Results in Control and Optimization 21 (2025) 100634

Contents lists available at ScienceDirect

Results in Control and Optimization

journal homepage: www.elsevier.com/locate/rico

Local path planning based on Bi-population Swarms optimization algorithms

Asmaa Shareef [a], Salah Al-darraji [a], Suhaib Al-Ansarry [a], Zaid Ameen Abduljabbar [b,c,d], Vincent Omollo Nyangaresi [e,f], Ali Hasan Ali [g,h,i], Husam A. Neamah [j,*]

[a] Department of Computer Science, College of Computer Science and Information Technology, University of Basrah, Basrah, 61004, Iraq
[b] Department of Computer Science, College of Education for Pure Sciences, University of Basrah, Basrah 61004, Iraq
[c] Department of Cyber Security, College of Science, Al-Kunooze University, Basrah 61004, Iraq
[d] Shenzhen Institute, Huazhong University of Science and Technology, Shenzhen 518000, China
[e] Department of Computer Science and Software Engineering, Jaramogi Oginga Odinga University of Science and Technology, Bondo 40601, Kenya
[f] Department of Applied Electronics, Saveetha School of Engineering, SIMATS, Chennai, Tamilnadu, 602105, India
[g] Department of Mathematics, College of Education for Pure Sciences, University of Basrah, Basrah 61004, Iraq
[h] Technical Engineering College, Al-Ayen University, Thi-Qar 64001, Iraq
[i] Institute of Mathematics, University of Debrecen, Pf. 400, H-4002, Debrecen, Hungary
[j] Department of Electrical Engineering and Mechatronics, Faculty of Engineering, University of Debrecen, Debrecen, 4028, Otemeto u.4-5, Hungary

ARTICLE INFO

Keywords:
Local robot path planning
Dynamic obstacles
Grasshopper Optimization Algorithm (GOA)
Waypoints
Bi-population Swarms algorithms

ABSTRACT

Robot navigation and path planning are among the most critical challenges facing mobile robots. Modern techniques have surpassed traditional methods in reducing high complexity by utilizing probabilistic or optimal solutions, such as algorithms based on swarm intelligence. This work proposes a novel global path planning using a bipopulation grasshopper optimization algorithm (BiGOA). Superior performance results were obtained compared to the original grasshopper optimization algorithm regarding time, cost, and path length for an average of 1000 times executions in environments of varying complexities. On the other hand, the optimal path, local path planning, and dynamic obstacle avoidance are done by the Dynamic Window Approach (DWA) algorithm that works within the follow-waypoint technique in mobile vehicles in the Robot Operating System (ROS). During robot movement, it tends to find a shorter and optimal path using a modified version of the waypoint method. A state machine determines the optimal path using the improved follow-waypoint algorithm during the robot's real-time movement, thereby controlling the path's smoothing, shortening, and continuity. This study compares the BiGOA algorithm with an optimal path method based on arrival time over ten trials. The results show that BiGOA paths were, on average, 33 s longer. Additionally, the robot's performance using the proposed method was tested in a dynamic and complex environment, where the results indicated a reduction in both time and effort.

* Corresponding author.
E-mail address: husam@eng.unideb.hu (H.A. Neamah).

https://doi.org/10.1016/j.rico.2025.100634
Received 9 July 2025; Received in revised form 11 October 2025; Accepted 19 November 2025
Available online 20 November 2025

1. Introduction

Path planning for an autonomous robot is one of the most critical challenges in robot navigation. Many studies have been presented regarding robot movement from starting to goal without colliding with obstacles in the robot's environment [1].

Global–local path planning constitutes one of the most important challenges, as it provides solutions to the problems of global path planning when obstacles are known (static) or unknown (dynamic). Therefore, innovative hybrid methods should be presented to address these challenges. Yao [2] proposed a hybrid A* algorithm for global path planning with an improved artificial potential field algorithm for local path planning. Other studies highlight the challenges robots face in crowded environments, leading to the development of numerous algorithms to address these challenges [3,4].

