Enhanced Autonomous Mobile Robot Navigation Using a Hybrid BFO/PSO Algorithm for Dynamic Obstacle Avoidance

Amina Makhlouf*, Abdelmadjid Benmachiche, Ines Boutabia
LIMA Laboratory, Dept. of Computer Science, faculty of science and technology, House of Artificial Intelligence, Chadli Bendjedid, University, El-Tarf, Algeria, PB 73, 36000
E-mail: makhlouf-amina@univ-eltarf.dz, benmachiche-abdelmadjid@univ-eltarf.dz, i.boutabia@univ-eltarf.dz
*Corresponding author

Keywords: particle swarm optimization, bacteria foraging optimization algorithm, mobile robot, obstacle avoidance, ambulance path planning, and evolutionary robotics

Received: July 19, 2024

Determining path planning for mobile robots has received a lot of attention over the last three decades, with the goal of identifying safe and effective paths between starting points and destinations. The investigation of diverse navigation methodologies exhibits potential in expanding the range of uses for autonomous mobile robots. This research introduces a novel navigation strategy by combining the Particle Swarm Optimization (PSO) method with the Bacteria Foraging Optimization Algorithm (BFO). Utilizing the benefits of PSO, this bio-inspired method is used to optimize paths for mobile robots, drawing inspiration from the foraging behavior of bacteria, namely E. coli. The principal aim of this study is to design routes that are practical and safe. The proposed model features a robot that emulates bacterial foraging behavior to determine the most optimal path through environments densely populated with obstacles, effectively linking a starting point to a specified destination. This strategy has been implemented and rigorously evaluated across a variety of scenarios. Experimental results reveal that the hybrid BFO-PSO algorithm achieves a success rate exceeding 90% in navigating dynamic environments. Furthermore, the algorithm significantly outperforms traditional methods, offering approximately 50% less computational speed, efficiency, and superior security metrics, confirming its effectiveness and potential for real-world applications.

Povzetek: Študija uvaja hibridni BFO-PSO algoritem za avtonomno navigacijo mobilnih robotov, ki preprečuje trke z dinamičnimi ovirami, izboljša varnost ter hitrost.

1 Introduction

As we step into the new era, the advancement of robotics and computer technologies sets the stage for a heightened dependence on autonomous machines across a wider spectrum of tasks. Mobile robots, in particular, offer the potential for executing diverse operations, especially in environments where human presence is unsuitable, such as nuclear plants, hazardous waste sites, and chemical industries. They excel at handling repetitive tasks like parts delivery within manufacturing plants, as well as complex endeavors like space and underwater exploration.

In this domain, a crucial aspect is the independent navigation capability of robots, especially in exploration scenarios where they must traverse terrains that are only partially understood at best. Achieving full autonomy in navigation requires progress in multiple fields including engineering, computer science, and applied mathematics.

In the past two decades, significant strides have been made in addressing navigation and obstacle avoidance, critical concerns in mobile robotics. Path planning [1], a complex aspect of mobile robot navigation, entails finding a collision-free route between two points within a given environment, this involves determining a suitable path for a robot, often by optimizing criteria such as minimizing travel distance. For instance, during emergencies like earthquakes or fires, planning ambulance movements involves challenges like choosing the shortest route to reach injured individuals, preventing collisions, and avoiding road obstacles.

While various approaches have been proposed, recent attention has shifted towards bacterial foraging optimization (BFO), a swarm intelligence technique inspired by the foraging behavior of E. coli colonies [2]. Although effective in addressing engineering and mobile robotics challenges, BFO suffers from drawbacks such as getting stuck in local minima and deviating from optimal solutions, especially in the presence of obstacles [3].

To address these challenges, researchers have explored different methods, including the use of heuristics like Particle Swarm Optimization (PSO) [4] to avoid collisions. PSO, known for its simplified structure and ease of implementation [5], has been widely applied to path planning due to its elegant mathematical analysis and simplicity, on the other hand, BFO is recognized for its straightforward and efficient search algorithm.

To leverage the strengths of both BFO and PSO while mitigating their drawbacks, this study introduces a hybrid method. By combining these two approaches, the proposed method aims to optimize path planning while ensuring secure navigation. The paper is structured as follows: the introduction provides an overview in section

1, followed by a discussion of previous related works in section 2. Section 0 briefly outlines the theory of the two algorithms used and presents the general architecture, detailing each component to navigate securely using a hybrid model for optimized path planning. Simulation results are evaluated in section 4 and discussed in section 5, followed by conclusions and future work in the final section 6.

2 Related works

The navigation problem of an autonomous mobile robot has been widely investigated in past decades, there has been an increasing interest in the conception and development of an autonomous mobile robot using several soft computing techniques. The goal of navigation is to search for an optimal path from the beginning point to the objective point with hindrance evasion skills. Essentially, the mobile robot navigation has been finished by the Deterministic algorithm and Nondeterministic (Stochastic) algorithm. These days, the hybridization of both algorithms is being utilized to solve the mobile robot navigation problem.

Many methods have been developed for deterministic navigation, i.e. Dijkstra algorithm [6], Visibility graph [7], Grids and Cell decomposition method [8], in 1986, Khatib introduced the Artificial Potential Field (APF) approach to address mobile robot navigation. In this method, the goal and obstacles act as charged surfaces, generating a total potential that exerts an imaginary force on the robot. This force attracts the robot towards the goal while repelling it from obstacles. To avoid obstacles and reach the target, the robot follows the negative gradient. Borenstein et al [9]. addressed local minimum conditions by considering the dynamic properties of robot navigation. Authors in [10] developed a mechanism for velocity control to navigate moving obstacles in real-time environments. Authors in [11] introduced superior potential and repulsive potential functions to avoid local minima and achieve global optima. Researchers in [12] tackled issues like oscillation and conflicts in mobile robot navigation using the APF approach, presenting an enhanced version to minimize these problems, especially when the goal is close to an obstacle.

However, nondeterministic methods are better suited for locating the best solutions for any kind of objective function since they are not dependent on the mathematical characteristics of a specific function. However, the weaknesses of the first type are its dependence on the gradient, local optima inefficient in large-scale search space, and cannot to solve discrete functions.

Nature-inspired algorithms are a class of efficient non-deterministic approaches that imitate the actions of specific animals or insects (birds, ants, bees, flies, and even germs!). These paradigms are currently in widespread use across numerous technical domains. Some of these algorithms are:

Genetic algorithm (GA) [13], The GA is a widely employed optimization tool based on genetic principles and natural selection, GA has found applications in various scientific and technological domains, including robot navigation. It is designed to address complex optimization problems involving the minimization of a function under specific constraints. Despite its random nature, GA outperforms random local search by leveraging historical information. Its application to mobile robot navigation in static environments has been demonstrated, with a focus on path planning around polygonal obstacles. Extensions of GA have tackled dynamic environments, including navigation around moving obstacles. Hybrid approaches aim to enhance robot path planning. Multi-robot path planning using GA has been explored for collision avoidance in both static and dynamic environments [14]. GA's versatility is evident in its application to diverse scenarios, spanning 3D path planning for underwater and aerial robots, 2D path planning for humanoid robots, and defense equipment.

