Journal of Engineering Management and Systems Engineering
https://www.acadlore.com/journals/JEMSE

Optimizing Path Planning for Smart Vehicles: A Comprehensive Review of Metaheuristic Algorithms

Osinachi Mbah, Qasim Zeeshan

Department of Mechanical Engineering, Eastern Mediterranean University, 99628 Famagusta, North Cyprus via Mersin 10, Turkey

* Correspondence: Qasim Zeeshan (qasim.zeeshan@emu.edu.tr)

Received: 10-10-2023 **Revised:** 11-28-2023 **Accepted:** 12-10-2023

Citation: O. Mbah and Q. Zeeshan, "Optimizing path planning for smart vehicles: A comprehensive review of metaheuristic algorithms," *J. Eng. Manag. Syst. Eng.*, vol. 2, no. 4, pp. 231–271, 2023. https://doi.org/10.56578/jemse020405.

Abstract: In the realm of smart vehicle navigation, both in known and unknown environments, the crucial aspects encompass the vehicle's localization using an array of technologies such as GPS, cameras, vision systems, laser, and ultrasonic sensors. This process is pivotal for effective motion planning within the vehicle's free configuration space, enabling it to adeptly avoid obstacles. The focal point of such navigation systems lies in devising a path from an initial to a target configuration, striving to minimize the path length and the time taken, while simultaneously circumventing obstacles. The application of metaheuristic algorithms has been pivotal in this regard. These algorithms, characterized by their ability to exploit initial solutions and explore the environment for feasible pathways, have been extensively utilized. A significant body of research in robotics and automation has focused on evaluating the efficacy of population-based algorithms including Genetic Algorithm (GA), Ant Colony Optimization (ACO), Particle Swarm Optimization (PSO), Firefly Algorithm (FA), and Whale Optimization Algorithm (WOA). Additionally, trajectory-based methods such as Tabu Search (TS) and Simulated Annealing (SA) have been scrutinized for their proficiency in identifying short, feasible paths among the plethora of solutions. There has been a surge in the enhancement and modification of these algorithms, with a multitude of hybrid metaheuristic algorithms being proposed. This review meticulously examines various metaheuristic algorithms and their hybridizations, specifically in their application to the path planning challenges faced by smart vehicles. The exploration extends to the comparison of these algorithms, highlighting their distinct advantages and limitations. Furthermore, the review delves into potential future directions in this evolving field, emphasizing the continual refinement of these algorithms to cater to the increasingly complex demands of smart vehicle navigation.

Keywords: Metaheuristic algorithms; Path planning; Smart vehicle navigation; Robotics; Automation

1 Introduction

The advent of technologies such as big data, cloud computing, 5G networks [1], the Internet of Things (IoT) [2], and artificial intelligence (AI) [3, 4] has been instrumental in the evolution of smart vehicles. These vehicles leverage these technologies to mitigate human error in driving, navigate traffic in self-driving modes, assist in industrial logistics and manufacturing processes as Automated Guided Vehicles (AGVs), and operate in challenging terrains as Unmanned Ground Vehicles (UGVs). Furthermore, their applications extend into healthcare, domestic settings, and e-commerce through Autonomous Mobile Robots (AMRs).

In the domain of self-driving vehicles, technologies like Light Detection and Ranging (LIDAR) sensors [5], cameras integrated with deep learning algorithms [6], and ultrasonic sensors [7]are employed for vehicle and object detection, traffic alerts, zebra crossing recognition, and collision avoidance with pedestrians. AGVs, utilized in warehouse logistics and material handling, navigate using lasers, magnets, vision cameras, or by following marked lines or wires. The role of AI, particularly reinforcement learning, has become prominent in route planning for

AGVs [8]. AMRs have a wide array of applications including inspection [9], surveillance [10], monitoring [11], logistics, and service [12–15]. They are categorized into holonomic and non-holonomic types [16, 17], with holonomic robots having controllable degrees of freedom equal to their total degrees of freedom [18], allowing movement in any direction within their configuration space. Non-holonomic mobile robots, on the other hand, have constraints on their velocities and derivatives of position [19].

For effective navigation, smart vehicles must comprehend the nature of their environment to adapt their actions for optimal goal attainment. Critical to this process are three fundamental components: mapping, localization, and path planning [20]. Mapping involves the creation or retrieval of environmental maps, providing location and orientation data for the vehicles. Localization is essential for vehicles to ascertain their position on the map, a task accomplished using cameras, GPS, and various sensors like laser, vision, and ultrasonic sensors. The location may be expressed in absolute coordinates (longitude, latitude, altitude), as a reference relative to the environment, or as topographical coordinates (e.g., in a room). Path planning is the process of determining a viable, obstacle-free route in typically congested real-world environments [21].

2 Literature Review

2.1 Path Planning Methods and Algorithms for Smart Vehicles

Path planning in the context of smart vehicles is categorized into two primary approaches: global and local. Global path planning is concerned with deriving the optimal path using extensive environmental data. This approach is most effective in static environments that are well-defined and familiar to the smart vehicle. Here, path planning algorithms generate a complete route from an origin to a destination, thereby determining the optimal trajectory for the vehicle. In contrast, local path planning is pertinent in environments that are either unfamiliar or subject to change. It involves real-time computation while the vehicle is in motion, utilizing data from onboard local sensors. This enables the smart vehicle to adaptively generate new routes in response to dynamic environmental changes. A wide array of path planning methods and algorithms have been explored in the field of robotics. Factors influencing the selection of an appropriate algorithm include the kinematics and dynamics of the environment, the computational capabilities of the smart vehicles, the type of sensors employed, and the availability of other sourced information. The decision-making process regarding the choice of an algorithm also involves considering the trade-offs between algorithmic performance and complexity, which vary depending on the specific application [22]. As illustrated in Figure 1, path planning methods and algorithms can be divided into several categories, such as classical methods like cell decomposition, metaheuristic algorithms including the genetic algorithm, machine learning approaches like reinforcement learning, and sampling methods exemplified by probabilistic roadmaps [22–25].

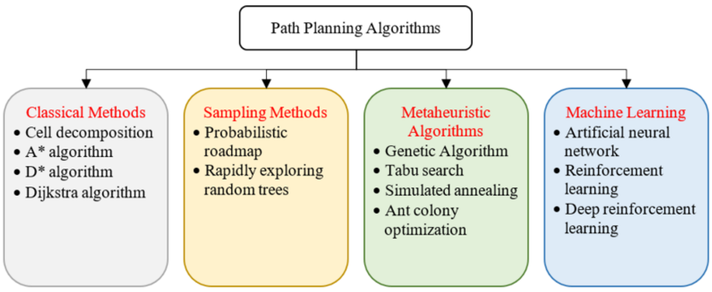

Figure 1. Classification of path planning algorithms

2.2 Metaheuristic Algorithms

Metaheuristic algorithms represent a class of high-level heuristic approaches that are designed to provide suitable solutions to optimization problems, particularly those characterized by incomplete information or limited computational resources. When applied to path planning, metaheuristic algorithms demonstrate proficiency in managing environments that are partially known or contain moving obstacles. This is in contrast to classical algorithms, which typically necessitate prior comprehensive knowledge of the environment [23]. Metaheuristic algorithms are categorized into two main groups: population-based methods like Particle Swarm Optimization and trajectory-based methods such as Simulated Annealing, as depicted in Figure 2 [26]. Population-based metaheuristics operate by generating multiple points within the search space, whereas trajectory-based methods progress through the search space by navigating a trajectory via a single point at each time step.

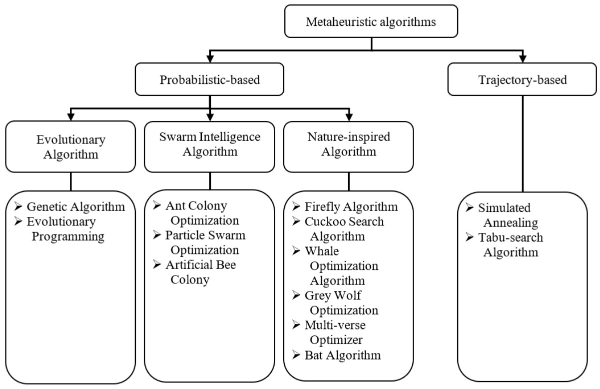

Figure 2. Classification of metaheuristic algorithms

Table 1. Pseudocode of GA

Genetic Algorithm
Choose encode method
$G \leftarrow 0$
$G_{max} \leftarrow Maximum\ generation$
Initialize population
for ($G < G_{max}$) **do**
for (*i=1 to maximum population*) **do**
Evaluate fitness of individual i
end for
Selection
Crossover
Mutation
Move new individuals to population G + 1
$G \leftarrow G + 1$
end for
return *best individual*

2.2.1 Genetic Algorithm (GA)

The GA, an optimization methodology, draws upon the principles of genetics and natural selection, first conceptualized by Bremermann [27]. Holland was the pioneer in adapting the genetic algorithm to computer science [28]. Its applications have since permeated various domains, including robot navigation and numerous scientific and technological fields. This algorithm focuses on optimizing complex problems where the objective function needs to be maximized or minimized within specified constraints. The method starts by defining a population size, where chromosomes (sets of genes) are formulated based on the given problem. Each chromosome in the population is assigned a fitness value, contingent upon the objective function. Chromosomes are then selected based on their fitness, allowing them to propagate their genes to subsequent generations through crossover processes. Mutation is employed to maintain population diversity and avert premature convergence. Table 1 elucidates the pseudocode of the genetic algorithm. The algorithm concludes its process once the population has converged [29].

The recent focus on GA-based methods, particularly in the realm of optimization problems like path planning, highlights the potential of GA in addressing these challenges [6]. This is evidenced by the success of GA in various applications, as discussed in Table 2, which outlines different studies that have employed GA for path planning. This table includes the variables considered in each study and provides insights into their findings. Moreover, the hybridization of GA with other intelligent algorithms has been an area of considerable research interest. Notable examples include the integration of GA with Fuzzy Logic [30], Intelligent Water Drop [31], and Neural Network [32], aiming to enhance the efficacy of the solutions. In the utilization of GA-based methods for path planning, distance often emerges as a common parameter [33–37], alongside other considerations such as path smoothness and clearance [34, 36, 37], energy evaluation [38], and factors related to robot speed. A noteworthy study by Liu et al. [39] presented an improved GA to tackle the appointment order allocation and route planning issues of Cainiao unmanned vehicles. Additionally, Wang et al. [40] proposed an optimized approach using GA to implement the Multi-Objective Evolutionary Algorithm (MOEA) for planning the trajectory of a mobile robot in a known environment. This experiment involved a two-wheeled mobile robot using the ArUco system within the Robotic Operating System (ROS). However, it's important to note the limitations of this method, particularly its unsuitability for rough terrains due to the omission of the mobile robot's dynamics in the planning process. Furthermore, the algorithm's deployment on a console computer, rather than within the robot's embedded system, is attributed to the limitations of the embedded system's low-end hardware.

Table 2. GA for path planning of smart vehicles

Types	Initial Population Generation Method	Population Size	Reproduction Operators	Fitness Function
Improved GA	Random	100		$F = \frac{1}{C+MP}$
Novel GA	CBPRM	$20, 25, 50$	Crossover: Ordinary Mutation: Change, smooth and shortcut operators	$\mathrm{C(k) = L(k) \times S(k)}$
hTetre-GA	Random	$25, 50, 100$	Crossover: Single-point crossover operator Mutation: Classic GA mutation operator Removal Operator: Translational motion command, Non-translational motion command Rearrangement Operator	$F = \frac{1}{1+W_{A*}\left(\mathrm{POS}\left(p_{l_{p-1}}\right)\right)}$
Novel GA	Random	20	Crossover: Single-point with crossover rate $(\mathrm{pc}) = 1.0$ Mutation: Bitwise flipping with mutation rate $(\mathrm{pm}) = 0.1(1/\text{string length})$	
GA	Random		Crossover: single-point	$F = N - (\alpha_1 \sum_{i=1}^{n} d_i + \alpha_2 * m)$

Types	Sorting and Selection Technique	Number of Generations	Type of Vehicle	Type of Obstacle
Improved GA	Optimization guidance factor and Roulette selection	500	Multiple	Static
Novel GA		50	Single	Static
hTetro-GA	NSGA-II technique	50	Single (Reconfigurable tilted robot)	Static (H-Shaped, Spiral and 3-Slit) Dynamic (Perpendicular and Parallel)
Novel GA	Parent Selection: Rank-based roulette wheel Survivor selection: Elitism + crowding distance (NSGA-II)	2000	Single (Two wheeled mobile robot running on STM32 microcontroller)	Static
GA	Roulette	Between 100 and 500 generations	Single	Static (Special, Regular, Irregular multiple)

Types	Type of Map	Software	Remarks	Ref
Improved GA	Topological	MATLAB R2018a	Order allocation and route planning problem is modelled to obtain efficient picking of orders. Provides the optimal solutions to unmanned vehicle inputs and their path planning.	[39]
Novel GA	Geometrical	CGAL 3.3.1 [41]	Minwoski sum is used as a computational geometry based approach instead of cell based methods. CBPRM speeds up the evolutionary process.	[42]
hTetro-GA	24×24 Grid	Robotic Operating System (ROS)	NSGA-II technique is implemented to determine best motion command sequence. hTetro-GA algorithm can be implemented for reconfigurable robots during a rescue task.	[43]
Novel GA	10×10 Grid	Robotic Operating System (ROS) and Python 3	Experimented mobile robot is localized by the ArUco system through a bird's view camera. Mobile robot can't operate on rough environment because its dynamic behaviour is not considered in the work.	[40]
GA	100×100 Geometrical		Large turning angle problem in basic genetic algorithm is overcome. Genetic algorithm implementation is separate from path smoothing process. Experimental results are compared with Dijkstra algorithm	[44]

Note: NSGA-II: Non-dominated sorting genetic algorithm-II; CBPRM: Clearance Based Probabilistic Roadmap Method; hTetro-Ga: hinged-Tetromino-Genetic Algorithm

2.2.2 Ant Colony Optimization (ACO)

The concept of ACO was first introduced by Dorigo in his Ph.D. dissertation in 1992 [45]. This algorithm is inspired by the sophisticated social behaviors exhibited by ants during their search for food. A key element of this behavior is the deposition of pheromones, which serve to guide other ants by creating a trail to the food source. The trail's pheromone concentration intensifies as more ants traverse it, thereby increasing the likelihood of it being followed by additional ants. Notably, the shortest route to the food source becomes the most popular among the ants, as it can be traversed in the least amount of time. This phenomenon was first observed in the renowned Double Bridge experiment [46], where ants consistently selected the shortest path over time when presented with multiple routes to a food source. Pheromone evaporation also plays a crucial role in this process. It serves as a mechanism to prevent the ants from getting trapped in locally optimal solutions [47]. As the pheromone evaporates, the attractiveness of a given path diminishes, reducing the likelihood of it being selected by other ants. Additionally, on the shortest path, the rate of pheromone deposition surpasses its rate of evaporation, ensuring that a high pheromone level is maintained. In the context of ACO algorithms, this concept is applied to the selection of paths between nodes. The probability of an ant, situated at node i, choosing to move to another node j in the network, is influenced by the level of pheromone deposition on the potential paths, as described in reference [47].

$$\mathrm{p}_{ij}^{k}=\begin{cases}\frac{\left(\tau_{ij}^{k}\right)^{\alpha}\left(\eta_{ij}^{k}\right)^{\beta}}{\sum_{i\in \mathrm{N}_{i}^{k}}\left(\tau_{ij}^{k}\right)^{\alpha}\left(\eta_{ij}^{k}\right)^{\beta}} & \text{if } j\in \mathrm{N}_{i}^{k}\\ 0 & \text{if } j\notin \mathrm{N}_{i}^{k}\end{cases} \tag{1}$$

where, τ_{ij}^{k} denotes pheromone levels. Analogous to the natural tendencies of ants, paths with elevated pheromone concentrations are more likely to attract ants in the algorithm, leading to a preference for these paths over others with lower pheromone levels [48].

Table 3. Pseudocode for ACO

Ant Colony Optimization
Initialize nodes and necessary parameters
Initialize pheromone level of each node
Define maximum iterations ITR
while (*ITR>0*) **do**
for *each ant k* **do**
$\eta_j \leftarrow$ *heuristic function of the search space (fitness value)*
Transition_probability $[j] \leftarrow p_{ij}^{k}(t)$
Select node with the highest $p_{ij}^{k}(t)$
Update pheromone level $\tau_{ij}(t+1)$
end for
ITR=ITR-1
end while
Best solution $\leftarrow$ *solution with best* η_j
return *Best solution*

The heuristic function: $\eta_{ij}^{k}=\frac{1}{d_{je}}$.
The pheromone update:

$$\tau_{ij}(t+1)=(1-\rho)\tau_{ij}(t)+\rho*\Delta\tau_{ij}(t)+q*\Delta\tau_{ij}^{b}(t) \tag{2}$$

$$\Delta\tau_{ij}(t)=\sum_{k=1}^{m}\Delta\tau_{ij}^{k}(t) \tag{3}$$

$\Delta\tau_{ij}^{k}(t)=\begin{cases}\frac{Q}{L_k}, & \text{if ant } k \text{ passes node } i \text{ and } j\\ 0, & \text{otherwise}\end{cases}$ is the quantity of pheromone deposited, where Q is a constant and L_k is the total length of the path that ant k travels. A pseudocode of this algorithm is presented in Table 3.

$$L_k=\sum_{i=1}^{n-1}\sqrt{\left(x_{i+1}-x_i\right)^2+\left(y_{i+1}-y_i\right)^2} \tag{4}$$

Numerous scholars have conducted comprehensive research on the operational mechanics, structural design, and optimal parameterization of ACO, proposing various enhancements to address these areas, as detailed in Table 4. Additionally, a range of variables, as listed in Table 5, have been considered in these studies. Liu et al. [49] optimized cross-path nodes in the path search process using the ant colony algorithm combined with geometric optimization, which improved the algorithm's effectiveness and path quality via pheromone updates. You et al. [50] developed a novel heuristic operator to augment the diversity and convergence of the population search. Dai et al. [51] addressed issues related to global convergence speed and path smoothing by enhancing an A* algorithm-based ACO and the maximum-minimum ant system, incorporating a retraction mechanism to circumvent deadlocks. Jiao et al. [52] proposed an adaptive state transfer and pheromone update method, enhancing the significance of heuristic information and pheromone strength in the iterative process of the algorithm, thereby improving its adaptability to diverse environments and its capacity to escape local optima. Akka and Khaber [53] refined the state transfer formula to prioritize the selection of neighbor nodes with the most exits as the subsequent node. This enhanced algorithm introduces diversity to the search process and mitigates the impact of ineffective pheromones by dividing the multi-heuristic function, separately rewarding and penalizing the worst path, as outlined by Yang et al. [54].