In recent years, methods inspired by natural behaviors, such as ant colony optimization (ACO) and particle swarm optimization (PSO), have shown notable effectiveness in navigating complex environments [5]. These algorithms are also extensively applied in optimization tasks, where they contribute to achieving a better balance between exploration and exploitation in search space [6]. The Grasshopper Optimization Algorithm (GOA) is an optimization algorithm introduced by Saremi et al. in 2017 [7]. It is an artificial swarm intelligence algorithm and belongs to population-based and meta-heuristic algorithms. It has proven its superiority in many fields including medical, engineering, industrial and others [8–11]. The GOA is particularly significant due to its ability to optimize multiple objectives related to path planning and robot navigation.

Elmi and Efe [12] proposed the GOA to optimize multiple objectives related to robots, including finding paths and minimizing distances. Moreover, in a separate publication [13], the authors have applied GOA to prevent collision with obstacles in online path planning; one of the disadvantages of both methods is that the path has not been planned globally. Shareef and Al-Darraji [14] improved the original GOA model to manage global path planning through environments with static obstacles, but the path was not planned locally.

Zhao et al. [15] proposed a bidirectional association algorithm (BALA) in which start and end points are connected to obtain the global shortest path to ensure efficient planning in diverse environments. Ge et al. [16] investigated a bidirectional technique in path planning for the RRT* and APF algorithms with a B-spline curve used for path smoothing generated in complex dynamic environments. Al-Ansarry et al. [17] implemented a bidirectional approach to enhance efficiency and reduce the time and cost associated with the conventional Rapidly Exploring Random Tree (RRT) algorithm. They incorporated enhancements to transform it into a connected adaptive algorithm. In the field of optimization algorithms [18], the bipopulation method has been used to overcome the local optimum problem in the Particle Swarm Optimization (PSO) algorithm for path planning. However, it has not been evaluated in more realistic or dynamic environments.

Guti'errez et al. [19] proposed a mobile vehicle controller for automated driving systems. They made the path smooth and continuous using the 'waypoint'. However, a drawback of this work is that it does not consider path length. In recent years' numerous studies and research endeavors have presented various techniques for identifying dynamic obstacles and facing challenges in navigating without colliding obstacles, necessitating localized path planning. Specific investigations have offered resolutions by devising sensors embedded within vehicles. These efforts concentrate on identifying obstacle types, diverse heights, and the multiple causes of collisions [20].

On the other hand, various studies and algorithms have been introduced for effective and practical obstacle collision avoidance, the most prominent of which is the dynamic window approach (DWA) algorithm [21]. Alternatively, some studies employed the DWA as a local planner. Furthermore, the A* algorithm was utilized in [22,23] to maneuver dynamic obstacles. While several studies have enhanced the Dynamic Window Approach (DWA) algorithm to improve local navigation and path smoothing for mobile robots, these improvements often come at the cost of increased algorithmic complexity [24].

Robot operating systems have witnessed progress in the field of path planning. Sometimes, they have been used in global path planning, as in [25], where the optimal path was planned by developing backpropagation on a neural network to find the shortest path based on the ROS platform in the parking field. On the other hand, ROS was used to develop the navigation of mobile cleaning robots based on the DWA algorithm [26]. Based on the studies mentioned earlier, several challenges are stated in the realm of robot navigation and mobility, including global path planning, obstacle detection, and collision avoidance. This research introduces a novel and intriguing approach to address both global and local path planning in static, complex, and dynamic environments for autonomous mobile robots. Our contributions can be outlined as follows:

1. Develop a bi-population grasshopper optimization algorithm (BiGOA) based on the GOA, which reduces the global GOA path's randomness and produces a near-optimal path by publishing two populations of grasshoppers around both sides, the starting and the target point. This method minimizes the chaotic points that make the trajectory zigzagging and unsuitable for robot movement.
2. Enhance the global path of BiGOA by optimizing it and ensuring its shortness, smoothness, and continuity. This is achieved by evaluating all feasible direct paths to each waypoint on the BiGOA path with the assistance of the waypoints [19]. It is a well-known, widely used tool in robotics for path planning and navigation.
3. Enhance the follow waypoint algorithm to shorten the planned path by eliminating unnecessary waypoints.
4. When dealing with dynamic obstacles that disrupt the global trajectory of a robot, the DWA algorithm is employed to avoid collisions. This involves treating each point as a sub-goal and locally planning the path, ensuring an efficient and secure navigation process.

