The Role of Block Particles Swarm Optimization to Enhance The PID-WFR Algorithm

Heru Suwoyo
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Abdurohman
School of Mechatronic Engineering and Automation, Shanghai University, China

Yifan Li
School of Mechatronic Engineering and Automation, Shanghai University, China

Andi Adriansyah
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Yingzhong Tian
School of Mechatronic Engineering and Automation, Shanghai University, China

Muhammad Hafizd Ibnu Hajar
Department of Electrical Engineering, Universitas Mercu Buana, Indonesia

Abstract: In the conventional Proportional Integral Derivation (PID) controller, the parameters are often adjusted according to the formulas and actual application. However, this empirical method will bring two disadvantages. First, testing the program takes much time and usually needs help to reach the optimal solution. Second, the PID parameters will not adapt to the new environment when the situation changes. This paper proposed a method by employing a Block Particles Swarm Optimization (BPSO) to enhance the conventional Proportional Integral Derivation (PID) algorithm to overcome the mentioned disadvantages. The genetic algorithm (GA) first optimized the PID parameters. However, its optimization time is relatively long. Then, a Block Particle Swarm Optimization (BPSO) algorithm is designed to solve the problem of long optimization time. This method was then applied to the wall-following robot problem by realistically simulating it to confirm the performance. After Compared with conventional methods, the proposed method shows a relatively stable solution.

Keywords: PID controller, Block Particle Swarm Optimization, Wall Following Robot, Genetic Algorithm

Introduction

Currently, the PID algorithm is widely used in industrial process control because of its ease. The essential idea is to construct a PID controller made up of the proportional, integral, and derivative of the system deviation linearly to control the object (Adriansyah et al., 2019). It has good control results, high security, and good stability. Since the invention of the PID controller (mainly due to Elmer's invention of ship autopilot) in 1910 and the Ziegler-Nichols PID parameter tuning method (Z-N method) in 1942 (Joseph et al., 2022) the applications of PID controllers have been developed and popularized extremely (Guo et al., 2022). PID parameters tuning includes the engineering and theoretical calculations tuning methods (Lawrence et al., 2022). The engineering tuning method tunes the parameters according to engineering experience directly. It often requires much experience, and different objective functions correspond to different experiences; the theoretical calculations tuning method obtains the PID parameters based on the mathematical model of the system. It takes much time to calculate and test the program, which usually cannot reach the optimal solution. PID parameter tuning is a complex and tedious process, and the research on the PID parameter tuning method has been an essential issue in the control field (Kvascev & Djurovic, 2022; Song et al., 2022).

Many intelligent optimization algorithms have been applied to PID parameter tuning and have achieved a good control effect to improve the efficiency of the PID controller parameter tuning. Such as genetic algorithm, particle swarm optimization, ant colony optimization algorithm, etc. However, there are various problems with the algorithms mentioned above. Some have slow convergence speeds, some can only get local optimal solutions, and some have high error rates. This paper uses the block particle swarm optimization algorithm (BPSO) for PID parameter tuning. It is an optimization algorithm that combines the concept of initial population fragmentation and the traditional particle swarm algorithm. It has a small population size, fast convergence speed and solid global search capabilities.

The remainder of the paper is organized as follows. In Section 2, Two forms of PID algorithm and their application will be mentioned briefly (Wang et al., 2022). Then take the genetic algorithm and particle swarm optimization algorithm as examples to illustrate the optimization of parameters and explain their working mechanism. Last but not least, the BPSO algorithm will be introduced to apply to the simulation of a wall-following robot. In Section 3, the running results of the BPSO algorithm on the simulation of a wall-following robot are analysed and compared with other optimization algorithms in terms of convergence speed, running time, and an optimal solution. Section 4 provides the concluding remarks and prospects (Li et al., 2022; Pervaiz et al., 2022; Wei et al., 2017).

Research Method

PID Algorithm

The PID controller consists of a proportional unit P, an integral unit I and a differential unit D. Set the three parameters of Kp, Ki, and Kd. The PID controller is mainly suitable for systems where the essential linearity and dynamic characteristics do not change with time. A proportional segment is an error between the actual and expected values (Behera & Choudhury, 2022; Havaei & Sandidzadeh, 2022; Wiangtong & Sirapatcharangkul, 2017).

$error = (target\ value) - (sensor\ reading)$

An integral segment is the sum of previous errors used for adjustment when the error is small.

$integral = integral + error * dT$

The differential segment is the change in error used to predict what the following error might be.

$derivative = ((current\ error) - (pervious\ error))/dT$

PID control system is mainly composed of PID controller and controlled object, and a typical PID control system can be described in Figure 1.