Artificial Neural Networks (ANN) [15], ANN represent intelligent systems composed of interconnected processing elements that convey information through dynamic state responses to external inputs. These networks consist of organized layers of interconnected nodes, each featuring an activation function. ANN's attributes, such as generalization ability, massive parallelism, distributed representation, learning capability, and fault tolerance, make it a valuable tool in the domain of mobile robot navigation. In the context of wheeled mobile robot navigation in a partially unknown environment, Janglova [16] employed ANN-based mechanisms for collision-free path generation. The first mechanism identifies free space using sensory data, while the second determines a safe trajectory by avoiding the nearest obstacle. Researchers in [17] integrated ANN into the fast Simultaneous Localization and Mapping technique to address error accumulation issues in odometry models and SLAM nonlinear function linearization. This integration enhances mobile robot navigation in unknown environments while ensuring collision avoidance. Authors in [18] suggested a hybrid strategy, merging Neural Networks and Fuzzy Logic for multiple mobile robot navigation in disorderly conditions, specifically analyzing its effectiveness in a static obstacle architecture. The authors in [19] presented a self-learning strategy for mobile robots based on ANN. ANN's versatility extends to path planning for humanoid, industrial, underwater, and aerial robots.

Fuzzy Logic (FL) [18], has become widely utilized in diverse research and development domains due to its effectiveness in dealing with high uncertainty, complexity, and nonlinearity. It finds applications in pattern recognition, automatic control, decision-making, and data classification. FL significantly eases the task of system designers and computers, proving valuable for obtaining accurate information in complex scenarios. Its application extends to the mobile robots' path planning, particularly in unstructured static and dynamic environments.

Researchers have developed fuzzy-based navigation systems for omnidirectional mobile robots, introducing automatic fuzzy rule generation for obstacle avoidance to enhance navigation efficiency. FL has been integrated with sensor-based navigation techniques to improve

incremental learning in new environments. Furthermore, FL has been synergistically employed with various algorithm-based navigation techniques. This integration aims to achieve optimal environmental perception, enabling robots to navigate challenging scenarios, including dead-end situations. Abadi et al. [20] designed a FL controller for wheeled mobile robots to track moving objects, employing a hybrid approach with Particle Swarm Optimization (PSO) for optimal parameter selection. Al-Jarrah et al. [21] focused on path planning for multiple mobile robot systems and active motion coordination, utilizing a probabilistic fuzzy controller with Artificial Neural Networks (ANN) [22]. The work in [23] has enhanced multi-autonomous underwater robot formation control by integrating event-triggered mechanisms and model predictive control, alongside neural network optimization. Their method effectively minimized errors and resource usage, demonstrating efficient and accurate obstacle avoidance. The authors in [24] have resolved the trajectory-tracking problem of non-holonomic mobile robots using Random Inertia Weight Particle Swarm Optimization (RNW-PSO) based optimal Mamdani-type fuzzy controller. The motion problem of the mobile robots on uneven terrain has been addressed. The work in [25] has developed the neuro-fuzzy controller for navigation of a non-holonomic differential drive mobile robot.

Ant Colony Optimization (ACO) [26], stands out as a swarm intelligence algorithm devised, drawing inspiration from the foraging behavior of ants. Initially crafted for addressing combinatorial optimization problems, ACO has found diverse applications, including its utilization in mobile robot navigation. In the realm of navigation for multiple mobile robots within a static environment, the authors in [27] introduced a collision avoidance strategy, incorporating a specialized function to enhance selective strategy and address challenges like dead corners. The authors in [28] proposed an innovative RA-ACO-based strategy for navigating humanoid robots through cluttered spaces. In the context of dynamic environments, The scientists in [29] applied ACO for mobile robot navigation in unknown dynamic environments, leveraging its capabilities for both selecting and optimizing fuzzy rules.

Particle Swarm Optimization (PSO) emerges as a nature-inspired metaheuristic algorithm, drawing inspiration from the collective behavior observed in fish schools and bird flocks. Unlike some algorithms, it mimics the cooperative behavior of social animals, with members communicating and collectively moving towards a solution. In the domain of mobile robot navigation, PSO has been extensively employed to tackle diverse challenges. Researchers have applied PSO to address mapping and localization challenges in unknown environments, leveraging a multi-agent particle filter for enhanced stability and reduced computation.

The authors in [30] contributed the Self-Adaptive Learning Particle Swarm Optimization approach, specifically designed for solving robot path planning problems in intricate environments with various constraints. This approach transformed the path planning problem into a minimization multi-objective optimization problem, considering factors such as path length, collision risk degree, and smoothness. Researchers in [31] explored the use of PSO in a complex 3D environment, employing the combined PSO-UFastSLAM approach to enhance estimation accuracy. PSO has demonstrated success in diverse applications, including aerial robots in unknown environments, humanoid robots, and industrial robots for navigation.

The Bacterial Foraging Optimization (BFO), established in 2002 by Passino [2], is modeled after the foraging behavior of bacteria such as M. Xanthus and E. coli. This algorithm incorporates four fundamental principles: chemotaxis, swarming, reproduction, and elimination dispersal. The BFO algorithm addressed mobile robot navigation in a static environment; introducing variable velocity based on uniform, Gauss, and Cauchy distributions. This strategy was extended to handle scenarios with multiple obstacles in a static environment by subsequent researchers. In the realm of improving wheeled robot performance in path planning, scientists in [28] developed an enhanced BFO algorithm. This model incorporates an Artificial Potential Field (APF) technique to represent the environment, integrating attractive forces toward the goal and repulsive forces from obstacles. The authors in [32] utilized the BFO algorithm to tackle the navigation problem for an Unmanned Aerial Vehicle (UAV). The BFO was combined with a Proportional Integral Derivative controller in this application, enabling the determination of optimal search parameters in 3D space. Below is a table summarizing the various discussed approaches, mentioning their pros and cons.