Table 1 summarizes related works, highlighting their advantages and limitations compared to the proposed BiGOA-DWA approach.

The rest of this paper is structured as follows: Section 2 presents the methodologies. The details of the experiments and outcomes derived from the proposed approaches are provided in Section 3. The summary and conclusion of this paper are introduced in Section 4.

Table 1
Comparison of related works with the proposed BiGOA-DWA approach.

Ref.	Year	Technique used	Advantages	Limitations compared to this work
[2]	2023	Hybrid A* + Improved APF	Combines global and local planning	Limited effectiveness in dynamic environments
[5,6]	2023, 2024	ACO/PSO| GWO Optimization	Balances exploration and exploitation in search	No direct integration with local planning
[7,12]	2017, 2018	Grasshopper Optimization Algorithm (GOA)	Multi-objective optimization, swarm-based	No hybrid or local dynamic planning
[13]	2021	GOA for dynamic obstacle avoidance	Online collision avoidance with GOA	No global planner; path not optimal
[14]	2022	Improved GOA for static global planning	Effective for static environments	Not suitable for local dynamic planning
[15]	2021	BALA (Bidirectional search)	Efficient shortest global path	Lacks obstacle avoidance in dynamic scenes
[16]	2021	RRT* + APF + B-spline	Smooth path generation in complex environments	High computation, lacks swarm intelligence
[17]	2023	Bidirectional adaptive RRT	Faster and cost-effective planning	No waypoint optimization or local collision handling
[18]	2024	Bi-population PSO	Avoids local optima in optimization	Not tested in realistic dynamic scenarios
[19]	2020	Waypoint-Based planning	Ensures smooth continuous path	Ignores total path length
[20]	2012	Sensor-based obstacle detection	Detects various obstacle types	Requires external hardware and preprocessing
[21]	2002	Dynamic Window Approach (DWA)	Real-time obstacle avoidance	Only local planning; no global optimization
[22,23]	2025,2020	ROS + A*/DWA	Reactive path planning in dynamic environments	No global optimization; lacks swarm-based methods
[25,26]	2020, 2023	Neural Networks + ROS	Learns optimal paths from data	Limited to static or pre-learned conditions
The proposed work	**2025**	**BiGOA + Waypoint Optimization + DWA**	**Global–local integration; smooth, short, optimized paths; dynamic collision avoidance**	–

2. Research methodology

The proposed method, Bipopulation Grasshopper Optimization Algorithm (BiGOA), is a novel approach that integrates the concept of bidirectional search with a swarm intelligence algorithm, the Grasshopper Optimization Algorithm (GOA). The bidirectional search addresses limitations of traditional search algorithms such as RRT and A* [17], which rely heavily on tree expansion or graph traversal techniques. Unlike traditional algorithms that require complex merging criteria, BiGOA introduces a direct communication mechanism: it identifies a path between two grasshoppers if a direct line of sight exists between them. This early connection strategy significantly accelerates the path selection process and reduces the chaotic wandering typically associated with swarm-based methods. Moreover, the attractive and repulsive forces inherent in the grasshopper optimization algorithm facilitate continuous interaction between the two populations. This transforms the search process from a simple bidirectional expansion into an adaptive heuristic exploration, enabling the algorithm to flexibly navigate complex and irregular environments by expanding the effective search space. The integration of BiGOA with the Dynamic Window Approach (DWA) adds an important layer of synergy between global and local planning. While BiGOA generates an efficient global path, DWA is employed to locally adjust the robot's trajectory in response to dynamic obstacles encountered during navigation. This integration makes the BiGOA-DWA framework a comprehensive and effective solution for real-time path planning in dynamic and crowded environments. The proposed approach, BiGOA-DWA can be summarized into the following steps:

1. Initializing the fundamental parameters of the (GOA), defining the population size, Cmin, Cmax, and the iteration count. Also, initializing the environmental aspects, including the number and sizes of obstacles, and establishing the starting and ending points for path planning.
2. Commencing from the initial position, we release a population of grasshoppers to navigate towards the goal, aiming to explore optimal solutions or positions.
3. Starting from the endpoint (goal), we release a population of grasshoppers directed towards the starting point, seeking to identify optimal solutions or positions.
4. A direct connection occurs when two grasshoppers can see each other without obstacles. This will result in the establishment of a global path for BiGOA.
5. The parameters of the waypoints are initialized with the ROS framework.

