
IB-PSO: an inversed Butterworth particle swarm optimisation algorithm for multi-agent path planning

Ferial Laassami*, Mohammed El Habib Souidi
and Abdeldjalil Ledmi

ICOSI Lab,
Department of Computer Science,
University of Khenchela,
40004 Khenchela, Algeria
Email: ferial.laassami@univ-khenchela.dz
Email: souidi.mohammed@univ-khenchela.dz
Email: abdeldjalil.ledmi@univ-khenchela.dz
*Corresponding author

Abstract: Path planning has an immense impact on the agents' coordination. This problem becomes more challenging to solve due to the various constraints. Recently, many optimisation techniques were proposed to solve the multi-agent path planning. However, among the most reliable techniques are those based on metaheuristic principles. In this paper, we propose a novel multi-agent path planning based on an enhanced version of the particle swarm optimisation (PSO). The inertia weight is a PSO parameter allowing the control of the exploration and the exploitation of the search space. Therefore, we propose a new inertia weight calculation method known as the inversed Butterworth particle swarm optimisation (IBPSO). The main objective of this approach is to provide an adequate transition between the exploitation and the exploration of the search space. To reflect the feasibility of this extension, we compared it with different PSO versions. The showcased results prove the efficiency of the proposed approach.

Keywords: metaheuristic; multi agent systems; path planning; particle swarm optimisation; inertia weight.

Reference to this paper should be made as follows: Laassami, F., Souidi, M.E.H. and Ledmi, A. (2024) 'IB-PSO: an inversed Butterworth particle swarm optimisation algorithm for multi-agent path planning', *Int. J. Systems, Control and Communications*, Vol. 15, No. 4, pp.387–410.

Biographical notes: Ferial Laassami obtained her Master's in Computer Science from the University of Khenchela, Algeria in 2020. Since March 2023, she is a PhD student at the University of Khenchela, Algeria. Her main areas of research include distributed artificial intelligence, multi-agent systems, path-planning, and metaheuristics.

Mohammed El Habib Souidi is Lecturer in the Department of Mathematics and Computer Science, Faculty of Sciences and Technology, University of Khenchela, Algeria. He received his PhD in Computer Science from the Harbin Institute of Technology (China), in 2017. His main areas of research include multi-agent task coordination, reinforcement learning, game theory and path planning.

Abdeldjalil Ledmi received his Engineering degree in Computer Science from the University of Khenchela, Algeria in 2010. He also received his Master's in Computer Science from the same university in 2014, and PhD in Computer Science from the Larbi Tebessi University, Algeria in 2019. He is working as Lecturer in the Department of Mathematics and Computer Science at the University of Khenchela, Algeria. Moreover, he is affiliated as a researcher in ICOSI Lab.

1 Introduction

Multi-agent system MAS is considered as one of the major research topics in artificial intelligence. MAS can model knowledge, be consistent with human behaviour and make decisions by reasoning through a set of intelligent entities known as agents. These entities interact with each other in distributed way to solve complex problems, which cannot be solved by a single entity. The main goal of the distributed artificial intelligence is to model the interactions between the intelligent agents by allowing them to exchange data and cooperate in different coalitions (Ferber, 1999). In MAS, the cooperative aspect of the agents reflects the decentralisation of the problem resolution. MASs have usually been studied as a part of artificial intelligence. Furthermore, they are also used to learn and predict new behaviours, such as the performance of natural systems (Ferber, n.d.). The most common applications of MAS are often related to manufacturing systems (González-Briones et al., 2018) traffic and transportation (Burmeister et al., 1997), pursuit evasion (McArthur et al., 2007) and optimisation problems.

In MAS, the agents can cooperate with each other and share a common purpose in heterogeneous environments. Unlike single systems, MASs have high efficiency due to the use of several mechanisms, such as negotiation, coordination, and cooperation. On the one hand, MAS negotiation can be defined as the interaction that enables groups of agents to conclude a mutual agreement regarding some belief, goal, or plan (Beer et al., 1999). On the other hand, MAS coordination can be seen as the process enabling various agents, each with different tasks, to make effective decisions and collaborate in a synchronised manner to achieve common objectives. According to Doran et al. (1997), cooperative agents must achieve not only their own goals, but also those of the other agents to achieve the common objective. In Ferber (1999), the authors concluded that the cooperation exists when the addition of new agents improves the situation of the system. Specifically, the agents share information or data, exchange skills or tasks, and collaborate as a group.

Path planning is considered as a problem that involves finding the future location of multiple agents and locating a path free of collisions in the shortest period possible. These agents must find the shortest path from the starting point to the final point and go to their desired positions (Rafai et al., 2022). In relation to robotics, path planning is considered as a basic task of robot navigation, especially for autonomous robots (Cruz and Yu, 2017). Since agents navigate independently, using interactions and coordination, MAS path planning becomes much more crucial compared to single agent path planning. Furthermore, the relevance of path planning in autonomous systems makes research on the topic is still ongoing (Wahab et al., 2020; Alam et al., 2015).

The challenge of multi-agent path planning becomes more difficult when the number of agents is significant. Metaheuristic techniques are hugely used to solve complex optimisation problems such as, path planning (Sood and Panchal, 2020). According to Glover and Kochenberger (2003), metaheuristics are considered as approaches that orchestrate an interaction between local improvement procedures and higher level strategies, to create a process capable of escaping local optima and performing a robust search of a solution space. These methods are used when the problem cannot be solved using traditional methods known as heuristics (Nesmachnow, 2014). The main objective of metaheuristics is to approximate the optimal solution by maximising or minimising the objective function according to the processed problem.

Particle swarm optimisation (PSO) algorithm is considered as a nature-inspired and a social swarm-based metaheuristic approach. PSO is based on a swarm of particles that update their positions and velocities in order to seek for optimal solutions. Precisely, PSO simulates a population of agent interacting with each other and with their environment to achieve a collective goal (Altshuler, 2023). During the algorithm's iterations, the group of particles must adjust their positions to find the best solution. In other words, each particle tries to update its position in relation to its personal best solution and the global best solution found in the search space. The strength of the PSO algorithm lies in its ability to combine and balance the exploration of the search space and the exploitation of the discovered solutions. Moreover, there exist several research activities aiming to balance the exploration and the exploitation concepts in PSO (Ghalia, 2008; Hussain et al., 2019). In relation to MAS, PSO is usually used in order to provide a task coordination mechanism (Raposo and Fuks, 2002) or path planning algorithm (Alam et al., 2015).

Inspired by the earlier research activities, the main objective of this paper is to propose a new MAS path planning based on an inversed Butterworth inertia weight (IW) in the PSO algorithm, known as IB-PSO. The aim of the proposed algorithm is to provide a novel inertia calculation that provides an adequate balance between the exploration and the exploitation of the search space. The main goals of this paper can be summarised as follows:

- The proposition of a MAS path planning algorithm based on an extension of PSO algorithm.
- The introduction of a new IW calculation method aiming to balance the exploration and the exploitation of the search space.
- The comparison of the proposed PSO version with recent approaches aiming to control the IW parameter through their application to path planning problem.

The paper is organised as follows: in Section 1, we introduce the processed problem and key concepts aiming to provide a clear comprehension of the proposed solution. In Section 2, we discuss the recent research activities regarding the applications of metaheuristics in MAS by focusing on the PSO application. Section 3 provides descriptions regarding the MAS path planning and the PSO algorithm. Section 4 reflects a detailed description of the proposed path planning algorithm. In Section 5, we analyse the main obtained simulation results in comparison with related approaches. Finally, the conclusion in Section 6 summarises the main contributions of this paper.

2 Related works

The aim of this section is to provide an analysis of the previous research activities related to the proposed work in this paper. Specifically, we will take into consideration the most important and recent MAS research activities based on metaheuristic approaches.

There exist diverse applications of metaheuristic algorithms for solving optimisation problems in real-life systems with complex configurations. In Lopes Silva et al. (2018), the authors proposed a multi-agent model based on metaheuristics via reusable codes (MOF for metaheuristic optimisation framework), facilitating the development of applications for solving optimisation problems. The study focused on the traits of hybridisation, collaboration and the integration of multi-agent systems. Building upon the work proposed in Lopes Silva et al. (2019), the authors introduced a different MAS optimisation approach for routing and scheduling, comparing various methods. In the same context, a recent research using metaheuristics to address the same problem was proposed by Kamali et al. (2023). Precisely, the authors presented a novel approach to enhance scheduling efficiency by integrating immune-based concepts with MAS in order to contribute in the growing field of optimisation solutions.

Concerning MAS task coordination, metaheuristics are usually proposed. In this kind of problem, each agent has a local search space in order to collaborate and find a global solution. For example, in Kaur and Saini (2021), the educational timetable problem was solved with metaheuristic methods based on distributed MAS. Moreover, the authors discussed the challenges and advancements in this field, while summarising the used metaheuristic techniques.

Swarm intelligence techniques are often used in combination with MAS in order to solve different problems. In Kusuma and Novianty (2023), the authors introduced a novel metaheuristic known as the total interaction algorithm (TIA). This method is a combination of five swarm intelligence algorithms developed through the coordination their competencies. Specifically, TIA depends on the agents' interactions to find the best answers to the processed complex problems.