Table 4. Applied ant Colony Optimization Algorithm and its hybrids for path planning of smart vehicles

Types	m	α	β	ρ	Q	N_{max}	φ	γ	ξ	υ	δ	Selection of Next Node
An improved ant colony algorithm (ACO-PD)	10	1.1	$\frac{p}{12}$	0.5		200		0.01	$\sqrt{2}$			
DL-ACO [PEACO and TPOA]	20 10	1 0.3	3 0.8	0.03 0.1		100 100	1	0.9				Roulette wheel selection
LF-ACO	50	1	3	0.3	100	50	1		10	5	20	Roulette wheel
APACA	120	1	5	0.9	100	200						
RMACA	50	1	5	0.5	10	54 % lower than [55]						
IACA	30	1	5		2	100						
Ant Colony Optimization and Fuzzy Control	80	1	9	0.5	1	100						Roulette wheel
IACO-A*	50	0,1, 2,3, 4,5	0,1, 3,5, 7,9	0,0.1, 0.3,0.5, 0.7,0.9	1	100						
IAACO	50	1	7		2.5	100			1			

Types	Type of Vehicle	Type of Obstacles	Type of Maps	Software
An improved ant colony algorithm (ACO-PD)	Single	Static	Grid	
DL-ACO [PEACO and TPOA]	Single (Rikirobot)	Static	Grid	
LF-ACO	Multiple robot	Static	Grid	MATLAB and Robotic Operating System (ROS)
APACA	Single (Smart wheelchairs)	Static	20 × 20 Grid (in subgraph (a) of Figure 3)	
RMACA	Single	Static	Grid. (Common, tumnel,trough, baffle maps)	MATLAB
IACA	Single	Static	Grid	MATLAB
Ant Colony Optimization and Fuzzy Control	Single	Static	20 × 20 Grid	MATLAB
IACO-A*	Single	Static	20 × 20 Grid (in subgraph (c) of Figure 3)	MATLAB R2020b
IAACO	Single	Static	20 × 20 Grid in subgraph (b) of Figure 3)	

Types	Remarks	Ref.
An improved ant colony algorithm (ACO-PD)	• Proposed method solves convergence speed problem in ACO. • Geometric optimization method is implemented to improve path generated by ACO.	[49]
DL-ACO [PEACO and TPOA]	• PEACO and TPOA is combined to generate path and avoid obstacles. • Proposed method gives better results in path distance, smoothness and good solution rate when compared to APACA and MO-FA. • Experimentation is done with Rikirobot powered by Raspberry Pi and Rplidar A1. • Implements piecewise B-Spline to smoothen path.	[56]
LF-ACO	• Proposed method aims to solve multi-robot path planning. • Pheromone update in ACO incorporates pheromones of leader and follower ant. • Maximum-minimum ant approach is employed for global search. • Generated path is optimized by TPOA and dynamic cut-point method.	[54]
APACA	• Implementation of Direction determining Method to speed up convergence rate for global optimal search. • Proposed method shows better outcomes in number of iterations and path length when compared to IACA and GPACA.	[52]
RMACA	• Retraction mechanism is employed to avoid local minimum. • Improved Maximum-minimum ant approach performs global search. • Estimation function in A* improves search efficiency of ACO. • RMACA is better in convergence rate and bending suppression effect.	[51]
IACA	• Stimulating probability is introduced to improve transition rule. • Unlimited step length principle is used as heuristic information for path search efficiency. • Dynamic change in evaporation rate increases convergence speed.	[53]
Ant Colony Optimization and Fuzzy Control	• Fuzzy algorithm controls α and β to adjust evaporation rate. • Proposed critical obstacle influence factor generates initial pheromone distribution.	[57]
IACO-A*	• Proposed modelled environment is based on geometric modelling and Monte Carlo calculation. • Proposed method gives optimum results in terms of path length, cumulative radiation dose and energy consumption when compared to other algorithms.	[58]
IAACO	• The transition probability is induced with angle guiding factor and obstacle exclusion factor to enhance path search efficiency. • Heuristic information adaptive adjustment factor and adaptive pheromone volatilization factor are introduced into the pheromone update rule for optimum global search.	[59]

Note: DL-ACO: Double Layer-ACO; PEACO: Parallel Elite Ant Colony Optimization; TPOA: Turning Point Optimization Algorithm; APACA: Adaptive Polymorphic Ant Colony Algorithm; Retraction Mechanism Ant Colony Algorithm; IACA: Improved Ant Colony Optimization Algorithm; MO-FA: Multiobjective Firefly Algorithm; IACO-A*: Improved Ant Colony Optimization algorithm-modified A*

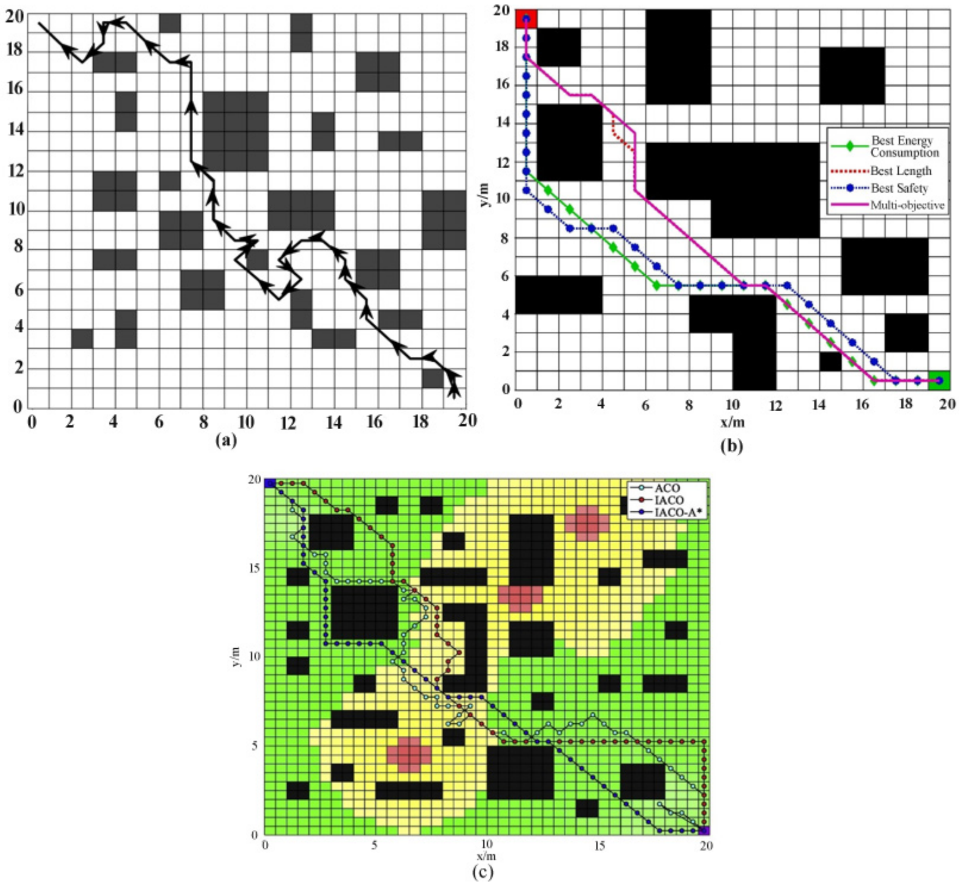

Figure 3. Sample maps implemented for ant Colony Optimization Algorithm: (a) APACA 20×20 grid map [52]; (b) IAACO 20×20 grid map [59]; (c) IACO-A* 20×20 grid map [58]

Table 5. Variables used in various ACO

Variables	Description
m	Ant's population size
N_{max}	Maximum iteration number
α	Weight of Pheromone
β	Weight of Heuristic information
ρ	Pheromone evaporation ratio
Q	Pheromone's Intensity
τ_{ij}	Pheromone on the path between i and j
η_{ij}	Heuristic information on j
ξ	Distance factor coefficient
φ	Distance correction parameter
v	Ant's importance on moving straight
δ	Parameter to update Pheromone
γ	Diffusion coefficient

2.2.3 Particle Swarm Optimization (PSO)

PSO is inspired by the collective behavior of animals like birds, tetrapods, or fish in their pursuit of food. This approach mirrors the natural group dynamics observed in these species, where there is no singular leader guiding the group to the food source [60]. In such a group, each individual may not know the precise location of the food, but they can approximate their proximity to it. Independent efforts by each animal to reach the food source would be inefficient, leading to extended time frames and chaos. Consequently, the most effective strategy is for the members to follow those closest to the food source [29]. In PSO, each individual animal is analogous to a solution, possessing two critical pieces of information: Their fitness value, derived from the objective function; The velocities that guide the solution towards the target location.

The algorithm commences with a set of solutions or particles. Each particle navigates the solution space, returning their fitness value, known as pbest, in each iteration. The best pbest value from each iteration is recorded as the global best value, gbest. Based on these two values, the algorithm updates the velocity and position of each particle. The search process concludes either upon reaching the maximum number of iterations or upon identifying the optimal solution. The formulas for updating the velocity and position of particles in PSO are outlined subsequently.

$$v_{id}(t+1) = w * v_{id}(t) + c_1 * r_1 * (p_{id} - x_{id}(t)) + c_2 * r_2 * (p_{gd} - x_{id}(t)) \tag{5}$$

$$x_{id}(t+1) = x_{id}(t) + \eta * v_{id}(t+1) \tag{6}$$

The inertia weight is denoted as w, c_1, c_2 are the learning factors, r_1, r_2 are normal distribution random numbers within the interval $[0, 1]$, η represents the velocity constraint proportional factor, v_{id} represents the velocity of the i-th particle in d dimension, and x_{id} represents the position of the i-th particle in d dimension. The procedure to implement this algorithm is described in Table 6.

Table 6. Pseudocode of particle swarm optimization

Particle Swarm Optimization

```
Initialize particle population size S
GEN ← 0
Initialize GEN_max
for (each particle i = 1,...,S) do
    Initialize particle's position x_i
    Initialize particle's best known position to initial position:  p_i ← x_i
    Initialize particles velocity v_i
    if fitness(p_i) > fitness(g) then
        g ← p_i
    end if
while GEN < GEN_max do
    for each particle i = 1,...,S do
        for each dimension d = 1,...,n do
            r_1, r_2 ← random normal distribution in [0,1]
            Update particle's velocity v_id
            Update particle's position x_id
        end for
        if fitness(x_i) > fitness(p_i) then
            p_i ← x_i
            if fitness(p_i) > fitness(g) then
                g ← p_i
            end if
        end if
    end for
    GEN=GEN+1
end while
Best solution ← solution with best η_j
return Best solution
```

The PSO is analogous to the GA in its initiation with a randomly formed population set, subsequently evaluated based on fitness values. This methodology has been effectively applied in various navigation contexts, such as aerial robot navigation in unknown three-dimensional environments [61], humanoid robot navigation [62], and industrial robot navigation [63]. The performance efficacy of PSO is contingent on the precision in adjusting, controlling, and updating its parameters.

Since its inception in 1995, numerous approaches have been suggested to refine these aspects and enhance the overall functionality of PSO. Traditional techniques for parameter adjustment and control include Fixed Inertia Weight (FIW) [64, 65], Linearly Decreasing Inertia Weight (LDIW) [64–67], Time Varying Acceleration Coefficient (TVAC) [65, 68], Random Inertia Weight (RANDIW) [64–66, 68], Random Acceleration Coefficients (RANDAC) [69], and Fixed Acceleration Coefficients (FAC) [64–66, 68].

Table 7 compiles various studies that have proposed improvements and hybridizations of PSO to address path planning challenges. Dewang et al. [70] introduced an adaptive particle swarm optimization (APSO) that dynamically alters the inertia weight in each iteration, initiating the search with a high inertia weight to avoid local minima, and gradually reducing it to focus on exploitation as iterations progress. This strategy yielded superior results in comparison to standard PSO in terms of path length and planning time, as demonstrated in the environment depicted in subgraph (a) of Figure 4. Chai et al. [71] combined PSO with the Probabilistic

Roadmap Method (PRM) to enhance sampling in PRM. This hybrid method leverages knowledge of sample points in obstacle-laden regions to refine sampling in open spaces, particularly in narrow passages, thereby improving connectivity. Masehian and Sedighizadeh et al. [72] utilized PSO to derive the shortest and smoothest feasible paths. Particles are initialized based on points identified in free space by a robot's laser sensor, with the optimal particle's position determined by the sensor readings. PRM serves as the local planner for obstacle avoidance, and simulation results indicate a more efficient runtime compared to the basic PRM approach. Li et al. [73] presented an improved PSO that initializes particles through uniform random distribution, employs an exponentially decaying inertia weight to enhance planning efficiency, and integrates cubic spline interpolation for path smoothing. This variant was benchmarked against other PSO variants, with comparisons based on performance indices and path planning metrics, where IPSO showed promising results. Song et al. [74] developed a fractional-order PSO variant (FOPSO) that introduces adaptive fractional-order velocity and utilizes Bezier curves for path smoothing. Its performance was evaluated against other PSO variants using benchmark functions. Finally, Alam et al. [75] implemented PSO for random sampling along grid lines between start and goal points, with an initial spacing of points along the Euclidean path. The effectiveness of this approach was validated through simulations in various environments with static obstacles.

Table 7. Applied Particle Swarm Optimization and its hybrids for path planning of smart vehicles

Types	**Particle Size**	**Inertia Weight (w)**	**Cognitive Factor (c1)**	**Social Factor (c2)**	**Number of Generations**
Hybrid PSO	20	10%	1.5	1.5	500
EDPSO	150	1	0.4	0.4	150
PSO-AWDV	200	0.9, 0.5	2.0, 1.0	1.0, 2.0	100
Improved PSO	10	0.4, 0.9	2	2	3000
FIMOPSO	50	0.4 to 0.9			100

Types	**Type of Vehicle**	**Type of Obstacles**	**Type of Map**	**Software**
Hybrid PSO	Single (Mobile robot)	Dynamic	200 × 200 Geometrical	MATLAB 2018b
EDPSO	Single	Static and Dynamic	20 × 20 Geometrical (see in subgraph (d) of Figure 4)	
PSO-AWDV	Single	Static	11 × 11 Geometrical (see in subgraph (c) of Figure 4)	
Improved PSO	Single	Static	18 × 18 Geometrical	MATLAB R2016a
FIMOPSO	Single	Static and dynamic	210 × 178 Geometrical (in subgraph (b) of Figure 4)	MATLAB

Types	**Remark**	**Ref.**
Hybrid PSO	•The performance of hybrid PSO and ACO on shortest path and least time constraint is measured against PSO and ACO separately. •Although it's observed that PSO outperforms ACO, the hybrid gives a superior results.	[76]
EDPSO	•Peaks of diversity in population gives room for more exploration in the search space. •Can be applied to smart vehicles with slow movements. •Comparison was made on all the cited meta-heuristics using 10 complex multi-model functions from the CEC 2019 benchmarking suite.	[77]
PSO-AWDV	•Quartic Bezier transition curve with three control points is implemented to smoothen planned path.	[78]
Improved PSO	•Proposed mutation operation increases particle diversity. •Proposed de-redundant algorithm removes needless points, thus improving generated path.	[79]
FIMOPSO	•Constraints to be minimized are path length, motor torque, travel time, robot acceleration; obstacle avoidance is maximized. •Obstacle avoidance problem is solved with Fuzzy inference system.	[80]

Note: EDPSO: Enhanced Diversity Particle Swarm Optimization; PSO-AWDV: Particle Swarm Optimization - Adaptive Weighted Delay Velocity; FIMOPSO: Fuzzy enhanced Improved Multi-objective Particle Swarm Optimization

2.2.4 Artificial Bee Colony (ABC)

The ABC algorithm, conceived by Dervis Karaboga for addressing polynomial mathematical problems [81], is inspired by the foraging behavior of honey bees. In their natural environment, honey bees use pheromones and a waggle dance to communicate information about food sources. When a bee finds a food source, it evaluates the nectar quantity, returns to the hive, and performs a waggle dance to convey information about this source. The quality of the food source is indicated by the vigor of the waggle dance. In the context of the ABC algorithm, the location of a food source represents a potential solution to an optimization problem, and the quality of the solution is analogous to the nectar content of the food source. The ABC algorithm categorizes bees into three roles: employed bees, onlooker bees, and scout bees. It is assumed that for each food source position, there is one corresponding employed bee. Employed bees share information about the location and quality of food sources with onlooker bees through the waggle dance. Onlooker bees then select food sources based on their perceived quality, meaning that higher-quality sources are more likely to be chosen. If employed bees abandon a food source, they transition into scout bees, embarking on the search for new food sources. Scout bees memorize the quality of discovered food spots and compare them with known sources to identify the most promising ones. The ABC algorithm's pseudocode is illustrated in Table 8. Initially, the ABC algorithm establishes a population of food source positions (SN), where SN represents the population size. Each food source, or solution, is a D-dimensional vector, with D being the number of optimization parameters. Each food source is linked to a probability value p_i, influencing the decision-making of onlooker bees.

Table 8. Pseudocode of ABC

Artificial Bee Colony Algorithm

```
Initialize bee population size SN = number of employed bees = number of observer bees
Evaluate fitness of each bee f(sol)
Set best solution, solBest ← sol
ITR ← 0
Initialize ITR_max
while ITR < ITRmax do
        for each employed bee i = 1 ,..., SN do
            Select random solution and apply random neighbourhood structure
            Determine the probability of each solution, p_i
        end for
        for each employed be do
            sol ← select solution with highest probability
            apply random neighbourhood structure
            If f(sol) < f(solBest) then
                solBest ← sol
            end if
        end for
        ITR = ITR + 1
end while
return Best solution, solBest
```

$$p_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{7}$$

where, fit_i is the fitness of solution i, and SN is number of employed bees (population size). The generation of a new food source position from an existing one is determined using the following expression:

$$v_{ij} = x_{ij} + \emptyset_{ij} \left(x_{ij} - x_{kj} \right) \tag{8}$$

where, k and j are random values in sets $\{1, 2 \ldots, \text{SN}\}$ and $\{1, 2, \ldots, \text{D}\}$ respectively. k should be different from i. $\emptyset_{ij} \in [-1, 1]$ controls the production of a neighbour food source position around x_{ij}.

Research in the field of ABC for path planning has led to a variety of advancements and hybrid approaches, as detailed in Table 9. One notable development in the navigation of smart vehicles is the combined Artificial Bee Colony and evolutionary programming approach proposed by Contreras-Cruz et al. [82]. In this methodology, the ABC algorithm serves as the local search method, while evolutionary programming is employed to enhance the potential paths obtained. This approach was initially applied in multi-robot systems and later refined for use in unfamiliar environments with dynamic obstacles [83]. However, certain shortcomings were identified, such as neglecting the distance between new bee positions and obstacles. To address these issues, an improved ABC-Evolutionary Programming (ABC-EP) approach was proposed by Kumar and Sikander [84]. Further advancements include the Adaptive Dimension Limit-Artificial Bee Colony Algorithm (ADL-ABC), introduced by Kamil et al. [85] for optimizing the global path of mobile robots. This algorithm demonstrated its efficacy by finding solutions with fewer iterations and reduced computational time. Another development, the Directed Artificial Bee Colony algorithm, was shown to yield better results in dense environments, such as maps with numerous static obstacles, compared to other leading algorithms [86]. In an effort to curtail computational time, particularly crucial in real-time path planning scenarios, Szczepanski and Tarczewski [87] proposed a hybrid approach combining the ABC and Dijkstra algorithms. This amalgamation aimed to balance the comprehensive search capabilities of the ABC algorithm with the efficiency of Dijkstra's algorithm, particularly in environments where quick computation is vital.

Table 9. ABC for path planning of smart vehicles

Types	Population Size	Number of Iterations	Type of Vehicle	Type of Obstacles	Type of Map
ADL-ABC	100	1000	Single	Static	10×10 Geometrical
ABC-EP	10	500 generations (for EP)	Single (Xidoo-Bot, mobile robot Pioneer 3-AT)	Static	Various Geometrical and Grid maps. (see Figure 5)
Improved ABC-EP	200, 400, 600, 800, 1000	100	Single	Static and Dynamic	100m×100m Geometric
Enhanced ABC	25	100	Single	Static	100×100 Grid

Types	Software	Remark	Ref.
ADL-ABC	MATLAB	•Implemented dynamic control limit reduces computational time and number of iterations. •Generated path is smoothened using cubic polynomial through three via points. •Better results are observed when proposed method is compared to ABC.	[85]
ABC-EP	C language	•While ABC builds a path in collision free space, EP improves the path using mutation to produce a short path. •Proposed approach was implemented in some benchmark maps. •ABC-EP is deployed to an experimental platform to show its feasibility.	[82]
Improved ABC-EP	MATLAB 2018b	•Best food points among randomly distributed ones which aid in finding optimum path are selected its distance to goal point and nearest obstacle. •When determining the optimum path, takes into account the distance between the new bee position (best node) and any surrounding barriers.	[84]
Enhanced ABC		•Cubic Ferguson spline is introduced to smoothen path generated by ABC.	[88]

Note: ADL-ABC: Adaptive Dimension Limit – Artificial Bee Colony; ABC-EP: Artificial Bee Colony – Evolutionary Programming.