6. To augment the count of points on the global path BiGOA, we partition the path into multiple segments. The points situated within these segments are selected and treated as waypoints.
7. Setting up the TurtleBot3 robot inside the environment to simulate traversing the BiGOA path, reducing the length and angle of LIDAR. This setup enables the robot to navigate through narrow corridors effectively, in addition to the robot's ability to maintain its speed and path by reducing the number of stops.
8. The robot moves from the starting point to achieve the optimal path. The path is generated instantly by guiding the robot directly towards the goal without colliding with any obstacle in the direction of the robot.
9. Employing the DWA algorithm enables the robot to prevent collisions with dynamic obstacles, treating each waypoint as a sub-goal. We repeated this process whenever the robot encountered a new dynamic obstacle. Fig. 1 presents a block diagram that illustrates a set of mechanisms created for the Grasshopper Optimization Algorithm, used in both global and local path planning of autonomous mobile robots.

2.1. Global path planning BiGOA

The grasshopper optimization algorithm, introduced in 2017, is a swarm intelligence and population spread algorithm [7]. It proceeds through two phases, corresponding to the grasshopper's life cycle: exploitation and exploration. The exploitation phase occurs when the grasshopper lacks wings and can only walk, while the exploration phase occurs when the grasshopper grows wings and can jump. Each phase demonstrates various advantages and disadvantages in path planning, as detailed in Fig. 2.

The local minimum is one of the problems faced in the exploitation phase when they are spread out in close spaces within U-shaped obstacles or crowded environments [27]. Consequently, the algorithm completes its iterations without reaching the best position close to the target. On the other hand, in the exploration phase, the grasshoppers jump into wide and separate spaces, thus causing slow convergence and failing to reach the best positions.

Theoretically, it is possible to address the two-phase problem in path planning. Two grasshopper populations are published around each of the two positions (the starting and the target). The connection is made between the first two positions that meet directly, as shown in Fig. 3

The complexities of each GOA and BiGOA are nearly identical. Suppose $N = 10$ grasshoppers in one swarm of GOA and $N = 5$ grasshoppers in each swarm of the BiGOA.

GOA	BiGOA
O(N)	O(2N)

Complexity of GOA ≃ Complexity of BiGOA.

Mathematically, the optimal position requires the use of various equations that influence the movement of the grasshopper, including the equations for wind and gravity as in Eqs. (1) and (2) [7].

$$X_i = c\left(\sum_{\substack{j=1 \\ j\neq i}}^{N} c\,\frac{UB_d - LB_d}{2}\, s\left(\left|x_j^d - x_i^d\right|\right)^{\frac{x_j - x_i}{d_{ij}}}\right) + \hat{T}_d, \tag{1}$$

$$c = c_{\max} - l\,\frac{c_{\max} - c_{\min}}{L} \tag{2}$$

Numerous studies have been conducted to determine the optimal parameter values for the main equations. In the field of global path planning, a bias factor has been introduced to balance between the optimal position and the target [14]. The complexity of the problem is directly related to the size of the search space. In global path planning, the search space can be visualized as a sphere expanding from the start when a single population is used. However, when two populations are initialized at both the starting and target points, two expanding spheres are formed. The probability of these spheres intersecting is significantly higher than that of a single sphere covering the entire search space. This increases the chances of finding a path more quickly, reduces the overall search effort, and minimizes the behavioral randomness of the grasshoppers' swarm. Moreover, this is particularly beneficial in large or complex environments where a one-population search might get trapped in local optima or take a very long time to find a viable path. The direct communication mechanism between the two populations serves as an early stopping criterion, further improving the global path, as illustrated in Fig. 4.