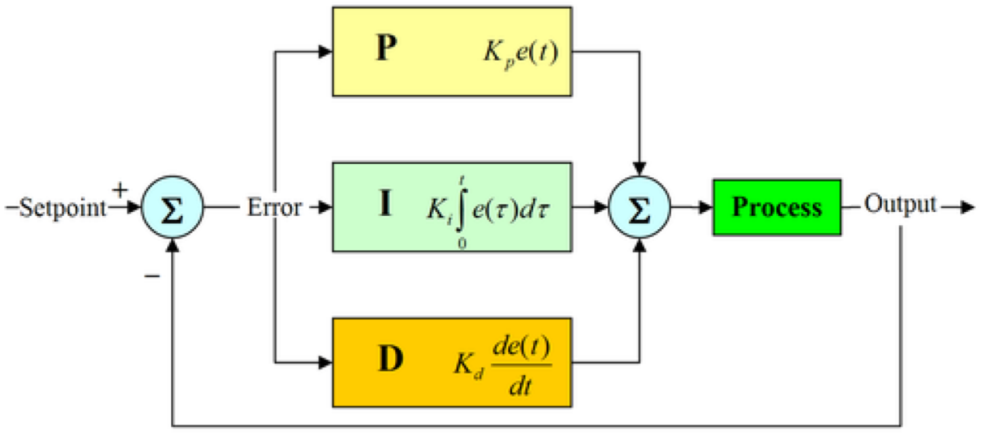

Figure 1 PID Control System

Position PID algorithm

PID control is a two orders linear controller. The original formula of the PID algorithm is equation 1:

$$u(t) = K_p \left[e(t) + \frac{T}{T_i} \int_0^t e(t)dt + \frac{T_d}{T} \frac{de(t)}{t} \right] \quad (1)$$

Where (it) is the output of control systems; (it) is the input of control systems; in the above formula, (it) is the output of the control system; (it) is the input of the control system, generally the difference between the set quantity and the controlled quantity, that is, $e(t) = r(t)- e(t)$; K PIs the proportional coefficient of the control system; T_i is the integral time of the control system; T_d is the differential time of the control system; T is the sampling period of the system (Kotb et al., 2022).

Discretization formula equation 2:

$$u(k) = K_p \left\{ e(k) + \frac{T}{T_i} \sum_{i=0}^{k} e(i) + \frac{T_d}{T} [e(k) - e(k-1)] \right\} \tag{2}$$

The position-type PID control algorithm is suitable for the actuator without an integral element. The action position of the actuator is one-to-one, corresponding to its input signal. According to the deviation $e(t)$ between the sampling result of the k-th controlled variable and the set value, the controller calculates the output control variable after the k-th sampling.

The disadvantage of the position PID algorithm is that sampling output is related to any previous state and not independent control quantity. When calculating, the accumulator should be used to accumulate $e(t)$ quantity, which is very large. At the same time, the output of the control system $u(t)$ is relative to the actual link of implementing equipment. Once the computer breaks down, $u(t)$ will change significantly, leading to the implementation of equipment position Dramatic changes (Liu et al., 2022).

Incremental PID Algorithm

Incremental PID algorithm control is the output increment of the control quantity (represented by $\Delta u(k)$) control system. When the algorithm is applied, the output control amount $\Delta u(k)$ is relative to the position increment of the equipment implemented this time, not to the actual position of the equipment implemented. Therefore, the algorithm needs the equipment to accumulate the control amount increment to realize the control of the controlled system. The cumulative function of the system can be realized by hardware circuit and software programming method, for example, by using the formula $\Delta u(k) = u(k) - u(k-1)$ to realize programming (Mok & Ahmad, 2022).

From Equation 2, Equation 1 below:

$$u(k-1) = K_p \left\{ e(k-1) + \frac{T}{T_i} \sum_{i=0}^{k-1} e(i) + \frac{T_d}{T} [e(k-1) - e(k-2)] \right\} \tag{3}$$

Subtract Equation 2 from Equation 3, Equation 2 below:

$$\Delta u(k) = K_p \left\{ e(k) - e(k-1) + \frac{T}{T_i} \sum_{i=0}^{k-1} e(i) + \frac{T_d}{T} [e(k) - 2e(k-1) + e(k-2)] \right\} \tag{4}$$

The advantages of using the incremental PID algorithm: there is no accumulation link in the formula, so a large amount of calculation is not needed; the control increment value $\Delta u(k)$ is related to the latest three sampling values of the system, which is convenient to use weighted processing to achieve good control effect; each time the computer output is only the control increment, that is, the change of relative execution equipment position. If the machine breaks down, the impact on the system will be small, and the production process will not be seriously hindered (Carlucho et al., 2019).