Recently, the PSO algorithm has been applied to several MASs-based problems. In Nouri et al. (2018), the authors proposed a PSO-solving method based on distributed systems, by proving its effectiveness in solving the flexible job-shop scheduling problems (FJSP). The main objective of the discussed approach is to manage a large number of entities by using a multi-objective particle swarm optimisation method (MOPSO). In addition, Dai et al. (2019) solved a smart electric vehicles charging station problem by using solar energy. In other words, the authors have based on physical agents and the application of MAS-PSO with the aim of optimising electricity and energy consumption. In another context (Souidi et al., 2023), the authors solved the multi-pursuers multi-evader game (MPMEG) via the use of a discrete approach of the PSO algorithm. The objective of the discussed study is to enhance the pursuers' reward acquisition, which is a measure of their learning rate during the task execution. Moreover, the authors tried to find a dynamic pursuers' grouping by comparing the results obtained with recent approaches. In another work (Khan et al., 2017), the authors introduced a combination of the PSO algorithm with other methods, which are the ant colony optimisation (ACO) and K-opt operation. In the sense that the role of the PSO comes after ACO, which is providing the swarm a sufficient number of times to find an optimal path in comparison with the existing algorithms to address the travelling salesman problem.

MAS path planning problems have been hugely processed by employing several PSO approaches. In this context, Biswas et al. (2017) used a new vectorised PSO extension (VPSO), for autonomous particles path planning and obstacle avoidance. Furthermore, the authors proved that their proposed optimisation algorithm provides a safe navigation ability by guarantying a collision free path. In addition, the unmanned aerial vehicles (UAVs) path-planning problem is becoming a very important research area. In this kind of problems, the goal is to optimise the flight trajectory of drones. In Shao et al. (2019), the authors introduced a coordination mechanism for multiple UAVs to meet at a location in space and time using distributed cooperative particle swarm optimisation (DCPSO). Specifically, the discussed proposal focused on the study of the algorithm initialisation, the update of velocities and positions, and the fitness function. In addition, they also provided an evaluation of the time complexity as well as simulation results for both 2-D and 3-D formation.

In relation to the proposed approach in this paper, several works have based on the modification of the IW calculation in order to improve the PSO efficiency. Nayeem et al. (2021) focused on enhancing the PSO algorithm for UAV path planning by providing a comparative analysis of different versions of IW that have been proposed, such as constant value, time-varying, single variable, and multivariable IW. The authors also illustrated which inertia calculation approach produces the best UAV path planning among all algorithms and strike a better balance between exploration and exploitation parameters. Moreover in Li et al. (2020), an improved PSO (IPSO) was proposed with the aim of finding the best path with collision-free using the minimal number of iterations possible and providing a balance between local and global optimisation skills. Also, the authors have based on uniform initialisation of the particles' positions in the search space, exponential attenuation IW by using an exponential function, cubic spline interpolation function, and learning factor of enhanced control aiming to decrease the cosine function of IW. The results were compared with classical PSO. Additionally, two other relevant works deal with the different kinds of IW in Han et al. (2010) and Kumar et al. (2018). The first approach (Han et al., 2010) proved that various IW strategies, such as simulated annealing IW and linearly decreasing IW can enhance PSO's convergence speed and search performance through their application to the path-planning problem. The second approach (Kumar et al., 2018) focused on the resolution of the economic load dispatch problem by applying the different IW functions.

Other research activities have focused on the increased IW calculation, such as Bansal et al. (2011) and Taherkhani and Safabakhsh (2016). In Bansal et al. (2011), Conducts a comparative analysis of 15 IW strategies applied to five optimisation problem, such as linearly decreasing IW PSO, fuzzy adaptive IW PSO, and random number IW PSO. Also in Taherkhani and Safabakhsh (2016), the proposed algorithm, known as adaptive inertia weight PSO (SAIW-PSO), achieves relevant performance and outperforms other PSO extensions. Moreover, the authors described how increasing IW can be advantageous. Also, Zheng et al. (2003) proved that an increasing IW outperforms a decreasing approach in some case and according to the processed problem.

3 Problem description

In this section, we describe the principles of the PSO algorithm, its different equations in Subsection 3.1. Moreover, we introduce the path-planning problem in Subsection 3.2 and describe how agents change their locations.

3.1 PSO description

There exist a wide range of optimisation problems, from simple optimisation problems to more challenging ones. In many fields, optimisation is considered as essential process aiming to improve the solution and to reduce the costs, especially for complex problems. In other words, optimisation concerns the process aiming to find the best solution among a set of feasible solutions. The main objective of the optimisation is to determine the maximum or minimum value that the objective function may achieve while adhering to a set of constraints or limitations, as shown in equation (1):

$$\begin{aligned} &\text{Minimise or Maximise } f(x), x \in A \text{ and } x = (x_1, x_2, x_N) \\ &\text{Subject to } g(x) \leq 0 \text{ and } h(x) = 0 \end{aligned} \quad (1)$$

where $f(x)$ is the objective function; A is a set of findable solutions, which can be discrete (specific elements) or continuous (numbers inside a range); $g(x)$ are the inequality constraints; $h(x)$ are the equality constraints; N is the problem dimension size.

Recently, metaheuristic approaches are often used as optimisation methods in complex problems. PSO is considered a metaheuristic approach aiming to provide an equilibrium between the exploration and the exploitation of the search space. The PSO algorithm (Kennedy and Eberhart 1995) was inspired by the social behaviour of bird flock. This method is a population-based optimisation technique based on a population of particles. Each particle represents an agent in MAS. The main goal of the particles is to reach the best possible solution in the search space. The particles update their positions $X_i = (x_1, x_2, \dots, x_N)$ in each iteration and navigate in their shared environment with the velocity $V_i = (v_1, v_2, \dots, v_N)$ established by their own experience or the experiences of the swarm's neighbours. The velocity describes how the particle updates its position during each iteration. In PSO, the particle's velocity update provides a certain balance between the personal best solution found by the concerned particle and the global best position found in the swarm. A high velocity value reinforces exploration of the search space. However, a low velocity value reinforces exploitation process. The particle's position is updated during each iteration as follows (Gupta and Srivastava, 2020):

$$x(t+1) = x(t) + v(t+1) \quad (2)$$

where x is the position and v is the velocity of the particle at time t .

The velocity is calculated as follows (Gupta and Srivastava, 2020):

$$v(t+1) = wv(t) + c_1r_1(pbest(t) - x(t)) + c_2r_2(gbest(t) - x(t)) \quad (3)$$

where $pBest$ (personal best) refers to the best solution or position found by the particle, it is updated through a comparison with the new particle's position at each iteration by taking into account the fitness values. $gBest$ (global best) refers to the best solution found by the group of particles. $gBest$ position refers to the position with the highest fitness

value found until the current iteration. $c1$ and $c2$ are the acceleration coefficients. $r1$ and $r2$ are stochastic random constants between 0 and 1; w is the IW aiming to control the exploration and the exploitation of the PSO algorithm.

Knowing that the main objective of this paper is to propose a new IW calculation, we have seen the usefulness to discuss the main IW calculation methods of the literature review. The IW can be represented by a constant value or adjusted dynamically during the execution process. IW aims to measure how the previous velocity of the particle impacts its current velocity. This parameter also controls the balance between exploration and exploitation of the search space. On the one hand, a high IW value favors the exploration of the search space. On the other hand, a low IW value favors the exploitation process. Usually, the IW values used are in the range of 0.4 to 0.9.

Figure 1 Particle motions in PSO algorithm (see online version for colours)



In order to enhance the PSO algorithm, different extensions based on the IW control were proposed:

- Constant IW (Han et al., 2010; Zhu and Wang, 2018): in this PSO extension, the IW is considered as static during the algorithm's iterations. The IW is calculated as follows:

$$w = \frac{1}{2 * \ln(2)} = 0.729 \quad (4)$$

- Linearly decreasing IW (Kumar et al., 2018; Bansal et al., 2011; Zhu and Wang, 2018): in this version, the IW value linearly decreases during the execution from $w_{\max}$ to $w_{\min}$ by taking into consideration the maximum number of iterations ($MaxIter$):

$$w_t = w_{\max} - \frac{w_{\max} - w_{\min}}{MaxIter} * t \quad (5)$$

where: $w_{\min} = 0.4$, $w_{\max} = 0.9$, t : the current iteration.

- Natural exponent strategy 1 (e1-PSO) IW (Kumar et al., 2018): in this case, the IW decreases in a nonlinear way according to the following equation:

$$w_t = w_{\min} + (w_{\max} - w_{\min}) * e^{-[t/(MaxIter/10)]} \quad (6)$$

where: $w_{\min} = 0.4$ and $w_{\max} = 0.9$.

- Natural exponent strategy 2 (e2-PSO) IW (Kumar et al., 2018): this version has the same principles of e1-PSO, however, the IW is differently calculated as follows:

$$w_t = w_{\min} + (w_{\max} - w_{\min}) * e^{-[t/(MaxIter/4)]^2} \quad (7)$$

where: $w_{\min} = 0.4$, $w_{\max} = 0.9$.