2.2.5 Firefly Algorithm (FA)

The FA, introduced by Yang [89], is inspired by the bioluminescent communication of fireflies. The key aspect of this communication is the distinctive light patterns emitted by fireflies, where the intensity of the light is indicative of a firefly's attractiveness. In the context of the FA, this attractiveness is analogous to the fitness value of a solution. The algorithm operates on the principle that among any two fireflies, the one emitting brighter light will attract the one with dimmer light. The attractiveness of a firefly is quantified as per the following equation [90]:

$$\beta = \beta_0 e^{-\gamma {r_{ij}}^2} \tag{9}$$

where, β denotes the attractiveness, β_0 represents the maximum attractiveness, which is typically set to $1, \gamma$ is the light absorption coefficient ranging between $[0.1, 10]$, and r_{ij} is the distance between firefly i and firefly j, calculated using the standard Euclidean distance formula:

$$x_i = x_i + \beta\left(x_j - x_i\right) + \alpha(\phi - 0.5) \tag{10}$$

where, x_i and x_j are the positions of firefly i and firefly j in the d-dimensional space, respectively; α and ϕ are random numbers in the distribution $[0, 1]$. The pseudocode for FA is described in Table 10.

Table 10. Pseudocode of FA

Firefly Algorithm
Objective function $f(\mathrm{x}), \mathrm{x} = (x_1, x_2 \ldots, x_d)$
Initialize population size $x_{\mathrm{i}}(i = 1, 2, \ldots, n)$
Determine the intensity (I) of each firefly determined by $f(x)$
Initialize GEN_{max}
while (GEN $<$ GEN_{max}) **do**
for (i = 1 to n) **do**
for (j = 1 to i) **do**
If $(I_j > I_i)$ **then**
Vary attractiveness with distance r via exp $(-\gamma r)$
move firefly i towards j
Evaluate new solutions and update light intensity
end if
end for
end for
Rank fireflies and find the current best
GEN = GEN + 1
end while
return Best firefly

The FA has been implemented in diverse research areas, including robotics [91, 92], machine learning [93], journalism [94], and cloud computing [95]. Table 11 presents a compilation of techniques proposed for FA's application in smart vehicle navigation. In an effort to enhance the convergence speed and local search accuracy of the standard FA, Chen et al. [96] introduced a modified version (PPMFA) that incorporates a Gaussian random number into the fixed step size. This addition enhances the diversity of the firefly population, helping to prevent the algorithm from stagnating in dead-end zones. Additionally, a novel path center technique was employed to calculate distances between fireflies, essentially representing paths. This method involves connecting geometric centers of path segments to form new segments and repeating the process until a single segment remains, termed as the path center. The distance between two path centers is used as the measure of distance between two fireflies. The PPMFA showed superior performance in accuracy and convergence speed compared to PSO and the Standard SFA. Duan et al. [97] proposed a Developed Firefly Algorithm (DFA) to address multi-objective navigation challenges. The algorithm extends the grid map to create feasible paths and employs the Pareto dominance relationship for path comparison and segregation. Non-dominant fireflies are stored in an elite record library for comparison during iterations. DFA includes an evolutionary stage for optimizing paths by adding, removing, or swapping points. When tested against NSGA-II on a ZDT1 instance, the DFA demonstrated enhanced efficiency. Patle et al. [98] utilized the firefly algorithm for mobile robot navigation in dynamically obstacle-laden environments. An AI mechanism navigates the robot, while a controller based on FA detects and avoids obstacles, generating paths (fireflies) and selecting the optimum one using Euclidean distance from the nearest obstacle. This approach outperformed other intelligent methods in terms of path length in various scenarios. Experiments with the Khepera-II robot showed a deviation of about 5.7% from simulated results. Hassan and Fadhil [99] developed a modified firefly algorithm for path planning of mobile robots in 3D sphere-like, partially dynamic environments. In this approach, each firefly is viewed as an agent navigating around obstacles, with the generated paths considered potential solutions. The optimal path is selected based on path length and completeness. This modified approach was noted for its minimal memory requirement and effective performance in spherical spaces.

Table 11. FA for path planning of smart vehicles

Types	Population Size	Generation	γ	β_0	α	Type of Vehicle	Type of Obstacles
FFA FAMCPSO MO-FA	200	150	1	2	0.5	Single Single Single	Static Static Static
FA-TPM	5 - 100	50 – 100	0.1 - 1	0.1-1	0.1-1	Single (Fire Bird V robot- NEX Robotics and Embedded Real-Time Systems Lab, CSE IIT Bombay)	Dynamic

Type	Type of Map	Software	Remark	Ref.
FFA	10×10 Geometrical		•A* algorithm is implemented to obtain the shortest path. •Cubic polynomial spline is interpolated on the generated path to produce smooth trajectory using iterative random selection.	[91]
FAMCPSO	600cm×800cm Geometrical	MATLAB 2018b	•Proposed method is a combination of MCPSO and FA. •Consideration of inverse dynamic and kinematic modelling to obtain optimum torque and velocity for wheels of AMR. •The recommended hybrid method shows good results when compared to various algorithms in different environments.	[100]
MO-FA	Grid (see subgraph (a) of Figure 6)	C/C++ language	•Path safety, the path length, and the path smoothness are considered in the design of proposed method.	[101]
FA-TPM	Grid (see in subgraph (b) of Figure 6)	Microsoft Visual C++, 2010 with OpenGL	•While TPM searches for obstacle free path, FA performs obstacle avoidance.	[102]

Note: FAMCPSO: Firefly Algorithm Modified Chaotic Particle Swarm Optimization; AMR: Autonomous Mobile robot; MCPSO: Modified Chaotic Particle Swarm Optimization; FA-TPM: Firefly Algorithm-Three Path Method.

Table 12. Pseudocode of CSA

Cuckoo Search Algorithm

Objective function $f(\mathrm{x}), \mathrm{x} = (x_1, x_2 \ldots, x_d)$
Initialize population of n host nests
ITR $\leftarrow$ 0
Initialize maximum number of generations ITR_{max}
while (ITR $< ITR_{max}$) **do**
    i $\leftarrow$ Get a cuckoo randomly by levy flight
    Evaluate fitness(i)
    j $\leftarrow$ choose a nest
    if fitness(i) $>$ fitness(j) **then**
        replace j by the new solution
    end if
    Abandon a fraction (Pa) of worst nest and build new ones
    Keep the best nests
    Rank the nests and find the current best
    Pass the current best solutions to the next generation
    ITR = ITR + 1
end while
return Best nest

2.2.6 Cuckoo Search Algorithm (CSA)

Developed in 2009, the CSA is a nature-inspired optimization technique designed to tackle complex problems [103]. Its conceptual framework is based on the intriguing brood parasitism behavior observed in certain cuckoo species. These birds are known for their strategy of laying eggs in the nests of other host species. Cuckoos often adapt the appearance of their eggs, mimicking the color and pattern of the host species' eggs [104], thereby reducing the likelihood of the host species detecting the foreign egg. In cases where the host species identifies and removes the cuckoo egg or abandons its nest to start anew, the cuckoo must find another host nest. Notably, cuckoo eggs typically hatch faster than those of the host species, allowing the young cuckoo to dispose of the host's eggs and monopolize the food supplied by the unwitting host [105]. The behavioral rules of the cuckoo birds, as translated into the CSA, can be summarized as follows:

Each cuckoo randomly selects a nest in which to lay its egg.

The nests that successfully retain cuckoo eggs, escaping detection and eviction by the host species, are carried forward to the next generation.

The probability of a host bird discovering an alien egg is denoted by $P \in [0, 1]$, and the total number of available host eggs or nests within the search space remains constant.

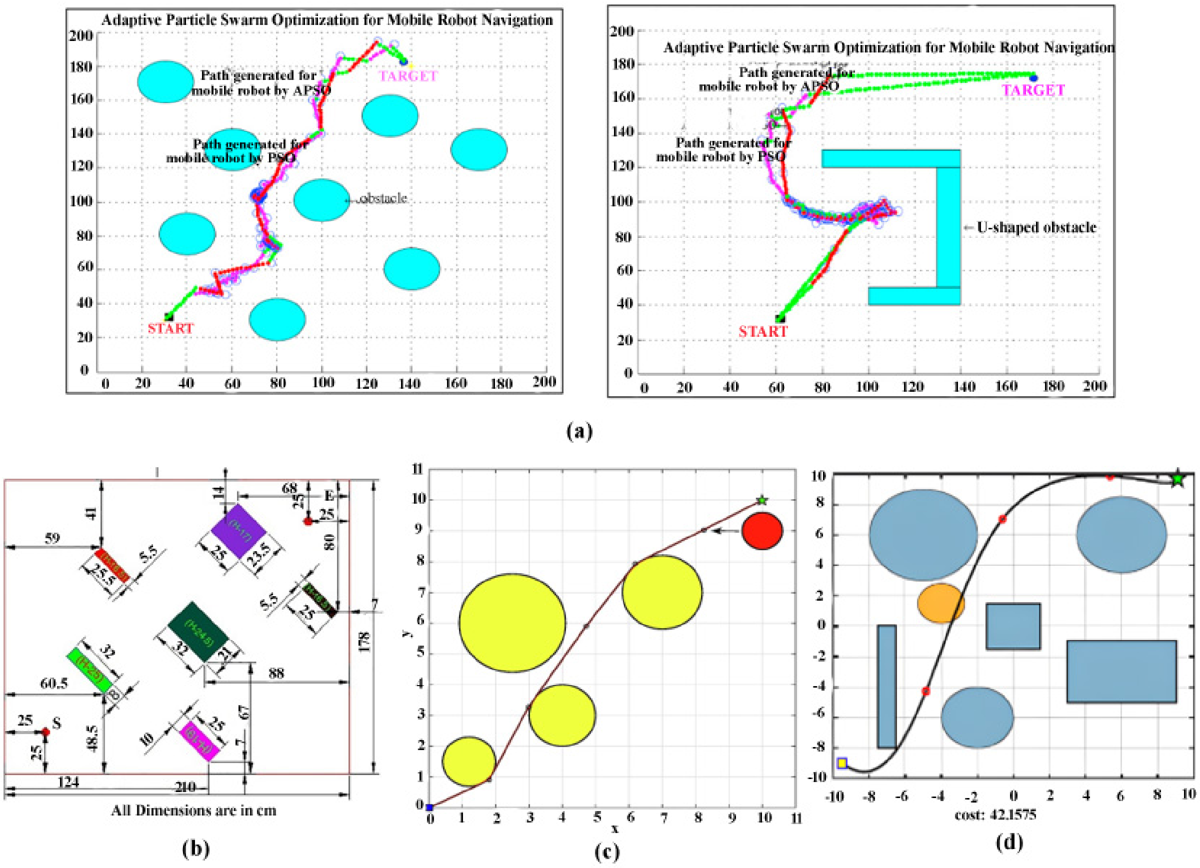

Figure 4. Sample maps implemented for particle swarm optimization: (a) APSO 200×200 map [70]; (b) FIMOPSO 210×178 map [80]; (c) PSO-AWDV 11×11 map [78]; (d) EDPSO 20×20 map [77]

The CSA primarily employs a random walk strategy for nest searching, with Levy flight [103] being the most commonly used method due to its efficiency in exploring the search space. In the context of path planning problems, the nests and eggs within the CSA framework can be metaphorically viewed as solutions, where host eggs in a nest represent current solutions and a cuckoo egg symbolizes a new, potentially superior solution x^{t+1}. The objective is to replace less effective solutions with more viable ones (represented by the cuckoo's egg).

$$x_i^{t+1} = x_i^t + \alpha \oplus \text{Levy}(\gamma) \tag{11}$$

where, i_i represents the i-th particle, t stands for the iteration cycle, $\alpha > 0$ is the step size, and $\oplus$ denotes entrywise multiplication. Step lengths of Levy flight are distributed according to this probability.

$$\text{Levy}(\gamma) = L^{-\gamma}, (1 < \gamma \leq 3) \tag{12}$$

where, L represents step size length and γ denotes the variance. P, γ, and α are are critical to the algorithm's performance and require careful tuning to enhance solution quality. One of the advantages of the cuckoo search algorithm is its minimal need for parameter fine-tuning, coupled with its ability to handle multi-modal objective functions effectively. The implementation of the Cuckoo Search Algorithm is further elucidated in a pseudocode format, as illustrated in Table 12.

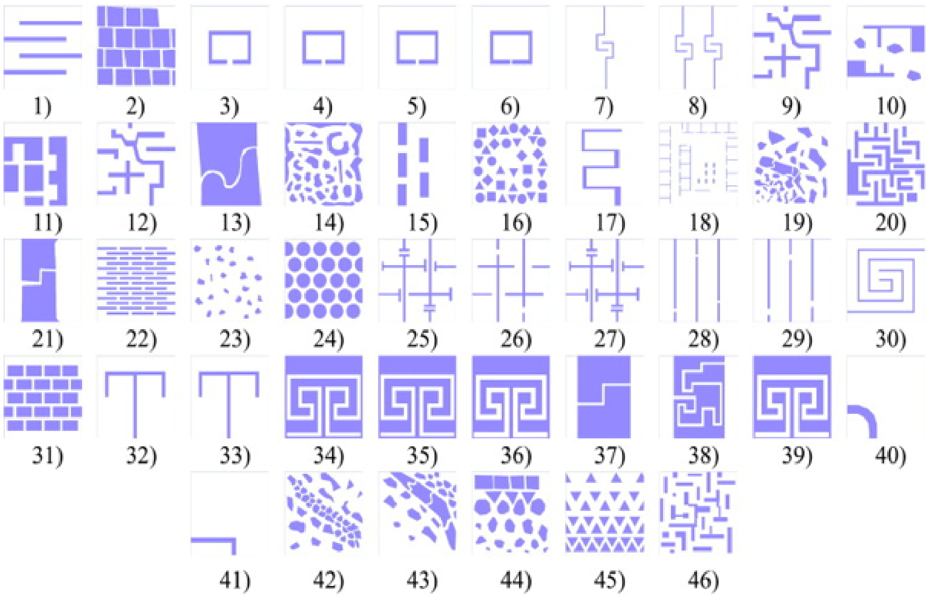

Figure 5. List of benchmark maps used in ABC-EP [82]

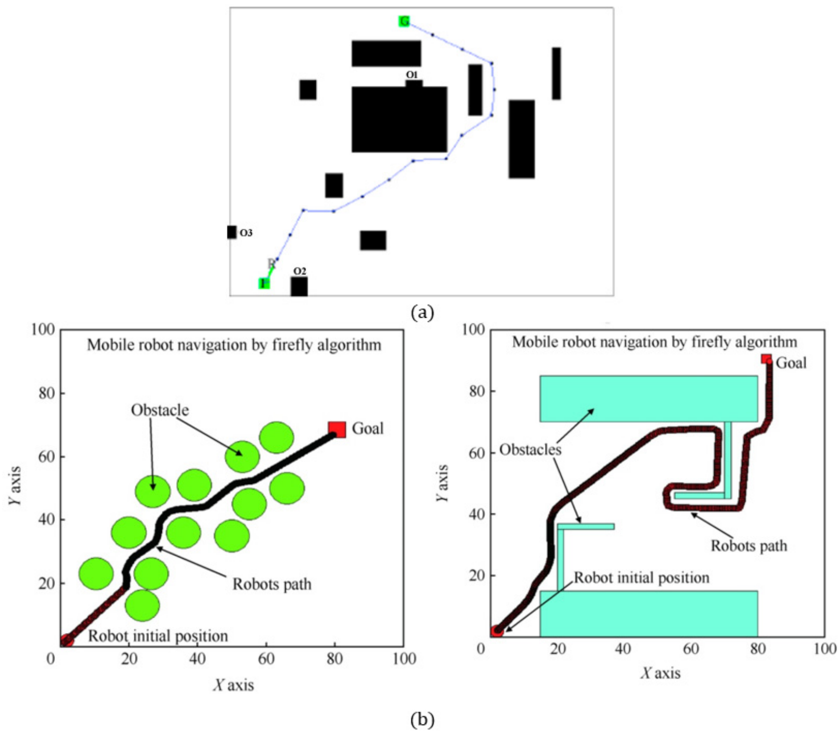

Figure 6. Sample maps implemented for firefly algorithm: (a) FA-TPM [102]; (b) FA 100×100 map [98]

The implementation of the CSA spans a wide array of fields, demonstrating its versatility and effectiveness. This includes applications in vehicle routing [106, 107], neural networks [108], scheduling [109, 110], medical fields [111, 112], cloud computing [113], and notably in robotics, particularly in the navigation of smart vehicles. Table 13 showcases a range of hybrid approaches combining Cuckoo Search with other methods to address path planning challenges. In the realm of multi-robot collaboration and navigation within densely obstacle-laden maps, Sahu et al. [114] introduced a Modified Cuckoo Search as a novel solution. This adaptation of the CSA was specifically designed to enhance collaborative strategies and navigation efficiency in complex environments. A comparative study by Ab Wahab et al. [23] assessed the performance of the cuckoo search against other metaheuristic algorithms and traditional path planning methods across various scenarios. The study highlighted CSA's strengths and potential areas for integration with other techniques. To address the dual goals of minimizing computational costs and maximizing efficiency in mobile robot path planning, Garip et al. [115] proposed a hybrid algorithm that combines the principles of cuckoo search with firefly and particle swarm optimization. This hybrid approach aimed to leverage the strengths of each method to produce a more robust and efficient path planning solution. In the context of quantum computing, Kundra et al. [92] utilized cuckoo search to prevent premature convergence in the proposed quantum firefly algorithm. This application underscores CSA's utility in enhancing the stability and performance of other advanced algorithms. Additionally, to optimize robot paths in environments with dynamic obstacles, Kumar et al. [116] implemented a Modified Cuckoo Search algorithm. This version of CSA was tailored to process obstacle distance and heading angle data from robot sensors, enabling more adaptive and responsive path planning in changing conditions.

Table 13. CSA for path planning of smart vehicles

Types	P	γ	α	Population Size	Number Generations	Type of Vehicle	Type of Obstacles
MCS-SCA-PSO	0.25			30		Multiple (Epuck robot)	Static and dynamic
Improved CSA	0.25			30		Single	Dynamic
Hybrid CSA-BA				20	500	Single	Static
Hybrid genetic-cucking	0.25		1	400	40	Single	Static
CSA-PSO-FA		1	0.2	20	1000	Single and multiple (Kobuki mobile robot)	Static
CSA-BA	0 - 1			30	500	Single	Static

Type	Type of Map	Software	Remark	Ref.
MCS-SCA-PSO	450 × 450 Geometrical (see subgraph (a) of Figure 7)	C language	•PSO performs local search, CSA performs global search, and sine cosine algorithm implements greedy approach.	[117]
Improved CSA	Topological	MATLAB 2014a	•Global search ability is improved by introducing mutation and crossover. •Convergence rate and optimization accuracy of algorithm are tested using unimodal and multimodal functions.	[118]
Hybrid CSA-BA	12 × 12 Geometrical (see in subgraph (b) of Figure 7)	MATLAB	•The proposed approach is implemented in two maps with various positions of circular obstacles.	[119]
Hybrid genetic-cucking	10000 × 8000 × 5000 Geometrical	MATLAB	•Using intelligent algorithms in path planning of 3D environment is studied. •Spherical obstacles are implemented.	[120]
CSA-PSO-FA	200 × 160 and 100 × 100 Grid	MATLAB, ROS	•CS-PSO-FA algorithm is investigated both in simulation and experimentally.	[115]
CSA-BA	12 × 12 Geometrical	MATLAB 2015b	•CSA-BA obtains a better result in path length than CSA and BA.	[121]

Note: CSA-BA: Cuckoo Search Algorithm – Bat Algorithm; CSA-PSO-FA: Cuckoo Search Algorithm – Particle Swarm Optimization – Firefly Algorithm; MCS-SCA-PSO: Modified Cuckoo Search - Sine Cosine Algorithm - Particle Swarm Optimization.

2.2.7 Whale Optimization Algorithm (WOA)

The WOA, introduced by Mirjalili and Lewis [122], is an innovative algorithm that emulates the bubble-net hunting behavior of humpback whales [123]. These whales, known for their social nature, often forage in groups, preying primarily on krill and small fish located near the water's surface. A notable aspect of their hunting technique involves diving approximately 12 meters deep and then engaging in a unique strategy to encircle their prey. This involves creating a ring of bubbles as they spiral upward towards the surface, effectively trapping the prey. This hunting method, particularly the encircling maneuver and the spiral bubble-net movement, has been translated into a mathematical model for the WOA. The algorithm draws inspiration from these distinct whale behaviors to devise a search and optimization strategy. The spiral path and bubble-net formation are key elements that have been abstracted and applied in the algorithm to simulate the whale's approach to localize and encircle the prey effectively. The mathematical representation of these behaviors enables the WOA to efficiently explore and exploit the search space in various optimization problems.