Genetic Algorithm

A genetic algorithm (GA) is a method for searching for the optimal solution. It simulates reproduction, mating, and mutation in natural selection and genetic evolution. It has the advantages of easy implementation and fast convergence speed when used in path planning. GA performs well on simple maps but poorly on complex maps and maps with many obstacles. Because GA can only search randomly in the solution space, it does not use the given information on the map (Yichen et al., 2020).

The flow chart of GA shows in Figure 2.

a) Generate a random population.

b) According to the strategy, we can judge whether the individual's fitness conforms to the optimization criteria. If it conforms, we can output the best individual and optimal solution and end the program. Otherwise, proceed to the next step.

c) Parents are selected according to their fitness, the individuals with high fitness are assigned with high probability, and the low fitness individuals are eliminated.

d) The parent's chromosomes were used to intersect to generate offspring according to specific methods.

e) Variation of offspring chromosomes.

f) A new generation population is generated by crossover and mutation, and step 2 returns until the optimal solution is generated.

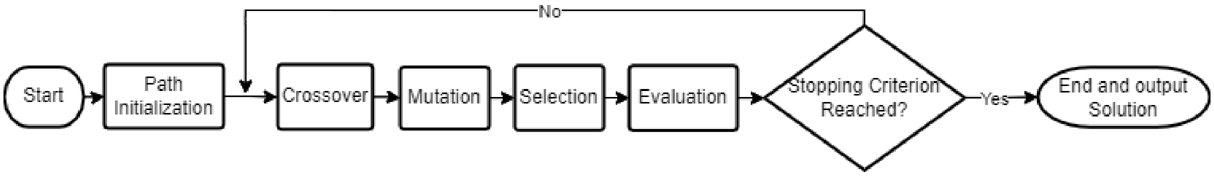

Figure 2 The flowchart of GA

Particle Swarm Optimization Algorithm

The algorithm is initially inspired by the regularity of the birds' cluster activities, and then a simplified model is established using group intelligence. Based on the observation of the behaviour of animal clusters, particle swarm optimization (PSO) uses information sharing among individuals to move the whole population from disorder to order in the problem-solving space to obtain the optimal solution (Zhang et al., 2022).

The mathematical language of the PSO algorithm is described as follows. Suppose there is a population composed of N particles, the position of each particle i at any time is an n-dimensional vector of decision space. At the iteration time t, the position of the particle can be recorded as $x\ (t) = [x\ (t), x\ (t), \ldots, x\ (t)]$, $(i = 1, 2, \ldots, N)$. The fitness of each particle can be obtained by substituting $x_i\ (t)$ into the objective function or fitness function, and the particle's fitness value determines the particle's quality. Comparing the current position of each particle with the position in the previous iterations, the optimal historical position of particle i in the t-th iteration is obtained, which is called individual extremum and is recorded as $x_i^{gb}(t) = [x_{i1}^{gb}(t), x_{i2}^{gb}(t), \ldots x_{i3}^{gb}(t)]^T$(the local extremum is not unique, for convenience, the symbol corresponding to the individual extremum is still used, but in this formula, i is no longer a symbol for particles.) The corresponding velocities of each particle i in the t-th iteration are recorded as v_i(t) = $[v_{i1}(\mathrm{t}), v_{i2}(\mathrm{t}), \ldots v_{in}(t)]^T$,$(i\ =\ 1, 2, \ldots, N)$ Each dimension of the individual extremum $(might\ consist)$ of particle the following formula updates me in iteration t, Equation 3 below:

$$x_{id}^{pb}(t+1) = \begin{cases} x_{id}(t+1), F(x_{id}(t+1)) < \ F(X_{id}^{pb}(t)) \\ X_{id}^{pb}(t), F(x_{id}(t+1)) \geq \ F(X_{id}^{pb}(t)) \end{cases} \tag{5}$$

where the fitness function is $F(x)$.

Each dimension of the global extremum $x_i^{gb}(t)$of all particles in the t-th iteration is selected as follows Equation 6:

$$x_{id}^{gb}(t+1) = arg\{minF(x_{id}^{gb}(t+1)\} \tag{6}$$

Where arg is the value of the independent variable corresponding to the function value $f\ (x), x_i^{gb}(t+1)$ is the optimal global position of particles in the $t+1$ iteration. In the $t+1$ iteration, each dimension of position $x\_i\ (t+1)$ and velocity $v\_i\ (t+1)$ of particle I are updated as follows, Equation 7:

$$\begin{gathered} v_{id}(t+1) = v_{id}(t) + c_1 c_{i1,t}(x_{id}^{pb}(t) - x_{id}(t)) + c_2 c_{i2,t}(x_{id}^{gb}(t) - x_{id}(t)) \\ x_{id}(t+1) = x_{id}(t) + v_{id}(t+1) \end{gathered} \tag{7}$$