- Simulated annealing IW (Kumar et al., 2018): this decreasing version takes into consideration a constant value (λ) instead of the maximum number of iterations. The IW is calculated during each iteration as follows:

$$w_t = w_{\min} + [(w_{\max} - w_{\min}) * \lambda^{-1}] \quad (8)$$

where: $\lambda = 0.95$.

- Standard IW (Jain et al., 2022; Marini and Walczak, 2015): this version refers to the original PSO algorithm, in which the IW is assumed to be neutral (IW = 1).
- Butterworth IW (Zhu and Wang, 2018): this nonlinear decreasing IW version is inspired by the curve of Butterworth filter in electrical engineering. This type of filter is often used in signal processing (Zhu and Wang, 2018). The Butterworth IW is calculated as follows:

$$w_t = w_{\max} * \frac{1}{1 + \left(\frac{t}{p1}\right)^{p2}} + w_{\min} \quad (9)$$

where $p1 = \frac{MaxIter}{3}$, $p2 = 10$, $w_{\max} = 0.5$; $w_{\min} = 0.4$.

- Linearly increasing IW (Bansal et al., 2011; Taherkhani and Safabakhsh, 2016; Zheng et al., 2003): in this case, the IW value linearly increases in relation to the algorithm's iterations. IW is calculated as follows:

$$w_t = w_{\min} - \frac{w_{\min} - w_{\max}}{MaxIter} * t \quad (10)$$

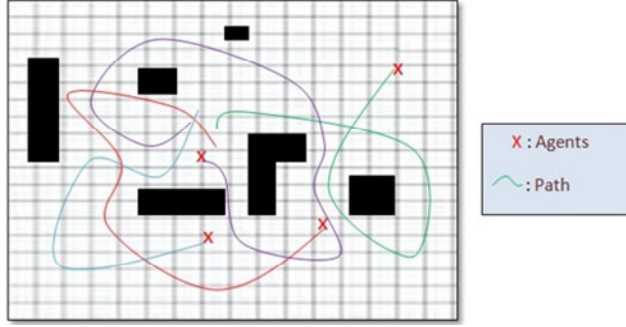
where: $w_{\min} = 0.4$, and $w_{\max} = 0.9$.

3.2 Collaborative path planning description

Collaborative path planning is a method used in multi-agent systems in order to coordinate the movements of the agents in an unknown and shared area, as shown in Figure 2. The agents need to adapt their paths to avoid collisions with each other as well as with the existing obstacles. In other words, the agents must synchronise their trajectories through the use of a path planning approach. MAS path planning can be used in centralised and decentralised way. On the one hand, centralised path planning refers to a method of defining optimal paths for multiple agents in a centralised way, where a single agent has complete information regarding the environment and the other agents. This agent, often known as a planner or a controller, provides the paths for the existing agents simultaneously, taking into consideration different constraints and objectives. On

the other hand, decentralised path planning is an approach where each agent makes its path planning decisions independently, without relying on a central planner. Specifically, each agent has access to local information regarding its environment, such as sensor data or through the communication with nearby agents. The agent utilises this information to plan its own path while considering its own objectives and constraints.

Figure 2 Path planning in MAS with obstacle avoiding (see online version for colours)



The primary goal of multi-agent path planning is to determine the most efficient trajectory for every agent in the environment. To address this issue, PSO is used in order to represent each agent as a particle that indicates possible solutions through its dynamic positions. In other words, each particle (agent) iteratively updates its position based on its Cartesian coordinates and according to the PSO principles as follows:

$$v_x(t+1) = wv(t) + c_1r_1(pBest(t) - x(t)) + c_2r_2(gBest(t) - x(t)) \quad (3)$$

$$v_y(t+1) = wv(t) + c_1r_1(pBest(t) - y(t)) + c_2r_2(gBest(t) - y(t)) \quad (3)$$

$$(x(t+1), y(t+1)) = \begin{cases} x(t+1) = x(t) + v_x(t+1) \\ y(t+1) = y(t) + v_y(t+1) \end{cases} \quad (11)$$

(x, y) : the Cartesian coordinates of the concerned agent.

In order to move to its new environment position, the agent uses the magnitude of its velocity as follows:

$$Move-to(x, y) = \sqrt{v_x \times v_x + v_y \times v_y} \quad (12)$$

The PSO takes the fitness score as a measure of a path's quality. Specifically, each position in the environment is assigned to a fitness value in the range of 0 to 1. The main objective of the PSO path planning is to allow the agents to achieve the positions with the highest fitness degree. Moreover, we note that only one environment position returns the maximum fitness degree. As a first step, a random fitness value in the range of 0 to 1 is generated in each environment position. After that, the fitness degree (f) of each environment position is updated as follows:

$$\begin{aligned} \theta &= \text{Maximise Val}(x_i) \\ f(x_i) &= \max_fitness * (x_i - \min_val) / (\max_val - \min_val) \end{aligned} \quad (13)$$

where:

Θ the objective function.

max_fitness it is used to normalise the position value between 0 and 1 (*max_fitness* = 1).

min_val the minimum fitness value randomly generated in the environment.

Max_val the maximum fitness value randomly generated in the environment.

4 The proposed algorithm

In this section, we introduce the proposed MAS path planning based on the new PSO version. Specifically, we detail how this approach is used in order to allow the agents to achieve the objective state in minimum time.

Algorithm 1 An inversed Butterworth PSO algorithm for multi-agent path planning

```

Inputs: Found-Target = false;
setup-environment;
Initialise (population-size);
Initialise (target-position);
Initialise (number-iteration-max);
Initialise (IW, c1, r1, c2, r2);
Foreach Agenti do
    Initialise (positioni);
    Initialise (velocityi);
    PBesti ← positioni;
End for;
searching-time ← 0;
target-value ← Evaluate-fitness(target-positioni);
Gbest ← Max (PBest);
while number-iteration ≠ number-iteration-max or Found-Target ≠ true do
    Foreach Agenti do
        fitness-valuei ← Evaluate-fitness(positioni);
        PBest-valuei ← Evaluate-fitness(PBesti);
        Communicate-with-the-environment(Agenti);
        If fitness-valuei = target-value then
            Found-Target ← true;
        Elseif fitness-valuei is better than PBest-valuei then
            Update PBesti;
            Gbest ← Max (Pbest);
        End if;
    If target-valuei = false and number-iteration ≠ number-iteration-max then
        Positioni ← Move-toward(Agenti, GBest);

```

```

    IW ← Calculate-inertia (IW); //using the new equation
    update-velocity(IW, positioni);
  End if;
End for;
searching-time ← searching-time + 1;
end while;
Outputs: searching-time, Found-Target;

```

The different algorithm steps can be explained as follows:

The main objective of this algorithm is to generate the trajectory for all agents in order to find the location of the global optimum, which is known as the location with the best fitness-value. At the beginning of the proposed path planning, the agents are created and placed randomly in the environment. In relation to the PSO principles, each agent represents a particle ($i \in [1, \text{population size}]$). Thus, the population size represents the number of existing agents. The population size is initialised and fixed at the beginning of each algorithm's episode. At each iteration, each agent updates its velocity according to their Cartesian coordinates as shown in equation (3). Known that, the initial velocity of the agents is assumed to be zero. According to the obtained velocity, the position of each agent in the environment is updated as detailed in equation (11). Moreover, we note that the constant parameters $c1$ and $c2$ used for calculating velocity are initialised at the beginning. However, the dynamic parameters $r1$, $r2$, as well as the IW are updated during each iteration.

During each iteration, IW is updated according to the proposed function known as the inversed Butterworth inertia weight (IBPSO), detailed in equation (14). The proposed IW calculation is inspired by the BPSO reflected in equation (9). However, we have adapted an inversed process in order to allow the exploitation of obtained solutions at the first iterations and the exploration of the search space in the last iterations. Specifically, in BPSO, the IW parameter decreases from 0.9 to 0.4 during the different iteration. On the other hand, in the proposed IBPSO, the IW parameter increases from 0.4 to 0.9 as follows:

$$W = w_{\max} \div \left(\frac{1}{1 + \left(\frac{p1}{t} \right)^{p2}} \right) + w_{\min} \quad (14)$$

where: $p1 = (\text{MaxIter})/3$, $p2 = 10$, $w_{\max} = 0.5$; $w_{\min} = 0.4$

Increasing the IW parameter during the algorithm's iterations improves the exploration of the search space. In the path planning case, the exploration principle allows the agents to move far away from their current positions to discover a broader range of possibilities. In Malik et al. (2007), the authors indicated that an increasing IW parameter in PSO reflects an aggressive movement towards the solution region. In addition, a high IW parameter allows the particles to escape problems related to the local minima. Also, increasing IW improves the PSO convergence by preventing the algorithm from deviating too far from the optimal direction. Furthermore, the strategy of increasing IW parameter is generally competitive and outperforms a decreasing IW one in terms of convergence speed and solution precision (Bansal et al., 2011; Zheng et al., 2003; Malik et al., 2007).

