The theoretical research of the PSO algorithm includes improved strategy research of PSO algorithm. Among them, the theoretical study of PSO algorithms focuses on the dynamic characteristics of such algorithms and the topological structure of PSO algorithm. Furthermore, the research studies the control parameters of PSO algorithms. The improved strategy deployed therein mainly focused on adaptive PSO control parameters, improved learning strategies, and mixed PSO strategies [20].

The problems of the multi-objective PSO algorithm are as follows:

1). How to select "leader" particles in the optimization process to lead the entire population to quickly approach the Pareto front under the premise of retaining some individual information, that is, the optimal particle selection strategy;

2). In PSO, the population individuals are affected by the "optimal" particles, and the rapid convergence leads to "precocity". This then presents the question of how to guide the particles to "jump out" of the local optimal, that is, the diversity preservation mechanism;

3). As the number of non-dominated solutions in the external storage concentration increases rapidly, then how to guide the population to further improve the search efficiency under the premise of ensuring the diversity of the population, so as to strengthen the advantages of the algorithm in the convergence rate, that is, the means to improve the convergence;

4). How to dynamically coordinate the relationship between the whole development and local search in different stages of the optimization process to obtain the best optimization results, that is, the balance method of diversity and convergence.

In order to improve the performance of the algorithm, iterative formula improvement, dynamic tuning of important parameters and adjustment of information interaction between particles are carried out, that is, the iterative formula, parameters and topology improvement scheme [21]. The algorithm improvement measures for these problems include: the selection of optimal particles, the preservation of diversity, the improvement of convergence, the balance of diversity and convergence, and the improvement of iterative formula, parameters and topology structure.

As a group optimization algorithm, in order to improve the search ability and avoid falling into local optimal, the PSO should maintain diversity in the early stage of the algorithm. In the later stages of the algorithm, it is particularly important to improve its convergence ability [22]. Common diversity control methods are as follows: diversity control based on particle spacing, diversity control based on exclusion and disturbance, diversity control based on multiple subgroups, and population diversity detection.

The improvement measures of the PSO algorithm are as follows [23]: 1) Increase the inertia weight and convergence factor to improve the convergence speed; 2) PSO with a selection mechanism will inhibit the control effect of a few super particles and improves the success rate of convergence; 3) PSO with a mutation operator can adjust the search speed of particles and improve the convergence speed and accuracy; 4) Based on the improvement of neighborhood operators and topology structure, different neighborhood topologies are used to study the performance of guaranteed convergence PSO; 5) A new particle structure or group structure is constructed to effectively improve the global convergence ability of the algorithm; 6) Improve or use the new position/speed update formula and improve the global and local search ability, thereby avoiding the algorithm falling into the local optimal; 7) The organic combination of other evolutionary optimization technology PSO algorithm.

Two traditional particle swarm optimization algorithms

Particle swarm optimization

PSO is an algorithm that simulates the foraging behavior of birds. If a flock of birds in a random search of their area for food, and all the birds do not know the exact location of food, but they know that others in their location will be searching together for the flock to find the nearest food, then during the process of search and to avoid any adjacent bird collision, they match both the surrounding birds' speed, and their own vector to the target.

The foraging behavior of birds is the origin of the idea of the PSO algorithm. A particle is used to represent a bird, then the current position of each particle is a feasible solution of the problem to be solved. The particle searches in the n-dimensional space, and the search process of the particle is factored into the flight process. The velocity is adjusted according to the historical optimal solution of the particle itself and the optimal solution of the population. The adjustment method is shown in Fig 1:

In Fig 1, $\vec{x}(t)$ is the position of the particle at time t, $\vec{x}(t+1)$ is the position of the particle at the next unit of time (time t+1), $\vec{v}(t)$ is the velocity of the particle at time t, $\vec{v}(t+1)$ is the velocity of the particle at time t, $\vec{p}(t)$ is the optimal solution of the particle before time t, and $\vec{g}(t)$ is the historical optimal solution of the whole particle swarm. Therefore, the formula for particle I to update its own velocity and position is shown in (1) and (2):

$$v_i(t+1) = w \times v_i(t) + c_1 r_1 (p_i(t) - x_i(t)) + c_2 r_2 (g(t) - x_i(t)) \tag{1}$$

$$x_i(t+1) = x_i(t) + v_i(t) \tag{2}$$

In (1), w is the inertia factor; c_1,c_2 are learning factors, also known as the acceleration constants; and r_1,r_2 are each a uniform random number within the range [0, 1].

It is assumed that the feasible solution space is D- dimensional, and the particle swarm consists of N particles. The relevant symbols are as follows:

Position of the i^{th} particle:

$$X_i = (x_{i1}, x_{i2}, \ldots, x_{iD}), i = 1, 2, \ldots, N \tag{3}$$

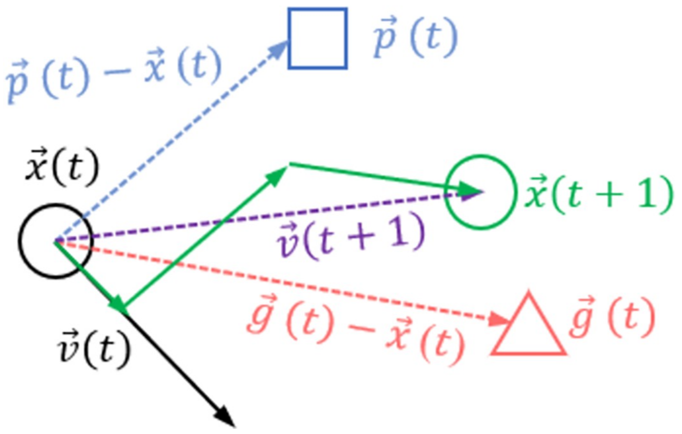

Fig 1. Particle flight velocity diagram.

https://doi.org/10.1371/journal.pone.0260231.g001

The flight speed of the i^{th} particle:

$$V_i = (v_{i1}, v_{i2}, \ldots, v_{iD}), i = 1, 2, \ldots, N \tag{4}$$

The historical optimal solution of the i^{th} particle is the individual extreme value:

$$P^i_{best} = (p_{i1}, p_{i2}, \ldots, p_{iD}), i = 1, 2, \ldots, N \tag{5}$$

The historical optimal solution of the entire particle swarm is the global extremum:

$$g_{best} = (p_{g1}, p_{g2}, \ldots, p_{gD}) \tag{6}$$

The degree of the solution represented by the particle is judged according to the fitness function value.