Equation 4

$$\begin{cases} v_{id} = v^{max}, v_{id} > v^{max} \\ v_{id} = -v^{max}, v_{id} > -v^{max} \end{cases} \tag{8}$$

Where the v is a constant, $c\_1\ \ c\_2$ is a learning factor, which is a non-negative constant; $r1r2$ is a random number between 0 and 1; from Equation 7, we can see that the velocity renewal formula of particles can be divided into three parts: the first part, $v\_id\ (t)$ represents the $d-Dimension$ of velocity $v\_i\ (t)$ of particle I in the t-iteration process, which shows that particle can explore new areas and develop its own so that the algorithm has the capability of global search, It is often used to balance the global and local search ability of particles; the second part represents the self-learning ability of particles, so that particles have strong local search ability; the third part can be seen as the ability of particles to learn from other members of the population, reflecting the information sharing between particles.

The steps of the original particle swarm optimization algorithm are as follows (Pinto et al., 2013) :

STEP 1: initialize particles randomly;

STEP 2: Calculate the fitness of each particle;

STEP 3: According to the formula, the individual extremum of particles is updated;

STEP 4: Update the optimal global position according to the formula;

STEP 5: Update the particle's speed and position according to the formula;

STEP 6: If the termination condition is not reached (usually the preset maximum number of iterations or the preset fitness value), Step 2 is returned. Otherwise, the algorithm is terminated.

BPSO Algorithm

The initial population data in the range is set, and the range is divided into several equal parts in this algorithm. The initial population number is equally distributed in each block.

Take the function $f(x)$ as an example, Equation 5:

$$f(x) = 1 + (2.1(1 - x + 2x^2)e^{(\frac{-x^2}{2})}) \tag{9}$$

Where $x \in [-5,5]$, the number of initial particle swarms is set to 12 and divided into four parts. Each particle swarm are distributed in the interval [- 5, - 2.5], [- 2.5,0], [0,2.5], [2.5,5], with an average of three particles in each block. (See Figure 3).

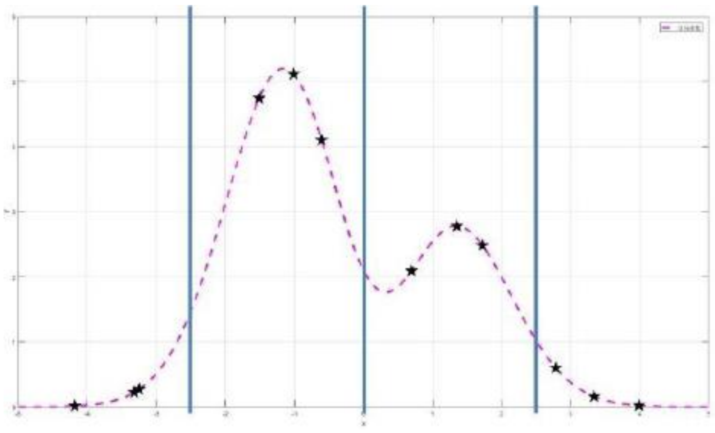

Figure 3 Block of Initial Particles

The following process is the same as the particle swarm optimization algorithm. The particle is substituted into the fitness function to calculate the fitness value of each particle. After selection, the particle with the smallest function value is the benchmark to let other particles move towards it at a certain speed. The particles finally gather near a point in many iterations to get the optimal global solution. The output parameters of particles are Kp, Ki and Kd.

Application

In this section, the wall following robot will be simulated in MATLAB. It can measure the distance from the wall and use the PID algorithm to adjust its moving direction and realize the following process. The BPSO algorithm will optimize three parameters of the PID algorithm.

Establishment of The Simulation Model

Firstly, we assume the climbing robot is an equilateral triangle and establish the robot's pose coordinate system, as shown in Figure 4.