The GOA algorithm employs several parameters, including population size, which, when increased, enhances exploration but also increases computational cost. Likewise, increasing the max_iterations improves the optimization performance, though with diminishing returns beyond a certain threshold. Furthermore, the parameters $C_{\max}$ and $C_{\min}$ regulate the attractive and repulsive forces among grasshoppers. In this study, these parameter values were adopted from previous swarm optimization literature to achieve an optimal balance between exploration and exploitation. In Algorithm 1, the best positions for both populations are calculated in lines 12–21, excluding those that intersect obstacles (lines 22–35). The bipopulation structure of BiGOA effectively reduces the search distance required for each grasshopper, as individuals only need to encounter a grasshopper from the opposite population without obstacles (lines 36–39), rather than reaching the final goal of a single population of origin.

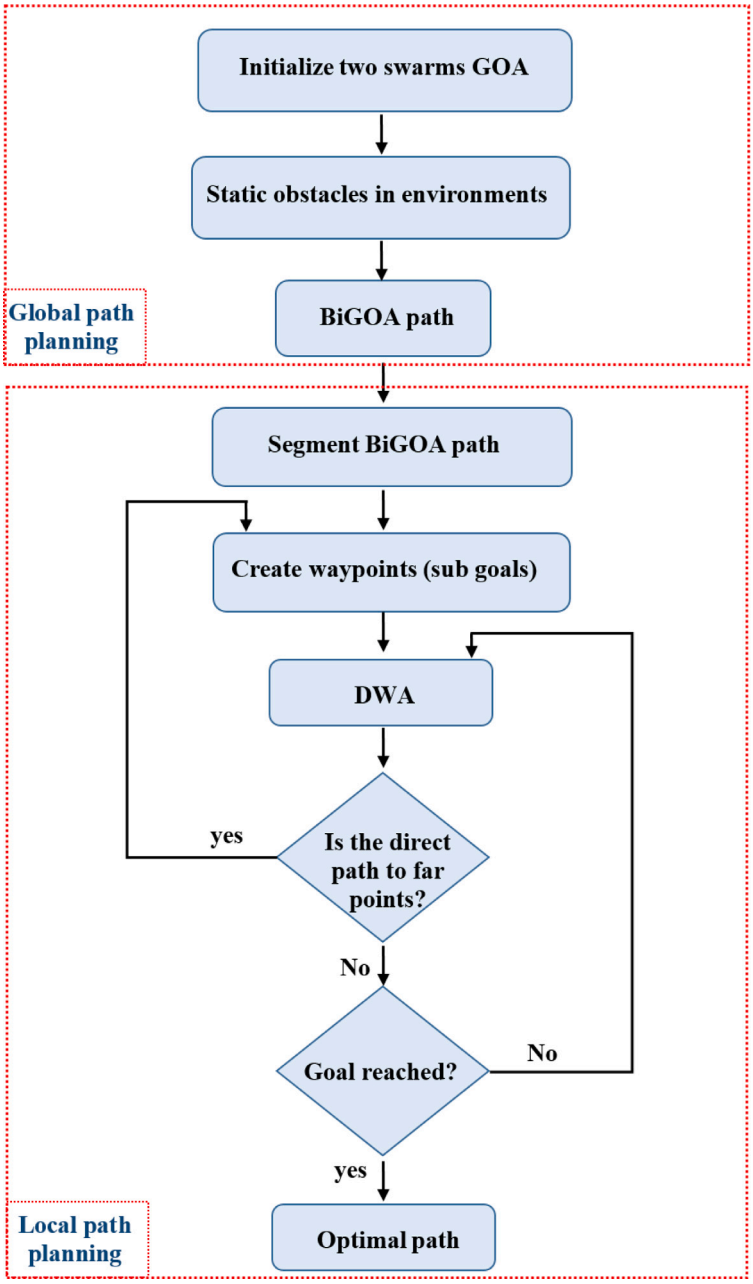

Fig. 1. Global and local paths.

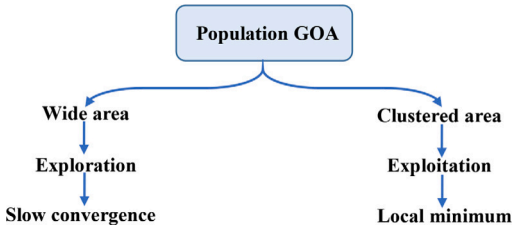

Fig. 2. Population of GOA.