The steps of the PSO algorithm are as follows:

Step 1: Initialization of the population, including size N of the particle swarm, position X_i and velocity V_i of each particle;

Step 2: Set the maximum number of iterations *CycleMax*, and make the current number of iterations $t = 1$;

Step 3: Calculate the fitness value $F_i(t)$ of each particle;

Step 4: Compare the fitness value $F_i(t)$ of each particle with the fitness value F^i_{pbest} of pbest P^i_{best}. If $F_i(t) < F^i_{pbest}$, replace P^i_{best} with $X_i(t)$ and F^i_{pbest} with $F_i(t)$.

Step 5: Compare the fitness value $F_i(t)$ of each particle with the fitness value F^i_{gbest} of g^i_{best}. If $F_i(t)<F_{gbest}$, replace g_{best} with $X_i(t)$ and F_{gbest} with $F_i(t)$.

Step 6: Update the velocity vector of each particle according to (1);

Step 7: Update the position vector of each particle according to (1);

Step 8: Judge whether the maximum number of iterations *CycleMax* or the accuracy requirements are reached. If yes, the simulation ends. Otherwise, return to Step 3.

The flow chart of particle swarm optimization algorithm is shown in Fig 2.

Predator-prey particle swarm optimization

Because the original PSO algorithm easily falls into the local optimum, the PSO algorithm was optimized according to the predation behavior [16–18]. This optimization idea was was inspired by the swarm behavior of sardines and herring. Particles are divided into two categories, predator and prey. Predators (particles that prey on) pursue their prey and move toward the center of the prey group; The prey (the escaping particle) escapes from the predator within the range of feasible solutions, and the prey adopt different escape behaviors by weighing the predator risk and energy. The velocity and position of predator and prey in the particle swarm

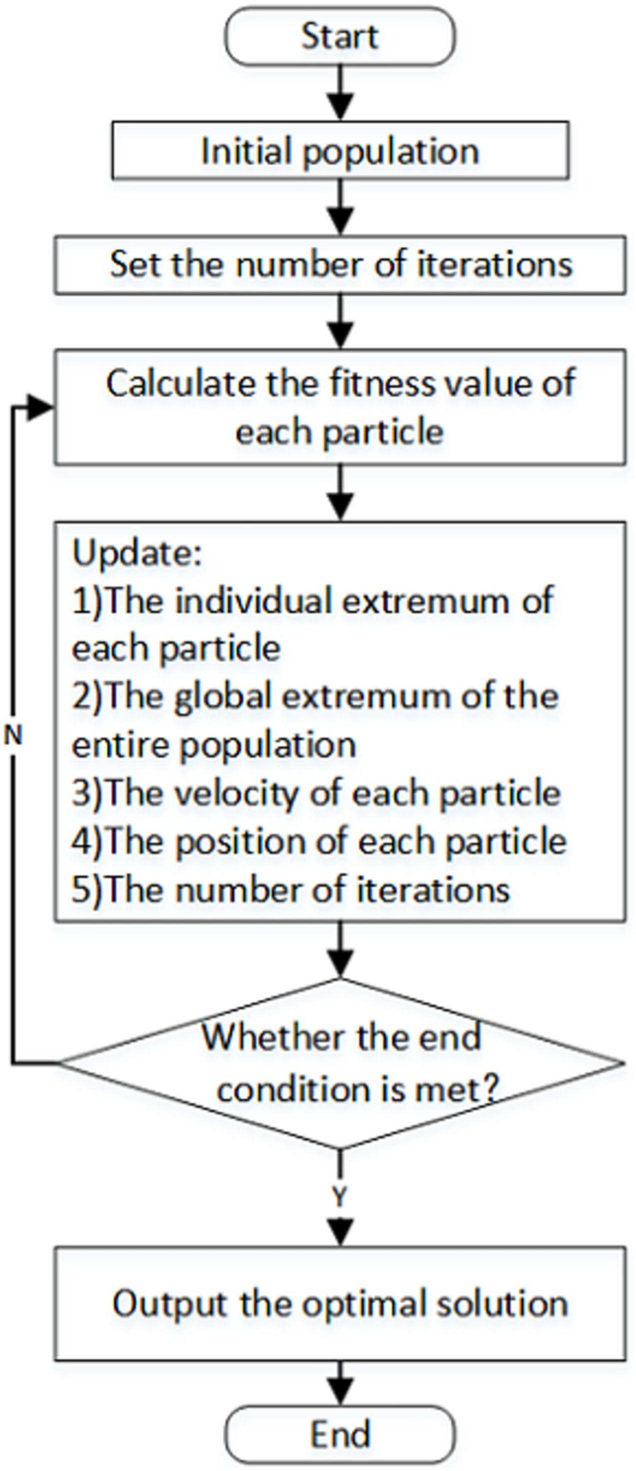
Start
Initial population
Set the number of iterations
Calculate the fitness value of each particle
Update:
1)The individual extremum of each particle
2)The global extremum of the entire population
3)The velocity of each particle
4)The position of each particle
5)The number of iterations
Whether the end condition is met?
N
Y
Output the optimal solution
End

Fig 2. Flow chart of the particle swarm optimization.

https://doi.org/10.1371/journal.pone.0260231.g002

optimization algorithm are defined as (7) and (8):

$$\begin{aligned} v_{di}(t+1) = \quad & w_d v_{di}(t) + c_1 r_1 (p_{di}(t) - x_{di}(t)) + \\ & c_2 r_2 (g_d(t) - x_{di}(t)) + c_3 r_3 (g(t) - x_{di}(t)) \end{aligned} \tag{7}$$

$$\begin{aligned} v_{ri}(t+1) = \quad & w_r v_{ri}(t) + c_4 r_4 (p_{ri}(t) - x_{ri}(t)) + c_5 r_5 (g_r(t) - x_{ri}(t)) + c_6 r_6 (g(t) - x_{ri}(t)) \\ & -P \cdot a \cdot sign[x_{dI}(t) - x_{ri}(t)] \cdot \exp(-b|x_{dI}(t) - x_{ri}(t)|) \end{aligned} \tag{8}$$

$$x_{di}(t+1) = x_{di} + v_{di}(t+1) \tag{9}$$

$$x_{ri}(t+1) = x_{ri} + v_{ri}(t+1) \tag{10}$$

The d and r in the subscripts of (6) and (7) denote predator and prey, respectively.

p_{di} is the historical best solution for the i^{th} predator (individual extremum), p_{ri} is the historical best location for the i^{th} prey, g_d is the historical best solution of the predator population (population extremum), g_r is the historical best solution of prey population (population extremum), g is the best position (global extremum) found in the whole particle swarm so far. The definitions of w_d and w_r are shown in Eqs (11) and (12).