Table 1: Summary of the various navigation approaches

Method	Advantages	Drawbacks	Performance Insights	Benefits of Hybrid BFO-PSO
APF [9-12]	- Simple to implement and good for local obstacle avoidance. - Light in terms of computation. Fast result.	- May encounter issues with local minima; less effective in highly dynamic or complex environments. - Not optimal	- Effective for local pathfinding; struggles with complex or dynamic scenarios.	- Provides local obstacle avoidance as part of a broader strategy in the hybrid model.
A*[1]	- Guarantees the shortest path in static environments.	- Struggles with dynamic changes, leading to high computation costs due to constant re-planning.	- Best for static environments; less effective in dynamic settings.	- Enhances performance in dynamic scenarios, reducing re-planning needs.
GA [1, 13-14]	- Flexible and effective for complex, multi-objective problems. - Incorporate probabilistic transition rules to circumvent local optima. - Facilitates multi-objective optimization.	- Can converge early on suboptimal solutions; performance depends on precise parameter tuning. - GA necessitates less data about the problem, yet formulating an objective function and acquiring suitable representations and operators can pose challenges. - GA consumes considerable time.	- Effective in many complex cases; may be slow to converge.	- Speeds up convergence and reduces risk of early convergence.
ANN [15-19]	- Learns and adapts to complex patterns, versatile in various scenarios. - Computing precise navigation routes. - Applied for addressing problems related to the target function, which could yield outcomes that are discrete, real-valued, or comprised of multiple attributes that are either discrete or real-valued.	- Needs large amounts of training data and computational power; risk of overfitting.	- Highly adaptable with sufficient training; resource-intensive.	- Lessens reliance on extensive datasets and reduces computational burden.
FL [1, 20-25]	- It excels at handling uncertainty and making smooth decisions, as it better reflects real-world problems compared to classical logic. - It employs the most human-like reasoning, enabling it to provide more true-to-life results, even when dealing with ambiguous or imprecise data. - Create a set of rules based on human experience. - It is light in terms of computation, time savings, and memory space.	- Effectiveness depends on the quality of the rule set, which can be difficult to design and tune. - The navigation path is not ideal due to the approximate inference method where comprehensive assurance and verification are required. - Operations are limited by these rules and require an expert.	- Effective under uncertainty; performance varies with rule set quality.	- Enhances decision-making under uncertainty and complements other methods.
ACO [26-29]	- Solves complex, dynamic problems effectively, inspired by natural processes. - Exhibits exceptional adaptability and is suited for dynamic applications. - Favorable feedback facilitates rapid identification of satisfactory solutions. - Distributed computation mitigates premature convergence.	- Can be slow to converge; may require many iterations for optimal solutions. - It is not universally applicable to all problem types and may encounter blocking states.	- Useful in dynamic settings; slower convergence can be a downside.	- Accelerates convergence and improves global search capabilities.
PSO [30-31]	- Rapidly converges and works well in dynamic settings. - Requires very few algorithm parameters and is indifferent to scaling variations in design variables. - Easily parallelizable, allowing for concurrent processing, and is highly efficient in global search.	- Prone to getting trapped in local optima in high-dimensional spaces; performance can vary depending on problem complexity.	- Effective in simpler scenarios; performance can be inconsistent in complex ones.	- Improves consistency and avoids suboptimal solutions when combined with BFO.
BFO [2, 28-32, 34]	- Robust in finding global solutions in noisy and dynamic environments. - BFO enriches the velocity-swim operator, which enhances its convergence behavior. - The linear decreasing mechanism offers simplicity and low computational cost, contributing to overall performance. - Features strong parallel search capabilities and facilitates easy escape from local minima.	- Convergence can be slow; performance is sensitive to parameter settings. - The mechanism for handling constraints is very precise but difficult to generalize so users must specify a large number of parameter values. - Weak interconnection among bacteria increases the risk of converging to local optima rather than global optima.	- Effective for noisy conditions; slower convergence is a limitation.	- Speeds up convergence and enhances performance in complex scenarios.
Hybrid BFO-PSO (Proposed)	- Combines the rapid convergence of PSO with BFO's global search capabilities, highly adaptable.	- More complex to implement and requires careful tuning of parameters.	- Achieves over 90% success rate in dynamic environments, faster and more efficient.	/

APF is known for its simplicity in implementation and its effectiveness in avoiding obstacles locally, making it useful for straightforward navigation tasks. However, APF can struggle with getting stuck in local minima,

where the robot may end up on suboptimal paths, and it tends to be less effective in environments with rapidly changing obstacles or high complexity. While it works well for local navigation, it may not be as effective in more complex or dynamic settings.

A* is recognized for its ability to find the shortest path between two points in stable environments, ensuring that the route is optimal according to its cost function. In dynamic settings where conditions change frequently, however, A* can become inefficient because it requires constant recalculations to adapt to new obstacles. Although A* performs excellently in static scenarios, its effectiveness diminishes in dynamic situations due to the high computational costs associated with continuous updates.

GA are valued for their adaptability and capability to address complex optimization problems with multiple objectives [33]. Their flexibility allows them to handle a broad spectrum of problems. Nonetheless, GAs can sometimes converge on suboptimal solutions prematurely, and their effectiveness depends heavily on how well their parameters are tuned, which can be challenging. GAs are effective for complex problems but may experience slower convergence rates and variable performance depending on problem specifics and parameter adjustments.

ANN are effective at learning and adapting to complex patterns, making them versatile and powerful for modeling intricate problems. They require large amounts of data and substantial computational resources for training, and there is a risk of overfitting if not managed properly. While NN is highly adaptable and effective in various applications, their resource-intensive nature and dependency on extensive training data can be a limitation.

PSO is known for its fast convergence and effectiveness in environments that change dynamically. It is particularly useful when quick solutions are required. However, PSO can sometimes become trapped in local optima, leading to less optimal solutions, and its performance may vary depending on the problem's complexity. While PSO is effective for simpler problems and dynamic settings, it may face challenges with consistency in more complex scenarios.

BFO is robust in finding global solutions, especially in noisy and dynamic environments, and is effective against disruptions. However, it tends to converge slowly and is sensitive to parameter settings, which can limit its performance in more complex scenarios. Although BFO is well-suited for handling noisy conditions and achieving global optimization, its slower convergence can be a notable drawback, particularly in fast-moving applications.

FL is effective in managing uncertainty and making smooth decisions, especially in situations where precise data is not available. The performance of Fuzzy Logic depends on the quality and design of the rule set, which can be difficult to develop. Fuzzy Logic is useful for dealing with uncertainty and works well in conjunction with other methods to enhance overall performance and decision-making.

ACO effectively addresses dynamic and complex problems by mimicking natural processes. It excels in challenging environments but can be slow to converge and may require many iterations to find the best solution. While ACO is effective for dynamic settings, its slower convergence rate can be a limitation, requiring careful balancing of trade-offs.

The proposed Hybrid BFO-PSO approach merges the strengths of PSO's quick convergence with BFO's global optimization capabilities. This combination results in a method that is both adaptable and efficient in dynamic environments. Although implementing this hybrid approach is more complex and requires careful parameter tuning, it achieves over 90% success in navigating dynamic settings, offering both faster computation and improved efficiency compared to traditional methods.

3 Our proposed system

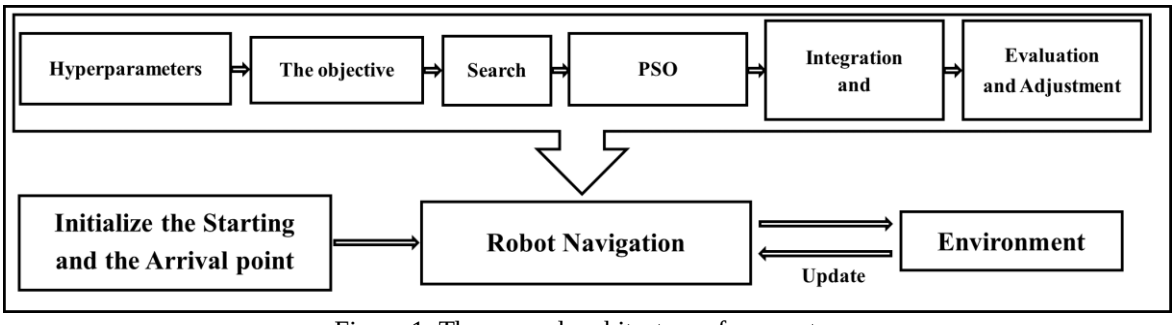

Figure 1: The general architecture of our system

3.1 BFO algorithm

The BFO algorithm [1] is a relatively new nature-inspired meta-heuristic algorithm, which imitates the foraging behavior of E. coli found in the human intestine. The process of finding the regions of high nutrients by the bacteria can be modeled as an optimization process. BFO has been applied in various optimization problems associated with the real world and therefore already

gained the attention of researchers in the domain. The swarm of bacteria S foraging behavior includes four main steps: chemotaxis (tumble and swimming), swarming, reproduction, and elimination-dispersal.