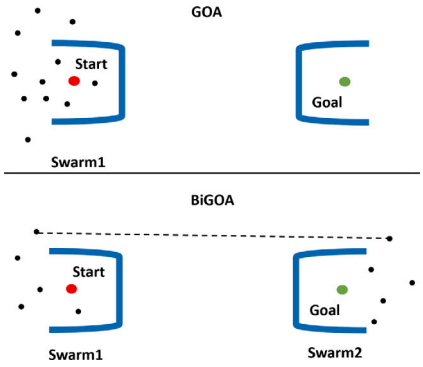

Fig. 3. GOA and BiGOA.

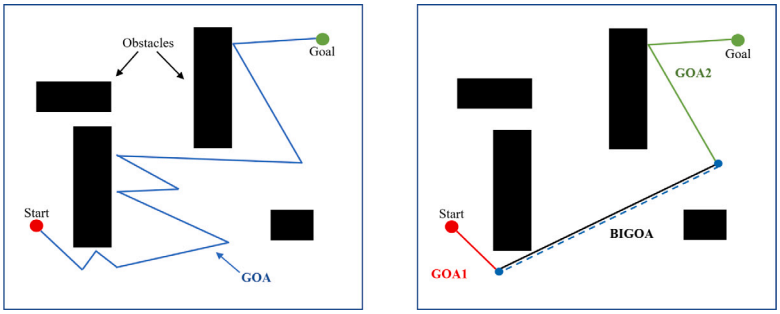

Fig. 4. Global paths (a) GOA path, (b) BiGOA path.

2.2. Local path planning (Improved Waypoint)

Local path planning focuses on short-range obstacle avoidance and motion adjustment based on real-time sensor data. Rather than depending on a complete model of the environment, it often employs biologically inspired strategies [28] or intelligent control [29] and learning systems [30], which have played a significant role in advancing local navigation capabilities. On the other hand, the Dynamic Window Approach (DWA) is a widely adopted local navigation strategy [21] that generates feasible trajectories by evaluating a range of velocity commands while accounting for the robot's dynamic constraints, as shown in Eq. (3).

$$G(v,w) = \alpha \cdot heading(v,w) + \beta \cdot dist(v,w) + \gamma \cdot velocity(v,w) \tag{3}$$

where $G(v,w)$: Dynamic window searches in constraints (acceleration and collision). α, β, and γ are weight factors to balance between reaching the goal, safety, and speed.

Initially, the focus is on identifying safe speeds to avoid collision with obstacles, thus narrowing the search space. In the subsequent stage, the algorithm selects the most suitable speeds to achieve the goal. Sensors are essential for this process to collect geometric information about obstacle locations; some robots utilize their cameras to transform visual data into proximity information for obstacle detection [31–33]. DWA is superior in path planning for autonomous vehicles in complex environments, determining the optimal path based on factors such as the cost function, distance of arrival, and proximity to obstacles. However, the limitation of DWA is the inability to find the global path. Therefore, alternative methods should be explored to follow the global path while considering dynamic obstacles [34]. In addition, DWA tries to find an obstacle-free path between two points. In the case of global path planning, the resulting path consists of a set of points (sub-goals). A known method to follow these subgoals is ROS, which provides some named libraries, 'follow-waypoints', for this purpose [35,36]. This work proposes processes to improve the global path obtained by BiGOA and to find solutions through local path planning if dynamic obstacles appear on the robot path. These processes include:

- *T*he range and angle of the sensor are important factors in the global path-tracking process. The range of the robot and its angle restrict its ability to navigate through narrow corridors. When the robot moves, it tracks the points in the global path obtained by the BiGOA algorithm; it should shorten and reduce the range and angle of the sensor to keep the robot moving in crowded areas and narrow corridors.
- *F*ollowing waypoints in BiGOA proved to be particularly challenging, primarily due to the limited number of hinge points generated by the grasshoppers' jumps. This limitation motivated the development of a modified version of the BiGOA waypoint-following algorithm, which incorporates path segmentation. Unlike simple partitioning, path segmentation in this context is an

