http://ejournal.stipjakarta.ac.id

METEOR STIP MARUNDA

pISSN : 1979 – 4746
eISSN : 2685 - 4775

Maritime Institute of Jakarta

Literature Review of Particle Swarm Optimization

[1] *Edi Kurniawan,* [1] *Diana Alia,* [1] *Henna Nurdiansari,* [1] *Sofyan Putra Wijaya*

[1] *Maritime Polytechnic of Surabaya, Surabaya, Indonesia*

email: edi.kurniawan@poltekpel-sby.ac.id

Submitted on : 09/10/2024 Revised : 05/11/2024 Accepted : 20/12/2024

ABSTRACT

Optimization methods are crucial methods in a process because optimization methods can solve complex problems. One of the most effective optimization methods to achieve optimal solutions is Particle Swarm Optimization (PSO), an algorithm inspired by the social behavior of animals. Where, the PSO algorithm is a particle (parable an animal) that has been initialized will move continuously updating its position based on a combination of two factors, namely the attraction towards the individual's best position (pBest) and the attraction towards the global best position (gBest) until it reaches the position optimal. Particle movement is influenced by three main control parameters, namely cognitive coefficient (c1), social coefficient (c2), and inertial weight (ω) in order to produce optimal values without being trapped in local solutions. The advantages of PSO compared to other optimal methods such as the Firefly Algorithm (FA) and Gray Wolf Optimizer (GWO) are its convergence speed and ability to handle non-linear problems with noise. This makes PSO good for applying to complex problems such as solving non-linear mathematical model problems, optimizing fuzzy controllers, optimizing exhaust gas emission parameters and engine performance on ships.

Keywords: *optimization, pBest, gBest, PSO*

INTRODUCTION

The need for humans to solve problems in the most efficient and effective way has been a matter of interest since ancient times. Beginning with classical geometry in ancient Greece, concerning the isoperimetric problem, which focuses on finding the shape with the maximum area for a given perimeter, this demonstrates that even in ancient times, humans were already considering ways to achieve optimal outcomes under limited conditions. Optimization has continued to evolve, from calculus optimization methods, linear and nonlinear optimization, to metaheuristics that can now solve highly complex problems where classical approaches are no longer efficient.

Optimization can be defined as a process of selecting the most efficient and effective solution from all possible options. To solve an optimization problem, the following steps can be taken: First, identifying the problem; second, formulating the objective function and constraints; third, selecting an optimization algorithm; and fourth, evaluating solutions until an optimal solution is reached.

Metaheuristics are robust and flexible approaches for solving complex optimization problems by leveraging principles derived from

nature [1]. Metaheuristics explore a vast search space and can identify optimal solutions for an optimization problem.

More than 40 years ago, the first recognized metaheuristic, Simulated Annealing (SA), was introduced. This algorithm was inspired by the physical annealing process in metallurgy, where material is heated and then slowly cooled to minimize defects and achieve a more stable structure. Since then, various other metaheuristic methods have emerged, such as the Grey Wolf Optimizer (GWO), an algorithm inspired by the hunting behavior of grey wolf packs. GWO mimics how wolves collaborate—alpha, beta, delta, and omega wolves—to find, encircle, and chase prey. The Whale Optimization Algorithm (WOA) is based on the hunting behavior of humpback whales using the bubble-net feeding technique. This algorithm imitates the spiral movement patterns humpback whales use to pursue prey underwater. The Harris Hawk Optimization (HHO) is inspired by the group hunting behavior of Harris's hawks, which use siege tactics to capture prey. Elephant Herding Optimization (EHO) draws from the social behavior of elephant herds, where elephants live in family groups led by older females, imitating the herd's migration patterns and task allocation. Particle Swarm Optimization (PSO) is based on the social behavior of organisms that live in groups, such as bird flocks or fish schools, which coordinate when searching for food or grouping to avoid predators. PSO can identify an optimal solution to a problem with fewer evaluations than other optimization methods [2].

PSO

a. Basic Concept

Particle Swarm Optimization (PSO) is a swarm-based algorithm, known for its simple, nature-inspired design, developed by Russell C. Eberhart, an electrical engineer, and James Kennedy, based on the flocking behavior of birds [3]. A single "bird" represents a solution within the problem space, where the term "bird" here refers to a "particle." Compared to other methods, PSO can identify optimal solutions with fewer evaluations and generally operates more efficiently and effectively [4]. PSO is also easy to implement for various problems [5]. Due to its simple model, PSO has attracted the attention of many researchers and has consequently been widely published to demonstrate its efficient performance across various application fields [3], [6].