From a biological perspective, E.coli bacteria exhibit two primary modes of movement: they can either swim consistently in one direction for a period or tumble continuously throughout their lifetime. In the conventional BFO model, a random change in direction signifies a "tumble," while maintaining the same direction as the previous step represents a "run."

Chemotaxis, the process by which bacteria navigate their environment based on chemical gradients, heavily influences their foraging behavior. This involves a sequence of tumbles interspersed with runs. Within the BFO framework, the algorithm simulates this chemotactic process through position updates, as outlined in (1)
(1):

$$\theta_i^{j+1} = \theta_i^j + C(i).\emptyset(i)$$

(2):

$$\emptyset(i) = \frac{\Delta(i)}{\sqrt{\Delta^T(i)\Delta(i)}}$$

The position of the i^{th} bacterium in the j^{th} chemotaxis step is given by equation (1), where C(i) is the step length in the j_i^{th} chemotaxis and ∅(i) is a unit vector indicating the direction of swimming following a tumble. It can be created using (2), where Δi is a vector created at random with the same issue dimension. The bacteria first developed a tumbling direction in each chemotactic phase. The bacteria then proceed in that direction by (1). It will run one more stage in the same direction if the concentration of nutrients in the current place is higher than that of the previous position. Until the maximum run step is reached or the nutrition gets worse, this process is repeated. Ns is a parameter that controls the maximum run iteration.

Assume θ^i(j, k, l) represents the bacterium at j^{th} chemotactic, kth reproductive, and lth elimination-dispersal step. C(i) is the chemotaxis step size during each tumble or run (the length of unit walks), and ∅(j)is the direction angle of the j^{th} step. Then the movement of the i^{th} bacterium can be modeled as:
(3):

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)\emptyset(j)$$
$$\theta^i(j+1,k,l) > \theta^i(j,k,l) \text{ Swimming in which}$$
$$\emptyset(j) = \emptyset(j-1)$$
$$\theta^i(j+1,k,l) >$$
$$\theta^i(j,k,l) \text{ Tumbling in which } \emptyset(j) \epsilon [0, 2\pi]$$

Where C(i), (i = 1, 2,…., S) is the size of the step made in the direction indicated by the tumble, which is random. The cost at the site of the i^{th} bacterium θi(j, k, l) is represented by the fitness, J(i, j, k, l). Another step of size C(i) will be taken in the same direction if the cost J (i, j + 1, k, l) is better (inferior) than at θi(j, k, l).

Otherwise, Bacteria exhibit a behavior where they move by taking steps of size C(i) in random directions, aiming to locate more favorable nutrient environments. This movement results in the clustering of bacteria in regions abundant with nutrients.

3.1.1 Swarming

This phenomenon, known as swarming, involves intriguing collective actions observed in various motile bacteria species such as E. coli. To mimic this behavior, a communication mechanism between cells is established to emulate the organic conduct of swarming bacteria. Each bacterium releases attractants as it moves, signaling other bacteria to converge towards it, while simultaneously emitting repellents to deter them from coming too close. BFO models this social interaction by encapsulating both the attraction and repulsion among cells. The combined effects of cell-to-cell attraction and repulsion can thus be represented as follows:
(4):

$$J_{cc}(\theta, P(j,k,l) = \sum_{i=1}^{s} J_{cc}^i \left(\theta, \theta^i (j,k,l)\right)$$

(5):

$$J_{cc}(\theta, P(j,k,l)$$
$$= \sum_{i=1}^{s} [-d_{att} \exp\left(-w_{att} \sum_{m=1}^{p} (\theta_m - \theta_m^i\right)^2)]$$
$$+ \sum_{i=1}^{s} [-d_{rep} \exp\left(-w_{rep} exp \sum_{m=1}^{p} (\theta_m - \theta_m^i\right)^2)]$$

Where datt and watt are respectively the depth and the width of the attractant released by the cell. Likewise, hrep and wrep are the height and measure of the width of the repellent effect.

3.1.2 Reproduction

After every N_c chemotactic step, a reproduction phase occurs within the bacterial population. The bacteria are sorted based on the nutrients they have acquired during preceding chemotactic processes, with those in the upper half considered to have obtained adequate nutrients for reproduction. Each bacterium in this group undergoes division, making two copies at the same place. On the other hand, bacteria in the lowest half of the population die and are eliminated, therefore the population size remains constant. This selective process ensures that individuals with higher nutrient levels survive and replicate, thereby enhancing the thorough exploration of potential optimal areas. Notably, reproduction occurs after all chemotactic steps. The current health status of the i^{th} bacterium can be described as:

(6):

$$J_{he}^{i} = \sum_{j=1}^{N_c+1} J(i,j,k,l)$$

3.1.3 Elimination and dispersal

In natural environments, fluctuations such as sudden changes in temperature, nutrient levels, or water flow can significantly impact the behavior and survival of bacterial populations. To mimic this phenomenon, the BFO algorithm incorporates an eliminate-dispersal mechanism. After every N_{re} reproduction step, an eliminate-dispersal event occurs. During this event, each bacterium is subjected to a random number generator within the range of zero to one. If the generated number falls below a predetermined threshold, denoted as Pe, the old bacterium is eliminated and a fresh one is released into the surroundings. One way to think of this process is as moving the bacterium to a randomly chosen location. Eliminating-dispersal events have the potential to impede chemotactic development, but they can also enhance solutions by placing bacteria in more favorable locations. Unlike reproduction, this operator enhances algorithm diversity. In the BFO algorithm, eliminate-dispersal events are repeated for N_{ed} iterations.

3.2 Particle swarm optimization (PSO)/ the bacterial foraging optimization (BFO)

The question arises why the PSO algorithm?

The PSO algorithm has the potential to enhance the parameters of the BFO algorithm in solving the shortest path problem by leveraging its unique strengths within the optimization process, in which:

- PSO is renowned for its capacity to conduct global exploration within the search space. By incorporating PSO alongside BFO, the amalgamated algorithm can effectively traverse a bsroad spectrum of potential solutions, ensuring thorough exploration of the search space and avoiding the oversight of promising regions. This global exploration capability facilitates the identification of diverse paths and prevents the algorithm from becoming trapped in local optima.
- PSO algorithms typically exhibit faster convergence rates compared to BFO, particularly in specific scenarios. By integrating PSO, the hybrid algorithm can leverage the speed and convergence properties of PSO, thereby expediting the identification of optimal or near-optimal paths in the shortest path problem. Consequently, this may lead to reduced computational time and heightened efficiency.
- PSO encourages diversity throughout the search process by maintaining a swarm of candidate solutions. This diversity proves advantageous in forestalling premature convergence and fostering the discovery of varied paths amidst dynamic obstacles. Through the integration of PSO with BFO, the algorithm can ensure the exploration of a broad array of paths, thereby augmenting the resilience of the solution.
- PSO algorithms inherently possess adaptability to dynamic environments owing to their adeptness at swiftly responding to alterations in the search landscape. When employed in conjunction with BFO for the shortest path problem, PSO facilitates the algorithm's adaptation to changes in obstacle positions or environmental conditions, empowering the robot to navigate adeptly within dynamic scenarios.
- PSO can be employed for optimizing BFO algorithm parameters, such as step sizes, chemotaxis rates, or other control parameters. By incorporating PSO for parameter optimization, the hybrid algorithm can dynamically adjust these parameters during the optimization process, thereby enhancing the solution's performance and robustness. Overall, the integration of PSO algorithms with BFO in the shortest path problem presents a synergistic approach that merges PSO's global exploration capabilities with BFO's adaptive search mechanisms, culminating in improved path planning performance within dynamic environments.