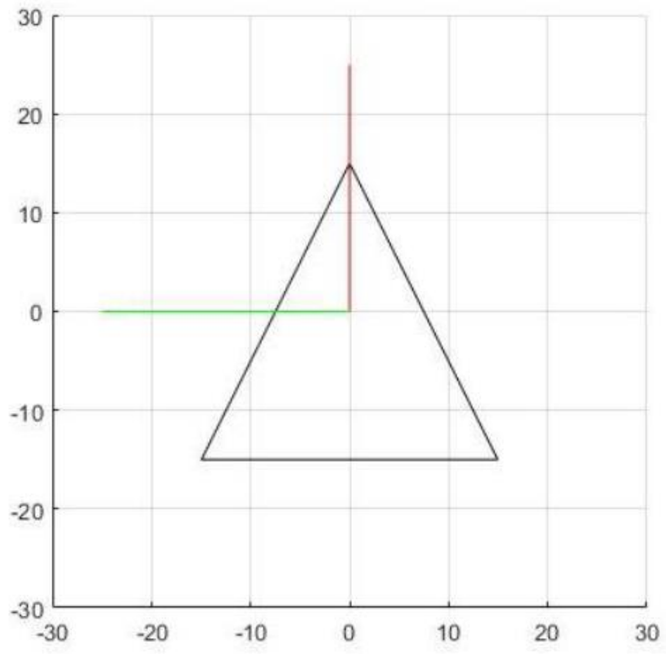

Figure 4 The initial position of the robot

Then the displacement matrix of the robot consists of two parameters: translation and rotation. Because a continuous function cannot control the robot's pose, the robot is driven by discrete differences. The position can be estimated for this robot starting from a known position by integrating the movement (summing the increment travel distances). Assuming Robot configuration $q_k = [x_k, y_k, \theta_k]$ and constant velocity inputs v_k and w_k are known at discrete time t_k then using Euler integration , Equation 6:

$$\begin{cases} x_{k+1} = x_k + v_k T_s cos\theta_k \\ y_{k+1} = y_k + v_k T_s sin\theta_k \\ \theta_k = \theta_k + \omega_k T_s \end{cases} \tag{10}$$

Where (x_k, y_k) is the current position of the robot, and $(x_{k+1}, y_{k+1})^T$ is the position at next time. Meanwhile, θ_k is updated by time t, angular velocity ω_k and current angular displacement θ_k. All these parameters can be represented by the following Figure 5 (Left). Meanwhile, the movement and rotation of the robot can be set at short intervals. The robot's trajectory can be accumulated through these discrete points to achieve a period movement trajectory, as shown in Figure 5 (Right).

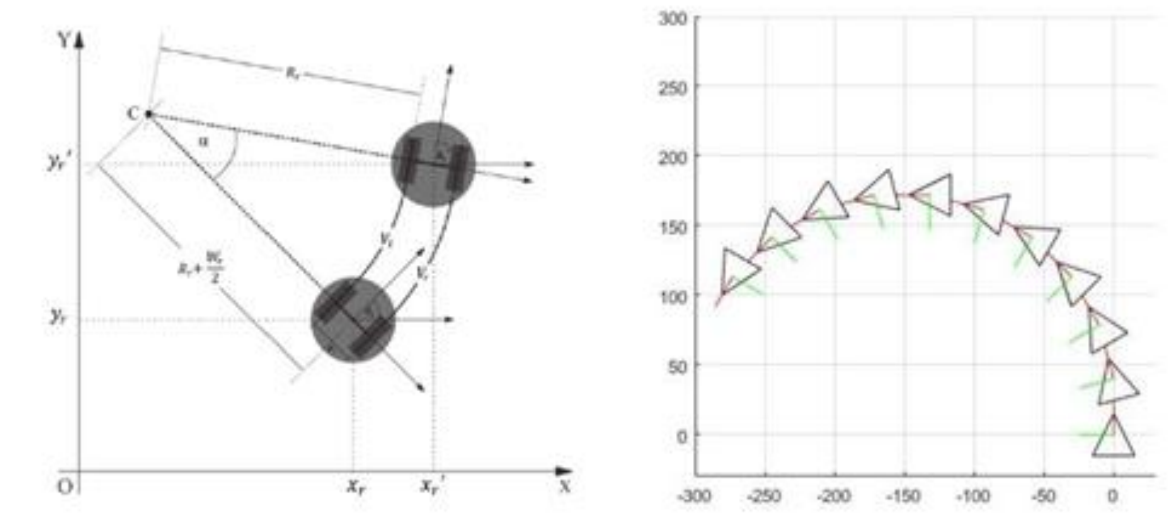

Figure 5 The initial position of the robot (Left) and The trajectory of the robot (Right)

Initialization of Particles

Firstly, set the particle swarm size to 28 (multiple of 4), set the three parameters KP, Ki, and KD of PID as independent variables, and define particles' acceleration and inertia factors. Because the wall-climbing robot optimizes RMSE by reading the distance from the wall and using three parameters Kp, Ki and Kd in PID algorithm, it is a multi-function optimization problem. The block particle swarm optimization (PSO) algorithm divides these three parameters into four average blocks from 0 to 0.6. The advantage of this method is that it can avoid omitting the optimal local solution and reduce the number of iterations. The codes are as follows:

```
individual1 = 0.15 * rand(particlesize/4,narvs);
individual2 = 0.15 + 0.15 * rand(particlesize/4,narvs);
individual3 = 0.30 + 0.13 * rand(particlesize/4,narvs);
individual4 = 0.45 + 0.15 * rand(particlesize/4,narvs);
x = [individual1;individual2;individual3;individual4];
```