The PSO computational method aims to optimize a problem iteratively, beginning with a set or population of candidate solutions known as a swarm of particles. Each particle is aware of both the global best position within the swarm and its individual best position discovered thus far during the search process in the problem space [7].

At each iteration, the velocity and position of each particle in the swarm, represented by a d-dimensional vector, are influenced by individual experiences and acquired information. This guides the iterative movement of particles through the potential solution space to search for the optimal solution until the desired criteria are met.

The velocity of particles in the swarm is updated at each iteration using as in (1) [8]:

$$\vec{V}_{t+1}^{i} = \vec{V}_{t}^{i} + \varphi_1 R_{1_t}^{i}\left(\vec{p}_t^{i} - \vec{x}_t^{i}\right) + \varphi_2 R_{2_t}^{i}\left(\vec{g}_t - \vec{x}_t^{i}\right) \quad (1)$$

where φ_1 and φ_2 are real acceleration coefficients, known respectively as cognitive and social weights, which control the extent to which the global best and individual best positions influence the velocity and trajectory of the particles.

b. PSO Algorithm

As shown below, is the pseudo code for PSO, which initializes by randomly generating values for the particles. Each particle will consider its individual best value (pBest) and the global best value (gBest) based on its position.

1. Start
2. Initialize PSO parameters:
    a. Number of particles (n)
    b. Randomly initialize the position of each particle in the search space
    c. Randomly initialize the velocity of each particle
    d. Initialize the best individual position of each particle (pBest)
    e. Initialize the global best position among all particles (gBest)
    f. Set the maximum number of iterations
3. Repeat until the maximum iteration is reached or the stopping criterion is met:
    a. For each particle:
        1) Calculate the fitness value of the particle at its current position
        2) If the fitness value at the current position is better than the pBest value, update pBestp with the current position value
        3) If the fitness value at the current position is better than the gBest value, update gBest with the current position value
    b. For each particle:
        1) Update the particle velocity based on the formula:

$$v[i] = v[i] + c_1 * r_1 * (pBest[i] - position\ [i] + c_2 * r_2 * (gBest - position[i])$$

Where, $v[i]$ is the velocity of particle i at the previous iteration, c_1 and c_2 are acceleration constants, r_1 and r_2 are random numbers between 0 and 1, pBest$[i]$ is the best position of particle i, gBest is the global best position, position $[i]$ is position number i.

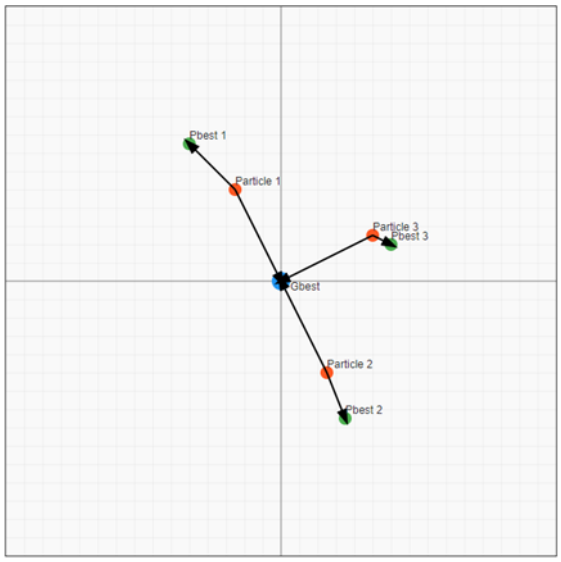
2) Update the position of the particle based on the formula:

$$position[i] = position[i] + v[i]$$

c. Check if the stopping criteria have been met

Finish: gBest is the best solution found.

To provide further clarity, here is a simple Cartesian diagram illustrating how PSO works.



Each initialized particle will continuously update its position based on a combination of two factors: attraction toward its individual best position (pBest) and attraction toward the global best position (gBest). The vectors shown in the diagram illustrate the direction of the particle's movement toward both pBest and gBest. Particles will continue to move until they reach an optimal position or until their movements become minimal, indicating convergence.

c. Control Parameter

There are three main control parameters for PSO: the cognitive coefficient (c_1), the social coefficient (c_2), and the inertia weight (ω) [9], [10]. PSO is highly sensitive to these three control parameters, which significantly affect the algorithm's performance in finding solutions [2].