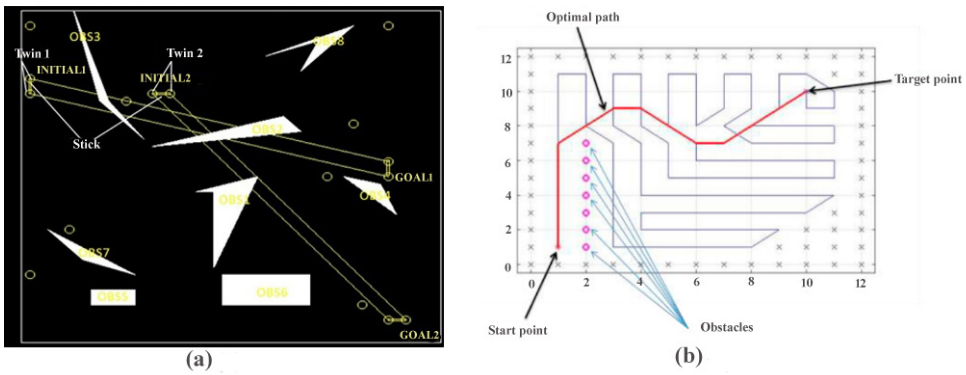

Figure 7. Sample maps implemented for cuckoo search algorithm: (a) MCS-SCA-PSO 450×450 map [117]; (b) Hybrid CSA-BA 12×12 map [119]

(1) Encircling prey

In the WOA, the behavior of a humpback whale encircling its prey is a key mechanism. When a whale identifies the location of its prey, it begins to encircle it. In the context of WOA, this is modeled by assuming that the best current solution in the search space is akin to the whale closest to the target prey. Consequently, other search agents (whales) in the algorithm adjust their positions relative to this best agent, simulating the encircling behavior. This behavior is represented mathematically in the following equations:

$$D = |C \cdot X^*(t) - X(t)| \tag{13}$$

$$X(t+1) = X^*(t) - A \cdot D \tag{14}$$

$$A = 2a \cdot r - a \tag{15}$$

$$C = 2r \tag{16}$$

where, A and C are coefficients, t denotes current iteration, X^* is the best whale position, X is the current whale position, a is a decreasing constant that linearly reduces from 2 to 0 over the course of iterations and is given by $a = 2 - \frac{2t}{M}$ (M: maximum number of iteration), $r \in [0, 1]$ is a random value.

(2) Spiral bubble-net manoeuvre (Exploitation phase)

As the value of a decreases, the radius of encircling the prey diminishes. With A being a random value within the range $[-a, a]$, search agents can discern the relationship between their current position and the optimal position when A is reduced to the interval $[-1, 1]$. Additionally, during the spiral movement phase, the position of the whale relative to the prey is updated. The distance D' between the i-th whale and the prey (which represents the best solution obtained so far) is calculated accordingly:

$$X(t+1) = D' \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t) \tag{17}$$

$$D' = |X^*(t) - X(t)| \tag{18}$$

where, b is a constant that defines the shape of the logarithmic spiral, and l is again a random value in the range [-1, 1]. The spiral movement is an integral part of the whale's hunting strategy, where it combines the encircling maneuver with a simultaneous inward spiral motion towards the prey. In the WOA, the whale's position is updated based on a probability of 50% to either continue encircling the prey or to engage in the spiral movement. This probabilistic approach enables the algorithm to balance between exploration and exploitation, effectively mimicking the hunting behavior of humpback whales. The decision between the two behaviors is governed by the following equation:

$$x(t+1) = \begin{cases} X^*(t) - A \cdot D, & p < 0.5 \\ D' \cdot e^{bl} \cdot \cos(2\pi l) + X^*(t), & p \geq 0.5 \end{cases} \tag{19}$$

(3) Searching for prey (Exploration phase)

In the exploration phase of the WOA, a whale updates its position based on a randomly selected agent, rather than the current best agent. This phase is crucial for global search within the solution space. When $|A| < 1$, the algorithm switches to exploration mode. In this case, given X_{rand} as the position of a random agent, the updated position of a whale is determined using the following equations:

$$D = |C \cdot X_{\text{rand}} - X| \tag{20}$$

$$X(t+1) = X_{\text{rand}} - A \cdot D \tag{21}$$

Table 14. Pseudocode of WOA

Whale Optimization Algorithm
Initialize the whale population $X_i (i = 1, 2, \ldots, n)$
Calculate the fitness of each whale
X_{best} = the best search agent
while ($t <$ maximum number of iterations)
for each search agent
Update a, A, C, l and p
if $(p < 0.5)$ **then**
if $(\lvert A \rvert < 1)$ **then**
Update current agent via Encircling Prey
else
Select a random agent (X_rand)
Update current agent via Search for Prey
else
pdate search agent via Spiral Bubble-net
end for
Amend the position of whales that are outside the search space
Calculate the fitness of each search agent
Update X_{best} if there is a better solution
$t = t + 1$
end while
return X_{best}

The pseudocode detailing the WOA is illustrated in Table 14. This algorithm has been widely applied across various research domains. It has been utilized for image segmentation [124], in the validation of welded Al/Cu bimetal sheets [125], for intelligent facial emotion recognition [126], to enhance power system stabilizers [127], and in task scheduling for microprocessor systems [128].

In robotics, WOA has been instrumental in planning the joint trajectory of robotic arms [129], enhancing robotic manufacturing processes [130], planning navigation of unmanned vehicles [131], aiding in multiple robot space exploration [132, 133], and optimizing deep neural networks [134]. Table 15 presents an overview of various studies where WOA has been implemented in smart vehicle navigation.

Table 15. WOA for path planning of smart vehicles

Types	Population Size	Number of Iterations	Type of Vehicle	Type of Obstacles	Type of Map
WOA		100	Single (Khepera II mobile robot)	Static	Geometrical
Improved WOA based on GA		100	Single	Static	20×0 Grid.
MWOA	100	500	Single	Static	300×500 pixels Geometrical
MO-WOA	50,80,100,150	50,70,90,110	Multiple	Static	15×15 Grid (see Figure 8)
NWOA		1000	Single (Raspberry Pi (3B+))	Static and dynamic	1800×1800 Geometrical
Updated WOA		500	Single	Static	8×8 Geometrical

Types	Software	Remark	Ref.
WOA	MATLAB	•Algorithm solves robot scheduling problem in a manufacturing environment. Novel mathematical model is proposed and experimentally tested on 26 benchmark functions.	[135]
Improved WOA based on GA		• Proposed method can be implemented for logistic mobile robot. Efficiency of proposed algorithm is improved by 10.71% compared to traditional WOA.	[136]
MWOA		•Distance and smooth path functions are minimized.• The pareto front-optimal solution gives the optimal solution for MWOA. The proposed method has a lower error rate than the Multi-Objective Genetic Algorithm (MOGA) method [137].	[138]
MO-WOA	MATLAB	•At 130 iterations and 150 way-points, proposed algorithm outperforms compared deterministic and hybrid stochastic exploration algorithm. Map exploration and minimum time map enhancing accuracy is the idea behind proposed algorithm.	[139]
NWOA	Python	• Adaptive technology, enhanced potential field factors and virtual obstacles are introduced to optimize the convergence rate of the algorithm. NWOA performance better in convergence rate when compared to WOA, GA-WOA, and EGE-WOA.	[140]
Ypdated WOA		Proposes a changed whale advancement calculation based Mobile robot way determination.	[141]

NOTE: MWOA: Multiobjective Whale Optimization Algorithm; MO-WOA: Multi-Objective Whale Optimization Algorithm; NWOA; Novel Whale Optimization Algorithm

2.2.8 Grey Wolf Optimization (GWO)

Proposed by Mirjalili et al. [142], GWO is an algorithm inspired by the social hierarchy and hunting techniques of grey wolves. Grey wolves are apex predators that typically live in packs of 5 to 12 members, each adhering to a strict social structure as depicted in Figure 9.

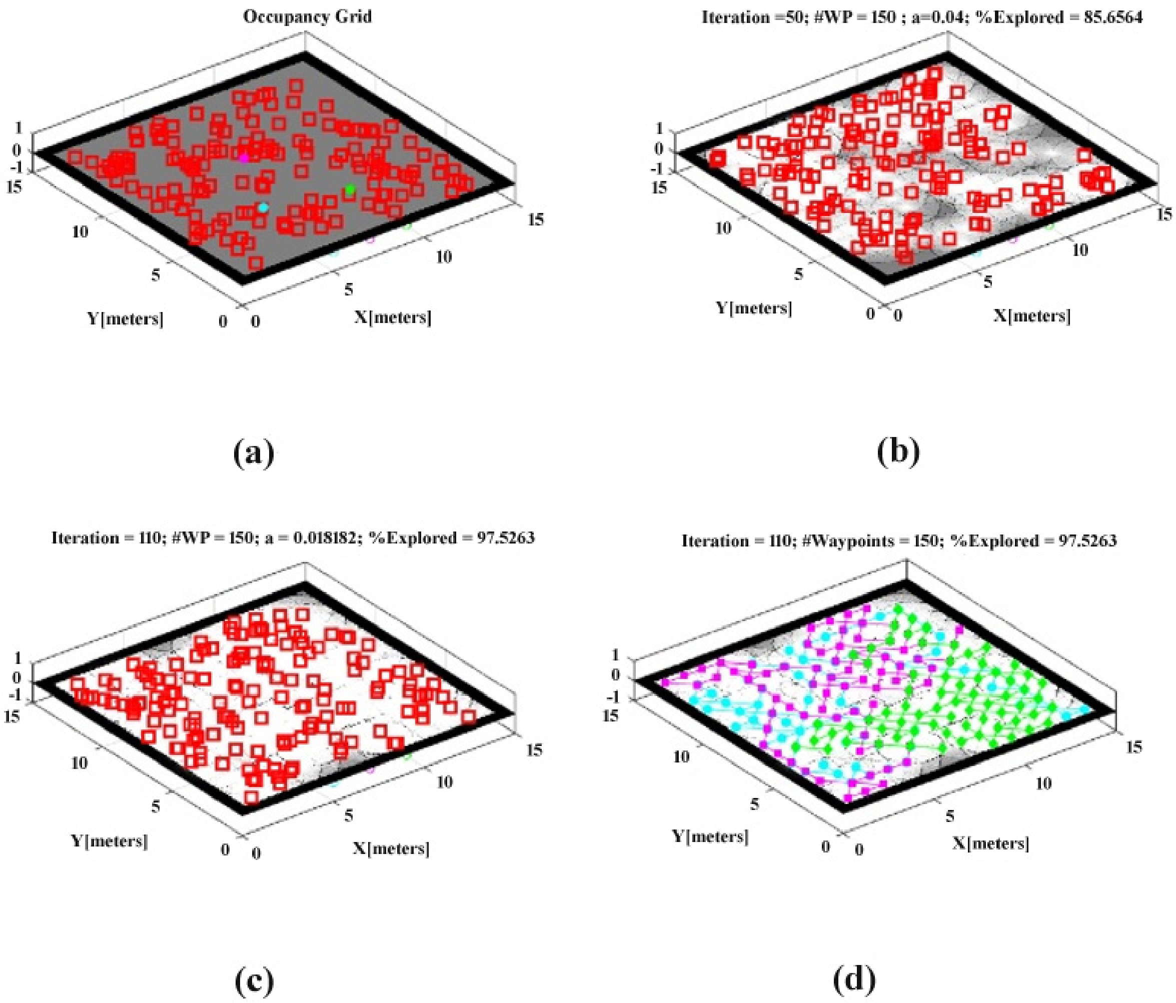

Figure 8. Sample map implemented for whale optimization algorithm: MO-WOA 15×15 map [139]

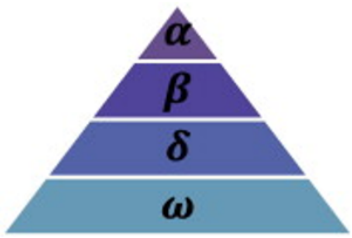

Figure 9. Social hierarchy of grey wolves

In this hierarchy, the alphas (α)—a male and a female—serve as the leaders. Positioned at the top, they represent the fittest solution, and their directives are followed by the rest of the pack. The beta (β) wolves, ranked second, are considered the primary candidates for alpha status. Below them are the delta (δ) wolves, including scouts, sentinels, elders, hunters, and caretakers, who preside over the omega (ω) wolves, often viewed as the scapegoats of the pack. Grey wolves hunt collaboratively, starting with tracking, chasing, and approaching their prey. They encircle, pursue, and harass the prey until it weakens, then launch an attack. This hunting behavior is mathematically modeled in GWO, where the alpha, beta, and delta wolves correspond to the best, second-best, and third-best solutions, respectively, while the remaining solutions are considered omega wolves. The model encompasses several phases:

(1) Encircling the prey:

$$\vec{D} = \left|\vec{C}\overrightarrow{X_P}(t) - \overrightarrow{X_P}(t)\right| \tag{22}$$

$$\vec{X}(t+1) = \overrightarrow{X_P}(t) - \vec{A}\vec{D} \tag{23}$$

$$\vec{A} = 2\vec{a}\overrightarrow{r_1} - \vec{a} \tag{24}$$

$$\vec{C} = 2\overrightarrow{r_2} \tag{25}$$

where, $\vec{A}$ and $\vec{C}$ represent coefficient vectors, t is the present iteration, $\overrightarrow{X_P}$ is prey's position, $\vec{X}$ is the grey wolf position, $\overrightarrow{r_1}$ and $\overrightarrow{r_2}$ are random vectors within $[0, 1]$, and $\vec{a}$ decreases linearly from 2 to 0 in the course of iterations.

(2) Hunting:

Alpha, beta, and delta wolves update their positions first, assuming better knowledge of the prey's location:

$$\overrightarrow{D_a} = \left|\overrightarrow{C_1}\overrightarrow{X_a}(t) - \vec{X}(t)\right| \tag{26}$$

$$\overrightarrow{D_\beta} = \left|\overrightarrow{C_2}\overrightarrow{X_\beta}(t) - \vec{X}(t)\right| \tag{27}$$

$$\overrightarrow{D_\delta} = \left|\overrightarrow{C_3}\overrightarrow{X_\delta}(t) - \vec{X}(t)\right| \tag{28}$$

$$\overrightarrow{X_1}(t+1) = \overrightarrow{X_\alpha}(t) - \overrightarrow{A_1}\overrightarrow{D_\alpha} \tag{29}$$

$$\overrightarrow{X_2}(t+1) = \overrightarrow{X_\beta}(t) - \overrightarrow{A_2}\overrightarrow{D_\beta} \tag{30}$$

$$\overrightarrow{X_3}(t+1) = \overrightarrow{X_\delta}(t) - \overrightarrow{A_3}\overrightarrow{D_\delta}A \tag{31}$$

$$\vec{X}(t+1) = \left(\overrightarrow{X_1} + \overrightarrow{X_2} + \overrightarrow{X_3}\right)/3 \tag{32}$$

(3) Attacking the prey (Exploitation):

The mathematical representation of a prey attack in the GWO is characterized by the gradual decrease of a from 2 to 0 over the iterations. The attack phase is initiated when the value of $|A| < 1$, with $\vec{A}$ being a random value within the range of $[-2\vec{a}, 2\vec{a}]$.

(4) Searching for prey (Exploration)

When the value of $|\boldsymbol{A}| > \mathbf{1}$ in the GWO, the wolves are prompted to explore for more suitable prey. This exploration phase is influenced by the parameter $\vec{\boldsymbol{C}}$, which is a random value within the range $[0, 2]$. The role of C is to introduce stochasticity into the behavior of the grey wolves, either emphasizing $(\mathrm{C} > 1)$ or deemphasizing $(\mathrm{C} < 1)$ their predatory attack. This mechanism allows for a balanced exploration of the search space, mimicking the adaptive hunting behavior of grey wolves in the wild. The pseudocode detailing the GWO process is provided in Table 16.

GWO has been applied to solve optimization problems in a diverse range of fields. In medicine, it has been used for various optimization tasks [143, 144]. In manufacturing, it has been employed for operation sequencing [145]. The algorithm has also been adapted for use in unmanned aerial vehicle navigation [146], multi-agent systems [147], and robotics. For insights into the variety of GWO applications specifically in smart vehicle navigation, one can refer to Table 17. Gul et al. [148] proposed a hybrid algorithm combining PSO with

GWO. This hybrid PSO-GWO algorithm was designed to improve path length and ensure smoother trajectories for mobile robots. Furthermore, a mutation operator was introduced to refine the trajectory generated by the PSO-GWO algorithm (Figure 10) [149]. To address the challenge of local minima in GWO, Dong et al. [150] suggested a modified position update mechanism specifically tailored for robot path planning. This modification aimed to enhance the algorithm's ability to navigate complex environments more effectively. In addition, Kumar et al. [151] developed a Variable Weight GWO, aimed at increasing speed and reducing the distance of planned routes for mobile robots. This variant of GWO adjusts the algorithm's parameters dynamically to optimize performance in real-time navigation tasks.

Table 16. Pseudocode of GWO

Grey Wolf Optimization

```
Initialize the prey wolf population X_i (i = 1, 2, ..., n)
Initialize a, A, and C
Calculate the fitness of each search agent
X_α = the best search agent
X_β = the second best search agent
X_δ = the third best search agent
while (t<maximum number of iterations)
    of each search agent
        Update the position of the current search agent
    end of
    Update a, A, and C
    Calculate the fitness of all search agent
    Update X_α, X_β, X_δ
    t = t + 1
end while
return X_α
```

Table 17. Grey Wolf Optimization for path planning of smart vehicles

Type	Population Size	Iterations	Type of Vehicle	Type of Obstacles	Type of Map	Software	Remark	Ref.
IGWO	30	100	Single	Static	Geometrical	MATLAB R2018b	•The algorithm is tested on 20 benchmark functions.	[150]
VM-GWO	20,25 (For map 1) 25,30 (For map2)	35,20 (For map 1) 30,40 (For map 2)	Single	Static	$20 \times 10 \times 10$ Geometrical (3D map)	MATLAB 2018a	•Execution speed outperformed GWO.	[151]
HPSO-GWO-EA			Single	Static	Geometrical	MATLAB R2017a	•Mutation operator from evolutionary algorithms is introduced to solve path safety, length, and smoothness.	[148]
HPSO-GWO	100	500	Single	Static and Dynamic	100×100 Geometrical (see Figure 10)	MATLAB R2019b	•Frequency-based function is introduced to modify the search process of GWO.	[149]
HWGO	20	100	Single			MATLAB R2019	•Proposed algorithm is implemented in tuning parameters of fractional order PID controller.	[152]

Note: IGWO: Improved Grey Wolf Optimization; VM-GWO: Variable Weight - Grey Wolf Optimization; HPSO-GWO-EA: Hybrid Particle Swarm Optimization - Grey Wolf Optimization – Evolution Algorithm; HPSO-GWO: Hybrid Particle Swarm Optimization - Grey Wolf Optimization; HWGO: Hybrid Whale Grey Wolf Optimizer

2.2.9 Grey Wolf Optimization (GWO)

The MVO is an innovative population-based algorithm inspired by the multi-verse theory in physics, which posits the existence of multiple universes interacting within a multi-verse [153]. This algorithm incorporates three key cosmological concepts: white holes, black holes, and wormholes [154, 155]. In astrophysics, the Big Bang, considered the origin of the universe, is likened to a white hole [156], a concept representing regions emitting energy and matter. Conversely, black holes are known for their intense gravitational force, which attracts and absorbs matter, including light beams [157]. Wormholes are theorized as space-time passages that link different parts of a universe or even connect separate universes. The concept of universe expansion, driven by the inflation rate or eternal inflation [158], is also integrated into this model.