$$w_d = 0.2 \exp\left(-10 \frac{iteration}{iteration_{max}}\right) + 0.4 \tag{11}$$

$$w_r = w_{max} - \frac{w_{max} - w_{min}}{iteration_{max}} iteration \tag{12}$$

w_d and w_r are the inertia weights of predator and prey, respectively. The value that inertia weights plays an important role in the global and local search ability and algorithm convergence of the algorithm. $iteration_{max}$ represents the maximum number of iterations. w_{max} and w_{min} represent the maximum and minimum values of w_r, respectively. The definition of I in (8) is shown in (13)

$$I = \{k | \min_k(|x_{dk} - x_{ri}|)\} \tag{13}$$

I is the number of predators around the i^{th} prey. In (8), P indicates whether the prey is able to escape ($P = 0$(yes) or $P = 1$ (no)), and a and b are the parameters that determine how difficult it is for the prey to escape the predator. The closer the predator is to the prey, the harder it is for the prey to escape the predator. $sign$ is a symbolic function, defined as Formula (14):

$$sign(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} \tag{14}$$

Then the flow chart of Predator-Prey Particle Swarms Algorithm is shown in Fig 3.

The algorithm steps are as follows:

Step 1: Initialization of the population. Set particle swarm size N, predator group size D, prey group size R to satisfy $N = D + R$, the initial position and initial velocity of each particle;

Step 2: Set the maximum number of iterations to $iteration_{max}$, and set the current number of iterations to $iteration = 1$;

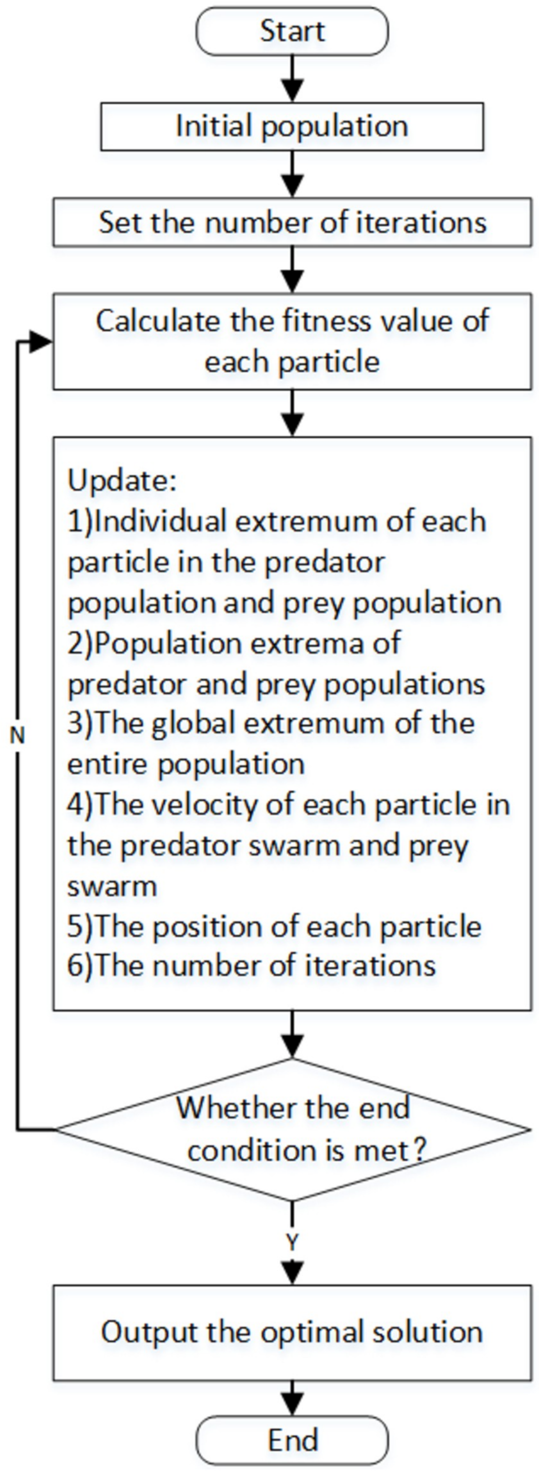
Start
Initial population
Set the number of iterations
Calculate the fitness value of each particle
Update:
1)Individual extremum of each particle in the predator population and prey population
2)Population extrema of predator and prey populations
3)The global extremum of the entire population
4)The velocity of each particle in the predator swarm and prey swarm
5)The position of each particle
6)The number of iterations
Whether the end condition is met?
N
Y
Output the optimal solution
End

Fig 3. Flow chart of predator-prey particle swarm optimization.

https://doi.org/10.1371/journal.pone.0260231.g003

Step 3: Calculate the fitness values $F_{di}(t)$ and $F_{ri}(t)$ of each particle in predator group and prey group respectively;

Step 4: Compare the fitness value $F_{di}(t)$ of each particle in the predator swarm with the fitness value F_{pbest}^{di} of the individual extreme value P_{di}^{best}. If $F_{di}(t) < F_{pbest}^{di}$, P_{di}^{best} will be replaced by $X_{di}(t)$, and F_{pbest}^{di} will be replaced by $F_{di}(t)$.

The fitness value $F_{ri}(t)$ of each particle in the prey group was compared with the fitness value F_{pbest}^{ri} of the individual extreme value P_{ri}^{best}. If $F_{ri}(t) < F_{pbest}^{ri}$, P_{ri}^{best} was replaced with $X_{ri}(t)$, and F_{pbest}^{ri} was replaced with $F_{ri}(t)$.

Step 5: Compare the fitness value $F_{di}(t)$ of each particle in the predator population with the fitness value F_{pbest}^{d} of the extreme population g_d of the predator population. If $F_{di}(t) < F_{pbest}^{d}$, then replace g_d with $X_{di}(t)$ and F_{pbest}^{d} with $F_{di}(t)$.

The fitness value $F_{ri}(t)$ of each particle in the prey group was compared with the fitness value F_{pbest}^{r} of the population extreme value g_r of the prey group. If $F_{ri}(t) < F_{pbest}^{r}$, g_r was replaced by $X_{ri}(t)$, and F_{pbest}^{r} was replaced by $F_{ri}(t)$.

Step 6: Compare the fitness value $F_i(t)$ of each particle in the population with the fitness value F_{gbest} of the global extreme value g_{best}. If $F_i(t) < F_{gbest}$, replace g_{best} with $X_i(t)$ and F_{gbest} with $F_i(t)$.

Step 7: Update the velocity vector of each particle in the predator swarm according to (7);

Step 8: Update the velocity vector of each particle in the prey group according to (8);

Step 9: Update the position vectors of each particle in the predator group and prey group according to (9) and (10) respectively;

Step 10: Determine whether the maximum number of iterations is $iteration_{max}$ or the accuracy requirements are met. If yes, output the optimal solution and end; otherwise, return to Step 3.