Algorithm 1 BiGOA Path Planning

Require: Input: start_point, target_point, obstacles, max_iterations, population_size, lower_bound, upper_bound
Ensure: Output: path

Initialize swarm$_1$ positions: X_i^1, $i = 1, 2, \ldots,$ population_size
Initialize swarm$_2$ positions: X_i^2, $i = 1, 2, \ldots,$ population_size
Initialize $C_{\min}$ and $C_{\max}$
$paths_1 \leftarrow [\,]$, $paths_2 \leftarrow [\,]$
Evaluate fitness of each grasshopper:
  $B_1 \leftarrow$ best in swarm$_1$, $B_2 \leftarrow$ best in swarm$_2$
$iter \leftarrow 1$
while $iter \leq$ max_iterations **do**
  $C \leftarrow C_{\max} - iter \cdot \frac{C_{\max} - C_{\min}}{\text{max\_iterations}}$
  for $i = 1$ to population_size **do**
    Normalize distance using global path equations
    $new\_position_1 \leftarrow$ calculate_new_position(X_i^1, C, B_1)
    $new\_position_2 \leftarrow$ calculate_new_position(X_i^2, C, B_2)
    if $outside(new\_position_1, lower\_bound, upper\_bound)$ **then**
      $new\_position_1 \leftarrow$ repair$(new_position_1)$
    end if
    if $outside(new\_position_2, lower\_bound, upper\_bound)$ **then**
      $new\_position_2 \leftarrow$ repair$(new_position_2)$
    end if
    $line_1 \leftarrow$ line_segment$(X_i^1, new_position_1)$
    $line_2 \leftarrow$ line_segment$(X_i^2, new_position_2)$
    if not $intersect(line_1, obstacles)$ **then**
      $X_i^1 \leftarrow new\_position_1$
      $paths_1[i].append(X_i^1)$
      if $fitness(X_i^1) < fitness(B_1)$ **then**
        $B_1 \leftarrow X_i^1$
      end if
    end if
    if not $intersect(line_2, obstacles)$ **then**
      $X_i^2 \leftarrow new\_position_2$
      $paths_2[i].append(X_i^2)$
      if $fitness(X_i^2) < fitness(B_2)$ **then**
        $B_2 \leftarrow X_i^2$
      end if
    end if
    if $(j, k) =$ connect$(paths_1, paths_2, obstacles)$ **then**
      $path \leftarrow paths_1[j] + paths_2[k]$
      return path
    end if
  end for
  $iter \leftarrow iter + 1$
end while
return `no_path_found`

adaptive process that accounts for the distribution of obstacles, with the resulting waypoints treated as subgoals. The robot then follows these waypoints as a sequence of connected points. The main objective of removing unnecessary waypoints is to shorten the path, smooth the trajectory, and reduce navigation effort while ensuring safety and reliability during movement. A line-of-sight criterion guides the removal process: if the path between two nonadjacent waypoints is free of obstacles, intermediate points are bypassed. This ensures that the robot only follows the most efficient path, eliminating redundant detours, ultimately generating an optimal path, as depicted in Fig. 5.

- The BiGOA path points serve as subgoals for the DWA algorithm. When a dynamic obstacle is detected, local planning is immediately activated. The DWA operates continuously and interactively by sampling linear and angular velocities within kinematic constraints defined by cost functions while remaining synchronized with the TurtleBot3 control loop. This makes the DWA particularly suitable for dynamic obstacle avoidance, as it provides high real-time computational efficiency by exploring the velocity space and generating candidate velocity pairs (v, w), which directly affect computation time. Within this process, predicting the robot's short-term is relatively lightweight, whereas collision verification is the most computationally demanding component. The accuracy of the map, the level of local occupancy, and the number of sampled velocities influence the overall computational load. However, these factors consistently stay within the control loop's limits, typically requiring only a fraction of a millisecond per cycle. Consequently, navigation is executed smoothly and interactively, with no noticeable