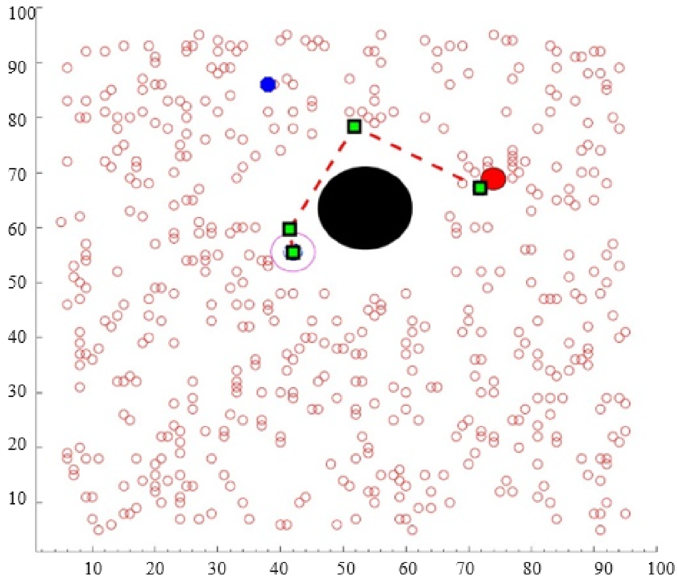

Figure 10. Sample maps implemented for grey wolf optimization: HPSO-GWO 100×100 [149]

In MVO, these astronomical phenomena are mathematically modeled to facilitate exploration (white holes), exploitation (wormholes), and local search (black holes) in the search space. Each 'universe' in MVO represents a potential solution, with the objects within a universe analogous to variables of that solution. The fitness value of a solution is equated to its inflation rate. Universes in MVO adhere to the following principles:

(1) High inflation rate leads to a high chance of having a white hole.

(2) Low inflation rate leads to a low chance of having a black hole.

(3) High inflation rate universes are likely to pass objects through white holes.

(4) Lower inflation rate universes have a tendency to get objects through black holes.

(5) Regardless of inflation rate, objects in all universes can move randomly towards an optimal universe via wormholes.

Assuming that U is a set of universes, where n is the number of possible solutions (universes) and d represents the number of parameters or variables:

$$U = \begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^d \\ x_2^1 & x_2^2 & \cdots & x_2^d \\ \vdots & \vdots & \vdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^d \end{bmatrix} \tag{33}$$

then each parameter can be represented as below:

$$x_i^j = \begin{cases} x_k^j, r1 < NI(Ui) \\ x_i^j, r1 \geq NI(Ui) \end{cases} \tag{34}$$

x_i^j is the j-th parameter of the i-th universe. $NI(Ui)$ indicates the normalized inflation rate of the i-th universe. Ui is the i-th universe. $r1$ is a random value in $[0, 1]$. x_k^j denotes the j-th variable of k-th universe chosen by a roulette wheel selection mechanism. Roulette wheel, which depends on normalized inflation rate, selects a universe and determines white holes for it. Through this mechanism, exploration is done. For exploitation, each universe is assumed to have wormholes connecting it to the best universe, facilitating the exchange of objects (parameters). The update for each parameter is thus given by:

$$x_i^j = \begin{cases} \begin{cases} \left(X_j + TDR \times \left((ub_j - lb_j) \times r4 + lb_j\right)\right), r3 < 0.5 \\ \left(X_j - TDR \times \left((ub_j - lb_j) \times r4 + lb_j\right)\right), r3 \geq 0.5 \end{cases} & r2 < WEP \\ x_i^j, & r2 > WEP \end{cases} \tag{35}$$

where, X_j represents the j-th parameter of the best universe formed so far, TDR (Travelling Distance Rate) and WEP (Wormhole Existence Probability) are coefficients, lb_j and ub_j denote the lower and upper bounds of the j-th parameter respectively, x_i^j represents the j-th parameter of the i-th universe and $r2$, $r3$, $r4$ are random values in $[0, 1]$. The formulas for the coefficients are as follows:

$$WEP = \min + l \times \left(\frac{\max - \min}{L}\right) \tag{36}$$

where, $\min$ and $\max$ are the minimum and maximum respectively, l tells the current iteration and L is the maximum iterations.

$$TDR = 1 - \frac{l^{1/p}}{L^{1/p}} \tag{37}$$

where, p denotes the exploitation accuracy over the iterations, with higher values leading to more accurate exploitation/local search. The Multi-Verse Optimizer algorithm is detailed in the pseudocode shown in Table 18.

Table 18. Multi-Verse Optimizer (MVO)

```
Multi-Verse Optimizer
Create random universes (U)
Initialize WEP, TDR, and Best Universe
SU ← Sorted universes
NI ← Normalize inflation rate (fitness) of the universes
while the end criterion is not satisfied do
    Evaluate the fitness of all universes
    for each universe indexed by i do
        Update WEP and TDR
        Black hole index ← i
        for each object indexed by j do
            r1 ← random([0, 1])
            if r1 < NI (U_i) then
                White hole index ← RouletteWheelSelection (−NI)
                U( Black hole index, j) ← SU( White hole index, j)
            end if
            r2 ← random([0, 1])
            if r2 < WEP then
                r3 ← random([0, 1])
                r4 ← random([0, 1])
                if r3 < 0.5 then
                    U(i, j) ← Best Universe (j) + TDR
                    × ((ub(j) − lb(j)) × r4 + lb(j))
                else
                    U(i, j) ←= Best Universe (j) − TDR
                    × ((ub(j) − lb(j)) × r4 + lb(j))
                end if
            end if
        end for
    end for
end while
return Best Universe
```

The MVO has been successfully implemented in a variety of domains to address complex optimization problems. Its applications range from project scheduling [159] to enhancing kernel extreme learning machines for medical diagnosis [160], modeling solar radiation [161], and solving economic dispatch problems in power systems [162]. In the field of robotics, MVO has demonstrated its versatility and effectiveness. It has been used for path planning in three-dimensional search spaces [163], tuning PID controllers [164, 165], planning the navigation of quadrotors [166], and devising routes for mobile robots [167]. However, the application of MVO in the navigation of smart vehicles is relatively nascent, with only a handful of researchers exploring its potential in this area, as detailed in Table 19.

Table 19. MVO for path planning of smart vehicles

Type	Type of Vehicle	Type of Obstacles	Type of Map	Software	Remark	Ref.
Evolutionary Multi-Verse Optimizer	Single			Python (3.7)	• Parameters of each solution are the weights and bias of implemented Multi-Layer perceptron Network.	[167]
MMVO	Single	Static	400×400 Geometrical (2D and 3D)		• 3D path planning in a modelled 3D environment is examined.	[163]

Note: MMVO: Modified Multi-Verse Optimizer

2.2.10 Bat Algorithm (BA)

The BA, inspired by the echolocation behavior of microbats, was developed by Yang [168]. Microbats emit short, loud bursts of sound at frequencies ranging from 25 kHz to 150 kHz and listen to the echoes bouncing back from nearby objects. This echolocation ability enables them to pinpoint the location of objects. Typically, the frequency of pulse emission and the loudness of the sound increase during prey search and decrease upon prey discovery. The BA is formulated based on idealized rules derived from this echolocation behavior:

(1) Bats estimate distance based on the echo of their sounds and can differentiate prey from other objects.

(2) Bats move randomly with a velocity v_i towards a prey at position x_i, emitting sounds at a frequency $f_{\min}$, with wavelength λ and loudness A_0.

(3) The loudness is assumed to be between a large positive value and its minimum value is defined as $A_{\min}$.

For the BA, the frequency which is assumed to be within $[0, f_{\max}]$, the new velocity and position are defined below.

$$f_i = f_{\min} + (f_{\max} - f_{\min})\,\beta \tag{38}$$

$$v_i^t = v_i^{t-1} + \left(x_i^t - x_*\right) f_i \tag{39}$$

$$x_i^t = x_i^{t-1} + v_i^t \tag{40}$$

where, x_* represents the current global best solution from all n bats and β is a vector within $[0, 1]$. The loudness (A), starts as any positive number, typically within the range $[1, 2]$, and is then updated by a constant $\alpha \in [0, 1]$ as shown in Eq. (36). $A = 0$ when a solution is found. The rate of pulse emission $\boldsymbol{r}_i^0 \in [0, 1]$ is controlled by a constant γ, which can be the same as α.

$$A_i^{t+1} = \alpha A_i^t, \quad r_i^{t+1} = r_i^0[1 - \exp(-\gamma t)] \tag{41}$$

For local search, new solutions for each bat are generated using a random walk strategy, once a solution is selected from the current best ones.

$$x_{\text{new}} = x_{\text{old}} + \epsilon A^t \tag{42}$$

where, $\epsilon \in [-1, 1]$ denotes a random number in $[-1, 1]$ and A^t represents average loudness of all bats at time t. The implementation of the Bat Algorithm is depicted in Table 20.

The implementation of the BA in the domain of mobile robot planning and navigation has yielded impressive results, as evidenced by numerous studies in the field. Table 21 provides an overview of various research efforts that have proposed enhancements to the BA, detailing the variables considered in each study. Among these advancements, Ajeil et al. [169] introduced a Modified Frequency Bat Algorithm (MFB) specifically designed to optimize the shortest path finding from a start to an end point, compared to the standard Bat Algorithm. This novel algorithm integrates obstacle detection and avoidance techniques, utilizing sensor data to dynamically plan new paths in response to moving obstacles. The algorithm's performance was evaluated through simulations conducted in a grid mat environment. The results demonstrated that the MFB outperformed the standard BA in terms of efficiency and effectiveness in pathfinding, particularly in environments with dynamic obstacles.

Table 20. Pseudocode of BA

Bat Algorithm
Objective function $f(\mathrm{x}), \mathrm{x} = (x_1, \ldots\ldots, x_d)^{\mathrm{T}}$
Initialize the bat population $x_i (i = 1, 2, \ldots, n)$ and v_i
Define pulse frequency f_{i} at x_{i}
Initialize pulse rates r_{i} and the loudness A_i
while $(t <$ Max number of iterations $)$ **do**
Generate new solutions by adjusting frequency,
and updating velocities and locations/solutions Eqs. (2)-(4)
if (rand $> r_{\mathrm{i}}$) **then**
Select a solution among the best solutions
Generate a local solution around by flying randomly
end if
Generate a new solution by flying randomly
if (rand $< A_i \& f(x_{\mathrm{i}}) < f(x_*)$) **then**
Accept the new solutions
Increase r_{i} and reduce A_i
end if
Rank the bats and find the current best x_*
end while
Post-process results and visualization

Table 21. BA for path planning of smart vehicles

Type	**Population Size**	**A(0)**	**r(0)**	α	γ	$f_{\min}$	$f_{\max}$	**Type of Vehicle**	**Type of Obstacles**
MFB	5	1	0.5	0.98	0.8	0	10	Single	Dynamic
Type-1 FLS-BA	20							Single	Static

Type	**Type of Map**	**Software**	**Remark**	**Ref.**
MFB	12×12 Grid	MATLAB	• Obstacle detection and avoidance method is integrated in the algorithm.	[169]
Type-1 FLS-BA			• BA modifies Type-1 FLS to generate optimum trajectory. Proposed method aims at obtaining the least mean square error in trajectory tracking.	[170]

Note: MFB: Modified Frequency Bat algorithm; FLS-BA: Fuzzy Logic System – Bat Algorithm

2.2.11 Tabu-Search Algorithm (TS)

The TS [171] is a method of optimization that utilizes constraint-based techniques to avoid local minima in a search space. It incorporates flexible memory cycles to intensify and diversify local search patterns, thereby facilitating the discovery of optimal solutions. During the exploration process, Tabu Search meticulously tracks information about both the current solution and those previously explored [172]. The algorithm operates by employing neighborhood search methods to progress from a current solution x to a feasible solution x'within the neighborhood of x. This iterative process continues until a predetermined stopping criterion is met. One of the key features of TS is its use of memory structures to record visited solutions, preventing the algorithm from getting trapped in local minima and encouraging the exploration of unvisited areas in the search space [173]. The memory structures, also known as the tabu list, contain a set of recently visited solutions that are temporarily banned from reconsideration, typically for n iterations (where n, the tabu tenure, specifies the length of the list). The procedure for implementing this algorithm is outlined in Table 22.

Table 22. Pseudocode of TS

Tabu-Search Algorithm

Generate initial solution (x_0)
Initialize tabu list (TL $\leftarrow$ *[ ])*
Current solution(x) $\leftarrow$ *initial solution* (x_0)
Best solution(x_{best}) $\leftarrow$ *current solution(x)*
Define maximum iteration (ITR_{max})
Iteration (ITR) $\leftarrow$ *0*
while *(ITR* $< ITR_{max}$*)* **do**
$S_N \leftarrow Get\ neighbours\ of\ x_{\text{best}}$
  for $S \in S_N$ **do**
    if $S \notin TL \&\&\ fitness(S) > fitness(x_{best})$ **then**
      $x_{\text{best}} = S$
    end if
  end for
  add *S to TL*
end while
return x_{best}

The application of the TS in mobile robot navigation is still relatively underexplored. However, some studies, as indicated in Table 23, have developed new hybrids that enhance the basic algorithm's efficiency in path planning. Xing et al. [174] introduced a novel TS tailored for routing multiple AGVs in warehouse settings. Châari et al. [175] developed a TS-based system model for global path planning on grid maps. Kumar et al. [176] modified the TS method for navigating mobile robots in complex environments. Khaksar et al. [177] integrated TS rules into a fuzzy controller designed to address online navigation challenges. Panda et al. [178] proposed a hybrid algorithm combining TS and PSO to optimize pathfinding for AMRs.

Table 23. TS for path planning of smart vehicles

Type	Iteration Number	Type of Vehicle	Type of Obstacles	Type of Map
TS-PATH	30	Single	Static	Grid (see Figure 11)
PSO-TABU	31	Multiple	Static	Geometrical
Modified Tabu Search		Single (Khepera-III robot)	Static	10×8cm Grid (simulation) and 400×300cm^2 Geometrical (experimental)
ANFIS		Single	Static	10×9, 10×10 and 10×14 Geometrical
Tabu Search	9	Single	Static	10×10 Grid
GSTLACA		Multiple	Dynamic	Grid
TS/FA	5-7	Single	Static	561×380 px^2 and 433×430 px^2 Grid

Type	Software	Remark	Ref.
TS-PATH	Designed a C++ simulation model	• The effectiveness of the tabu search for the global path planning problem is investigated.	[175]
PSO-TABU	Designed a C simulation environment	• In terms of solution quality and computation time, PSO-TABU, outperforms the basic PSO and TABU search algorithm.	[178]
Modified Tabu Search	V-REP simulation software	•Algorithm is verified in both simulation and experimental platform. Deviation between the simulation and experimental results is about 4%.	[176]
ANFIS	MATLAB R2010b	• Heuristic rules of Tabu search is infused in fuzzy controller. Fuzzy planner can handle online navigation task.	[177]
Tabu Search	MATLAB	•Algorithm generates trajectories to multiple end points using the shortest possible path.	[179]
GSTIACA	e-Puck architecture in the Webots simulation environment.	• Real-time simulation shows concurrent navigation and map building in dynamic environments.	[180]
TS/FA	MATLAB 2014a and V-rep simulator	• TS/FA is an offline hybrid algorithm. Bezier curve is used to smoothen generated path.	[181]

Note: TS-PATH: Tabu Search Path; PSO-TABU: Particle Swarm Optimization – Tabu Search; ANFIS: Adaptive Neuro-Fuzzy Inference System; GSTIACA: Genetic Shared Tabu Inverted Ant Cellular Automata; TS/FA: Tabu Search / Firefly Algorithm

2.3 Analysis

An important aspect to consider when evaluating the metaheuristic algorithms discussed in this work is their performance on various benchmark functions. Benchmark tests, comprising mathematical functions, are instrumental in assessing the algorithms' ability to find solutions in a given dimension d that lead to global optima [182]. These benchmark functions, as cataloged in Table 24, can be categorized into unimodal, multimodal, or combinatory types, which blend unimodal and multimodal characteristics. Unimodal functions are designed to lead to a single optimum solution, whereas multimodal functions yield multiple optimum solutions.

The significance of metaheuristic algorithms in research, particularly in addressing complex real-world problems, is underscored by several advantages noted by Gholizadeh and Barati [183]. Their high efficiency and flexibility are key attributes that make these algorithms increasingly valuable in solving complex challenges. Additionally, the popularity and impact of a metaheuristic algorithm can often be gauged by the number of citations it receives in academic literature. Table 25 provides a ranking of the algorithms utilized in this study based on their citation count.

Selecting the appropriate algorithm for path planning in the application of smart vehicles is a critical decision. The choice of algorithm significantly depends on the specific requirements of the smart vehicle's mission. For instance, the algorithmic needs for rescue missions and urgent tasks differ markedly from those required for surveillance or logistics operations. A key consideration in this decision-making process is the balance between exploration and exploitation, which are inherent trade-offs in optimization problems. Exploration entails an efficient search of the solution space, aiming to circumvent local optima in pursuit of global solutions. While this process is thorough, it often results in slower convergence speeds. On the other hand, exploitation focuses on rapidly converging to a solution, which enhances the speed but raises the risk of becoming trapped in local optima. Therefore, when choosing an algorithm for path planning in a particular context, it's crucial to weigh these factors: convergence speed and the ability to identify global optima. The chosen algorithm should align with the specific objectives of the task at hand, whether it requires rapid response times or a more comprehensive search of the solution space.

Table 24. Common benchmark problems [117, 150, 182]

Function Name	Equation	Objective Value	Modality
Spherical	$\sum_{i=1}^{i=d} {x_i}^2$	0	Unimodal
Schwefel 2.22	$\sum_{i=1}^{i=d} \lvert x_i\rvert + \prod_{i=1}^{i=n} \lvert x_i\rvert$	0	Multimodal
Schwefel 2.21	$\max_{1\leq i\leq n} \lvert x_i\rvert$	0	Unimodal
Rosenbrock	$\sum_{i=1}^{i=d} 100\left(x_{i+1}-{x_i}^2\right)^2+(1-x_i)^2$	0	Multimodal
Step	$\sum_{i=1}^{d} \lvert x_i+0.5\rvert^2$	0	Unimodal
Schwefel	$418.9829d-\sum_{i=1}^{i=d} -x_i \sin\sqrt{\lvert x_i\rvert}$	0	Multimodal
Rastrigin	$10*d+\sum_{i=1}^{i=d} {x_i}^2-10\cos(2\pi x_i)$	0	Multimodal
Auckley	$-20*\exp\sqrt{\frac{1}{d}\sum_{i=1}^{i=d} x_i^2}-\exp\left(\frac{1}{d}\sum_{i=1}^{i=d}\cos(2\pi x_i)\right)+e$	0	Multimodal
Griewank	$\sum_{i=1}^{d}\frac{x_i^2}{4000}-\prod_{i=1}^{d}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	0	Multimodal

Table 25. Citation ranking of algorithms used in this work (Retrieved 28 Nov, 2022, Google Scholar)

Rank	Year	Algorithm	Number of Citations
1	1995	Particle Swarm Optimization [60]	75041
2	1975	Genetic Algorithm (GA) [28]	74165
3	1992	Ant Colony Optimization [45, 184]	5685 (from 1992) 15447 (from 2006)
4	2014	Grey Wolf Optimization [142]	9881
5	1986	Tabu Search Algorithm [171, 185]	6320 (from 1986) 9716 (from 1989)
6	2005	Artificial Bee Colony [186]	8060
7	2009	Cuckoo Search Algorithm [103]	7217
8	2016	Whale Optimization Algorithm [122]	6649
9	2008	Firefly Algorithm [89]	6224
10	2016	Multi-Verse Optimizer [153]	1656

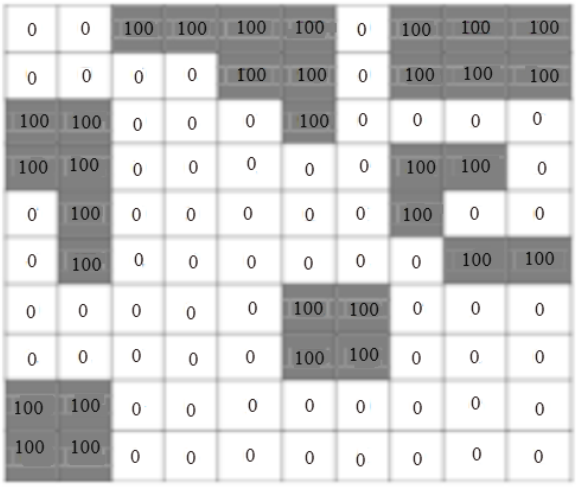

Figure 11. Sample maps implemented for tabu search algorithm: TS-PATH grid map [175]

2.3.1 Analysis on computational time and shortest path

In the realm of computational time and path optimization for smart vehicle applications, various metaheuristic algorithms exhibit distinct strengths. A hybrid CSA, for instance, has been shown to achieve optimal paths more rapidly than both PSO and GA [187]. PSO, on the other hand, outperforms the BA in terms of convergence when tuning omnidirectional mobile robots [188], while a hybrid PSO variant provides shorter paths in less time compared to modified BA and ABC [189].