Improved predator-prey particle swarm optimization

When two populations are used, the convergence speed of the algorithm is slow as it approaches the global optimal value, and the whole population cannot all converge to the global optimal value (that is, the average fitness value of the population cannot converge). According to the above shortcomings, the PP-PSO algorithm is improved as follows:

1) Improved initial predator and prey distribution. For the randomly distributed particles in the solution space, because predators can hunt prey close to the target, during initialization one calculates the population fitness function value for all the particles. The fitness value is set higher for prey particles, with the remainder of the poor fitness values set as predators, and predators (by chasing targets at the same time) and prey improve the convergence of the algorithm.

2) Improved inertia weights in predators and prey. In the PP-PSO algorithm, the inertia weight is related to the number of iterations. With an increase in the number of iterations, the inertia weight gradually decreases. However, the increase of the number of iterations does not mean that the algorithm has converged to the global optimal. Therefore, the average fitness value of the population particle is considered in the inertia weight value, that is, the closer the

particle is to the global optimal, the smaller the inertia weight is, thus improving the problem of "precocity" of the algorithm. The improved definitions of w_d and w_r are shown in (15) and (16):

$$w_d = \left(0.2\exp\left(-10\frac{iteration}{iteration_{max}}\right) + 0.4\right)\cdot\left(1 - \exp\left(\frac{-1}{D}\sum_{i=1}^{D}F_{di}(t)\right)\right) \tag{15}$$

$$w_r = \left(w_{max} - \frac{w_{max} - w_{min}}{iteration_{max}}iteration\right)\cdot\left(1 - \exp\left(\frac{-1}{R}\sum_{i=1}^{R}F_{ri}(t)\right)\right) \tag{16}$$

3) Improved particle velocity calculation formula. In the PP-PSO algorithm, the velocity formulas of predator and prey particles do not change in the iterative process. As the predator population is affected by the optimal solution of prey population and its own population, the whole population cannot converge to the global optimal value. According to this shortcoming, the velocity calculation formula of the particle is improved. When the fitness value of the particle is less than a certain threshold, the particle no longer considers the population optimization and approaches the global optimization. The threshold can be determined by Monte Carlo simulation thusly: Within the range of the fitness function, p as the step length, generate q threshold points (q = the range divided by p, round operation); The threshold points are obtained with equal probability, and simulation is performed every time a threshold point is obtained; The simulation runs n times; The number of iterations of global optimization calculated for the same threshold point is statistically processed, and the threshold point with the best performance is selected as the final threshold point of the entire algorithm. Then, the velocity definitions of predator and prey in the improved predator-prey particle swarm optimization algorithm are shown in Eqs (17) and (18):

$$v_{di}(t+1) = \begin{cases} w_d v_{di}(t) + c_1 r_1(p_{di}(t) - x_{di}(t)) + c_2 r_2(g_d(t) - x_{di}(t)) + & \\ c_3 r_3(g_r(t) - x_{di}(t)) + c_4 r_4(g(t) - x_{di}(t)) & F_{di}(t) > threshold \\ w_d v_{di}(t) + c_1 r_1(p_{di}(t) - x_{di}(t)) + c_3 r_3(g(t) - x_{di}(t)) & F_{di}(t) \le threshold \end{cases} \tag{17}$$

$$v_{ri}(t+1) = \begin{cases} w_r v_{ri}(t) + c_5 r_5(p_{ri}(t) - x_{ri}(t)) + c_6 r_7(g_r(t) - x_{ri}(t)) + & \\ c_7 r_7(g_r(t) - x_{ri}(t)) & F_{ri}(t) > threshold \\ w_r v_{ri}(t) + c_5 r_5(p_{ri}(t) - x_{ri}(t)) + c_7 r_7(g(t) - x_{ri}(t)) & F_{ri}(t) \le threshold \end{cases} \tag{18}$$

4) Add pathfinder to increase the diversity of the population. Through improving the distribution of the initial predator and prey inertia weight values while improving the particle velocity formula we can improve the convergence speed of algorithm in theory, but it may make the algorithm demonstrate a "premature" phenomenon. Therefore, introducing the pathfinder to a random distribution in the solution space of particles can also be used for the current global optimal solutions evolved particles. In this manner, the global searching ability of the algorithm is improved by means of pathfinder. According to the algorithm principle, the optimization degree of the individual extremum $\delta_p(t)$ and the global extremum $\delta_g(t)$ in the t iteration

can be obtained, as shown in (19) and (20):

$$\delta_p(t) = \sum_{i=1}^{D} F_{di}(t) + \sum_{i=1}^{R} F_{ri}(t) - \left(\sum_{i=1}^{D} F_{di}(t-1) + \sum_{i=1}^{R} F_{ri}(t-1)\right) \tag{19}$$

$$\delta_g(t) = F_{gbest}(t) - F_{gbest}(t-1) \tag{20}$$

If the individual optimization threshold $OptimalValue$ is set, when

$$\delta_p(t) < OptimalValue \times N \tag{21}$$

and

$$\delta_g(t) < OptimalValue \tag{22}$$

This then indicates that the optimization degree of the algorithm in the t iteration is low. Therefore, the pathfinder is added, and the pathfinder can evolve a new solution in the feasible solution space according to the current global extreme value, or generate a new solution randomly in the feasible solution space. Pathfinder evolves according to the evolutionary formula (23) of the evolutionary strategy:

$$X_{ti}^{k}(t) = g^{k}(t) + N(0, \sigma) \tag{23}$$

In (23), $X_{ti}^{k}(t)$ is the k^{th} component of the i^{th} pathfinder particle; $g^{k}(t)$ is the k^{th} component of the global extremum; and $N(0, \sigma)$ is a random number following a normal distribution, with a mean of zero and a standard deviation of σ.