Fitness Function

In this function, firstly, the KP, Ki and Kd values of all particles are input as a function, and then a multivariate function $u = f(K\_p, K\_i, K\_d)$ about KP, Ki and KD can be obtained. The root means square error of the robot's current and last positions can be calculated, and the root means square error sequence can be obtained by cycling. Then the minimum value in the sequence is read and compared with the global minimum value, and the smaller value is taken as the extreme global value. Furthermore, update the subscript of global extremum. The root means square error is expressed as follows, Equation 7:

$$u = \sqrt{\frac{\sum_{i=o}^{N}(x_{k+1} - x_k)^2}{N}} \tag{11}$$

where N represents the number of robot motions. $x_{k+1} - x_k$ represents errors.

Result and Discussion

Results

Figure 6 represents the particle's initial state and final motion position. It is well explained that the particle swarm is from the initial irregular discrete state after many iterations gathered in a range near a point. The calculated Kp, Ki and Kd values can be obtained from the three coordinate values of particles.

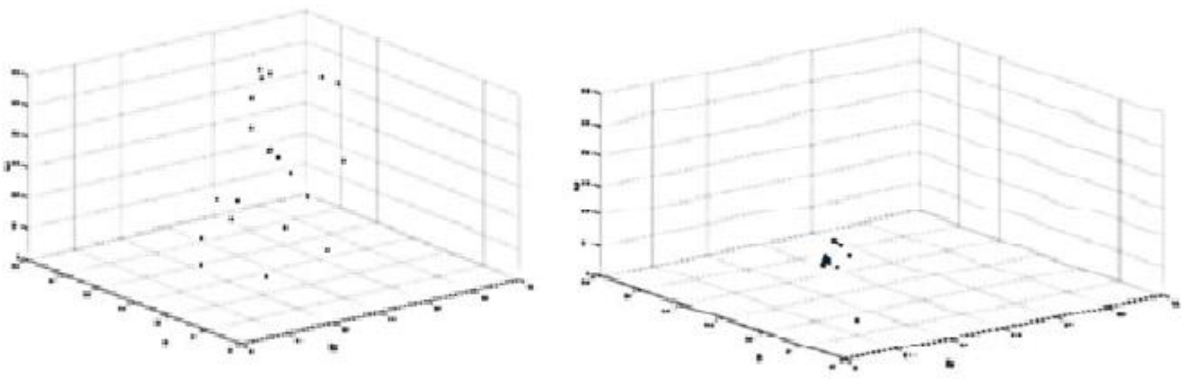

Figure 6 Results of particle swarm operation: the initial state of the particle (Left) and the final motion position of the particle (Right)

We can see this in Table 1 below. After many tests, we find that the PID value is not constant because it is a multivariate function problem that is even more complex than the travelling salesman problem in the ant colony algorithm. The coordinates of the three points can be changed, which leads to the various results of KP, Ki, and KD. However, the value of RMSE is stable between 9 and 10, which shows the reliability of the optimization method.

Table 1 Optimized Value of PID and RMSE

Tests	K_p	K_i	K_d	**RMSE**
1st Test	0.19643	0.15611	0.15874	9.2643
2nd Test	0.18991	0.15938	0.13627	9.3315
3rd Test	0.13317	0.2232	0.17449	9.2424
4th Test	0.13787	0.21424	0.16672	9.3168
5th Test	0.041398	0.03805	0.11515	9.3514

Comparison

Let the two algorithms iterate 500 times simultaneously, GA takes 148 seconds, and BPSO takes 64 seconds. It can be seen from the fitness function that the BPSO algorithm can significantly improve the operation speed and reduce the waiting time. It can also be seen from Figure 7 that BPSO only needs a few iterations to get the optimal solution, while GA needs at least 400 iterations.