To optimize the hyperparameters of the BFO algorithm with the PSO algorithm in the shortest path problem, we follow these specific steps:

1. BFO algorithm's hyperparameters to optimize: the size of the bacteria population, and chemotaxis parameters.
2. The objective function: is the total length of the path found by the algorithm. The goal is to minimize this length, which is equivalent to finding the shortest path between a start and an end while avoiding obstacles.
3. Establish the ranges of possible values for each hyperparameter to optimize. The size of the bacteria population could range from a few individuals to several dozen, and movement rates should be adjusted within a certain range of speeds v_i^t.
4. Configuration of the PSO algorithm with its hyperparameters: swarm size, inertia coefficients, maximum and minimum velocities with random values.
5. Performance evaluation: For each particle, evaluate your performance using BFO with the current chemotaxis rate, by updating positions and speeds.
6. Adjust the positions and speeds of each particle as a function of their $pbest$ position and the $gbest$ position.
7. Chemotaxis rate update: Use performance particles to update the chemotaxis rate.
8. Use the PSO algorithm to optimize the values of the BFO algorithm's hyperparameters by adjusting them iteratively towards optimal values. The position update equation for each particle in PSO is given by: (7):

$$x_i^{t+1} = x_i^t + v_i^t$$

Where x_i^t is the current position of particle *i* at iteration t, and v_i^t is the velocity of particle *i* at

iteration t. The velocity v_i^t is updated using the following:
(8):

$$v_i^{t+1} = wv_i^t + c_1 r_1 (pbest_i^t - x_i^t) + c_2 r_2 (gbest^t - x_i^t)$$

Where w is the inertia weight, c_1, and c_2 are acceleration coefficients, r_1 and r_2 are random numbers between 0 and 1, $pbest_i^t$ is the best position of particle i found so far, and $gbest^t$ is the best position found by any particle in the swarm at iteration t.

PSO's global exploration mechanism encourages particles to explore new regions of the search space. This is achieved through the velocity update equation, where the global best-known position $gbest^t$ influences the movement of particles towards promising regions of the search space.

While PSO primarily focuses on global exploration, it also incorporates a degree of local search through its velocity update equation. The term $pbest_i^t - x_i^t$ represents the difference between the current position of particle i and its best-known position, encouraging particles to move towards regions of the search space where improvements have been observed.

PSO exhibits adaptive behavior through its inertia weight w, which controls the balance between exploration and exploitation. By adjusting the inertia weight dynamically during the optimization process, PSO can adapt its search strategy based on the current state of the search space, ensuring effective exploration and exploitation.

PSO's convergence speed is influenced by its inertia weight w and acceleration coefficients c_1, and c_2. Higher values of w promote global exploration, while higher values of c_1, and c_2 enhance local search. By appropriately tuning these parameters, PSO can achieve faster convergence to optimal or near-optimal solutions.

9. During iterations of the PSO algorithm, evaluate the length of the path found by the BFO algorithm with the current hyperparameter values, and gradually adjust the hyperparameter values based on the total length of the path found.
10. Evaluate the performance of the BFO algorithm with the optimized hyperparameter values on different instances.
11. Adjust the parameters of the PSO algorithm and repeat the optimization process until the desired performance is achieved
12. Stopping Criterion: Continue iterating until achieve several maximum iterations.

PSO maintains population diversity through the swarm of particles exploring the search space simultaneously. This diversity prevents premature convergence by ensuring that the swarm explores a wide range of potential solutions. As a result, the hybrid approach benefits from diverse solutions generated by PSO, leading to improved robustness and adaptability in dynamic environments.

By integrating PSO with BFO and considering these equations, the hybrid approach can effectively improve the parameters of BFO in the shortest path problem, enabling efficient and robust path planning in dynamic environments.

4 Simulation and results

We have developed a control strategy to direct the robot along a path that avoids collisions, even in environments with obstacles. By conducting simulations with obstacles of different shapes and sizes, we have thoroughly assessed the efficacy of our proposed approach. Our findings indicate that the robot navigates to its destination safely, without experiencing any collisions.

4.1 Results

Numerous advancements have been made in research concerning robot navigation and path planning within dynamic environments, addressing various challenges from different perspectives. Some focus on tracking dynamic targets. However, literature related to solving mobile robot navigation problems in unknown environments often provides only simplified diagrams, typically featuring small working areas with few obstacles. Consequently, direct comparison data is scarce.

We have broadened our analysis to include a diverse range of environmental scenarios relevant to mobile robot navigation. This includes both static and dynamic obstacles as well as varying complexities in the environment. These additions provide a more comprehensive view of how the hybrid BFO-PSO algorithm performs under different conditions.

We conducted an ablation study to separately evaluate the performance of pure BFO, pure PSO, and the hybrid BFO-PSO algorithm. This study helps in understanding the specific advantages of combining these algorithms. The results from the ablation study are now detailed in the manuscript.

We have included a table that summarizes the performance metrics of each algorithm across different scenarios. Metrics such as success rate, average path length, and computation time are compared. The hybrid BFO-PSO approach shows a higher success rate and shorter path lengths compared to pure BFO and PSO in both static and dynamic environments.

Table 2: Comparison of performance with different scenarios of pure BFO, PSO and hybrid BFO-PSO

Algorithm	Scenario	Success Rate (%)	Average Path Length	Computation Time (s)
Pure BFO	Static Obstacles	85	12.3	4.5
Pure PSO	Static Obstacles	88	11.7	3.9
Hybrid BFO-PSO	Static Obstacles	92	10.5	4.0
Pure BFO	Dynamic Obstacles	78	14.2	5.2
Pure PSO	Dynamic Obstacles	82	13.0	4.8
Hybrid BFO-PSO	Dynamic Obstacles	91	11.0	4.3

We have also included a table presenting statistical analysis of the performance metrics, including mean values, standard deviations, and significance testing results. This analysis supports our findings and provides additional context for the performance of each algorithm.

Table 3: Comparison in terms of different metrics between pure BFO, PSO and hybrid BFO-PSO

Metric	Pure BFO	Pure PSO	Hybrid BFO-PSO
Mean Success Rate (%)	81.5	85.0	91.5
Mean Path Length	13.0	12.4	10.8
Mean Computation Time (s)	4.85	4.35	4.15
Std Dev Success Rate (%)	4.5	3.8	2.9
Std Dev Path Length	1.2	1.0	0.9
Std Dev Computation Time (s)	0.6	0.5	0.4

The results highlight that the hybrid BFO-PSO algorithm outperforms both pure BFO and PSO in terms of success rate and path length, particularly in dynamic environments. This demonstrates the effectiveness of the hybrid approach in handling complex navigation tasks.

The hybrid method also shows improved computational efficiency, with reduced computation times compared to pure BFO while maintaining competitive performance against pure PSO. This indicates that the hybrid approach not only enhances navigation accuracy but also offers better efficiency.