According to the above improvements, the flow chart of the improved predator-prey particle swarm optimization is shown in Fig 4. The steps of the algorithm are as follows:

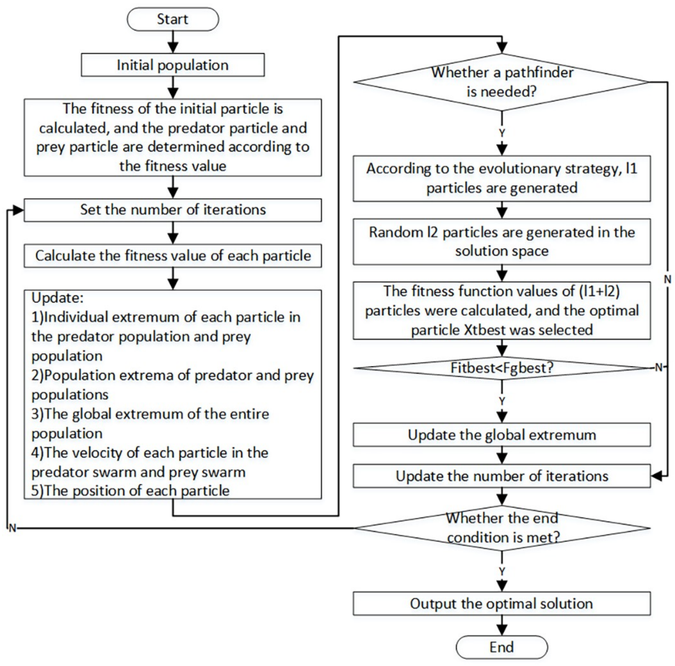
Start
Initial population
The fitness of the initial particle is calculated, and the predator particle and prey particle are determined according to the fitness value
Set the number of iterations
Calculate the fitness value of each particle
Update:
1)Individual extremum of each particle in the predator population and prey population
2)Population extrema of predator and prey populations
3)The global extremum of the entire population
4)The velocity of each particle in the predator swarm and prey swarm
5)The position of each particle
Whether a pathfinder is needed?
Y
N
According to the evolutionary strategy, l1 particles are generated
Random l2 particles are generated in the solution space
The fitness function values of (l1+l2) particles were calculated, and the optimal particle Xtbest was selected
Fitbest<Fgbest?
Y
N
Update the global extremum
Update the number of iterations
Whether the end condition is met?
N
Y
Output the optimal solution
End

Fig 4. Improved predator-prey particle swarm optimization flow chart.

https://doi.org/10.1371/journal.pone.0260231.g004

Step 1: Initialization of the population. Set particle swarm size N, predator group size D, prey group size R to satisfy $N = D + R$, giving the initial position and initial velocity of each particle.

Step 2: Calculate the fitness value $F_i(0)$ of each initial particle, and sort the fitness value from small to large. The particle corresponding to the fitness value of the first ranked R is prey, and the remaining particles are predators.

Step 3: Set the maximum number of iterations $iteration_{max}$, and set the current number of iterations to $iteration = 1$;

Step 4: Calculate the fitness values $F_{di}(t)$ and $F_{ri}(t)$ of each particle in predator group and prey group respectively;

Step 5: Compare the fitness value $F_{di}(t)$ of each particle in the predator group with the fitness value F^{di}_{pbest} of individual extreme value P^{best}_{di}, if $F_{di}(t) < F^{di}_{pbest}$, replace P^{best}_{di} with $X_{di}(t)$, and F^{di}_{pbest} with $F_{di}(t)$;

By comparing the fitness value $F_{ri}(t)$ of each particle in the prey group with the fitness value F^{ri}_{pbest} of individual extreme value P^{best}_{ri}, if $F_{ri}(t) < F^{ri}_{pbest}$, replace P^{best}_{ri} with $X_{ri}(t)$, and F^{ri}_{pbest} with $F_{ri}(t)$;

Step 6: Compare the fitness value $F_{di}(t)$ of each particle in the predator population with the fitness value F^{d}_{pbest} of the population extreme value g_d of the predator population, if $F_{di}(t) < F^{d}_{pbest}$, replace g_d with $X_{di}(t)$, and F^{d}_{pbest} with $F_{di}(t)$;

The fitness value $F_{ri}(t)$ of each particle in the prey group is compared with the fitness value F^{r}_{pbest} of the population extremum g_r of the prey group, if $F_{ri}(t) < F^{r}_{pbest}$, replace g_r with $X_{ri}(t)$, and F^{r}_{pbest} with $F_{ri}(t)$;

Step 7: Compare the fitness value $F_i(t)$ of each particle in the population with the fitness value F_{gbest} of the global extreme value g_{best}, if $F_i(t) < F_{pbest}$, replace g_{best} with $X_i(t)$, and F_{pbest} with $F_i(t)$;

Step 8: Update the velocity vector of each particle in the predator swarm according to (17);

Step 9: Update the velocity vector of each particle in the prey group according to (18);

Step 10: Update the position vectors of each particle in the predator group and prey group according to (9) and (10) respectively;

Step 11: Determine if Pathfinder added is needed. If "No", go to Step 12. If "yes", l_1 pathfinders are generated according to Formula (17), l_2 pathfinders are randomly generated in the solution space, and fitness function values of $l_1 + l_2$ pathfinders are calculated, among which the optimal individual X_{tbest} is selected and compared with the global extremum. If it is better than the global extremum, g_{best} is replaced by X_{tbest}, and F_{gbest} is replaced by F_{tbest}.

Step 12: Judge whether the maximum number of iterations $iteration_{max}$ or the accuracy requirements are met. If "yes", the optimal solution will be the output and the end will be concluded. If "No", go back to Step 3.

Simulation and analysis

The Nash equilibrium solution of the two-player non-cooperative game must be solved, and the payment matrix of both players needs to be calculated by the objective function [24–26].

The payoff matrix for player 1 is $MA_{m\times n}$. The payoff matrix for Player 2 is $MB_{m\times n}$. m is the number of pure strategies for Player 1, and n is the number of pure strategies for player 2. Set the mixed strategy vector of both players as:

$$X_i = (x_{i1}, \cdots, x_{im}, y_{i1}, \cdots, y_{in}) \tag{24}$$

In (24),

$$x_{ik} \geq 0 \tag{25}$$

$$y_{ik} \geq 0 \tag{26}$$

$$\sum_{k=1}^{m} x_{ik} = 1 \tag{27}$$

$$\sum_{k=1}^{n} y_{ik} = 1 \tag{28}$$

Therefore, for any mixed strategy X_i, the Nash equilibrium solution (X^*, Y^*) must satisfy the formula:

$$\begin{cases} MA(Y^*)' \geq MA(X_i(m+1:m+n))' \\ X^*MB \geq X_i(1:m)MB \end{cases} \tag{29}$$

Each particle is represented by a mixed strategy of all players in IPP-PSO algorithm. To solve the Nash equilibrium, the fitness function is defined as shown in formula (30) and formula (31):

$$\begin{aligned} f[X^t_{di}] = \quad & max\{\max_{1\leq j\leq m}\{MA(j,:) \times (X^t_{di,m+1:m+n})' - X^t_{di,1:m} \times MA \times (X^t_{di,m+1:m+n})'\}, 0\} \\ & + max\{\max_{1\leq j\leq n}\{X^t_{di,1:m} \times MB(:,j)' - X^t_{di,1:m} \times MB \times (X^t_{di,m+1:m+n})'\}, 0\} \end{aligned} \tag{30}$$

$$\begin{aligned} f[X^t_{ri}] = \quad & max\{\max_{1\leq j\leq m}\{MA(j,:) \times (X^t_{ri,m+1:m+n})' - X^t_{ri,1:m} \times MA \times (X^t_{ri,m+1:m+n})'\}, 0\} \\ & + max\{\max_{1\leq j\leq n}\{X^t_{ri,1:m} \times MB(:,j)' - X^t_{ri,1:m} \times MB \times (X^t_{ri,m+1:m+n})'\}, 0\} \end{aligned} \tag{31}$$