CSA has been proven to outperform BA in finding an optimal path [121]. When it comes to covering the search space effectively, a multi-objective GWO demonstrates superior performance over multi-objective PSO and GA [190]. ACO outshines GA in obtaining optimal paths [191], and a hybrid ACO-PSO algorithm yields more optimal robot paths than ACO alone [192]. Moreover, ABC is noted for achieving shorter paths than PSO, as evidenced by simulation results [88]. These findings underscore the importance of selecting appropriate algorithms for specific tasks in smart vehicle applications, as outlined in Table 26, where the best-fitted algorithms for various industrial activities such as logistics, material handling, and surveillance are detailed.

Table 26. Various tasks performed by smart vehicles

Tasks
Vehicle scheduling
Warehouse material handling
Unmanned ground vehicle (military)
Search operation
Security and surveillance
Cleaning and disinfection operation
E-commerce delivery

2.4 Simulation Platform

The simulation of metaheuristic algorithms is carried out on various platforms, each offering unique mathematical and graphical functionalities. While some platforms provide a basic graphical representation of simulation outputs, others, like Gazebo-ROS, offer a more immersive experience with 3D animated environments for visualizing simulated outcomes. Among the most popular choices for researchers is MATLAB, known for its comprehensive inbuilt functions that greatly facilitate the programming of metaheuristic algorithms. However, some researchers prefer custom-built solutions, creating their own simulation platforms using fundamental programming languages such as C and C++ [175, 178]. Table 27 presents a compilation of the languages and simulation platforms commonly employed in this field of research. Additionally, a quantitative comparison of some of these simulators has been conducted by Farley et al. [193], providing insights into their respective capabilities and suitability for different types of simulations.

Table 27. Common simulation platforms and programming languages

Platform/Programming Language	Remarks	Ref
Python	High-level user-friendly programming language.	[194]
C, C++	High-level programming language for general-purpose programming.	[195]
MATLAB	Modeling and simulation software built by Mathworks.	[196]
CoppeliaSim (formally V-REP)	Creates room for importing personally designed robots. Robotic models can be controlled using C, python, or MATLAB scripts including ROS node.	[197]
ROS with MoveIt	MoveIt is the primary simulator in ROS for motion planning, 3D perception, manipulation and control.	[198]
GazeboSim	Offers various libraries and cloud services for robot simulation.	[199]
Webots	Offers a complete development environment to simulate robots and mechanical systems.	[200]
MORSE	Modular Open Robot Simulator Engine based on Blender game engine. A 3D simulator that offers a set of standard sensors, actuators and robotic bases.	[201]
USARsim	Urban Search and Rescue simulator for multi-robot purposes.	[202]

3 Conclusion

This study provides a comprehensive review of various metaheuristic algorithms and their hybrids, as developed by researchers to address path planning and navigation challenges in smart vehicles. The classification of these algorithms is primarily into two categories: population-based (encompassing evolutionary, swarm intelligence, and nature-inspired algorithms) and trajectory-based algorithms. A detailed description of each algorithm is provided, followed by reviews of recent articles spanning the last 13 years (2010 - 2023), with a majority of the studies concentrated between 2017 and 2023. Key parameters considered in this review include the type of vehicle (single or multiple robots), obstacle nature (static or dynamic), map type (topological, geometrical, or grid map), and the simulation platforms used for analysis.

The analysis also focuses on computational time and the efficiency of these algorithms in finding the shortest optimum path. Various tasks performed by smart vehicles are enumerated, highlighting the diverse applications. A notable observation is that navigation for smart vehicles remains an ongoing challenge, particularly in optimizing path length and reducing travel time. Researchers have made significant improvements and updates to these algorithms to address observed anomalies [118, 150]. A prominent trend is the development of hybrids between different algorithms, either to fine-tune parameters [136] or to combine advantages for enhanced robustness [119]. Path smoothing has been a crucial consideration in some studies, with techniques like cubic polynomials [85, 91], Bezier curves [78, 181], B-spline curves [56], and Cubic Ferguson splines [88] being used to generate smooth paths.

While most reviewed studies focused on static environments, there is a growing need to explore the navigation of smart vehicles in dynamic settings, considering moving obstacles and human interactions. One case study demonstrates the potential of using object detection sensors and refining data through neural networks with finely tuned weights for path planning in dynamic environments [167]. Additionally, combining reinforcement learning with metaheuristic algorithms could offer novel solutions for path planning challenges. For instance, incorporating reinforcement learning agents to balance exploration and exploitation in population-based metaheuristic algorithms could significantly enhance navigation path planning.

Data Availability

The data used to support the research findings are available from the corresponding author upon request.

Conflicts of Interest

The authors declare no conflict of interest.

References

[1] Y. L. Shi, Q. Han, W. M. Shen, and X. F. Wang, “A multi-layer collaboration framework for industrial parks with 5G vehicle-to-everything networks,” *Eng.*, vol. 7, no. 6, pp. 818–831, 2021. https://doi.org/10.1016/j.eng.2020.12.021

[2] R. Gasmi and S. Harous, “Robust connectivity-based internet of vehicles clustering algorithm,” *Wirel. Pers. Commun.*, vol. 125, no. 4, pp. 3153–3185, 2022. https://doi.org/10.1007/S11277-022-09703-0

[3] S. Dadvandipour and A. G. Ganie, “An approach to implementation of autoencoders in intelligent vehicles,” in *Vehicle and Automotive Engineering*. Cham: Springer International Publishing, 2022, pp. 3–10. https://doi.org/10.1007/978-3-031-15211-5_1

[4] A. Madhav and A. Tyagi, “Explainable artificial intelligence (XAI): Connecting artificial decision-making and human trust in autonomous vehicles,” in *Lecture Notes in Networks and Systems*, 2023, pp. 123–136. https://doi.org/10.1007/978-981-19-1142-2_10

[5] R. Théodosc, D. Denis, C. Blanc, T. Chateau, and P. Checchin, “Vehicle detection based on deep learning heatmap estimation,” in *IEEE Intelligent Vehicles Symposium (IV)*, Paris, France, 2019, pp. 108–113. https://doi.org/10.1109/IVS.2019.8814285

[6] H. Thadeshwar, V. Shah, M. Jain, R. Chaudhari, and V. Badgujar, “Artificial intelligence based self-driving car,” in *Proceedings of the 4th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, Chennai, India, 2020, pp. 1–5. https://doi.org/10.1109/ICCCSP49186.2020.9315223

[7] C. Ciuciu, D. Bărbuţă, S. Săsăujan, and E. Şipoş, “Autonomous scale model car with ultrasonic sensors and Arduino board,” *Acta Tech. Napocensis*, vol. 58, no. 4, pp. 6–11, 2017.

[8] Z. Zhang, J. Chen, and Q. Guo, “Application of automated guided vehicles in smart automated warehouse systems: A survey,” *CMES-Comput. Model. Eng. Sci.*, vol. 134, no. 3, pp. 1529–1563, 2023. https://doi.org/10.32604/CMES.2022.021451

[9] R. Anvo, A. Kaur, and T. P. Sattar, “Wireless communication with mobile inspection robots operating while submerged inside oil storage tanks,” in *Robotics for Sustainable Future: CLAWAR 2021*. Springer International Publishing, 2022, pp. 154–164. https://doi.org/10.1007/978-3-030-86294-7_14

[10] P. Coandă, M. Avram, V. Constantin, and B. Grămescu, “Indoor GPS system for autonomous mobile robots used in surveillance applications,” in *International Conference Innovation in Engineering*, Guimarães, Portugal, 2021, pp. 90–98. https://doi.org/10.1007/978-3-030-79168-1_9

[11] B. Grămescu, A. Cartal, A. S. Hashim, and C. Niţu, “Dynamic analysis of a robot locomotion for an external pipe inspection and monitoring,” in *International Conference Innovation in Engineering*, Guimarães, Portugal, 2021, pp. 189–200. https://doi.org/10.1007/978-3-030-79168-1_18

[12] Ş. Yıldırım and S. Savaş, “Design of a mobile robot to work in hospitals and trajectory planning using proposed neural networks predictors,” in *International Conference on Reliable Systems Engineering*, Bucharest, Romania, 2022, pp. 32–45. https://doi.org/10.1007/978-3-030-83368-8_4

[13] Z. Zhou, L. Li, A. Fürsterling, H. J. Durocher, J. Mouridsen, and X. Zhang, “Learning-based object detection and localization for a mobile robot manipulator in SME production,” *Robot. Comput. Integr. Manuf.*, vol. 73, p. 102229, 2022. https://doi.org/10.1016/j.rcim.2021.102229

[14] T. Dellmann and K. Berns, “Toward a realistic simulation for agricultural robots,” in *Agriculture Digitalization and Organic Production, Proceedings of the First International Conference, ADOP 2021*, St. Petersburg, Russia, 2021, pp. 3–13. https://doi.org/10.1007/978-981-16-3349-2_1

[15] R. Galati, G. Mantriota, and G. Reina, “Mobile robotics for sustainable development: Two case studies,” in *International Workshop IFToMM for Sustainable Development Goals I4SDG 2021: Proceedings of I4SDG Workshop 2021*, 2022, pp. 372–382. https://doi.org/10.1007/978-3-030-87383-7_41

[16] J. Fu, F. Tian, T. Chai, Y. Jing, Z. Li, and C. Y. Su, “Motion tracking control design for a class of nonholonomic mobile robot systems,” *IEEE Trans. Syst., Man, Cybern. Syst.*, vol. 50, no. 6, pp. 2150–2156, 2018. https://doi.org/10.1109/TSMC.2018.2804948

[17] M. W. Mehrez, K. Worthmann, J. P. Cenerini, M. Osman, W. W. Melek, and S. Jeon, “Model predictive control without terminal constraints or costs for holonomic mobile robots,” *Robot. Auton. Syst.*, vol. 127, p. 103468, 2020. https://doi.org/10.1016/j.robot.2020.103468

[18] R. Holmberg and O. Khatib, "Development and control of a holonomic mobile robot for mobile manipulation tasks," *Int. J. Robot. Res.*, vol. 19, no. 11, pp. 1066–1074, 2000. https://doi.org/10.1177/02783640022067977

[19] J. P. Laumond, P. E. Jacobs, M. Taix, and R. M. Murray, "A motion planner for nonholonomic mobile robots," *IEEE Trans. Robot. Autom.*, vol. 10, no. 5, pp. 577–593, 1994. https://doi.org/10.1109/70.326564

[20] A. Koubaa, H. Bennaceur, I. Chaari *et al.*, "Introduction to mobile robot path planning," in *Robot Path Planning and Cooperation: Foundations, Algorithms and Experimentations*. Springer, 2018, pp. 3–12. https://doi.org/10.1007/978-3-319-77042-0_1

[21] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Robot. Auton. Syst.*, vol. 86, pp. 13–28, 2016. https://doi.org/10.1016/j.robot.2016.08.001

[22] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021. https://doi.org/10.3390/vehicles3030027

[23] M. N. Ab Wahab, S. Nefti-Meziani, and A. Atyabi, "A comparative review on mobile robot path planning: Classical or meta-heuristic methods?" *Annu. Rev. Control*, vol. 50, pp. 233–252, 2020. https://doi.org/10.1016/J.ARCONTROL.2020.10.001

[24] P. Verma, A. Alam, A. Sarwar, M. Tariq, H. Vahedi, D. Gupta, and A. Shah Noor Mohamed, "Meta-heuristic optimization techniques used for maximum power point tracking in solar pv system," *Electronics*, vol. 10, no. 19, p. 2419, 2021. https://doi.org/10.3390/electronics10192419

[25] S. Campbell, N. O'Mahony, A. Carvalho, L. Krpalkova, D. Riordan, and J. Walsh, "Path planning techniques for mobile robots a review," in *2020 6th International Conference on Mechatronics and Robotics Engineering (ICMRE)*, Barcelona, Spain, 2020, pp. 12–16. https://doi.org/10.1109/ICMRE49073.2020.9065187

[26] C. Blum and A. Roli, "Metaheuristics in combinatorial optimization: Overview and conceptual comparison," *ACM Comput. Surv.*, vol. 35, no. 3, pp. 268–308, 2003. https://doi.org/10.1145/937503.937505

[27] H. J. Bremermann, "The evolution of intelligence: The nervous system as a model of its environment," University of Washington, Department of Mathematics, Tech. Rep., 1958.

[28] J. H. Holland, *Adaptation in natural and artificial systems*. Ann Arbor: University of Michigan Press, 1975.

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

[30] D. K. Pratihar, K. Deb, and A. Ghosh, "Fuzzy-genetic algorithms and time-optimal obstacle-free path generation for mobile robots," *Eng. Optim.*, vol. 32, no. 1, pp. 117–142, 1999. https://doi.org/10.1080/03052159908941294

[31] S. Kumar, D. R. Parhi, K. K. Pandey, and M. K. Muni, "Hybrid IWD-GA: An approach for path optimization and control of multiple mobile robot in obscure static and dynamic environments," *Robotica*, vol. 39, no. 11, pp. 2033–2060, 2021. https://doi.org/10.1017/S0263574721000114

[32] N. B. Hui and D. K. Pratihar, "A comparative study on some navigation schemes of a real robot tackling moving obstacles," *Robot. C Int. Manuf.*, vol. 25, no. 4-5, pp. 810–828, 2009. https://doi.org/10.1016/j.rcim.2008.12.003

[33] T. P. Fries, "Evolutionary robot navigation using fuzzy terrain conditions," in *NAFIPS 2006-2006 Annual Meeting of the North American Fuzzy Information Processing Society*, Montreal, QC, Canada, 2006, pp. 535–540. https://doi.org/10.1109/NAFIPS.2006.365466

[34] Y. Hu and S. X. Yang, "A knowledge based genetic algorithm for path planning of a mobile robot," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04.*, New Orleans, LA, USA, 2004, pp. 4350–4355. https://doi.org/10.1109/ROBOT.2004.1302402

[35] L. Lei, H. Wang, and Q. Wu, "Improved genetic algorithms based path planning of mobile robot under dynamic unknown environment," in *2006 International Conference on Mechatronics and Automation*, Luoyang, China, 2006, pp. 1728–1732. https://doi.org/10.1109/ICMA.2006.257475

[36] J. Lu and D. Yang, "Path planning based on double-layer genetic algorithm," in *Third International Conference on Natural Computation (ICNC 2007)*, Haikou, China, 2007, pp. 357–361. https://doi.org/10.1109/ICNC.2007.546

[37] H. Mahjoubi, F. Bahrami, and C. Lucas, "Path planning in an environment with static and dynamic obstacles using genetic algorithm: A simplified search space approach," in *IEEE International Conference on Evolutionary Computation*, Vancouver, BC, Canada, 2006, pp. 2483–2489. https://doi.org/10.1109/CE

C.2006.1688617

[38] J. Yuan, T. Yu, K. Wang, and X. Liu, "Step-spreading map knowledge based multi-objective genetic algorithm for robot-path planning," in *IEEE International Conference on Systems, Man and Cybernetics*, Montreal, QC, Canada, 2007, pp. 3402–3407. https://doi.org/10.1109/ICSMC.2007.4413953

[39] M. L. Liu, X. Z. Yao, J. Y. Huang, and C. Zhang, "Optimization of unmanned vehicle scheduling and order allocation," *Int. J. Simul. Model.*, vol. 21, no. 3, pp. 477–488, 2022. https://doi.org/10.2507/IJSIMM21-3-613

[40] W. C. Wang, C. Y. Ng, and R. Chen, "Vision-aided path planning using low-cost gene encoding for a mobile robot," *Intell. Autom. Soft Comput.*, vol. 32, no. 2, 2022. https://doi.org/10.32604/iasc.2022.022067

[41] P. Alliez and A. Fabri, "CGAL: The computational geometry algorithms library," in *ACM SIGGRAPH 2016 Courses*, Anaheim, California, USA, 2016, pp. 1–8. https://doi.org/10.1145/2897826.2927362

[42] M. Naderan-Tahan and M. T. Manzuri-Shalmani, "Efficient and safe path planning for a mobile robot using genetic algorithm," in *IEEE Congress on Evolutionary Computation*, Trondheim, Norway, 2009, pp. 2091–2097. https://doi.org/10.1109/CEC.2009.4983199

[43] K. P. Cheng, R. E. Mohan, N. H. K. Nhan, and A. V. Le, "Multi-objective genetic algorithm-based autonomous path planning for hinged-tetro reconfigurable tiling robot," *IEEE Access*, vol. 8, pp. 121 267–121 284, 2020. https://doi.org/10.1109/ACCESS.2020.3006579

[44] Y. Fu, "Path planning method of smart mobile robot based on genetic algorithm," *J. Phys. Conf. Ser.*, vol. 2083, no. 4, p. 042014, 2021. https://doi.org/10.1088/1742-6596/2083/4/042014

[45] M. Dorigo, "Optimization, learning and natural algorithms," Ph.D. dissertation, Politec. Di Milano, 1992.