The ablation study shows that the hybrid algorithm effectively combines the strengths of both BFO and PSO, providing enhanced robustness and adaptability. The hybrid approach leverages the complementary strengths of the individual algorithms to achieve superior overall performance.

Our scalability analysis includes both a complexity evaluation and a performance assessment across various simulated scenarios.

- Complexity analysis

- ***Time complexity*****:** The time complexity is estimated at $\theta(n.(m+k))$ where n is the number of iterations, m represents the number of obstacles, and k denotes the number of particles in the PSO. This reflects the workload required for BFO's local search combined with PSO's global optimization within the simulation.
- ***Space complexity*****:** The space complexity is approximately $\theta(k.d)$, where d is the dimensionality of the search space. This metric reflects the storage requirements for particle positions and velocities in the simulated environment.

- Performance Metrics

The hybrid algorithm was tested in simulated environments of different sizes and varying obstacle densities. The following tables present the results:

Table 4: Performance metrics by environment size in simulations

Environment Size	Number of Obstacles	Success Rate (%)	Average Path Length	Computation Time (s)
100x100	50	92	208	5.1
200x200	100	89	251	7.4
300x300	150	85	291	10.2
400x400	200	81	326	13.8

Table 5: Performance metrics by obstacle density in simulations

Obstacle Density (%)	Environment Size	Success Rate (%)	Average Path Length	Computation Time (s)
10	200x200	93	251	6.2
20	200x200	88	285	7.1
30	200x200	84	315	8.4
40	200x200	80	347	9.9

The results indicate a slight decrease in success rate as the size of the simulated environment increases. This is likely due to the added complexity of navigating larger spaces with more obstacles. Nonetheless, the hybrid BFO-PSO algorithm maintains robust performance, with high success rates even in the largest simulated environments.

As obstacle density increases, both the success rate and computation time are affected. The algorithm adapts well to these conditions, but the trend suggests that performance optimization may be necessary for environments with very high obstacle densities.

The increase in computation time with larger environment sizes and higher obstacle densities reflects the growing complexity of the tasks. Despite this, the hybrid approach demonstrates efficiency in simulated environments, making it promising for complex real-world applications.

The robustness and adaptability of the hybrid BFO-PSO algorithm were tested under various dynamic scenarios in simulated environments. These scenarios included sudden obstacle appearances, path blockages, and other real-time changes. The results were analyzed in terms of the algorithm's ability to re-plan paths and maintain high success rates under these challenging conditions.

➢ Dynamic scenario testing

We conducted a series of simulations where the environment's conditions changed dynamically, challenging the algorithm to adapt in real-time. The following key scenarios were tested:

- ***Sudden obstacle appearance*:** An obstacle was introduced suddenly on the robot's path.
- ***Path blockage*:** A previously clear path was entirely blocked, requiring the robot to find an alternative route.
- ***Dynamic obstacle movement*:** Obstacles moved during the robot's navigation, simulating real-world dynamic changes.

➢ Performance metrics

The following tables summarize the algorithm's performance across these dynamic scenarios:

Table 6: Performance in sudden obstacle appearance scenarios

Environment Size	Success Rate (%)	Average Re-planning Time (s)	Computation Time (s)	Path Length Increase (%)
100x100	91	0.8	5.3	12.0
200x200	87	1.1	7.8	15.5
300x300	82	1.4	11.0	18.3
400x400	78	1.7	14.5	20.7

Table 7: Performance in path blockage scenarios

Environment Size	Success Rate (%)	Average Re-planning Time (s)	Computation Time (s)	Path Length Increase (%)
100x100	89	1.2	5.8	14.5
200x200	85	1.6	8.2	17.0
300x300	80	2.0	11.7	21.4
400x400	75	2.3	15.3	24.6

The hybrid BFO-PSO algorithm demonstrated robust adaptability by successfully re-planning paths in real-time when faced with sudden obstacle appearances and path blockages. The average re-planning time remained low, indicating the algorithm's capability to quickly adjust to dynamic changes.

Although there was a slight decrease in success rates and an increase in path length when navigating larger or more complex environments, the algorithm maintained a high degree of robustness. Even under extreme conditions, such as total path blockages, the algorithm was able to find alternative routes effectively.

The computation time increased with environment size and complexity, as expected. However, the algorithm's performance remained efficient, with re-planning times staying within acceptable limits for real-time applications. This demonstrates the practical relevance of the hybrid approach, particularly in scenarios requiring immediate adaptability.

The increase in path length in response to dynamic obstacles and blockages reflects the trade-offs inherent in real-time adaptability. While the paths may not always be the shortest, they are optimized for feasibility and safety, ensuring that the robot can navigate successfully even in highly dynamic environments.

To assess real-time performance of path planning, we compared running times in a standardized environment under identical conditions, employing a combination of PSO method with GA, Artificial NN algorithm, and ASTAR Algorithm. This comparison showcased the capability of our algorithm to achieve globally near-optimal paths.

We observed different algorithms guiding robots to their destination's collision-free. Firstly, we introduced a novel approach utilizing the PSO method with a BFO Algorithm for mobile robot path planning. Our method leveraged the core BFO algorithm, including PSO computation, resulting in significantly shorter execution times compared to other algorithms. The distinct parameter selection for path points contributed to this efficiency.

The BFO Algorithm, rooted in computational intelligence, offers advantages such as reduced computational burden, global convergence, and versatility in handling multiple objective functions. We also utilized GA for navigation control, noting its ability to generate progressively improved results despite initial solution generation time. Additionally, we explored Artificial Neural Networks (ANN), focusing on training the network using Q Learning and Backpropagation algorithms to enable obstacle avoidance in obscure environments.

Comparatively, the ASTAR algorithm efficiently optimized robot paths, even in chaotic environments, yielding good solutions in limited scenarios. However, its complexity increases with distant start and endpoints. ASTAR outperformed other algorithms in terms of speed on small-scale inputs, as indicated by output sequence analysis. However, its effectiveness varies based on map size and specific scenarios.

While the ASTAR Algorithm demonstrates effectiveness in low-size inputs, challenges arise with larger map sizes. GA exhibits high memory requirements compared to ANN. Despite challenges, the ASTAR Algorithm surpasses its counterparts in efficiency. The presented table illustrates the fitness variation over generations in genetic algorithms, highlighting the minimization of the fitness function. Notably, the application of the ANN Algorithm struggled in highly complex environments and semi-maze scenarios.

The advantages of the hybrid algorithm combining bacterial foraging optimization (BFO) and particle swarm optimization (PSO) compared to using a single PSO or single BFO algorithm:

- Balanced Exploration and Exploitation: The hybrid algorithm merges the global exploration capabilities of PSO with the local search and exploitation abilities of BFO. This fusion ensures a harmonious balance between exploration and exploitation, enhancing the efficiency of solution search within the shortest path problem's search space.