The above fitness function derivation can be found in Jia W et al [27] and Duan H et al [2]. $X^t_{di,1:m}$ and $X^t_{di,m+1:m+n}$ represent the mixed strategy of player 1 and player 2 in predator i respectively in (30).$X^t_{ri,1:m}$ and $X^t_{ri,m+1:m+n}$ represent the mixed strategy of player 1 and player 2 in prey i respectively in (31). The constraint conditions of the mixed strategy are shown in formula (32) and (33):

$$x_{di,j}(t) \geq 0, x_{ri,j}(t) \geq 0 \tag{32}$$

$$\begin{cases} \sum_{j=1}^{m} x_{di,j} = 1, \sum_{j=m+1}^{m+n} x_{di,j} = 1 \\ \sum_{j=1}^{m} x_{ri,j} = 1, \sum_{j=m+1}^{m+n} x_{ri,j} = 1 \end{cases} \tag{33}$$

Table 1. Two-person zero-sum game payment function.

	palyer2 strategy1	palyer2 strategy2	palyer2 strategy3
palyer1 strategy1	(8,-8)	(9,-9)	(3,-3)
palyer1 strategy2	(2,-2)	(5,-5)	(6,-6)
palyer1 strategy3	(4,-4)	(1,-1)	(7,-7)

https://doi.org/10.1371/journal.pone.0260231.t001

In order to verify the effectiveness of the proposed algorithm, the performance results of PSO, PP-PSO and IPP-PSO are compared by solving the Nash equilibrium solution of a zero-sum game and a non-zero-sum game.

The relevant initial parameters of the algorithm are set: the maximum number of iterations $iteration_{max} = 100$, particle swarm size $N = 30$, inertia weight $w_{max} = 0.9$, $w_{min} = 0.2$. In the PP-PSO and IPP-PSO algorithms, the population sizes of predators and prey are respectively $D = 20$ and $R = 10$, while in IPP-PSO algorithm, pathfinder $l_1 = 15$ and $l_2 = 15$.

The payment function of the two-person zero-sum game and non-zero-sum game [28] are shown in Tables 1 and 2 respectively:

In Tables 1 and 2, the first column and the first row represent the policies for Player 1 and Player 2, respectively. For example, in a zero-sum game, each player has three strategies that represent the number of rows and columns in the payoff matrix, respectively. Payments (or benefits) are set forth inside the table with the first number is the payout for the column player (Player 1), and the second is the payout for the row player (Player 2). To reduce statistical errors, each algorithm runs 100 independent tests on the two games. Figs 5 to 7 respectively present the simulation result curves of the three algorithms in terms of the global extreme value, the average fitness function value and the average error. Tables 3 and 4 show the average fitness function value, optimal fitness function value, mean minimum error, and the number of times of average error less than 0.0001 (the number of successes) of the three algorithms in 100 tests.

The definition of average error is shown in (34) and (35):

$$e_{pp}(t) = \frac{\sum_{i=1}^{D} ||X_{di}(t) - E_S|| + \sum_{i=1}^{R} ||X_{ri}(t) - E_S||}{D + R} \tag{34}$$

$$e_{p}(t) = \frac{\sum_{i=1}^{N} ||X_{i}(t) - E_S||}{N} \tag{35}$$

In (34), $e_{pp}(t)$ represents the average error of PP-PSO and IPP-PSO; in (35), $e_p(t)$ represents the average error of the original particle swarm optimization algorithm. E_S represents the

Table 2. Two-person zero-sum game payment function.

	palyer2 strategy1	palyer2 strategy2	palyer2 strategy3	palyer2 strategy4
palyer1 strategy1	(1,1)	(235,0)	(0,235)	(0.1,1.1)
palyer1 strategy2	(0,235)	(1,1)	(235,0)	(0.1,1.1)
palyer1 strategy3	(235,0)	(0,235)	(1,1)	(0.1,1.1)
palyer1 strategy4	(1.1,0.1)	(1.1,0.1)	(1.1,0.1)	(0,0)

https://doi.org/10.1371/journal.pone.0260231.t002

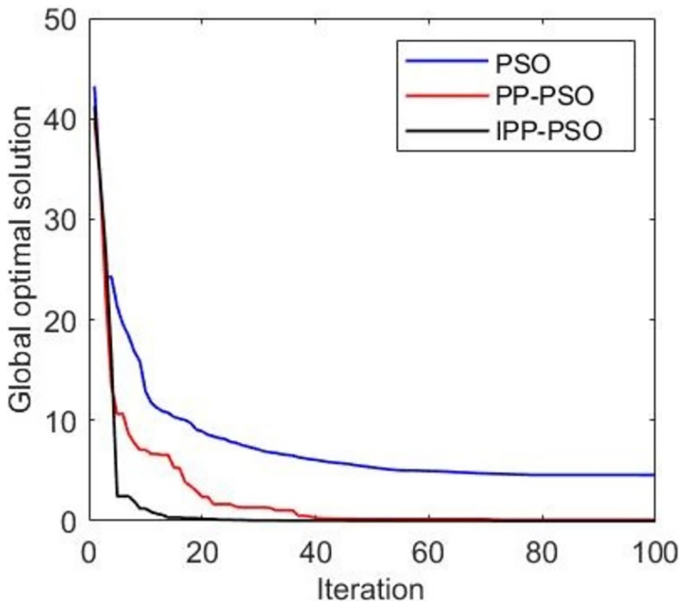

Fig 5. Figure of global optimal solution simulation results of two-person zero-sum game.

https://doi.org/10.1371/journal.pone.0260231.g005

mixed strategy Nash equilibrium solution. The E_S in Table 1 is:

$$E_S = \left[\frac{21}{52}, \frac{12}{52}, \frac{19}{52}, \frac{2}{13}, \frac{3}{13}, \frac{8}{13}\right] \tag{36}$$

The E_S in Table 2 is:

$$E_S = \left[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, 0\right] \tag{37}$$

Fig 5 shows that the global optimal solution is the global optimal fitness function value, and the smaller the fitness value, the better the result. IPP-PSO approximately converges to the global optimal solution after about 20 iterations, PP-PSO approximately converges to the global optimal solution after about 40 iterations, and the PSO algorithm does not converge to the global optimal solution in this simulation. The convergence speed of IPP-PSO algorithm is faster than that of PP-PSO and PSO algorithm, and it can converge to the global optimal solution. Fig 6 shows the change of the average fitness function value with the number of iterations. The average fitness value represents the average level of fitness values of all particles in the algorithm. It can be seen from Fig 6 that the IPP-PSO algorithm improves the problem that the average fitness function value of PP-PSO algorithm cannot converge, and the convergence speed is better than that of the PSO algorithm. Fig 7 shows the variation of the average error with the number of iterations. The PP-PSO algorithm and PSO algorithm fail to converge near the global optimum, and the average error is large. The improved IPP-PSO algorithm improves this problem, and the "precocity" problem of the algorithm is improved because pathfinder is added to improve the population diversity. This addition also speeds up the convergence speed of the algorithm.