After the update of the agents' positions, the *PBest* position of each agent will be updated. In order to update its *PBest* position, each agent compares the fitness value of its current position with the fitness value of its *PBest* position according to equation (15). Knowing that, the initial position of each agent is considered as its initial *PBest* position. The *GBest* position is also updated after each iteration. The *GBest* position is chosen among the *PBest* positions found by the swarm. Precisely, the *Pbest* position with the highest fitness degree is considered as the new *GBest* position according to equation (16).

$$PBest_i = \begin{cases} PBest_i & \text{if } (PBest\text{-value}_i > \text{fitness-value}_i) \\ position_i & \text{else} \end{cases} \quad (15)$$

where:

- *PBest-value_i*: represents the fitness value associated to the *PBest_i* position.
- *position_i*: refers to the current position of the agent *i*.

$$GBest = \begin{cases} GBest & \text{if } (GBest\text{-value} > \text{Max}(Pbest\text{-value})) \\ \text{Max}(Pbest) & \text{else} \end{cases} \quad (16)$$

where:

- *GBest-value*: is the fitness value associated with the *GBest* position.
- *Max (PBest)*: returns the position associated with the maximum *PBest* obtained by the swarm.
- *Max (Pbest-value)*: the maximum value of *PBest* obtained by the swarm.

The algorithm will be executed until the stopping conditions are met. In other words, the algorithm will stop in the case of reaching the maximum number of iterations or finding the position with the highest fitness degree. At the end, the proposed algorithm returns the used number of iterations as well as if the position with the highest fitness degree was reached:

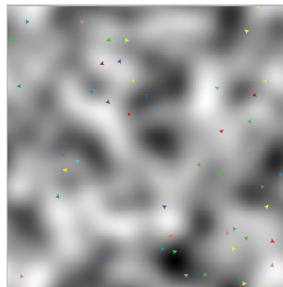
$$\text{found-target} = \begin{cases} \text{True} & \text{if } Gbest\text{-value} = 1 \\ \text{false} & \text{else} \end{cases} \quad (17)$$

5 Simulation results

In this section, we aim to compare the proposed path planning method with other approaches based on PSO principles. For this study, we employed the NetLogo (Tisue and Wilensky, 2004) Platform version 5.3.1 to achieve robust results. This platform facilitates the study of agent-based systems by employing a suite of visual tools. Moreover, NetLogo can model relationships and interactions within complex systems. In the proposed study, we implemented the different scenarios by considering each particle as an agent (an agent-based modelling approach). The studied case in this section is based on a population of 50 particles that must achieve a final optimal location by taking into account the number of used iterations. Noting that, each iteration represents a cycle of the proposed Algorithm 1. As shown in Figure 3, the initial state of the environment is a 2D

grid of 100×100 of environmental positions, known as patches. Each patch is characterised by its Cartesian coordinates as well as a fitness degree calculated according to equation 13. With the aim of providing a visual approximation of the patches' fitness degree, we used the patch colour provided by the NetLogo platform. Specifically, the brightness of the patch increases in connection with the increase of fitness degree, as shown in Figure 3. Knowing that, the agents' colours showed in Figure 3 are randomly chosen. During each iteration, each agent (particle) updates its Cartesian coordinates (position) according to equation 11. In the initial case, the agents' positions are randomly initialised. Also, the fitness degrees of the patches are updated from a simulation episode to another. The constant parameters $c1$ and $c2$ are fixed at the beginning of each simulation episode. However, the parameters $r1$ and $r2$ are randomly updated before each iteration.

Figure 3 A representation of the environment (see online version for colours)



Knowing that the proposed PSO version is based on the improvement of the IW parameter updated before each iteration according to equation (14), we have seen the usefulness of comparing it with improved PSO approaches based on the same parameter. In other words, the proposed approach is compared with eight approaches based on different equations for calculating IW parameters. The principles of the nine compared cases are well discussed in Sections 3 and 4 of this paper:

- Constant inertia weight (CIW) (Han et al., 2010; Zhu and Wang, 2018): in this case, the IW is during the different iterations ($IW = 0.729$).
- Linearly decreasing inertia weight (LDIW) (Kumar et al., 2018): in this case, the IW linearly reduces during iterations from 0.9 to 0.4 according to equation (5). LDIW prioritises the search space exploration before the exploitation of the obtained solutions.
- Natural exponent strategy 1 inertia weight (NE1IW) (Kumar et al., 2018): in this case, the value of IW nonlinearly decreases from 0.9 to 0.4 according to equation (6).
- Natural exponent strategy 2 inertia weight (NE2IW) (Kumar et al., 2018): this case has the same principle as the previous one (NE1IW), however, IW reduces according to equation (7).
- Simulated annealing inertia weight (SAIW) (Kumar et al., 2018): in this case, the IW decreases during the iterations according to equation (8).

- Butterworth inertia weight (BWIW) (Zhu and Wang, 2018): in this case, the IW parameter also decreases in a nonlinear way according to equation (9).
- Standard inertia weight (SIW) (Jain et al., 2022; Marini and Walczak, 2015): this case is based on the original PSO version, in which the IW parameter is considered as a neutral element ($IW = 1$).
- Linearly increasing inertia weight (LIIW) (Bansal et al., 2011; Taherkhani and Safabakhsh, 2016; Zheng et al., 2003): in this case, the IW parameter increases from 0.4 to 0.9 according to equation (10). LIIW aims to escape from local spaces in order to improve the search space exploration.
- Inversed Butterworth inertia weight (IBWIW): this case is based on the IW calculation proposed in this paper according to equation (14).

Figure 4 represents the development of IW values during the execution of 100 iterations of the proposed algorithm by using the compared IW calculation approaches. As we can constate, the IW values remain constant in CIW ($IW = 0.729$) and SIW ($IW = 1$) cases. However, in LDIW, NE1IW, NE2IW, SAIW, and IBWIW, the IW values decrease from 0.9 to 0.4. In LAIW and IBWIW, the IW values increase from 0.4 to 0.9.

Figure 5 represents the searching time obtained by the execution of 100 episodes of the proposed algorithm through the use of the compared IW methods. The searching time is reflected by the number of iterations effectuated by the particles with the aim of finding the target position. According to the reflected results, we can note that the 8 algorithms have outperformed the SIW case by lower values in terms of searching time average. The highest average searching time is 92.6 iterations, which is obtained via the use of the SIW parameter. Furthermore, we can note an important decrease of 44.51% in the average of BWIW (50.48 iterations) in relation to SIW case. In CIW, NE2IW, and SAIW cases the averages obtained were respectively 49.34, 49.05, and 49.47. In addition, we constate an approximation between NE1IW 43.12 and LDIW 45.97 cases. According to the shown results, we can note that the best average searching time 37.47 is obtained via the use of the proposed approach in this paper (IBWIW). Also, an interesting average 38.79 is obtained through the utilisation of the LLIW approach, which proves that the increasing IW methods have a positive impact on the MAS path planning problem.

Furthermore, a comprehensive analysis of the searching time obtained in 20 episodes is detailed in Figure 6. Knowing that in this case, each episode is composed of a maximum of 100 iterations. The searching time can be defined as the number of iterations generated by the particles with the aim of finding the goal position. In the case where the particles cannot reach the target at the end of an episode, the value 100 will be assigned as a result. The showcased results revealed that the SIW parameter significantly exhibits longer searching time in relation to the other studied cases, which is clearly represented by the bar chart in the figure. Across the majority of episodes, and in opposition to the other cases, the SIW based approach takes 100 iterations with an average of 91.65 without finding the target location. In the other hand, it is crucial to note that the IBWIW and LIIW take the shortest period with averages 37.47 and 38.79 respectively.