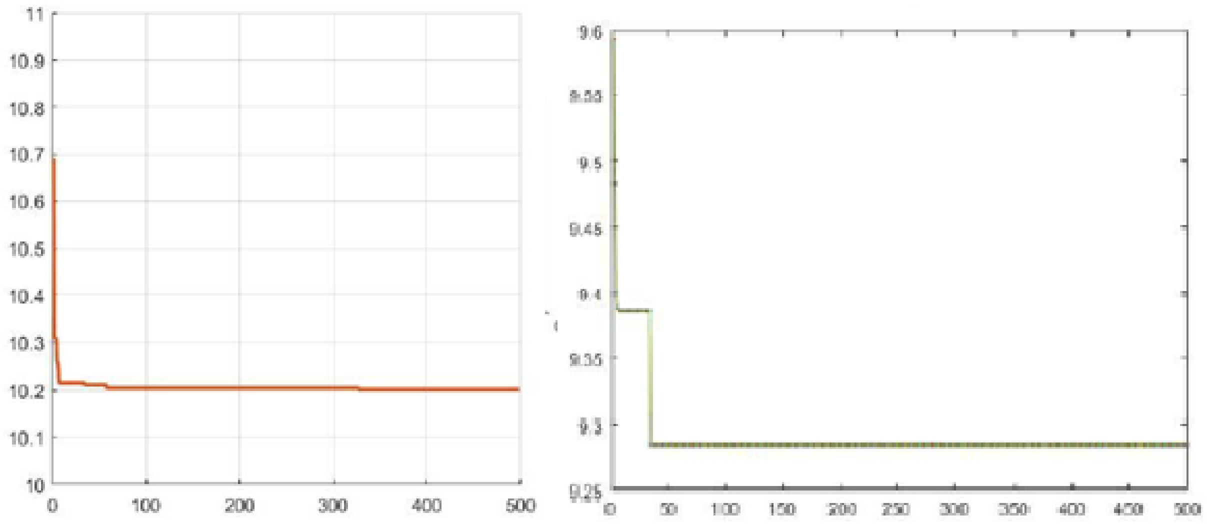

Figure 7 The fitness function of GA(Left) and BPSO (Right)

Figure 8 shows the comparison between BPSO and conventional PSO. Compared with the conventional PSO algorithm, the BPSO algorithm is easier to get the optimal solution under the same number of iterations. It can get the range of the aggregated solution. The solution of conventional PSO is more discrete.

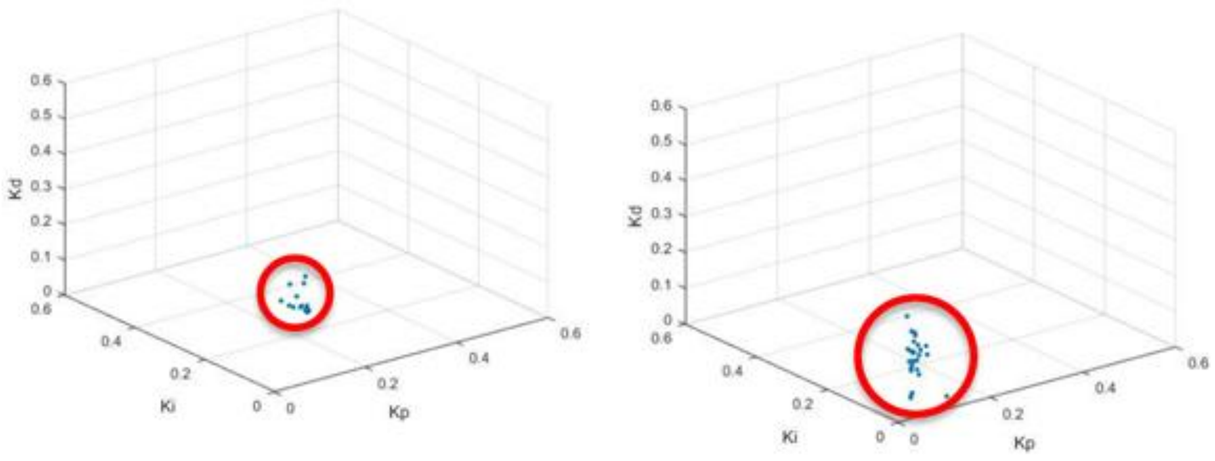

Figure 8 BPSO (Left) and Conventional PSO (Right) Optimal Solutions Comparison

Conclusions

In this study, a block particle swarm optimization algorithm is introduced to optimize the PID parameters of the wall-climbing robot. The function of the block is to make the initial particle swarm evenly distributed in the range of values as far as possible to avoid the occurrence of the optimal local solution and, thus, to get the optimal global solution. Simulation results show that BPSO can significantly reduce the number of iterations and convergence speed compared with GA and PSO mentioned in this study and make the aggregation of particle swarm stronger. In the later research, the parameters of the state transition equation (such as acceleration constant and inertia factor) in PSO can be estimated, so there will be more optimization schemes to optimize the three parameters of the PID algorithm in the future.

References

Adriansyah, A., Suwoyo, H., Tian, Y., & Deng, C. (2019). Improving the Wall-Following Robot Performance using PID-PSO Controller. Jurnal Teknologi, 81(3).