Tables 3 and 4 show the simulation results of 100 tests, which reduces the contingency of the algorithm in a simulation and can better show the real performance of the algorithm. The average fitness value is the average of the optimal fitness value of each test, the optimal fitness

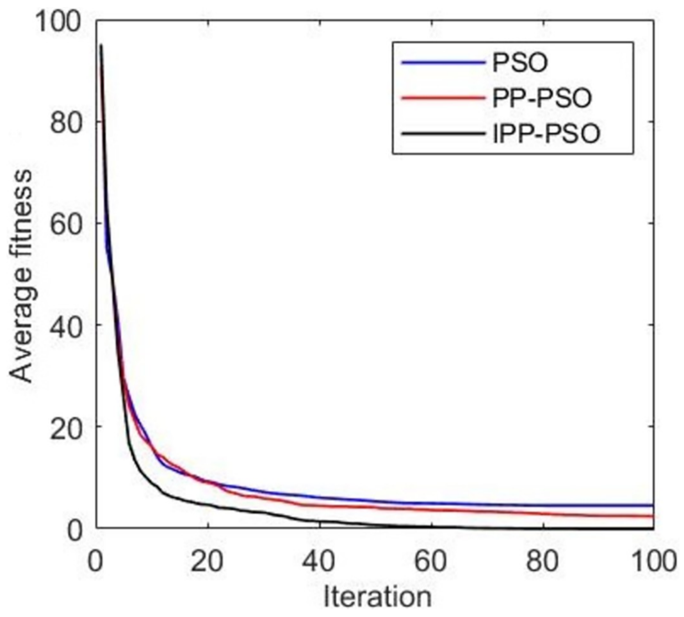

Fig 6. The average fitness function value of two—person zero—sum game simulation results.

https://doi.org/10.1371/journal.pone.0260231.g006

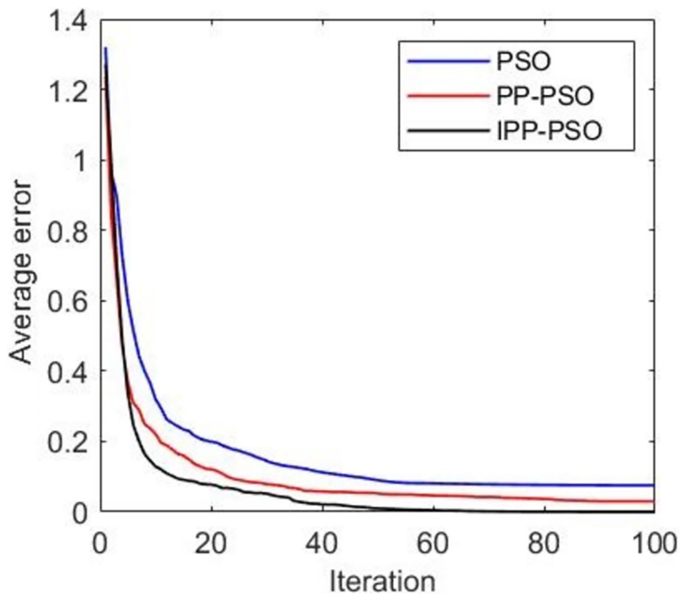

Fig 7. Average error simulation results of two-person zero-sum game.

https://doi.org/10.1371/journal.pone.0260231.g007

Table 3. Simulation results of two-person zero-sum game.

	Average fitness value	Optimal fitness value	Mean minimum error	Standard deviation	Number of successful
PSO	0.0675	4.8357e-8	0.0120	0.1279	4
PP-PSO	0.0128	5.9138e-4	0.0051	0.0731	0
IPP-PSO	0.0015	1.4794e-9	8.0634e-4	0.0047	97

https://doi.org/10.1371/journal.pone.0260231.t003

Table 4. Simulation results of two-person non-zero-sum game.

	Average fitness value	Optimal fitness value	Mean minimum error	Standard deviation	Number of successful
PSO	8.2190	2.0876e-8	0.0210	0.2070	3
PP-PSO	0.0831	0.0056	2.6151e-3	0.0202	0
IPP-PSO	1.1077e-5	3.7714e-9	1.3308e-5	5.4336e-4	94

https://doi.org/10.1371/journal.pone.0260231.t004

value is the minimum fitness value of the 100 tests, the mean minimum error is the average of the mean minimum error of each test, and the number of successes is the number of times the mean error is less than 0.0001. By comparing the three algorithms in Tables 3 and 4, it can be concluded that the IPP-PSO algorithm has the minimum average fitness value, minimum average error, minimum optimal fitness value and the most successful times, which further demonstrates that the IPP-PSO algorithm has better performance than both the PP-PSO algorithm and the PSO algorithm in solving Nash equilibrium problems.

According to the above analysis, the IPP-PSO algorithm is superior to the PP-PSO algorithm and the original PSO algorithm in terms of solving accuracy, convergence speed and reliability of Nash equilibrium calculation. It can be used to solve the Nash equilibrium of the game problem of attack and defense confrontation of clustered UAVs.

Conclusion

When we study the game model of attack and defense confrontation of swarm UAVs, we encounter the problem of choosing the best algorithm to solve the game model. In the process of trying to solve Nash equilibrium with PSO and PP-PSO, combined with the results of previous studies, we find the convergence and local optimality of these two algorithms. An IPP-PSO algorithm is proposed to solve the Nash equilibrium of the game. By improving the distribution of predator and prey, the formula of inertia weight and velocity of predator and prey particles, and adding pathfinder particles, the algorithm improves the problems of the original algorithm, such as "precocity", slow convergence and local optimum. The performance of PSO, PP-PSO and IPP-PSO algorithm in solving Nash equilibrium of a game is compared with simulation examples. Simulation results show that the IPP-PSO algorithm is convergent and effective, and the convergence speed and global optimal performance are better than the other two algorithms. The improved algorithm can be used to solve the game model of attack and defense confrontation of clustered UAVs.