Figure 4 Different kinds of IW executed in 100 iterations

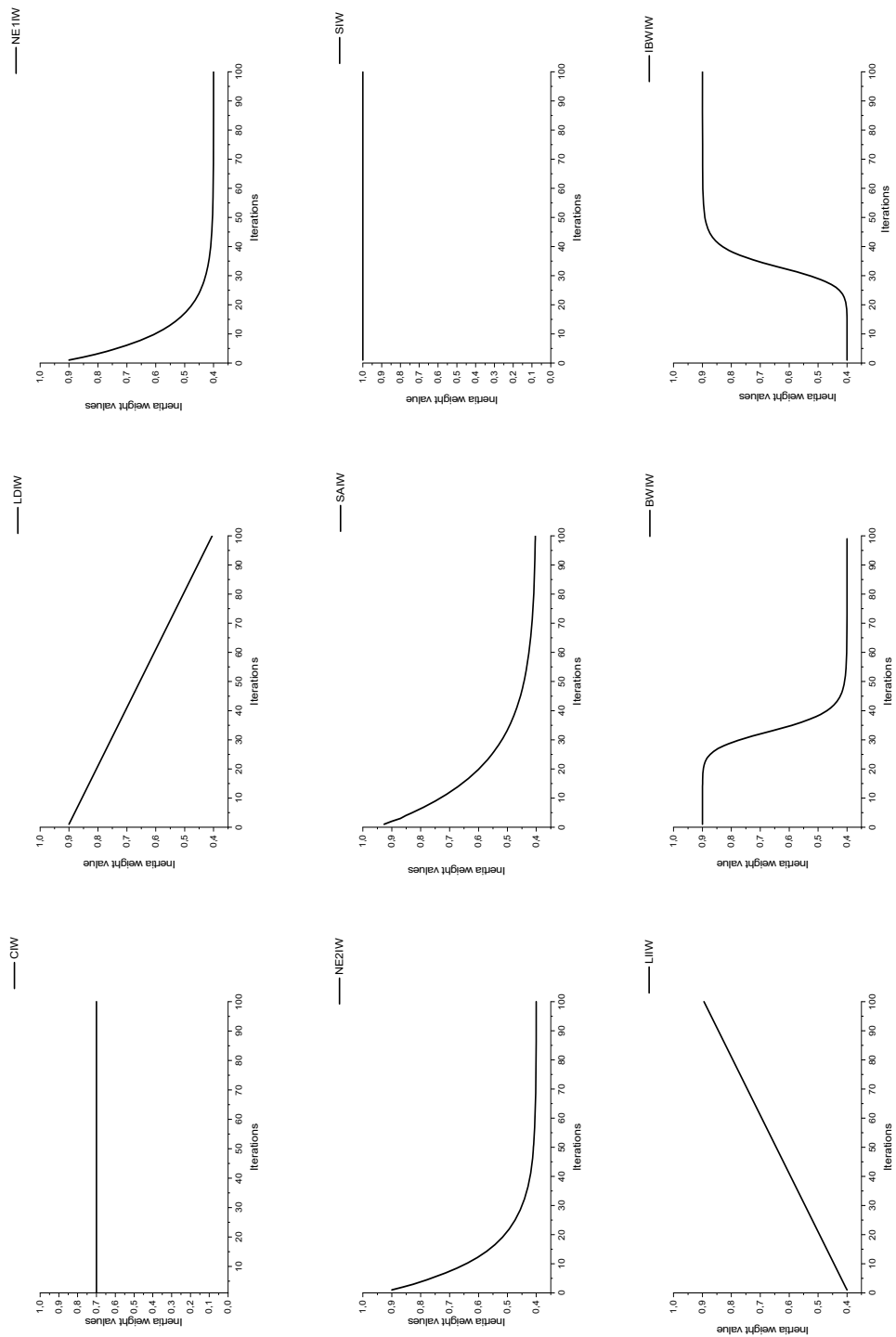


Figure 5 Average-searching time obtained (nine cases)

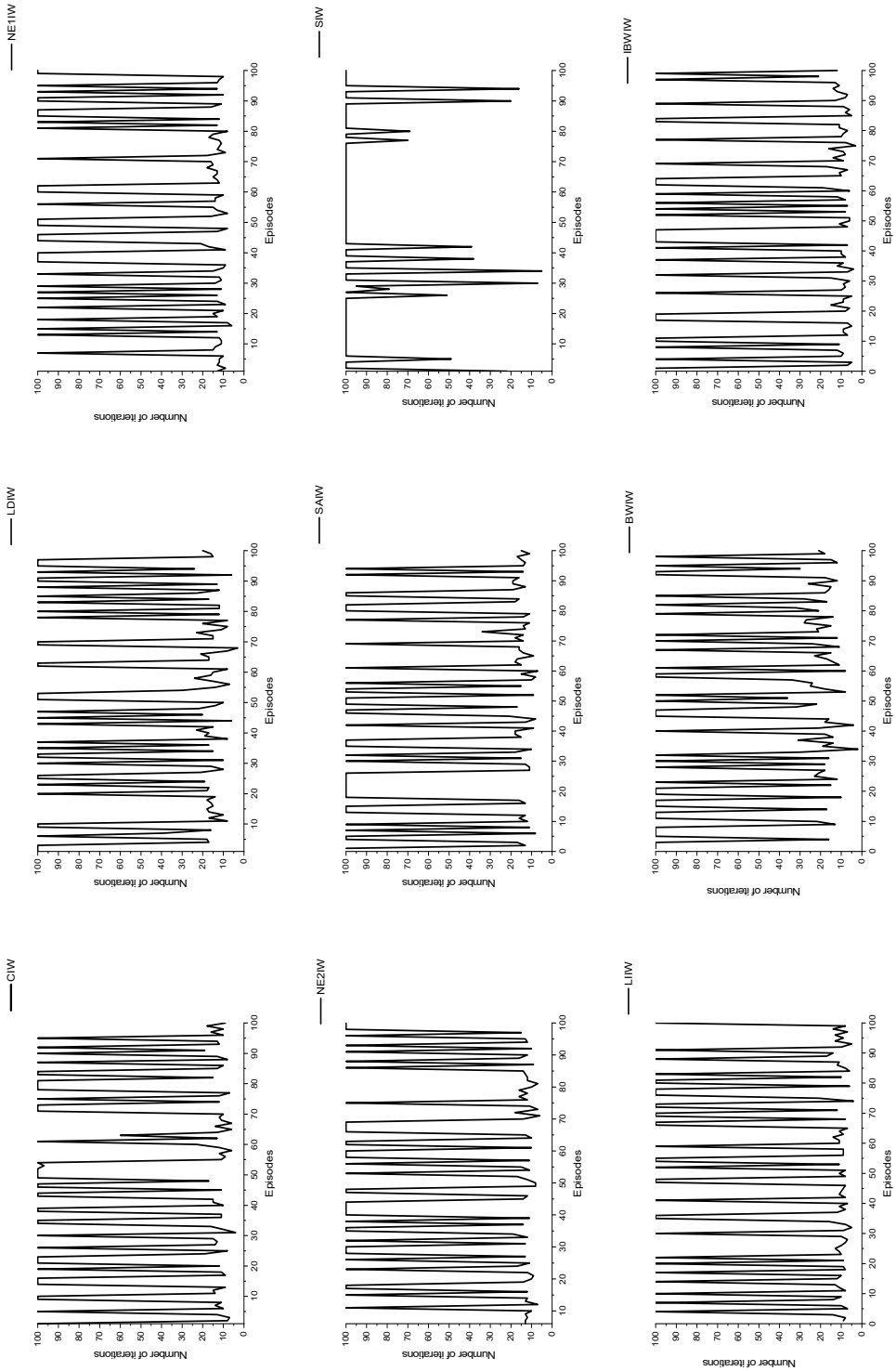


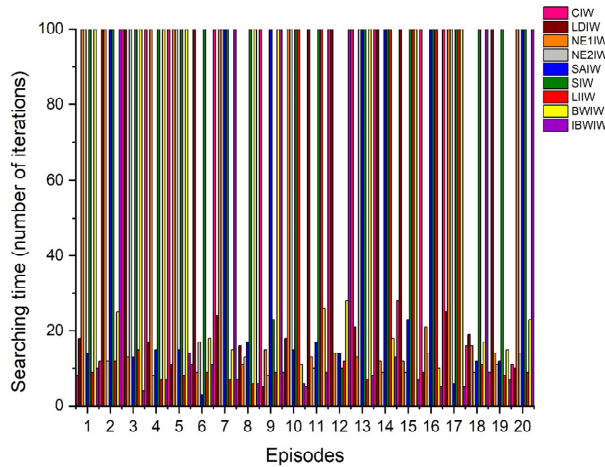
Figure 6 Average-searching time obtained in 20 episodes (see online version for colours)

Figure 7 represents the fitness development, which is indicated by the best value founded in a single episode. Each episode is composed of 100 iterations in order to find the desired location by utilising the compared IW methods. Knowing that, we have used the same agents' initial positions in the environment as shown in Figure 9 in order to equitably compare the studied cases. Based on the obtained results, we constate that the maximum fitness ($f = 1$) was obtained in the 9th iteration through the application of IBWIW parameter. In the LIW case, a fitness value of 0.95 was achieved after 13 iterations. In the CIW, SAIW, NE2IW and NE2IW cases, a fitness values between 0.82 and 0.92 were acquired in 10, 10, 19 and 13 iterations respectively. On the other hand, a fitness value of 0.75 was reached in the 41st iteration by the use of the SIW parameter. These results proved the goal orientation provided by the proposed IW calculation to MAS path planning problem. In other words, the proposed approach allows the agents to reach their objective position in minimum time in comparison with other improved PSO approaches.

Based on the results reflected in Figure 7, we have seen the usefulness to study the difference between the fitness obtained in the $N+1$ iteration and the N iteration during the complete episode, as shown in Figure 8. Generally, in the eight compared cases, the difference increases in the first iterations. In particular, the highest difference that the IBWIW case produced is 0.9234 in the first iteration and 0.02 in the third iteration. In another studied case, the NE1IW produced a difference of 0.72 in the first iteration then 0.13 in the third one. Also, The SIW and the SAIW cases reflected close differences of 0.07 and 0.09, respectively in the 41st and 7th iterations. The highest differences obtained in the SIW and SAIW cases were in the first iteration (0.48 and 0.79 respectively). Furthermore, the differences obtained in the LDIW and the NE2IW methods were up to 0.82 and 0.86 in the first iteration. Also, these last two cases produced differences of 0.04 in 34th and 5th iterations. On the other hand, the differences that the BWIW, CIW, and LIW cases returned in first iteration were 0.88, 0.80 and 0.92, respectively. At a later time, they returned between the third and the fourth iterations difference values between 0.01 and 0.02.