- Adaptability to Diverse Environments: Through the fusion of BFO and PSO, the algorithm gains the flexibility to adjust to various environmental conditions, whether static or dynamic. This adaptability proves invaluable, particularly in environments where the search landscape undergoes temporal changes.
- Mitigation of Premature Convergence Risks: By integrating the search mechanisms of PSO and BFO, the hybrid algorithm mitigates the risk of prematurely converging to local optima. This capability enables the algorithm to sustain exploration within the search space even after promising solutions are discovered, potentially yielding higher-quality outcomes.
- Enhanced Robustness and Reliability: The amalgamation of diverse optimization methodologies bolsters the algorithm's resilience and dependability. Leveraging multiple search mechanisms, the hybrid algorithm diminishes the likelihood of becoming trapped in suboptimal regions of the search space, thus facilitating the discovery of quality solutions under challenging circumstances.
- Performance Enhancement: Through meticulous parameter adjustments of both algorithms, the hybrid algorithm can be fine-tuned to maximize performance in addressing the specific challenges of the shortest path problem. This optimization process often yields superior solutions and reduced computational overhead compared to utilizing PSO or BFO in isolation.

In summary, the hybridization of BFO and PSO presents numerous advantages for tackling the shortest path problem, including achieving a balanced exploration-exploitation trade-off, heightened adaptability to diverse environments, mitigated risks of premature convergence, enhanced robustness and reliability, and optimized performance.

Table 8: A comparison between algorithms about time and path length in three different environments

	Environment	Theoretical distance	Path length (PL)	Convergence rate
BFO/PSO	[10X 10] Start: (0, 0) Goal: (9, 9) Obstacles: 10	12.72	22	**100%**
BFO			23	**95.65%**
PSO			22	**100%**
GA			22	**100%**
ANN & QL			23	**95.65%**
A*			22	**100%**
BFO/PSO	[100X 100] Start: (0, 0) Goal: (99, 99) Obstacles: 100	140.00	218	**97.24%**
BFO			235	**90.21%**
PSO			225	**94.22%**
GA			240	**88.33%**
ANN & QL			238	**89.07%**
A*			212	**100%**
BFO/PSO	[1000X 1000] Start: (0, 0) Goal: (999, 999) Obstacles: 1000	1412.79	2155	**86.96%**
BFO			2237	**83.77%**
PSO			2201	**85.14%**
GA			2269	**80.44%**
ANN & QL			2705	**68.28%**
A*			/	**Stack Overflow**

5 Discussion

The robot is able to move from the initial to the final position in a dynamic environment without collision with obstacles, the chosen path is optimum. This part clearly shows the effective working of our algorithm. The comparative analysis can be done easily by looking at the various runs. The ASTAR algorithm was much more efficient and exceeded its counterparts. It was better to find the results early. Although we know that the performance of various algorithms will change according to the parameters and input size, we can easily notice, by looking into the other algorithms, how they behave with different inputs. The genetic Algorithm is the least efficient for a maze situation. If the input size is immense, ANN will be the best. A better comparative analysis could be done in the future. Some new results are likely to come up for different input parameters that may be experimented in the future.

6 Conclusion

In environments characterized by dynamic obstacles obstructing the path of mobile robots, path-planning algorithms face intricate computational challenges demanding high-performance computing capabilities.

These challenges involve calculating new paths and determining trajectories promptly to meet the stringent demands of local controllability within short timeframes. In our research, we presented a hybrid autonomous navigation approach tailored for mobile robots navigating environments replete with obstacles. Our approach combines the Particle Swarm Optimization (PSO) algorithm with the Bacterial Foraging Optimization (BFO) Algorithm to efficiently identify the shortest path amidst dynamically moving obstacles, facilitating smooth navigation from initial to final positions without encountering collisions. By leveraging the strengths of both PSO and BFO, our method optimizes path planning in real-time, ensuring adaptive responses to changing environmental conditions.

Simulation results offer compelling evidence of the effectiveness of our approach, demonstrating the successful navigation of a robot through complex and dynamic environments while adeptly circumventing obstacles along its route. This accomplishment underscores the practical utility of our method in real-world scenarios, particularly in applications where swift and safe navigation is paramount, such as emergency response scenarios and industrial automation.

Furthermore, the versatility of our approach extends beyond robot navigation, encompassing a broad spectrum of applications. For instance, in robotic surgery, our method can assist in planning optimal paths for surgical instruments to navigate complex anatomical structures safely. In the realm of video game artificial intelligence, it can enhance the realism and strategic decision-making capabilities of virtual agents navigating dynamic environments. Similarly, in architectural design, our method can aid in optimizing pedestrian flow within complex architectural spaces, ensuring efficient circulation while minimizing congestion.

Looking ahead, our research trajectory includes plans to expand the applicability of our approach to accommodate scenarios involving multiple cooperating robots and mobile manipulators. By further refining our hybrid navigation framework and exploring collaborative strategies, we aim to unlock new possibilities for autonomous navigation in increasingly complex and dynamic environments.

References

[1] A. Benmachiche, B. Tahar, L. M. Tayeb, and Z. Asma (2016). A dynamic navigation for autonomous mobiles robots. *Intell. Decis. Technol.*, vol. 10, no. 1, pp. 81–91. https://doi.org/10.3233/idt-150239.

[2] A. Benmachiche, A. Makhlouf, and T. Bouhadada (2020). Optimization learning of hidden Markov model using the bacterial foraging optimization algorithm for speech recognition. *Int. J. Knowl.-Based Intell. Eng. Syst.*, vol. 24, no. 3, pp. 171–181. https://doi.org/10.3233/kes-200039.

[3] L. Zhang, Y. Zhang, and Y. Li (2020). Mobile robot path planning based on improved localized particle swarm optimization. *IEEE Sens. J.*, vol. 21, no. 5, pp. 6962–6972. https://doi.org/10.1109/jsen.2020.3039275.

[4] Q. Fan, Y. Zhang, and N. Li (2021). An autoselection strategy of multiobjective evolutionary algorithms based on performance indicator and its application. *IEEE Trans. Autom. Sci. Eng.*, vol. 19, no. 3, pp. 2422–2436. https://doi.org/10.1109/tase.2021.3084741.

[5] Ö. Ekrem and B. Aksoy (2023). Trajectory planning for a 6-axis robotic arm with particle swarm optimization algorithm. *Eng. Appl. Artif. Intell.*, vol. 122. https://doi.org/10.1016/j.engappai.2023.106099.

[6] M. A. Alam and M. O. Faruq (2019). Finding shortest path for road network using Dijkstra's algorithm. *Bangladesh J. Multidiscip. Sci. Res.*, vol. 1, no. 2, pp. 41–45. https://doi.org/10.46281/bjmsr.v1i2.366.

[7] I. Umay, B. Fidan and W. Melek (2019). An Integrated Task and Motion Planning Technique for Multi-Robot-Systems. *IEEE International Symposium on Robotic and Sensors Environments (ROSE)*, Ottawa, ON, Canada, pp. 1-7. https://doi.org/10.1109/rose.2019.8790413.

[8] B. R. Kiran *et al.* (2021). Deep reinforcement learning for autonomous driving: A survey. *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 6, pp. 4909–4926. https://doi.org/10.48550/arXiv.2002.00444.

[9] J. Borenstein and Y. Koren (1989). Real-time obstacle avoidance for fast mobile robots. *IEEE Trans. Syst. Man Cybern.*, vol. 19, no. 5, pp. 1179–1187. https://doi.org/10.1109/robot.1990.126042.

[10] S. Chehelgami, E. Ashtari, M. A. Basiri, M. T. Masouleh, and A. Kalhor (2023). Safe deep learning-based global path planning using a fast collision-free path generator. *Robot. Auton. Syst.*, vol. 163, pp. 104384.