Supporting information

S1 Data.
(XLSX)

S2 Data.
(XLSX)

S3 Data.
(XLSX)

S4 Data.
(XLSX)

S5 Data.
(XLSX)

S6 Data.
(XLSX)

S7 Data.
(XLSX)

S8 Data.
(XLSX)

S9 Data.
(XLSX)

S10 Data.
(XLSX)

S11 Data.
(XLSX)

S12 Data.
(XLSX)

S13 Data.
(XLSX)

S14 Data.
(XLSX)

S15 Data.
(XLSX)

S16 Data.
(XLSX)

S17 Data.
(XLSX)

S18 Data.
(XLSX)

Author Contributions

Conceptualization: Yufeng Meng, Jianhua He, Siqi Tao.

Data curation: Siqi Tao.

Formal analysis: Jiancheng Xu.

Funding acquisition: Jianhua He.

Investigation: Shichu Luo, Jiancheng Xu.

Methodology: Yufeng Meng, Siqi Tao.

Project administration: Jianhua He.

Resources: Shichu Luo.

Software: Siqi Tao.

Supervision: Jianhua He.

Validation: Jianhua He.

Visualization: Siqi Tao.

Writing – original draft: Yufeng Meng.

Writing – review & editing: Yufeng Meng.

References

1. Wang Y, Zhang W, Fu L, Huang D, Li Y. Nash Equilibrium Strategies Approach for Aerial Combat Based on Elite Re-election Particle Swarm Optimization. Control Theory and Applications. 2015; 32 (7):857–865.
2. Duan H, Li P, Yu Y. A Predator-Prey Particle Swarm Optimization Approach to Multiple UCAV Air Combat Modeled by Dynamic Game Theory. IEEE/CAA Journal of Automatica Sinica. 2015; 2(1):11–18. https://doi.org/10.1109/JAS.2015.7032901
3. Chen X, Li G, Zhao L. Research on UCAV Game Strategy of Cooperative Air Combat Task. Fire Control and Command Control. 2018; 43(11):17–23.
4. Luo H, Ma Y, Wang G, Hu X, Lei X, Zhu M, et al. Nash Equilibrium Game Method and System for Intelligent Decision Making of UAV in Adversarial Environment. CN110134138A; 2019.
5. Chen X, Zhao M, Xu G. Fuzzy Dynamic Game Based Operation Strategy for Multiple UAVs. Electronics Optics and Control. 2014; 21(6):19–23,34.
6. Rechenberg IE. Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution. Technical University of Berlin; 1971.
7. Schwefel HP. Evolutionsstrategie und Numerische Optimierung. Technical University of Berlin; 1975.
8. Schwefel HP. Numerical Optimization of Computer Models. Journal of the Operational Research Society. 1981; 33(12):1166.
9. Schwefel HP. Evolution and Optimum Seeking. Germany: DBLP; 1995.
10. Chen S, Sun Y, Wu Z. A Genetic Algorithm to Acquire the Nash Equilibrium. Systems Engineering. 2001; 19(5):67–70.
11. Dorigo M, Gambardella LM. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Transactions on Evolutionary Computation. 1997; 1(1):53–66. https://doi.org/10.1109/4235.585892
12. Wang Z, Han X, Xu W, Yang J. Nash Equilibrium Solution Based on Improved Ant Colony Algorithm. Computer Engineering. 2010; 36(14):166–168,171.
13. Yin J, Cheng K. Another Solving Method of Nash Equilibrium. Statistics and Decision. 2017; 15:70–72.
14. Kennedy J, Eberhart R. Particle Swarm Optimization. Proceedings of ICNN'95- International Conference on Neural Networks. 1995; 4:1942–1948.
15. Eberhart R, Shi Y. Particle Swarm Optimization: Developments, Applications and Resources. Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546). 2001;1:81–86.
16. Duan H, Yu Y, Zhao Z. Parameters Identification of UCAV Flight Control System Based on Predator-Prey Particle Swarm Optimization. Science China Information Sciences. 2013; 56(1):117–128. https://doi.org/10.1007/s11432-012-4754-9
17. Duan H, Li S, Shi Y. Predator-Prey Brain Storm Optimization for DC Brushless Motor. IEEE Transactions on Magnetics. 2013; 49(10):5336–5340. https://doi.org/10.1109/TMAG.2013.2262296
18. Pan F, Li X, Zhou Q, Li W, Gao Q. Analysis of Standard Particle Swarm Optimization Algorithm Based on Markov Chain. Acta Automatica Sinica. 2013; 39(4):381–389. https://doi.org/10.1016/S1874-1029(13)60037-3
19. Wang W. Research on Particle Swarm Optimization Algrithm and Its Application. PhD thesis, Southwest Jiaotong University, 2013.
20. Liu D. Research on Particle Swarm Optimization and Its Engineering Application. PhD thesis, Southwest Jiaotong University, 2013.
21. Feng Q, Li Q, Quan W, Pei X. Overview of Multi-objective Particle Swarm Optimization Algorithm. Chinese Journal of Engineering. 2021; 43(6):745–753.
22. Shi X. Research of Swarm Intelligence Algorithm Based Particle Swarm Optimization and Chicken Swarm Optimization. PhD thesis, Ningxia University, 2018.
23. Lin W. Research of Improved Particle Swarm Optimization and Its Applications. PhD thesis, East China University of Science and Technology, 2013.

24. Wu W, Meng X, Kang Q, Li Q. Particle Swarm Optimization Algorithm for Hybrid Strategy Approximation Nash Equilibrium. Application Research of Computers. 2014; 31(8):2299–2302.

25. Chen X, Wang Y. Study on Multi-UAV Air Combat Game Based on Fuzzy Strategy. Applied Mechanics and Materials. 2014; 49(1):1102–1105. https://doi.org/10.4028/www.scientific.net/AMM.494-495.1102

26. Qu Y, Zhang J, Song Y. Particle Swarm Optimization Algorithm for Solving Multiple Nash Equilibrium Solutions. Operations Research and Management. 2010; 19(2):52–55.

27. Jia W, Xiang S, Yang J, Hu W. Solving Nash Equilibrium for N-persons' Non-cooperative Game Based on Immune Particle Swarm Algorithm. Application Research of Computers. 2012; 29(1):28–31.

28. Yu Q, Wang X. Evolutionary Algorithm for Solving Nash Equilibrium Based on Particle Swarm Optimization. Journal of Wuhan University. 2006; 52(1):25–29.