[46] S. Goss, S. Aron, J. L. Deneubourg, and J. M. Pasteels, "Self-organized shortcuts in the Argentine ant," *Naturwissenschaften*, vol. 76, no. 12, pp. 579–581, 1989. https://doi.org/10.1007/BF00462870

[47] M. Dorigo and C. Blum, "Ant colony optimization theory: A survey," *Theor. Comput. Sci.*, vol. 344, no. 2-3, pp. 243–278, 2005. https://doi.org/10.1016/j.tcs.2005.05.020

[48] M. Brand, M. Masuda, N. Wehner, and X. H. Yu, "Ant colony optimization algorithm for robot path planning," in *2010 International Conference On Computer Design and Applications*, Qinhuangdao, China, 2010, pp. 436–440. https://doi.org/10.1109/ICCDA.2010.5541300

[49] J. Liu, J. Yang, H. Liu, X. Tian, and M. Gao, "An improved ant colony algorithm for robot path planning," *Soft Comput.*, vol. 21, pp. 5829–5839, 2017. https://doi.org/10.1007/s00500-016-2161-7

[50] X. You, S. Liu, and C. Zhang, "An improved ant colony system algorithm for robot path planning and performance analysis," *Int. J. Robot. Autom.*, vol. 33, no. 5, pp. 527–533, 2018. https://doi.org/10.2316/Journal.206.2018.5.206-0071

[51] X. Dai, S. Long, Z. Zhang, and D. Gong, "Mobile robot path planning based on ant colony algorithm with A* heuristic method," *Front. Neurorobot.*, vol. 13, p. 15, 2019. https://doi.org/10.3389/fnbot.2019.00015

[52] Z. Jiao, K. Ma, Y. Rong, P. Wang, H. Zhang, and S. Wang, "A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs," *J. Comput. Sci.*, vol. 25, pp. 50–57, 2018. https://doi.org/10.1016/j.jocs.2018.02.004

[53] K. Akka and F. Khaber, "Mobile robot path planning using an improved ant colony optimization," *Int. J. Adv. Robotic Syst.*, vol. 15, no. 3, 2018. https://doi.org/10.1177/1729881418774673

[54] L. Yang, L. Fu, P. Li, J. Mao, N. Guo, and L. Du, "LF-ACO: An effective formation path planning for multi-mobile robot," *Math. Biosci. Eng.*, vol. 19, no. 1, pp. 225–252, 2022. https://doi.org/10.3934/mbe.2022012

[55] J. Zhao, D. Cheng, and C. Hao, "An improved ant colony algorithm for solving the path planning problem of the omnidirectional mobile vehicle," *Math. Probl. Eng.*, vol. 2016, 2016. https://doi.org/10.1155/2016/7672839

[56] H. Yang, J. Qi, Y. Miao, H. Sun, and J. Li, "A new robot navigation algorithm based on a double-layer ant algorithm and trajectory optimization," *IEEE Trans. Ind. Electron.*, vol. 66, no. 11, pp. 8557–8566, 2018. https://doi.org/10.1109/TIE.2018.2886798

[57] Y. Tao, H. Gao, F. Ren, C. Chen, T. Wang, H. Xiong, and S. Jiang, "A mobile service robot global path planning method based on ant colony optimization and fuzzy control," *Appl. Sci.*, vol. 11, no. 8, p. 3605, 2021. https://doi.org/10.3390/app11083605

[58] D. Zhang, R. Luo, Y. Yin, and S. Zou, "Multi-objective path planning for mobile robot in nuclear accident environment based on improved ant colony optimization with modified A*," *Nucl. Eng. Technol.*, vol. 55, no. 5, pp. 1838–1854, 2023. https://doi.org/10.1016/J.NET.2023.02.005

[59] C. Miao, G. Chen, C. Yan, and Y. Wu, "Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm," *Comput. Ind. Eng.*, vol. 156, p. 107230, 2021. https://doi.org/10.1016/j.cie.2021.107230

[60] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, Perth, WA, Australia, 1995, pp. 1942–1948. https://doi.org/10.1109/ICNN.1995.488968

[61] M. A. Rend'on and F. F. Martins, "Path following control tuning for an autonomous unmanned quadrotor using particle swarm optimization," *IFAC-PapersOnLine*, vol. 50, no. 1, pp. 325–330, 2017. https://doi.org/10.1016/j.ifacol.2017.08.054

[62] P. B. Kumar, K. K. Pandey, C. Sahu, A. Chhotray, and D. R. Parhi, "A hybridized RA-APSO approach for humanoid navigation," in *2017 Nirma University International Conference on Engineering (NUiCONE)*, Ahmedabad, India, 2017, pp. 1–6. https://doi.org/10.1109/NUICONE.2017.8325611

[63] M. Gao, P. Ding, and Y. Yang, "Time-optimal trajectory planning of industrial robots based on particle swarm optimization," in *2015 Fifth International Conference on Instrumentation and Measurement, Computer, Communication and Control (IMCCC)*, Qinhuangdao, China, 2015, pp. 1934–1939. https://doi.org/10.1109/IMCCC.2015.410

[64] I. B. Aydilek, M. A. Nacar, A. GumuSCu, and M. U. Salur, "Comparing inertia weights of particle swarm optimization in multimodal functions," in *2017 International Artificial Intelligence and Data Processing Symposium (IDAP)*, Malatya, Turkey, 2017, pp. 1–5. https://doi.org/10.1109/IDAP.2017.8090225

[65] P. Raska and Z. Ulrych, "Testing different particle swarm optimization strategies," in *Proceedings of the 30th International Business Information Management Association Conference, IBIMA*, 2017.

[66] H. P. Dai, D. D. Chen, and Z. S. Zheng, "Effects of random values for particle swarm optimization algorithm," *Algorithms*, vol. 11, no. 2, p. 23, 2018. https://doi.org/10.3390/A11020023

[67] Y. Shi and R. C. Eberhart, "Parameter selection in particle swarm optimization," in *Evolutionary Programming VII: 7th International Conference*, San Diego, CA, USA, 1998, pp. 591–600. https://doi.org/10.1007/BFb0040810

[68] A. Ratnaweera, S. K. Halgamuge, and H. C. Watson, "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Trans. Evol. Comput.*, vol. 8, no. 3, pp. 240–255, 2004. https://doi.org/10.1109/TEVC.2004.826071

[69] N. Suryanto, C. Ikuta, and D. Pramadihanto, "Multi-group particle swarm optimization with random redistribution," in *2017 International Electronics Symposium on Knowledge Creation and Intelligent Computing (IES-KCIC)*, Surabaya, Indonesia, 2017, pp. 1–5. https://doi.org/10.1109/KCIC.2017.8228445

[70] H. S. Dewang, P. K. Mohanty, and S. Kundu, "A robust path planning for mobile robot using smart particle swarm optimization," *Procedia Comput. Sci.*, vol. 133, pp. 290–297, 2018. https://doi.org/10.1016/J.PROCS.2018.07.036

[71] Q. Chai, Y. Wang, Y. He, C. Xu, and Z. Hong, "Improved PRM path planning in narrow passages based on PSO," in *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, Guilin, China, 2022, pp. 41–46. https://doi.org/10.1109/ICMA54519.2022.9855913

[72] E. Masehian and D. Sedighizadeh, "A multi-objective PSO-based algorithm for robot path planning," in *2010 IEEE International Conference on Industrial Technology*, Via del Mar, Chile, 2010, pp. 465–470. https://doi.org/10.1109/ICIT.2010.5472755

[73] X. Li, D. Wu, J. He, M. Bashir, and M. Liping, "An improved method of particle swarm optimization for path planning of mobile robot," *J. Control Sci. Eng.*, vol. 2020, p. 3857894, 2020. https://doi.org/10.1155/2020/3857894

[74] B. Song, Z. Wang, and L. Zou, "An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve," *Appl. Soft Comput.*, vol. 100, p. 106960, 2021. https://doi.org/10.1016/j.asoc.2020.106960

[75] M. S. Alam, M. U. Rafique, and M. U. Khan, "Mobile robot path planning in static environments using particle swarm optimization," *arXiv preprint arXiv:2008.10000*, 2020. https://doi.org/10.48550/arXiv.2008.10000

[76] L. B. Amar and W. M. Jasim, "Hybrid metaheuristic approach for robot path planning in dynamic environment," *Bull. Electr. Eng. Inform.*, vol. 10, no. 4, pp. 2152–2162, 2021. https://doi.org/10.11591/EEI.V10I4.2836

[77] P. B. Fernandes, R. C. L. Oliveira, and J. F. Neto, "Trajectory planning of autonomous mobile robots applying a particle swarm optimization algorithm with peaks of diversity," *Appl. Soft Comput.*, vol. 116, p. 108108, 2022. https://doi.org/10.1016/J.ASOC.2021.108108

[78] L. Xu, M. Cao, and B. Song, "A new approach to smooth path planning of mobile robot based on quartic Bezier transition curve and improved PSO algorithm," *Neurocomputing*, vol. 473, pp. 98–106, 2022. https://doi.org/10.1016/J.NEUCOM.2021.12.016

[79] Y. Han, L. Zhang, H. Tan, and X. Xue, "Mobile robot path planning based on improved particle swarm optimization," in *2019 Chinese Control Conference (CCC)*, Guangzhou, China, 2019, pp. 4354–4358. https://doi.org/10.23919/CHICC.2019.8866634

[80] V. Sathiya, M. Chinnadurai, and S. Ramabalan, "Mobile robot path planning using fuzzy enhanced improved multi-objective particle swarm optimization (FIMOPSO)," *Expert Syst. Appl.*, vol. 198, p. 116875, 2022. https://doi.org/10.1016/J.ESWA.2022.116875

[81] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Glob. Optim.*, vol. 39, pp. 459–471, 2007. https://doi.org/10.1007/s10898-007-9149-x

[82] M. A. Contreras-Cruz, V. Ayala-Ramirez, and U. H. Hernandez-Belmonte, "Mobile robot path planning using artificial bee colony and evolutionary programming," *Appl. Soft Comput.*, vol. 30, pp. 319–328, 2015. https://doi.org/10.1016/j.asoc.2015.01.067

[83] A. Q. Faridi, S. Sharma, A. Shukla, R. Tiwari, and J. Dhar, "Multi-robot multi-target dynamic path planning using artificial bee colony and evolutionary programming in unknown environment," *Intell. Serv. Robot.*, vol. 11, pp. 171–186, 2018.

[84] S. Kumar and A. Sikander, "Optimum mobile robot path planning using improved artificial bee colony algorithm and evolutionary programming," *Arab. J. Sci. Eng.*, vol. 47, no. 3, pp. 3519–3539, 2022. https://doi.org/10.1007/s13369-021-06326-8

[85] R. T. Kamil, M. J. Mohamed, and B. K. Oleiwi, "Path planning of mobile robot using improved artificial bee colony algorithm," *Eng. Technol. J.*, vol. 38, no. 9, pp. 1384–1395, 2020. https://doi.org/10.30684/etj.v38i9a.1100

[86] N. H. Abbas and F. M. Ali, "Path planning of an autonomous mobile robot using the directed artificial bee colony algorithm," *Int. J. Comput. Appl.*, vol. 96, no. 11, pp. 11–16, 2014. https://doi.org/10.5120/16836-6681

[87] R. Szczepanski and T. Tarczewski, "Global path planning for mobile robot based on artificial bee colony and Dijkstra's algorithms," in *2021 IEEE 19th International Power Electronics and Motion Control Conference (PEMC)*, Gliwice, Poland, 2021, pp. 724–730. https://doi.org/10.1109/PEMC48073.2021.9432570

[88] P. Sudhakara, V. Ganapathy, and K. Sundaran, "Mobile robot trajectory planning using enhanced artificial bee colony optimization algorithm," in *2017 IEEE International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI)*, Chennai, India, 2017, pp. 363–367. https://doi.org/10.1109/ICPCSI.2017.8392316

[89] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. Luniver Press, 2010.

[90] X. Qi, S. Zhu, and H. Zhang, "A hybrid firefly algorithm," in *2017 IEEE 2nd Advanced Information Technology, Electronic and Automation Control Conference (IAEAC)*, Chongqing, China, 2017, pp. 287–291. https://doi.org/10.1109/IAEAC.2017.8054023

[91] S. M. Ali, J. F. Yonan, O. Alniemi, and A. A. Ahmed, "Mobile robot path planning optimization based on integration of firefly algorithm and cubic polynomial equation," *J. ICT Res. Appl.*, vol. 16, no. 1, pp. 1–22, 2022. https://doi.org/10.5614/ITBJ.ICT.RES.APPL.2022.16.1.1

[92] H. Kundra, W. Khan, M. Malik, K. P. Rane, R. Neware, and V. Jain, "Quantum-inspired firefly algorithm integrated with cuckoo search for optimal path planning," *Int. J. Mod. Phys. C*, vol. 33, no. 2, p. 2250018, 2022. https://doi.org/10.1142/S0129183122500188

[93] M. Mohammadi and M. I. Mobarakeh, "An integrated clustering algorithm based on firefly algorithm and self-organized neural network," *Prog. Artif. Intell.*, vol. 11, no. 3, pp. 207–217, 2022. https://doi.org/10.1007/S13748-022-00275-5

[94] M. Zivkovic, A. Petrovic, K. Venkatachalam, I. Strumberger, H. S. Jassim, and N. Bacanin, "Novel chaotic best firefly algorithm: COVID-19 fake news detection application," in *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems*, 2022, pp. 285–305. https://doi.org/10.1007/978-3-

031-09835-2_16

[95] E. Moharamkhani, M. Yahyaei Feriz Hendi, E. Bandar, A. Izadkhasti, and R. Sirwan Raza, “Intrusion detection system based firefly algorithm-random forest for cloud computing,” *Concurrency and Comput. Pract. Exper.*, vol. 34, no. 24, p. e7220, 2022. https://doi.org/10.1002/CPE.7220

[96] X. Chen, M. Zhou, J. Huang, and Z. Luo, “Global path planning using modified firefly algorithm,” in *2017 International Symposium on Micro-NanoMechatronics and Human Science (MHS)*, Nagoya, Japan, 2017, pp. 1–7. https://doi.org/10.1109/MHS.2017.8305195

[97] P. Duan, J. Li, H. Sang, Y. Han, and Q. Sun, “A developed firefly algorithm for multi-objective path planning optimization problem,” in *2018 IEEE 8th Annual International Conference on CYBER Technology in Automation, Control, and Intelligent Systems (CYBER)*, Tianjin, China, 2018, pp. 1393–1397. https://doi.org/10.1109/CYBER.2018.8688342

[98] B. K. Patle, A. Pandey, A. Jagadeesh, and D. R. Parhi, “Path planning in uncertain environment by using firefly algorithm,” *Defence Technol.*, vol. 14, no. 6, pp. 691–701, 2018. https://doi.org/10.1016/J.DT.2018.06.004

[99] A. K. A. Hassan and D. J. Fadhil, “Mobile robot path planning method using firefly algorithm for 3D sphere dynamic & partially known environment,” *J. Univ. Babylon Pure Appl. Sci.*, vol. 26, no. 7, pp. 309–320, 2018. https://doi.org/10.29196/jubpas.v26i7.1506

[100] N. A. K. Zghair and A. S. Al-Araji, “Intelligent hybrid path planning algorithms for autonomous mobile robots,” *Int. J. Intell. Eng. Syst.*, vol. 15, no. 5, pp. 309–325, 2022. https://doi.org/10.22266/IJIES2022.1031.28

[101] A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, and N. Pavón, “Solving the multi-objective path planning problem in mobile robotics with a firefly-based approach,” *Soft Comput.*, vol. 21, pp. 949–964, 2017. https://doi.org/10.1007/s00500-015-1825-z

[102] N. H. Singh, A. Laishram, and K. Thongam, “Optimal path planning for mobile robot navigation using FA-TPM in cluttered dynamic environments,” *Proc. Comput. Sci.*, vol. 218, pp. 612–620, 2023. https://doi.org/10.1016/J.PROCS.2023.01.043

[103] X. Yang and S. Deb, “Cuckoo search via Lévy flights,” in *2009 World Congress on Nature & Biologically Inspired Computing (NaBIC)*, Coimbatore, India, 2009, pp. 210–214. https://doi.org/10.1109/NABIC.2009.5393690

[104] A. S. Joshi, O. Kulkarni, G. M. Kakandikar, and V. M. Nandedkar, “Cuckoo search optimization-A review,” *Mater. Today Proc.*, vol. 4, no. 8, pp. 7262–7269, 2017. https://doi.org/10.1016/j.matpr.2017.07.055

[105] E. Valian, S. Mohanna, and S. Tavakoli, “Improved cuckoo search algorithm for global optimization,” *Int. J. Commun. Inf. Technol.*, vol. 1, no. 1, pp. 31–44, 2011.

[106] L. Xiao, A. Hajjam-El-Hassani, and M. Dridi, “An application of extended cuckoo search to vehicle routing problem,” in *2017 International Colloquium on Logistics and Supply Chain Management (LOGISTIQUA)*, Rabat, Morocco, 2017, pp. 31–35. https://doi.org/10.1109/LOGISTIQUA.2017.7962869

[107] K. Karagul and Y. Sahin, “An improved cuckoo search algorithm for the capacitated green vehicle routing problem,” in *Advances in Swarm Intelligence: Variations and Adaptations for Optimization Problems*. Springer, 2022, pp. 385–406. https://doi.org/10.1007/978-3-031-09835-2_21

[108] S. Chatterjee, N. Dey, S. Sen, A. S. Ashour, S. J. Fong, and F. Shi, “Modified cuckoo search based neural networks for forest types classification,” in *Information Technology and Intelligent Transportation Systems*. IOS Press, 2017, pp. 490–498. https://doi.org/10.3233/978-1-61499-785-6-490

[109] K. Bibiks, Y. F. Hu, J. P. Li, P. Pillai, and A. Smith, “Improved discrete cuckoo search for the resource-constrained project scheduling problem,” *Appl. Soft Comput.*, vol. 69, pp. 493–503, 2018. https://doi.org/10.1016/j.asoc.2018.04.047

[110] S. Prakash Tiwari and G. Singh, “Optimizing job scheduling problem using improved GA+CS algorithm,” in *International Conference on Innovative Computing and Communications Proceedings of ICICC 2022*, Delhi, India, 2022, pp. 291–297. https://doi.org/10.1007/978-981-19-2821-5_25

[111] M. Shehab, A. T. Khader, and M. A. Al-Betar, “A survey on applications and variants of the cuckoo search algorithm,” *Appl. Soft Comput.*, vol. 61, pp. 1041–1059, 2017. https://doi.org/10.1016/j.asoc.2017.02.034

[112] S. A. Lakshmi and K. Anandavelu, “Enhanced cuckoo search optimization technique for skin cancer diagnosis application,” *Intell. Autom. Soft Comput.*, vol. 35, no. 3, pp. 3403–3413, 2023. https://doi.org/10.32604/IASC.2023.030970

[113] D. Parkash and S. Mittal, "An enhanced secure framework using CSA for cloud computing environments," in *International Conference on Innovative Computing and Communications Proceedings of ICICC 2022*, Delhi, India, 2022, pp. 349–356. https://doi.org/10.1007/978-981-19-2535-1_27

[114] B. Sahu, P. Kumar Das, and M. Ranjan Kabat, "Multi-robot cooperation and path planning using modified cuckoo search," in *Next Generation of Internet of Things Proceedings of ICNGIoT 2022*, Odisha, India, 2022, pp. 369–382. https://doi.org/10.1007/978-981-19-1412-6_31

[115] Z. Garip, D. Karayel, and M. Erhan Çimen, "A study on path planning optimization of mobile robots based on hybrid algorithm," *Concurrency Computat.: Pract. Exper.*, vol. 34, no. 5, p. e6721, 2022. https://doi.org/10.1002/CPE.6721

[116] S. Kumar, D. R. K. Parhi, A. K. Kashyap, and Vikas, "Path optimization and control of mobile robot using modified cuckoo search algorithm," in *Applications of Computational Methods in Manufacturing and Product Design Select Proceedings of IPDIMS 2020*, 2022, pp. 125–133. https://doi.org/10.1007/978-981-19-0296-3_12

[117] B. Sahu, P. K. Das, and R. Kumar, "A modified cuckoo search algorithm implemented with SCA and PSO for multi-robot cooperation and path planning," *Cogn. Syst. Res.*, vol. 79, pp. 24–42, 2023. https://doi.org/10.1016/J.COGSYS.2023.01.005

[118] Y. Fan, X. Sun, G. Wang, and D. Mu, "Collision avoidance controller for unmanned surface vehicle based on improved cuckoo search algorithm," *Appl. Sci.*, vol. 11, no. 20, p. 9741, 2021. https://doi.org/10.3390/APP11209741

[119] M. Saraswathi, G. B. Murali, and B. B. V. L. Deepak, "Optimal path planning of mobile robot using hybrid cuckoo search-bat algorithm," *Procedia Comput. Sci.*, vol. 133, pp. 510–517, 2018. https://doi.org/10.1016/j.procs.2018.07.064

[120] J. Wang, X. Shang, T. Guo, J. Zhou, S. Jia, and C. Wang, "Optimal path planning based on hybrid genetic-cuckoo search algorithm," in *2019 6th International Conference on Systems and Informatics (ICSAI)*, 2019, pp. 165–169. https://doi.org/10.1109/ICSAI48974.2019.9010519

[121] B. Gunji, B. B. V. L. Deepak, M. B. L. Saraswathi, and U. R. Mogili, "Optimal path planning of mobile robot using the hybrid cuckoo–bat algorithm in assorted environment," *Int. J. Intell. Unmanned Syst.*, vol. 7, no. 1, pp. 35–52, 2019. https://doi.org/10.1108/IJIUS-07-2018-0021

[122] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, 2016. https://doi.org/10.1016/j.advengsoft.2016.01.008

[123] W. A. Watkins and W. E. Schevill, "Aerial observation of feeding behavior in Four Baleen Whales: Eubalaena glacialis, Balaenoptera borealis, Megaptera novaeangliae, and Balaenoptera physalus," *J. Mammal.*, vol. 60, no. 1, pp. 155–163, 1979. https://doi.org/10.2307/1379766

[124] M. Abdel-Basset, R. Mohamed, and M. Abouhawwash, "A new fusion of whale optimizer algorithm with kapur's entropy for multi-threshold image segmentation: Analysis and validations," *Artif. Intell. Rev.*, vol. 55, no. 8, pp. 6389–6459, 2022. https://doi.org/10.1007/s10462-022-10157-w

[125] A. A. Taheri and S. Golabi, "Thickness optimization and experimental validation of incremental collar forming of explosive welded Al/Cu bimetal sheet with an obround hole using finite element, whale algorithm and neural network," *Iran. J. Sci. Technol. Trans. Mech. Eng.*, vol. 46, pp. 885–900, 2022. https://doi.org/10.1007/s40997-021-00445-1