1) Cognitive Coeficient (c_1)

The parameter c1 determines the extent to which the individual best position (pBest) influences the movement of the particles. This parameter reflects the tendency of particles to follow their own successful paths. Setting a value of c_1 that is too high can cause particles to focus excessively on achieving pBest, thereby reducing their exploration capability within the solution space. Conversely, if c_1 is set too low, particles may not be sufficiently guided by their own experiences and may become overly reliant on the global best position (gBest), which can limit the discovery of new solutions. Therefore, the value of c_1 is typically set in the range of 1.5 to 2.0, providing an optimal balance between exploration and exploitation. Adjusting the value of c_1 through experimentation can help identify the optimal value for specific problems.

2) Social Coeficient (c_2)

The parameter c_2 functions to control the extent to which the global best position (gBest) influences the movement of particles, reflecting their tendency to follow the collective success of the swarm. A high value of c_2 will encourage particles to focus more on moving toward gBest, which can accelerate convergence to the optimal solution. However, if the value of c_2 is set too high, it may lead to a loss of diversity among the particles and cause them to become trapped in local solutions. Conversely, if c_2 is set too low, particles may be insufficiently influenced by the swarm's achievements and may not move quickly enough toward gBest, thereby slowing down the search process. Therefore, c_2 is typically set in the range of 1.5 to 2.0.

3) Inertia Weight (ω)

The inertia weight ω regulates the influence of a particle's previous velocity on its current velocity, functioning to balance exploration and exploitation. A high value of ω allows particles to maintain their previous speed, which can enhance exploration and broaden the coverage of the search space. Conversely, a low value of ω makes particles more responsive to pBest and gBest, accelerating the exploitation process. Typically, the value of ω is set in the range of 0.4 to 0.9.

COMPARISON TO OTHER ALGORITHMS

Every algorithm has its own advantages and disadvantages and is best suited for specific types of problems. An optimization algorithm is considered efficient if it can reach the global minimum with fewer iterations [11]. Here are some comparisons of PSO with other algorithms.

a. PSO vs Firefly Algorithm (FA)

PSO and FA were tested to solve noisy nonlinear optimization problems, and the results showed that PSO performed better in terms of convergence speed. This phenomenon may be attributed to the influence of completely different random number generation used in the iterative process of the algorithm [12].

b. PSO vs Grey Wolf Optimization

The battery autonomy testing using PSO and GWO in photovoltaic systems for solar panels aimed to maximize output, even in shaded areas. The tests showed that both algorithms yielded similar results and did not differ significantly during trials. However, PSO outperformed GWO in the time taken to reach the maximum power point [13].

c. PSO vs Bee Colony Optimization vs Bat Algorithm

In the conducted tests, PSO demonstrated its ability to solve complex problems and produce optimal solutions, particularly in the optimization of membership functions in fuzzy controllers. Based on comparisons of various PSO variants, combination of PSO and interval type-2 fuzzy system (IT2FS) variant, which utilizes IT2FS for dynamic parameter adaptation, exhibited superior performance compared to all other PSO variants and showed better performance than both BA and BCO [14].

APPLICATIONS

a. Non Linear Mathematical Model

Mathematical model functions sometimes have disturbances (noise). Therefore, optimization is needed for every mathematical model to find optimal results. Mathematical models are tried for optimization like [12]:

- Four peak function
- Parabolic function
- Camelback function
- Rastrigin function

The results of the tests indicated that, despite the presence of noise in the mathematical models, PSO was able to find optimal results for each mathematical model, with varying processing times and numbers of iterations depending on the specific problem.

b. Trajectory of autonomus mobile robot

The research aims to implement trajectory optimization for autonomous robots using PSO, BA, or BCO methods combined with modified fuzzy controllers (T1FS and IT2FS). The findings highlight that the integration of the PSO method with IT2FS is highly effective in addressing complex problems and achieving optimal solutions, particularly by enhancing the membership function optimization in fuzzy controllers, outperforming other methods. [14].

c. Main Engine

The pollution generated by ships has reached a significant level, primarily from their main engines, which emit harmful gases that contribute to global warming and climate change. To address this issue, many modern ships have switched to dual-fuel engines to reduce their reliance on diesel fuel. In efforts to lower emissions, Combination of artificial neural network and PSO algorithm plays a crucial role in optimizing emission parameters and engine performance [15].

PSO can be implemented by optimizing various operational parameters, such as the fuel mixture ratio and engine speed, which affect the emissions produced by the engine. With PSO, the system can achieve optimal engine performance, where the power output remains maximized while minimizing exhaust emissions to meet strict environmental standards.

CONCLUSION

PSO can be applied to optimized across various filed, including mathematical model optimization, trajectory planning for autonomus mobile robots, and emission control for main engines. Through various tests, PSO has demonstrated advantages over other algorithms such as FA and GWO, especially in terms of convergence speed and its ability to handle nonlinear problems with noise. Moreover, PSO is highly effective at solving complex problems and achieving optimal solutions, where its performance is superior compared to BA and BCO when applied on trajectory planning for autonomus mobile robots. Overall, PSO is an efficient and versatile algorithm with significant potential for adaptation in more complex optimization contexts. Future research can focus on developing more adaptive PSO variants and applying them to multi-objective optimization and optimize decision maker based on big data.