Figure 7 Fitness development during one episode of 100 iterations

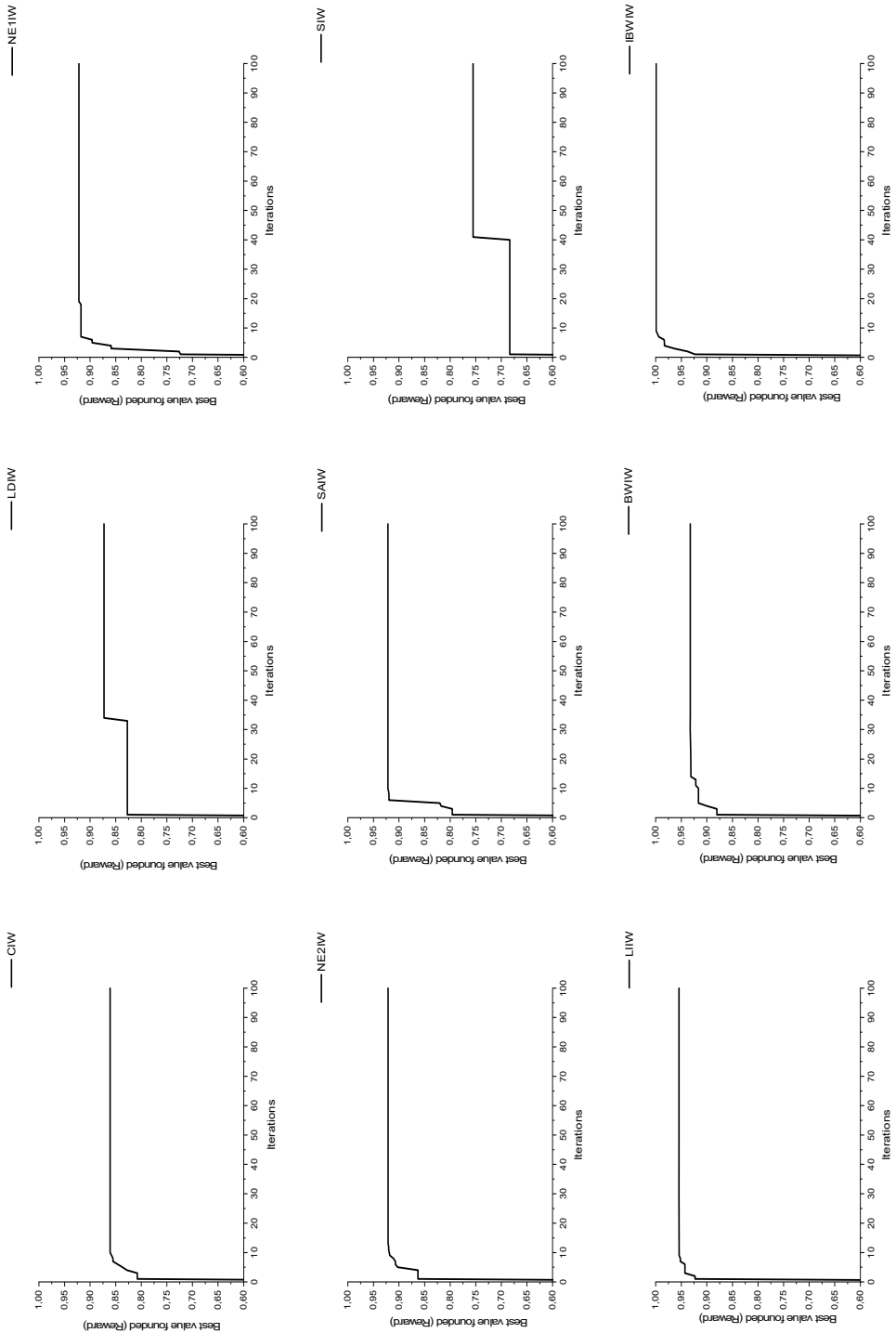
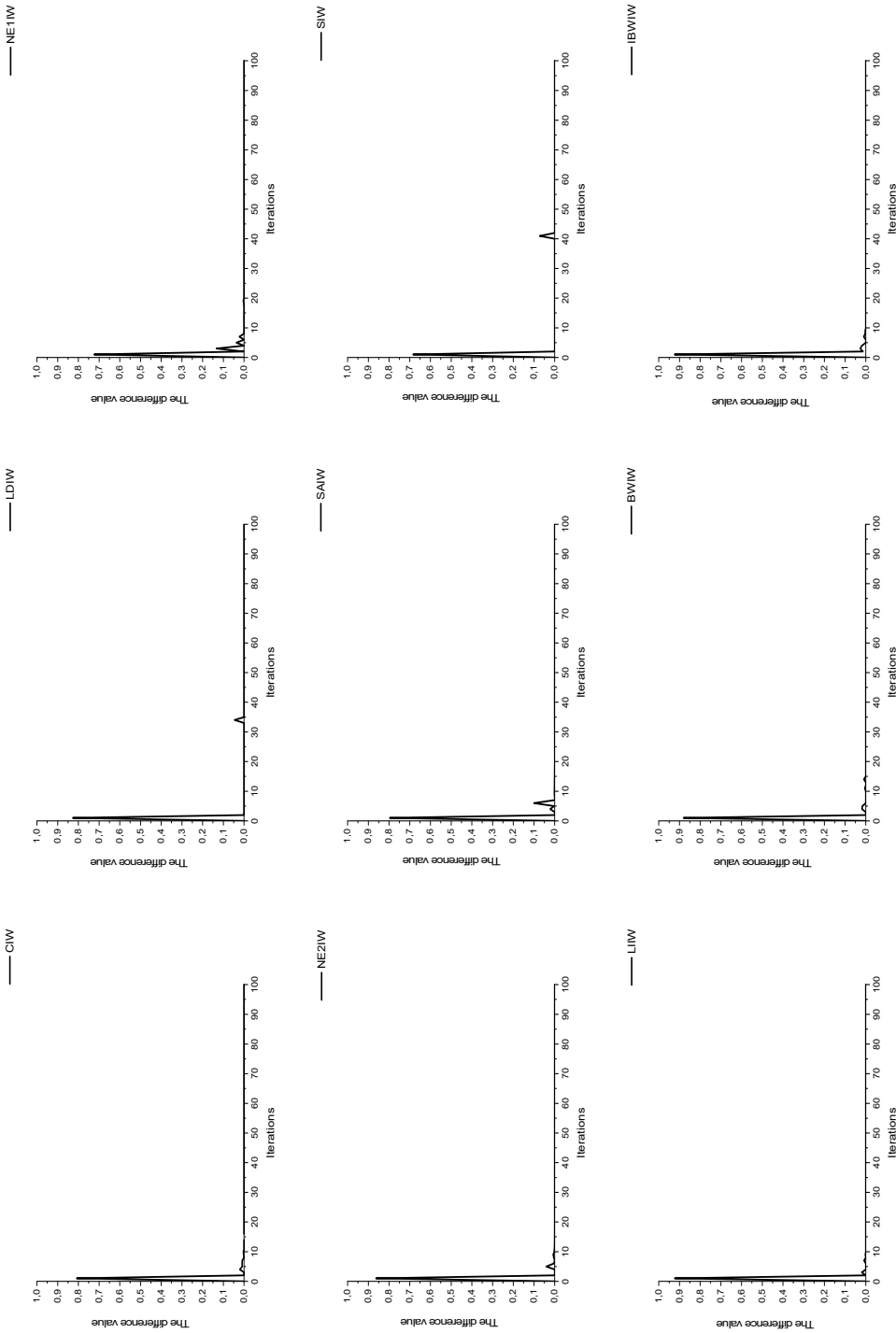


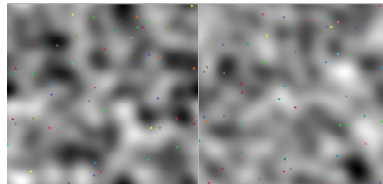
Figure 8 Difference between the N+1 iteration fitness and the N iteration fitness during one episode of 100 iterations



In comparison with SIW case, all the IW based on dynamic calculations provided better results in terms of fitness development, especially the proposed IBWIW. Precisely, the obtained results proved the efficiency of the PSO based on dynamic IW methods regarding the convergence towards the optimal solution. Moreover, the results demonstrate that the proposed IBIW is the most efficient approach to resolve the MAS path-planning problem.