[126] A. V. Lakshmi and P. Mohanaiah, "Intelligent facial emotion recognition based on hybrid whale optimization algorithm and sine cosine algorithm," *Microprocess. Microsyst.*, vol. 95, p. 104718, 2022. https://doi.org/10.1016/J.MICPRO.2022.104718

[127] D. Butti, S. K. Mangipudi, and S. R. Rayapudi, "An improved whale optimization algorithm for the design of multi-machine power system stabilizer," *Int. Trans. Electr. Energy Syst.*, vol. 30, no. 5, p. e12314, 2020. https://doi.org/10.1002/2050-7038.12314

[128] M. Abdel-Basset, D. El-Shahat, K. Deb, and M. Abouhawwash, "Energy-aware whale optimization algorithm for real-time task scheduling in multiprocessor systems," *Appl. Soft Comput.*, vol. 93, p. 106349, 2020. https://doi.org/10.1016/j.asoc.2020.106349

[129] X. Li, Q. Yang, H. Wu, S. Tan, Q. He, N. Wang, and X. Yang, "Joints trajectory planning of robot based on slime mould whale optimization algorithm," *Algorithms*, vol. 15, no. 10, p. 363, 2022. https://doi.org/10.3390/A15100363

[130] G. Li, Y. Cui, L. Wang, and L. Meng, "Automatic registration algorithm for the point clouds based on the

optimized ransac and IWOA algorithms for robotic manufacturing," *Appl. Sci.*, vol. 12, no. 19, p. 9461, 2022. https://doi.org/10.3390/APP12199461

[131] X. Zong, J. Liu, Z. Ye, and Y. Liu, "Whale optimization algorithm based on levy flight and memory for static smooth path planning," *Int. J. Mod. Phys. C*, vol. 33, no. 10, p. 2250138, 2022. https://doi.org/10.1142/S0129183122501388

[132] F. Gul, S. Mir, and I. Mir, "Multi robot space exploration: A modified frequency whale optimization approach," in *AIAA SCITECH 2022 Forum*, San Diego, CA, USA, 2022. https://doi.org/10.2514/6.2022-1416

[133] F. Gul, I. Mir, W. Rahiman, and T. U. Islam, "Novel implementation of multi-robot space exploration utilizing coordinated multi-robot exploration and frequency modified whale optimization algorithm," *IEEE Access*, vol. 9, pp. 22 774–22 787, 2021. https://doi.org/10.1109/ACCESS.2021.3055852

[134] A. Brodzicki, M. Piekarski, and J. Jaworek-Korjakowska, "The whale optimization algorithm approach for deep neural networks," *Sensors*, vol. 21, no. 23, p. 8003, 2021. https://doi.org/10.3390/s21238003

[135] M. Petrović, Z. Miljković, and A. Jokić, "A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm," *Appl. Soft Comput.*, vol. 81, p. 105520, 2019. https://doi.org/10.1016/j.asoc.2019.105520

[136] J. Zan, P. Ku, and S. Jin, "Research on robot path planning based on whale optimization algorithm," in *2021 5th Asian Conference on Artificial Intelligence Technology (ACAIT)*, Haikou, China, 2021, pp. 500–504. https://doi.org/10.1109/ACAIT53529.2021.9731150

[137] O. Castillo, L. Trujillo, and P. Melin, "Multiple objective genetic algorithms for path-planning optimization in autonomous mobile robots," *Soft Comput.*, vol. 11, pp. 269–279, 2007. https://doi.org/10.1007/s00500-006-0068-4

[138] T. K. Dao, T. S. Pan, and J. S. Pan, "A multi-objective optimal mobile robot path planning based on whale optimization algorithm," in *2016 IEEE 13th International Conference on Signal Processing (ICSP)*, Chengdu, China, 2016, pp. 337–342. https://doi.org/10.1109/ICSP.2016.7877851

[139] F. Gul, I. Mir, L. Abualigah, S. Mir, and M. Altalhi, "Cooperative multi-function approach: A new strategy for autonomous ground robotics," *Future Gener. Comput. Syst.*, vol. 134, pp. 361–373, 2022. https://doi.org/10.1016/J.FUTURE.2022.04.007

[140] Y. Dai, J. Yu, C. Zhang, B. Zhan, and X. Zheng, "A novel whale optimization algorithm of path planning strategy for mobile robots," *Appl. Intell.*, vol. 53, no. 9, pp. 10 843–10 857, 2023. https://doi.org/10.1007/S10489-022-04030-0/TABLES/4

[141] A. Chhillar and A. Choudhary, "Mobile robot path planning based upon updated whale optimization algorithm," in *2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence)*, Noida, India, 2020, pp. 684–691. https://doi.org/10.1109/Confluence47617.2020.9058323

[142] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Adv. Eng. Softw.*, vol. 69, pp. 46–61, 2014. https://doi.org/10.1016/j.advengsoft.2013.12.007

[143] T. Thanya and S. W. Franklin, "Grey wolf optimizer based deep learning for pancreatic nodule detection," *Intell. Autom. Soft Comput.*, vol. 36, no. 1, pp. 97–112, 2023. https://doi.org/10.32604/IASC.2023.029675

[144] S. Bourouis, S. S. Band, A. Mosavi, S. Agrawal, and M. Hamdi, "Meta-heuristic algorithm-tuned neural network for breast cancer diagnosis using ultrasound images," *Front. Oncol.*, vol. 12, p. 834028, 2022. https://doi.org/10.3389/FONC.2022.834028

[145] S. W. Nappu, J. D. Nappu, and C. Zhang, "Path-relinking grey wolf optimizer for solving operation sequencing problem," in *2022 IEEE International Conference on Mechatronics and Automation (ICMA)*, 2022, pp. 1375–1380. https://doi.org/10.1109/ICMA54519.2022.9856130

[146] C. Yang, Z. Liu, W. Zhang, and W. Yue, "Cooperative online path planning for UAVs based on DMPC and improved grey wolf optimizer," in *2022 41st Chinese Control Conference (CCC)*, Hefei, China, 2022, pp. 5008–5013. https://doi.org/10.23919/CCC55666.2022.9901663

[147] L. Wei, Z. Cai, and K. Zhou, "Multi-objective gray wolf optimization algorithm for multi-agent pathfinding problem," in *2022 IEEE 5th International Conference on Electronics Technology (ICET)*, Chengdu, China, 2022, pp. 1241–1249. https://doi.org/10.1109/ICET55676.2022.9824428

[148] F. Gul, W. Rahiman, S. S. N. Alhady, A. Ali, I. Mir, and A. Jalil, "Meta-heuristic approach for solving multi-objective path planning for autonomous guided robot using PSO–GWO optimization algorithm with evolutionary programming," *J. Ambient Intell. Humaniz. Comput.*, vol. 12, pp. 7873–7890, 2021. https:

//doi.org/10.1007/s12652-020-02514-w

[149] F. Gul, I. Mir, D. Alarabiat, H. M. Alabool, L. Abualigah, and S. Mir, "Implementation of bio-inspired hybrid algorithm with mutation operator for robotic path planning," *J. Parallel Distrib. Comput.*, vol. 169, pp. 171–184, 2022. https://doi.org/10.1016/J.JPDC.2022.06.014

[150] L. Dong, X. Yuan, B. Yan, Y. Song, Q. Xu, and X. Yang, "An improved grey wolf optimization with multi-strategy ensemble for robot path planning," *Sensors*, vol. 22, no. 18, p. 6843, 2022. https://doi.org/10.3390/S22186843

[151] R. Kumar, L. Singh, and R. Tiwari, "Comparison of two meta-heuristic algorithms for path planning in robotics," in *2020 International Conference on Contemporary Computing and Applications (IC3A)*, Lucknow, India, 2020, pp. 159–162. https://doi.org/10.1109/IC3A48958.2020.233289

[152] R. Euldji, N. Batel, R. Rebhi *et al.*, "Optimal Backstepping-FOPID controller design for wheeled mobile robot," *J. Eur. Syst. Autom.*, vol. 55, no. 1, pp. 97–107, 2022. https://doi.org/10.18280/JESA.550110

[153] J. Barrow, P. Davies, C. Harper, and D. Shortt, *Science and Ultimate Reality: Quantum Theory, Cosmology and Complexity*. Cambridge, UK: Cambridge University Press, 2004. https://doi.org/10.1017/CBO9780511814990

[154] S. Mirjalili, S. Mirjalili, and A. Hatamlou, "Multi-verse optimizer: A nature-inspired algorithm for global optimization," *Neural Comput. Appl.*, vol. 27, no. 2016, pp. 495–513, 2016. https://doi.org/10.1007/s00521-015-1870-7

[155] J. Khoury, B. Ovrut, N. Seiberg, P. Steinhardt, and N. Turok, "From big crunch to big bang," *Phys. Rev. D*, vol. 65, no. 8, p. 086007, 2002. https://doi.org/10.1103/PhysRevD.65.086007

[156] D. Eardley, "Death of white holes in the early universe," *Phys. Rev. Lett.*, vol. 33, no. 7, pp. 442–444, 1974. https://doi.org/10.1103/PhysRevLett.33.442

[157] R. Wald, "The thermodynamics of black holes," *Living Rev. Relativ.*, vol. 4, pp. 1–44, 2001. https://doi.org/10.12942/lrr-2001-6

[158] A. Guth, "Eternal inflation and its implications," *J. Phys. A Math. Theor.*, vol. 40, no. 25, p. 6811, 2007. https://doi.org/10.1088/1751-8113/40/25/S25

[159] L. Zhu, J. Lin, and Z. Wang, "A discrete oppositional multi-verse optimization algorithm for multi-skill resource constrained project scheduling problem," *Appl. Soft Comput.*, vol. 85, p. 105805, 2019. https://doi.org/10.1016/j.asoc.2019.105805

[160] J. Liu, J. Wei, A. Heidari, F. Kuang, S. Zhang, W. Gui, and Z. Pan, "Chaotic simulated annealing multi-verse optimization enhanced kernel extreme learning machine for medical diagnosis," *Comput. Biol. Med.*, vol. 144, p. 105356, 2022. https://doi.org/10.1016/j.compbiomed.2022.105356

[161] R. Ikram, H. Dai, A. Ewees, J. Shiri, O. Kisi, and M. Zounemat-Kermani, "Application of improved version of multi verse optimizer algorithm for modeling solar radiation," *Energy Reports*, vol. 8, pp. 12 063–12 080, 2022. https://doi.org/10.1016/J.EGYR.2022.09.015

[162] M. Iqbal, A. Bhatti, A. Butt, Y. Sheikh, K. Paracha, and R. Ashique, "Solution of economic dispatch problem using hybrid multi-verse optimizer," *Electric Power Syst. Res.*, vol. 208, p. 107912, 2022. https://doi.org/10.1016/j.epsr.2022.107912

[163] S. Liang, R. Zhang, and Y. Bai, "3D path planning based on MMVO," in *2021 33rd Chinese Control and Decision Conference (CCDC)*, Kunming, China, 2021, pp. 7385–7391. https://doi.org/10.1109/CCDC52312.2021.9601774

[164] E. Ghith, M. Sallam, I. Khalil, M. Serry, and S. Hammad, "Real time implementation for tuning PID controller based on advanced optimization techniques for micro robotics system," *Int. J. Eng. Adv. Technol.*, 2021. https://doi.org/10.35940/ijeat.f3073.0810621

[165] E. Ghith and F. Tolba, "Labview implementation of tuning PID controller using advanced control optimization techniques for micro-robotics system," *Int. J. Mech. Eng. Robot. Res.*, vol. 11, no. 9, pp. 653–661, 2022. https://doi.org/10.18178/IJMERR.11.9.653-661

[166] R. Jarray, M. Al-Dhaifallah, H. Rezk, and S. Bouallègue, "Path planning of quadrotors in a dynamic environment using a multicriteria multi-verse optimizer," *Comput. Mater. Continua*, vol. 69, no. 2, pp. 2159–2180, 2021. https://doi.org/10.32604/cmc.2021.018752

[167] S. Jalali, A. Khosravi, P. Kebria, R. Hedjam, and S. Nahavandi, "Autonomous robot navigation system using the evolutionary multi-verse optimizer algorithm," in *2019 IEEE International Conference on Systems, Man and Cybernetics (SMC)*, Bari, Italy, 2019, pp. 1221–1226. https://doi.org/10.1109/SMC.2019.8914399

[168] X. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*, 2010, pp. 65–74. https://doi.org/10.1007/978-3-642-12538-6_6

[169] F. Ajeil, I. Ibraheem, A. Humaidi, and Z. Khan, "A novel path planning algorithm for mobile robot in dynamic environments using modified bat swarm optimization," *J. Eng.*, vol. 2021, no. 1, pp. 37–48, 2021. https://doi.org/10.1049/TJE2.12009

[170] J. Perez, P. Melin, O. Castillo, F. Valdez, C. Gonzalez, and G. Martinez, "Trajectory optimization for an autonomous mobile robot using the bat algorithm," in *Fuzzy Logic in Intelligent System Design: Theory and Applications*, 2018, pp. 232–241. https://doi.org/10.1007/978-3-319-67137-6_25

[171] F. Glover, "Tabu search—Part I," *ORSA J. Comput.*, vol. 1, no. 3, pp. 190–206, 1989. https://doi.org/10.1287/ijoc.1.3.190

[172] W. Khaksar, T. Hong, M. Khaksar, and O. Motlagh, "Sampling-based tabu search approach for online path planning," *Adv. Robot.*, vol. 26, no. 8-9, pp. 1013–1034, 2012. https://doi.org/10.1163/156855312X632166

[173] R. Pradhan, S. Panigrahi, and P. K. Sahu, "Conformational search for the building block of proteins based on the gradient gravitational search algorithm (ConfGGS) using force fields: CHARMM, AMBER, and OPLS-AA," *J. Chem. Inf. Model.*, vol. 63, no. 2, pp. 670–690, 2023. https://doi.org/10.2307/3010402

[174] L. Xing, Y. Liu, H. Li, C. C. Wu, W. C. Lin, and X. Chen, "A novel tabu search algorithm for multi-AGV routing problem," *Mathematics*, vol. 8, no. 2, p. 279, 2020. https://doi.org/10.3390/math8020279

[175] I. Châari, A. Koubâa, H. Bennaceur, A. Ammar, S. Trigui, M. Tounsi, and H. Youssef, "On the adequacy of tabu search for global robot path planning problem in grid environments," *Procedia Comput. Sci.*, vol. 32, pp. 604–613, 2014. https://doi.org/10.1016/j.procs.2014.05.466

[176] S. Kumar, M. K. Muni, K. K. Pandey, A. Chhotray, and D. R. Parhi, "Path planning and control of mobile robots using modified Tabu search algorithm in complex environment," in *International Conference on Artificial Intelligence in Manufacturing & Renewable Energy (ICAIMRE)*, Odisha, India, 2019. https://doi.org/10.2139/ssrn.3539922

[177] W. Khaksar, T. Hong, K. Sahari, M. Khaksar, and J. Torresen, "Sampling-based online motion planning for mobile robots: Utilization of tabu search and adaptive neuro-fuzzy inference system," *Neural Comput. Appl.*, 2019. https://doi.org/10.1007/s00521-017-3069-6

[178] M. Panda, R. Priyadarshini, and S. Pradhan, "Autonomous mobile robot path planning using hybridization of particle swarm optimization and tabu search," in *2016 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC)*, Chennai, India, 2017. https://doi.org/10.1109/ICCIC.2016.7919636

[179] K. Balan and C. Luo, "Optimal trajectory planning for multiple waypoint path planning using tabu search," in *2018 9th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, New York, NY, USA, 2018. https://doi.org/10.1109/UEMCON.2018.8796810

[180] H. Lopes and D. Lima, "Surveillance task optimized by evolutionary shared tabu inverted ant cellular automata model for swarm robotics navigation control," *SRN Electron. J.*, 2021. https://doi.org/10.2139/ssrn.3962774

[181] H. Hliwa, M. Daoud, N. Abdulrahman, and B. Atieh, "Optimal path planning of mobile robot using hybrid tabu search-firefly algorithm," *Int. J. Comput. Sci. Trends Technol.*, vol. 6, no. 6, p. 7, 2018. http://www.ijcstjournal.org/volume-6/issue-6/IJCST-V6I6P6.pdf

[182] W. K. Wong and C. I. Ming, "A review on metaheuristic algorithms: Recent trends, benchmarking and applications," in *2019 7th International Conference on Smart Computing & Communications (ICSCC)*, Sarawak, Malaysia, 2019, pp. 1–5. https://doi.org/10.1109/ICSCC.2019.8843624

[183] S. Gholizadeh and H. Barati, "A comparative study of three metaheuristics for optimum design of trusses," *J. Print Media Technol. Res.*, vol. 1, no. 4, pp. 223–233, 2012.

[184] M. Dorigo, M. Birattari, and T. Stutzle, "Ant colony optimization," *IEEE Comput. Intell. Mag.*, vol. 1, no. 4, pp. 28–39, 2006. https://doi.org/10.1109/MCI.2006.329691

[185] F. Glover, "Future paths for integer programming and links to artificial intelligence," *Comput. Oper. Res.*, vol. 13, no. 5, pp. 533–549, 1986. https://doi.org/10.1016/0305-0548(86)90048-1

[186] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Ph.D. dissertation, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[187] D. Chen, Z. Wang, G. Zhou, and S. Li, "Path planning and energy efficiency of heterogeneous mobile robots using Cuckoo–beetle swarm search algorithms with applications in ugv obstacle avoidance," *Sustainability*,

vol. 14, no. 22, p. 15137, 2022. https://doi.org/10.3390/SU142215137

[188] O. Serrano-Pérez, M. Villarreal-Cervantes, J. González-Robles, and A. Rodríguez-Molina, "Meta-heuristic algorithms for the control tuning of omnidirectional mobile robots," *Eng. Optim.*, vol. 52, no. 2, pp. 329–342, 2019. https://doi.org/10.1080/0305215X.2019.1585834

[189] F. Ajeil, I. Ibraheem, M. Sahib, and A. Humaidi, "Multi-objective path planning of an autonomous mobile robot using hybrid PSO-MFB optimization algorithm," *Appl. Soft Comput.*, vol. 89, p. 106076, 2020. https://doi.org/10.1016/j.asoc.2020.106076

[190] M. Petrović, A. Jokić, Z. Miljković, and Z. Kulesza, "Multi-objective scheduling of a single mobile robot based on the grey wolf optimization algorithm," *Appl. Soft Comput.*, vol. 131, p. 109784, 2022. https://doi.org/10.2139/ssrn.4058009

[191] I. Chaari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and K. Al-Shalfan, "SmartPATH: An efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots," *Int. J. Adv. Robotic Syst.*, vol. 11, no. 7, p. 94, 2014. https://doi.org/10.5772/58543

[192] S. Xu, E. Ho, and H. Shum, "A hybrid metaheuristic navigation algorithm for robot path rolling planning in an unknown environment," *Mechatronic Syst. Control*, vol. 47, no. 4, pp. 216–224, 2019. http://doi.org/10.2316/J.2019.206-4649

[193] A. Farley, J. Wang, and J. Marshall, "How to pick a mobile robot simulator: A quantitative comparison of CoppeliaSim, Gazebo, MORSE and Webots with a focus on accuracy of motion," *Simul. Model. Pract. Theory*, vol. 120, p. 102629, 2022. http://doi.org/10.1016/J.SIMPAT.2022.102629

[194] G. van Rossum, *Python Language*, 1991. https://www.python.org/

[195] B. Stroustrup, *C/C++*, 1985. https://isocpp.org/

[196] J. Little, C. Moler, and B. Steve, *MathWorks*, 1984. https://www.mathworks.com/

[197] C. R. AG, *CoppeliaSim*, 2010. https://www.coppeliarobotics.com/

[198] P. Robotics, *Moveit*. https://moveit.ros.org/

[199] Gazebosim, *Gazebosim*, 2004. https://gazebosim.org/home

[200] O. Michel, *Webots*, 1996. https://cyberbotics.com/

[201] Morse, *Morse*, 2013. http://morse-simulator.github.io/

[202] S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper, "Usarsim: A robot simulator for research and education," in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation*, Rome, Italy, 2007, pp. 1400–1405. https://doi.org/10.1109/ROBOT.2007.363180