REFERENCES

[1] J. Del Ser et al., “Bio-inspired computation: Where we stand and what’s next,” Swarm Evol. Comput., vol. 48, no. April, pp. 220–250, 2019, doi: 10.1016/j.swevo.2019.04.008.

[2] M. Isiet and M. Gadala, “Sensitivity analysis of control parameters in particle swarm optimization,” J. Comput. Sci., vol. 41, p. 101086, 2020, doi: 10.1016/j.jocs.2020.101086.

[3] J. Kennedy and R. Eberhart, “Particle Swarm Optimization,” Nat. Comput. Ser., pp. 105–111, 1995, doi: 10.1007/978-3-031-17922-8_4.

[4] M. Isiet and M. Gadala, “Self-adapting control parameters in particle swarm optimization,” Appl. Soft Comput. J., vol. 83, p. 105653, 2019, doi: 10.1016/j.asoc.2019.105653.
[5] B. Ji, X. Lu, G. Sun, W. Zhang, J. Li, and Y. Xiao, “Bio-Inspired Feature Selection: An Improved Binary Particle Swarm Optimization Approach,” IEEE Access, vol. 8, pp. 85989–86002, 2020, doi: 10.1109/ACCESS.2020.2992752.
[6] R. Eberhart and J. Kennedy, “New optimizer using particle swarm theory,” Proc. Int. Symp. Micro Mach. Hum. Sci., pp. 39–43, 1995, doi: 10.1109/mhs.1995.494215.
[7] D. Freitas, L. G. Lopes, and F. Morgado-Dias, “Particle Swarm Optimisation: A historical review up to the current developments,” Entropy, vol. 22, no. 3, pp. 1–36, 2020, doi: 10.3390/E22030362.
[8] M. R. Bonyadi and Z. Michalewicz, “Particle swarm optimization for single objective continuous space problems: A review,” Evol. Comput., vol. 25, no. 1, pp. 1–54, 2017, doi: 10.1162/EVCO_r_00180.
[9] J. Wu, C. Song, C. Fan, A. Hawbani, L. Zhao, and X. Sun, “DENPSO: A Distance Evolution Nonlinear PSO Algorithm for Energy-Efficient Path Planning in 3D UASNs,” IEEE Access, vol. 7, pp. 105514–105530, 2019, doi: 10.1109/ACCESS.2019.2932148.
[10] K. R. Harrison, B. M. Ombuki-Berman, and A. P. Engelbrecht, An analysis of control parameter importance in the particle swarm optimization algorithm, vol. 11655 LNCS. Springer International Publishing, 2019. doi: 10.1007/978-3-030-26369-0_9.
[11] A. Gupta and S. Srivastava, “Comparative Analysis of Ant Colony and Particle Swarm Optimization Algorithms for Distance Optimization,” Procedia Comput. Sci., vol. 173, no. 2019, pp. 245–253, 2020, doi: 10.1016/j.procs.2020.06.029.
[12] S. K. Pal, C. . Rai, and A. P. Singh, “Comparative Study of Firefly Algorithm and Particle Swarm Optimization for Noisy Non-Linear Optimization Problems,” Int. J. Intell. Syst. Appl., vol. 4, no. 10, pp. 50–57, 2012, doi: 10.5815/ijisa.2012.10.06.
[13] H. Kraiem, F. Aymen, L. Yahya, A. Triviño, M. Alharthi, and S. S. M. Ghoneim, “A comparison between particle swarm and grey wolf optimization algorithms for improving the battery autonomy in a photovoltaic system,” Appl. Sci., vol. 11, no. 16, 2021, doi: 10.3390/app11167732.
[14] F. Olivas, L. Amador-Angulo, J. Perez, C. Caraveo, F. Valdez, and O. Castillo, “Comparative study of type-2 fuzzy Particle swarm, Bee Colony and Bat Algorithms in optimization of fuzzy controllers,” Algorithms, vol. 10, no. 3, 2017, doi: 10.3390/a10030101.
[15] C. Ma, C. Yao, E. Z. Song, and S. L. Ding, “Prediction and optimization of dual-fuel marine engine emissions and performance using combined ANN with PSO algorithms,” Int. J. Engine Res., vol. 23, no. 4, pp. 560–576, 2022, doi: 10.1177/1468087421990476.