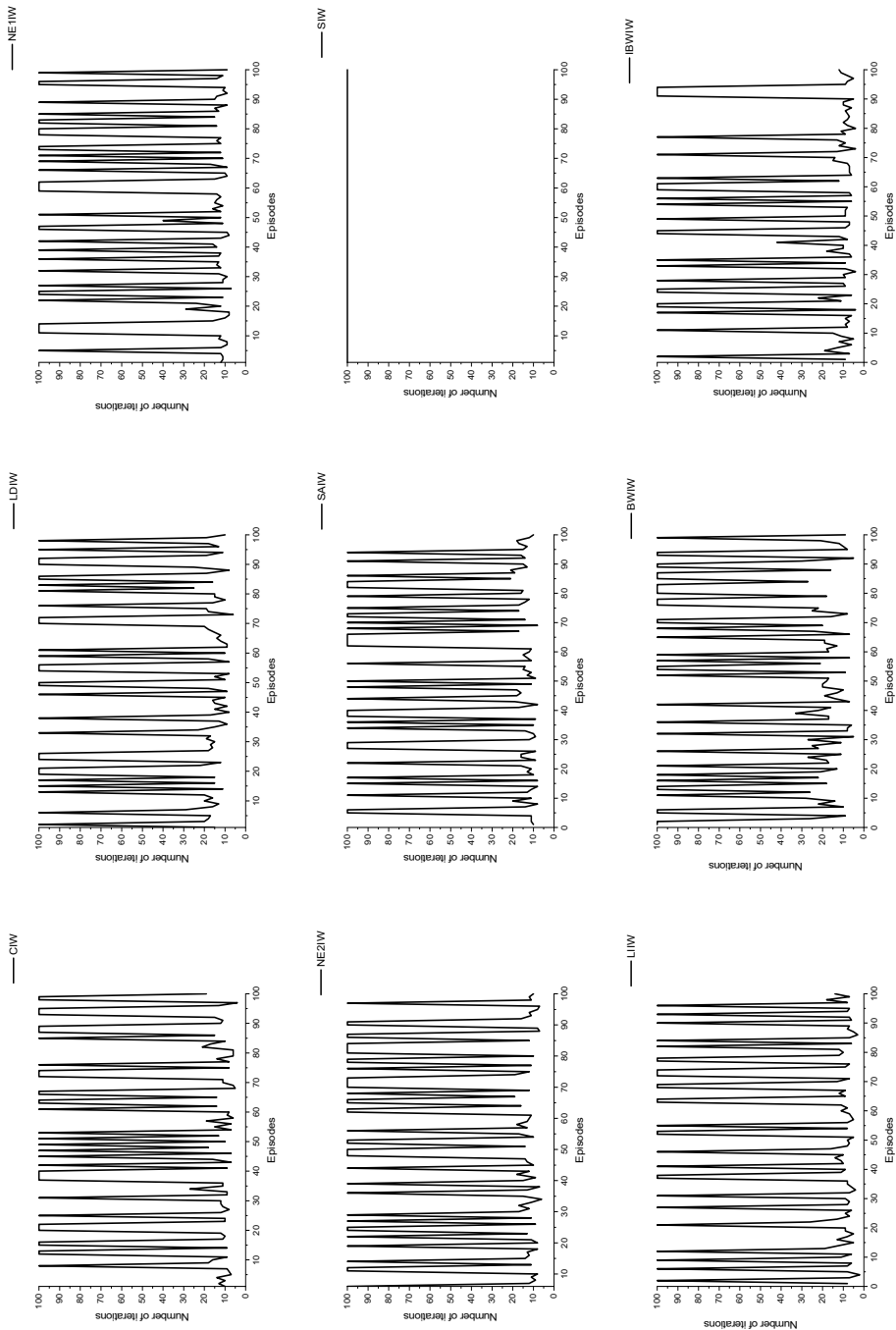
To verify the accuracy and reliability of the obtained results from the prior analysis, another simulation scenario is introduced. Figure 10 represents the searching time obtained through the application of the compared cases during 100 episodes as the first scenario. However, the same initial positions of particles are provided for all the IW variants (fixing the random initialisation of positions) as shown in Figure 9. Moreover, we note that the remaining parameters are unchanged. From the reflected results, we can constate that the dynamic IW based approaches achieved superior performances compared to the SIW in terms of searching time's average. Specifically, an average searching time of 100 was obtained by the use of the SIW parameter. Otherwise, the searching time decreases in BWIW and NE2IW with the averages of 48.38 and 48.10 respectively. In addition, the average of NE1IW case (43.30) is approximate to the averages of LDIW (43.98), CIW (44.95), and SAIW (43.65). Subsequently, the IBWIW case reflects the lowest average searching time (32.14) in comparison with the other cases. In the same context, the LIIW returned an average approximately equal to the IBWIW with 34.38. According to the obtained results, we can deduce that the increasing IW based methods are more efficient in the processing of the MAS path planning problem.

Figure 9 Example of two initialisations with the same positions (see online version for colours)



In this section, we introduced a comparative analysis of the proposed approach (IBPSO) in comparison with recent PSO versions based on IW improvement. Through these simulations, we studied the searching time obtained using the compared IW methods during 100 episodes by using random initial positions of the agents. Furthermore, we studied the fitness development obtained in an episode of 100 iterations. From the obtained results, we constated that all of the PSO improved versions, including the proposed one, outperformed the original PSO version. Moreover, we noted that the increasing IW methods provide a significant improvement in path planning problems. Specifically, PSO versions based on increasing IW enable agents to achieve their objectives in minimum time and efficiently converge to their optimal solutions. Moreover, to validate the obtained results, we studied the searching time through the use of the same the initial agents' positions. The results proved the effectiveness of the proposed approach as well as LIIW. We can have a better understanding of the increasing IW role by observing how solutions evolve during simulations. In other words, increasing IW encourages initial exploitation of the best solutions found, followed by an exploration of the search space.

Figure 10 Average-searching time obtained by initialising agents with the same positions for the nine cases



6 Conclusions

In this paper, we introduced a novel MAS path planning based on a modified PSO approach. As a metaheuristic approach, PSO allows the particles to move in the search space according to their personal best position as well as the global best position detected by the swarm. Moreover, PSO is based on the IW parameter, which allows to control the exploration and the exploitation of the search space. In the proposed PSO version, we have introduced an increasing nonlinearly IW control, known as IBIW in order to favorise the exploitation of the found solutions at the first iterations, and progressively the exploration of the search space. To apply this version to the MAS path planning, we have considered each agent as a particle moving in the environment to find the location with the highest fitness value. To prove the feasibility of the proposed method, we have compared it to other PSO versions based on IW improvement. According to the showcased results, we have constated that the IB-PSO allows to decrease the searching time and to increase the fitness acquisition in comparison with the other PSO versions. In future work, we will try to modify the proposed approach in order to be applicable in 3D search space environment. This initiative will be taken with the aim of considering reel world application based on the use of UAVs.

References

- Alam, M.S., Rafique, M.U. and Khan, M.U. (2015) 'Mobile robot path planning in static environments using particle swarm optimization', *International Journal of Computer Science and Electronics Engineering*, Vol. 3, No. 3.
- Altshuler, Y. (2023) 'Recent developments in the theory and applicability of swarm search', *Entropy*, Article 5, Vol. 25, No. 5, <https://doi.org/10.3390/e25050710>.
- Bansal, J.C., Singh, P., Saraswat, M., Verma, A., Jadon, S.S. and Abraham, A. (2011) 'Inertia weight strategies in particle swarm optimization', *2011 Third World Congress on Nature and Biologically Inspired Computing*, pp.633–640, <https://doi.org/10.1109/NaBIC.2011.6089659>.
- Beer, M., d'Inverno, M., Luck, M., Jennings, N., Preist, C. and Schroeder, M. (1999) 'Negotiation in multi-agent systems', *The Knowledge Engineering Review*, Vol. 14, No. 3, Article 3, <https://doi.org/10.1017/S0269888999003021>.
- Biswas, S., Anavatti, S.G. and Garratt, M.A. (2017) 'Obstacle avoidance for multi-agent path planning based on vectorized particle swarm optimization', in Leu, G., Singh, H.K. and Elsayed, S. (Eds.): *Intelligent and Evolutionary Systems*, Vol. 8, pp.61–74, Springer International Publishing, Cham, https://doi.org/10.1007/978-3-319-49049-6_5.
- Burmeister, B., Haddadi, A. and Matylis, G. (1997) 'Application of multi-agent systems in traffic and transportation', *IEE Proceedings – Software Engineering*, Article 1, Vol. 144, No. 1, <https://doi.org/10.1049/ip-sen:19971023>.
- Cruz, D.L. and Yu, W. (2017) 'Path planning of multi-agent systems in unknown environment with neural kernel smoothing and reinforcement learning', *Neurocomputing*, Vol. 233, pp.34–42, <https://doi.org/10.1016/j.neucom.2016.08.108>.
- Dai, Q., Liu, J. and Wei, Q. (2019) 'Optimal photovoltaic/battery energy storage/electric vehicle charging station design based on multi-agent particle swarm optimization algorithm', *Sustainability*, Article 7, Vol. 11, No. 7, <https://doi.org/10.3390/su11071973>.
- Doran, J.E., Franklin, S., Jennings, N.R. and Norman, T.J. (1997) 'On cooperation in multi-agent systems', *The Knowledge Engineering Review*, Article 3, Vol. 12, No. 3, <https://doi.org/10.1017/S0269888997003111>.
- Ferber, J. (1999) *Multi-agent Systems : An Introduction to Distributed Artificial Intelligence*, Addison-Wesley, Harlow.
- Ferber, J. (n.d.) *Les systèmes multi-agents : Un aperçu général*.