[11] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang (2023). Path planning techniques for mobile robots: Review and prospect. *Expert Syst. Appl.*, vol. 227. https://doi.org/10.1016/j.eswa.2023.120254.

[12] S. Mbakop, G. Tagne, S. V. Drakunov, and R. Merzouki (2021). Parametric ph curves model based kinematic control of the shape of mobile soft manipulators in unstructured environment. *IEEE Trans. Ind. Electron.*, vol. 69, no. 10, pp. 10292–10300. https://doi.org/10.1109/tie.2021.3123635.

[13] A. Benmachiche, A. Makhlouf, and T. Bouhadada (2019). Evolutionary learning of HMM with Gaussian mixture densities for Automatic speech recognition. *In Proceedings of the 9th International Conference on Information Systems and Technologies*, pp. 1–6. https://doi.org/10.1145/3361570.3361591.

[14] A. Benmachiche, A. A. Betouil, I. Boutabia, A. Nouari, K. Boumahni, and H. Bouzata (2022). A fuzzy navigation approach using the intelligent lights algorithm for an autonomous mobile robot. *In International Conference on Computing and*

Information Technology, Springer, pp. 112–121. https://doi.org/10.1007/978-3-031-25344-7_11.

[15] D. K. Mishra, A. Thomas, J. Kuruvilla, P. Kalyanasundaram, K. R. Prasad, and A. Haldorai (2022). Design of mobile robot navigation controller using neuro-fuzzy logic system. *Comput. Electr. Eng.*, vol. 101, pp. 108044. https://doi.org/10.1016/j.compeleceng.2022.108044.

[16] W. M. Hassen, S. H. Amin, and A. S. Al-Araji (2023). Hybrid Swarm Algorithm for Mobile Robot Path Planning. *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 11, no. 9, pp. 947-957. https://doi.org/10.17762/ijritcc.v11i9s.9996.

[17] B. Patle, A. Pandey, D. Parhi, A. Jagadeesh, and al. (2019). A review: On path planning strategies for navigation of mobile robot. *Def. Technol.*, vol. 15, no. 4, pp. 582–606. https://doi.org/10.1016/j.dt.2019.04.011.

[18] B. Hilali, M. Ramdani, and A. Naji (2023). Neuro-Fuzzy Combination for Reactive Mobile Robot Navigation: A Survey. *Indones. J. Electr. Eng. Inform. IJEEI*, vol. 11, no. 2, pp. 375–388. https://doi.org/10.52549/ijeei.v11i2.4009.

[19] K. Khnissi, C. Seddik, and H. Seddik(2018). Smart navigation of mobile robot using neural network controller. in *2018 International Conference on Smart Communications in Network Technologies (SaCoNeT)*, IEEE, pp. 205–210. https://doi.org/10.1109/saconet.2018.8585616.

[20] A. Mellouk and A. Benmachiche (2020). A survey on navigation systems in dynamic environments. in *Proceedings of the 10th International Conference on Information Systems and Technologies*, pp. 1–7. https://doi.org/10.1145/3447568.3448527.

[21] A. Loganathan and N. S. Ahmad (2023). A systematic review on recent advances in autonomous mobile robot navigation. *Eng. Sci. Technol. Int. J.*, vol. 40, p. 101343. https://doi.org/10.1016/j.jestch.2023.101343.

[22] C. E. Llorente-Peralta, L. Cruz-Reyes, and R. A. Espín-Andrade (2021). Knowledge discovery using an evolutionary algorithm and compensatory fuzzy logic. *Fuzzy Log. Hybrid Ext. Neural Optim. Algorithms Theory Appl.*, pp. 363–383. https://doi.org/10.1007/978-3-030-68776-2_21.

[23] X. Qi (2024). Event-Triggered Predictive Control Algorithm for Multi-AUV Formation Modeling. *Informatica journal*, vol. 48, pp. 127–142. https://doi.org/10.31449/inf.v48i9.5890.

[24] A. K. De, D. Chakraborty, and A. Biswas (2022). Literature review on type-2 fuzzy set theory. *Soft Comput.*, vol. 26, no. 18, pp. 9049–9068. https://doi.org/10.1007/s00500-022-07304-4.

[25] A. J. Muñoz-Vázquez, V. Parra-Vega, A. Sánchez-Orta, and J. D. Sánchez-Torres (2021). Adaptive fuzzy velocity field control for navigation of nonholonomic mobile robots. *J. Intell. Robot. Syst.*, vol. 101, no. 2, p. 38. https://doi.org/10.1007/s10846-020-01306-w.

[26] J. R. Sanchez-Ibanez, C. J. Pérez-del-Pulgar, and A. García-Cerezo (2021). Path planning for autonomous mobile robots: A review. *Sensors*, vol. 21, no. 23, p. 7898. https://doi.org/10.3390/s21237898.

[27] A. Kareem, O. Odeniyi, and N. Lawal (2023). Development of a COVID-19 Patients' Fatality Prediction System Using Swarm Intelligent Convolution Neural Network. *Asian J Res Comput Sci*, vol. 16, pp. 12–35. https://doi.org/10.9734/ajrcos/2023/v16i2336.

[28] J. A. Abdulsaheb and D. J. Kadhim (2023). Classical and heuristic approaches for mobile robot path planning: A survey. *Robotics*, vol. 12, no. 4, p. 93. https://doi.org/10.3390/robotics12040093.

[29] O. Mypati, A. Mukherjee, D. Mishra, S. K. Pal, P. P. Chakrabarti, and A. Pal (2023). A critical review on applications of artificial intelligence in manufacturing. *Artif. Intell. Rev.*, vol. 56, no. Suppl 1, pp. 661–768. https://doi.org/10.1007/s10462-023-10535-y.

[30] Y. Zhou, D. Wang, and L. Liu (2024). Exploring unknown environments: motivated developmental learning for autonomous navigation of mobile robots. *Intell. Serv. Robot.*, vol. 17, no. 2, pp. 197–219. https://doi.org/10.1007/s11370-023-00504-3.

[31] M. E. Hedroug, K. Guesmi, and al. (2024). Fuzzy predictive controller for trajectory tracking of a wheeled mobile robot. *Stud. Eng. Exact Sci.*, vol. 5, no. 1, pp. 449–472. https://doi.org/10.54021/seesv5n1-027.

[32] S. Darvishpoor, A. Darvishpour, M. Escarcega, and M. Hassanalian (2023). Nature-inspired algorithms from oceans to space: A comprehensive review of heuristic and meta-heuristic optimization algorithms and their potential applications in drones, *Drones*, vol. 7, no. 7, p. 427. https://doi.org/10.3390/drones7070427.

[33] Y. Yang (2024). The Impact of GA Optimization Model Under the Constraint of Maximum Inventory on the Logistics Cost Control of Automotive Parts Production in the Factory. *Informatica journal*, vol. 48, no. 11, pp. 1–14. https://doi.org/10.31449/inf.v48i11.5959.

[34] I. Alshawi, H. Al-badrei (2022). Secure Routing Protocol for WSNs Using Bacterial Foraging Optimization and Improved RC4. *Informatica journal*, vol. 46, no. 8, pp. 1–10. https://doi.org/10.31449/inf.v46i8.4277.