Behera, S., & Choudhury, N. B. D. (2022). Modelling and simulations of modified slime mould algorithm based on fuzzy PID to design an optimal battery management system in microgrid. Cleaner Energy Systems, 3. https://doi.org/10.1016/j.cles.2022.100029

Carlucho, I., De Paula, M., & Acosta, G. G. (2019). Double Q-PID algorithm for mobile robot control. Expert Systems with Applications, 137, 292-307. https://doi.org/10.1016/j.eswa.2019.06.066

Guo, J., Lu, Y., & Li, Z. (2022). PID parameter tuning algorithm of rotor UAV Based on Improved Particle Swarm Optimization. 2022 IEEE 6th Information Technology and Mechatronics Engineering Conference (ITOEC),

Havaei, P., & Sandidzadeh, M. A. (2022). Intelligent-PID controller design for speed track in automatic train operation system with heuristic algorithms. Journal of Rail Transport Planning & Management, 22. https://doi.org/10.1016/j.jrtpm.2022.100321

Joseph, S. B., Dada, E. G., Abidemi, A., Oyewola, D. O., & Khammas, B. M. (2022). Metaheuristic algorithms for PID controller parameters tuning: Review, approaches and open problems. Heliyon, e09399.

Kotb, H., Yakout, A. H., Attia, M. A., Turky, R. A., & AboRas, K. M. (2022). Speed control and torque ripple minimization of SRM using local unimodal sampling and spotted hyena algorithms based cascaded PID controller. Ain Shams Engineering Journal, 13(4). https://doi.org/10.1016/j.asej.2022.101719

Kvascev, G. S., & Djurovic, Z. M. (2022). Water Level Control in the Thermal Power Plant Steam Separator Based on New PID Tuning Method for Integrating Processes. Energies, 15(17), 6310.

Lawrence, N. P., Forbes, M. G., Loewen, P. D., McClement, D. G., Backström, J. U., & Gopaluni, R. B. (2022). Deep reinforcement learning with shallow controllers: An experimental application to PID tuning. Control Engineering Practice, 121, 105046.

Li, X.-L., Serra, R., & Olivier, J. (2022). A multi-component PSO algorithm with leader learning mechanism for structural damage detection. Applied Soft Computing, 116, 108315.

Liu, M., Zhang, H., Zhang, Y., & Yuan, C. (2022). Design and Performance Analysis of ZYNQ Based Incremental PID-PWM Controller 2022 IEEE International Conference on Electrical Engineering, Big Data and Algorithms (EEBDA),

Mok, R., & Ahmad, M. A. (2022). Fast and optimal tuning of fractional order PID controller for AVR system based on memorizable-smoothed functional algorithm. Engineering Science and Technology, an International Journal, 35. https://doi.org/10.1016/j.jestch.2022.101264

Pervaiz, S., Haider Bangyal, W., Ashraf, A., Nisar, K., Haque, M. R., Ibrahim, A., Asri, A., Chowdhry, B., Rasheed, W., & Rodrigues, J. J. (2022). Comparative research directions of population initialization techniques using PSO algorithm. Intelligent Automation & Soft Computing, 32(3), 1427-1444.

Pinto, A. M., Moreira, A. P., & Costa, P. G. (2013). A Localization Method Based on Map-Matching and Particle Swarm Optimization. Journal of Intelligent & Robotic Systems, 77(2), 313-326. https://doi.org/10.1007/s10846-013-0009-2

Song, L., Xu, C., Hao, L., Yao, J., & Guo, R. (2022). Research on PID Parameter Tuning and Optimization Based on SAC-Auto for USV Path Following. Journal of Marine Science and Engineering, 10(12), 1847.

Wang, W., Wu, K., Zhang, Y., Wang, M., Zhang, C., & Chen, L. (2022). The Development of an Electric-Driven Control System for a High-Speed Precision Planter Based on the Double Closed-Loop Fuzzy PID Algorithm. Agronomy, 12(4), 945.

Wei, J., Zhang, R., Yu, Z., Hu, R., Tang, J., Gui, C., & Yuan, Y. (2017). A BPSO-SVM algorithm based on memory renewal and enhanced mutation mechanisms for feature selection. Applied Soft Computing, 58, 176-192. https://doi.org/10.1016/j.asoc.2017.04.061

Wiangtong, T., & Sirapatcharangkul, J. (2017, 25-27 Sept. 2017). PID design optimization using flower pollination algorithm for a buck converter. 2017 17th International Symposium on Communications and Information Technologies (ISCIT),

Yichen, L., Bo, L., Chenqian, Z., & Teng, M. (2020, 10-12 July 2020). Intelligent Frequency Assignment Algorithm Based on Hybrid Genetic Algorithm. 2020 International Conference on Computer Vision, Image and Deep Learning (CVIDL),

Zhang, L., Yin, Q., Zhang, Z., Zhu, Z., Lyu, L., Hai, K. L., & Cai, G. (2022). A wind power curtailment reduction strategy using electric vehicles based on individual differential evolution quantum particle swarm optimization algorithm. Energy Reports, 8, 14578-14594. https://doi.org/10.1016/j.egyr.2022.10.442