- Ghalia, M.B. (2008) 'Particle swarm optimization with an improved exploration-exploitation balance', *2008 51st Midwest Symposium on Circuits and Systems*, pp.759–762, <https://doi.org/10.1109/MWSCAS.2008.4616910>.
- Glover, F. and Kochenberger, G.A. (Éds.) (2003) *Handbook of Metaheuristics*, Springer US, Boston, MA.
- González-Briones, A., De La Prieta, F., Mohamad, M., Omatu, S. and Corchado, J. (2018) 'Multi-agent systems applications in energy optimization problems : a state-of-the-art review', *Energies*, Article 8, Vol. 11, No. 8, <https://doi.org/10.3390/en11081928>.
- Gupta, A. and Srivastava, S. (2020) 'Comparative analysis of ant colony and particle swarm optimization algorithms for distance optimization', *Procedia Computer Science*, Vol. 173, pp.245–253, <https://doi.org/10.1016/j.procs.2020.06.029>.
- Han, W., Yang, P., Ren, H. and Sun, J. (2010) 'Comparison study of several kinds of inertia weights for PSO', *2010 IEEE International Conference on Progress in Informatics and Computing*, pp.280–284, <https://doi.org/10.1109/PIC.2010.5687447>.
- Hussain, K., Salleh, M.N.M., Cheng, S. and Shi, Y. (2019) 'On the exploration and exploitation in popular swarm-based metaheuristic algorithms', *Neural Computing and Applications*, Article 11, Vol. 31, No. 11, <https://doi.org/10.1007/s00521-018-3592-0>.
- Jain, M., Saihjal, V., Singh, N. and Singh, S.B. (2022) 'An overview of variants and advancements of PSO algorithm', *Applied Sciences*, Vol. 12, No. 17, p.8392, <https://doi.org/10.3390/app12178392>.
- Kamali, S.R., Baniroostam, T., Motameni, H. and Teshnehlab, M. (2023) 'An immune-based multi-agent system for flexible job shop scheduling problem in dynamic and multi-objective environments', *Engineering Applications of Artificial Intelligence*, Vol. 123, p.106317, <https://doi.org/10.1016/j.engappai.2023.106317>.
- Kaur, M. and Saini, S. (2021) 'A review of metaheuristic techniques for solving university course timetabling problem', in Goar, V., Kuri, M., Kumar, R. and Senju, T. (Éds.): *Advances in Information Communication Technology and Computing*, Vol. 135, pp.19–25, Springer, Singapore, https://doi.org/10.1007/978-981-15-5421-6_3.
- Kennedy, J. and Eberhart, R. (1995) 'Particle swarm optimization', *Proceedings of ICNN'95 – International Conference on Neural Networks*, Vol. 4, pp.1942–1948, <https://doi.org/10.1109/ICNN.1995.488968>.
- Khan, I., Maiti, M.K. and Maiti, M. (2017) 'Coordinating particle swarm optimization, ant colony optimization and K-opt algorithm for traveling salesman problem', in Giri, D., Mohapatra, R.N., Begehr, H. and Obaidat, M.S. (Éds.): *Mathematics and Computing*, Vol. 655, pp.103–119, Springer, Singapore, https://doi.org/10.1007/978-981-10-4642-1_10.
- Kumar, N., Pal, N., Kumar, P. and Kumari, A. (2018) 'Impact of different inertia weight functions on particle swarm optimization algorithm to resolve economic load dispatch problems', *2018 4th International Conference on Recent Advances in Information Technology (RAIT)*, pp.1–5, <https://doi.org/10.1109/RAIT.2018.8389065>.
- Kusuma, P.D. and Novianty, A. (2023) 'Total interaction algorithm : a metaheuristic in which each agent interacts with all other agents', *International Journal of Intelligent Engineering and Systems*, Article 1, Vol. 16, No. 1, <https://doi.org/10.22266/ijies2023.0228.20>.
- Li, X., Wu, D., He, J., Bashir, M. and Liping, M. (2020) 'An improved method of particle swarm optimization for path planning of mobile robot', *Journal of Control Science and Engineering*, Vol. 2020, pp.1–12, <https://doi.org/10.1155/2020/3857894>.
- Lopes Silva, M.A., De Souza, S.R., Freitas Souza, M.J. and Bazzan, A.L.C. (2019) 'A reinforcement learning-based multi-agent framework applied for solving routing and scheduling problems', *Expert Systems with Applications*, Vol. 131, pp.148–171, <https://doi.org/10.1016/j.eswa.2019.04.056>.
- Lopes Silva, M.A., De Souza, S.R., Freitas Souza, M.J. and De França Filho, M.F. (2018) 'Hybrid metaheuristics and multi-agent systems for solving optimization problems : a review of frameworks and a comparative analysis', *Applied Soft Computing*, Vol. 71, pp.433–459, <https://doi.org/10.1016/j.asoc.2018.06.050>.

- Malik, R.F., Rahman, T.A., Hashim, S.Z.M. and Ngah, R. (2007) 'New particle swarm optimizer with sigmoid increasing inertia weight', *International Journal of Computer Science and Security*, Vol. 1, No. 2, pp.35–42.
- Marini, F. and Walczak, B. (2015) 'Particle swarm optimization (PSO). A tutorial', *Chemometrics and Intelligent Laboratory Systems*, Vol. 149, pp.153–165, <https://doi.org/10.1016/j.chemolab.2015.08.020>.
- McArthur, S.D.J., Davidson, E.M., Catterson, V.M., Dimeas, A.L., Hatziaargyriou, N.D., Ponci, F. and Funabashi, T. (2007) 'Multi-agent systems for power engineering applications – part I: concepts, approaches, and technical challenges', *IEEE Transactions on Power Systems*, Article 4, Vol. 22, No. 4, <https://doi.org/10.1109/TPWRS.2007.908471>.
- Nayeem, G.M., Fan, M. and Akhter, Y. (2021) 'A time-varying adaptive inertia weight based modified PSO algorithm for UAV path planning', *2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST)*, pp.573–576, <https://doi.org/10.1109/ICREST51555.2021.9331101>.
- Nesmachnow, S. (2014) 'An overview of metaheuristics: accurate and efficient methods for optimisation', *International Journal of Metaheuristics*, Article 4, Vol. 3, No. 4, <https://doi.org/10.1504/IJMHEUR.2014.068914>.
- Nouiri, M., Bekrar, A., Jemai, A., Niar, S. and Ammari, A.C. (2018) 'An effective and distributed particle swarm optimization algorithm for flexible job-shop scheduling problem', *Journal of Intelligent Manufacturing*, Article 3, Vol. 29, No. 3, <https://doi.org/10.1007/s10845-015-1039-3>.
- Rafai, A.N.A., Adzhar, N. and Jaini, N.I. (2022) 'A review on path planning and obstacle avoidance algorithms for autonomous mobile robots', *Journal of Robotics*, p.2538220, <https://doi.org/10.1155/2022/2538220>.
- Raposo, A.B. and Fuks, H. (2002) 'Defining task interdependencies and coordination mechanisms for collaborative systems', in Blay-Fornarino, M., Pinna-Dery, A.M., Schmidt, K. and Zaraté, P. (Eds.): *Cooperative Systems Design*, Vol. 74, pp.88–103, IOS Press.
- Shao, Z., Yan, F., Zhou, Z. and Zhu, X. (2019) 'Path planning for multi-UAV formation rendezvous based on distributed cooperative particle swarm optimization', *Applied Sciences*, Article 13, Vol. 9, No. 13, <https://doi.org/10.3390/app9132621>.
- Sood, M. and Panchal, V.K. (2020) 'Meta-heuristic techniques for path planning: recent trends and advancements', *International Journal of Intelligent Systems Technologies and Applications*, Vol. 19, No. 1, pp.36–52, DOI: 10.1504/ijista.2020.105177.
- Souidi, M.E.H., Haouassi, H., Ledmi, M., Maarouk, T.M. and Ledmi, A. (2023) 'A discrete particle swarm optimization coalition formation algorithm for multi-pursuer multi-evader game', *Journal of Intelligent & Fuzzy Systems*, Article 1, Vol. 44, No. 1, <https://doi.org/10.3233/JIFS-221767>.
- Taherkhani, M. and Safabakhsh, R. (2016) 'A novel stability-based adaptive inertia weight for particle swarm optimization', *Applied Soft Computing*, Vol. 38, pp.281–295, <https://doi.org/10.1016/j.asoc.2015.10.004>.
- Tisue, S. and Wilensky, U. (2004) 'Netlogo: a simple environment for modeling complexity', in *International Conference on Complex Systems*, May, Vol. 21, pp.16–21.
- Wahab, M.N.A., Nefti-Meziani, S. and Atyabi, A. (2020) 'A comparative review on mobile robot path planning: classical or meta-heuristic methods?', *Annual Reviews in Control*, Vol. 50, pp.233–252, <https://doi.org/10.1016/j.arcontrol.2020.10.001>.
- Zheng, Y.-L., Ma, L.-H., Zhang, L.-Y. and Qian, J.-X. (2003) 'Empirical study of particle swarm optimizer with an increasing inertia weight', *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, pp.221–226, <https://doi.org/10.1109/CEC.2003.1299578>.
- Zhu, X. and Wang, H. (2018) 'A new inertia weight control strategy for particle swarm optimization', *AIP Conf. Proc.*, 040095, <https://doi.org/10.1063/1.5033759>.