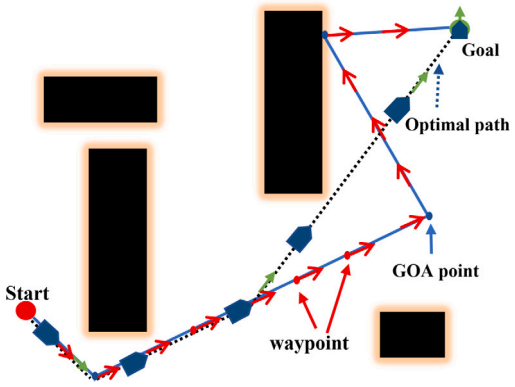

Fig. 5. Optimal path.

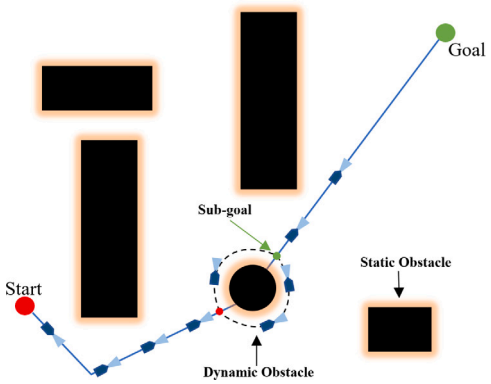

Fig. 6. Local path in dynamic obstacle.

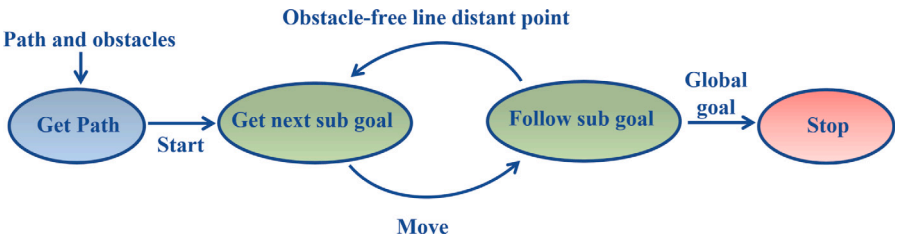

Fig. 7. Controller of state machine operation.

delay. Additionally, the algorithm realigns with the nearest available point (subgoal) from the global path, as illustrated in Fig. 6.

- The improved follow-waypoint module is implemented as a state machine to manage the overall flow of the node. Algorithm 2 presents the details of the improved follow-waypoint approach, which employs a formal finite state machine (FSM) model for path planning and navigation in the proposed BiGOA-DWA framework. In this proposed, the robot progresses through a sequence of states and transitions across defined nodes until the global goal is achieved, as illustrated in Fig. 7. The initial state, Get Path, initializes the process by providing the global path generated by BiGOA along with the environmental information. Upon receiving the Start signal, the FSM transitions to the next state. The second state, Get Next Subgoal, selects the next feasible waypoint along the planned trajectory, thereby decomposing the global path into smaller, manageable subgoals. Once a subgoal is identified, the FSM transitions to the Follow Subgoal state. In the third state, Follow Subgoal, the robot actively navigates towards the selected subgoal. If an obstacle-free line of sight exists to a more distant waypoint or directly to the global goal, the robot bypasses intermediate subgoals, thus optimizing the path. When the current subgoal is reached, the FSM returns to the Get Next Subgoal state. However, if the global goal is reached, the FSM transitions to the termination state.

Table 2
Experimental results comparison for path length (GOA and BiGOA).

	Map-1	Map-2	Map-3	Map-4	Map-5
GOA	5931	9620	9293	5437	5915
BiGOA	482	1342	853	520	450
EP (%)	91.87	86.05	90.82	90.44	92.39

Algorithm 2 Improved Follow Waypoint

Require: Input: path, obstacles
Ensure: Output: move robot on optimal path
segmented_path ← getSegmentedPath(path)
n ← last goal index
for $i = 0$ to n **do**
  subgoal p_i ← segmented_path[i]
  prepare DWA with subgoal p_i
  $j \leftarrow n$
  while not reached p_i **do**
    current_point ← getOdometry()
    while $j \geq i$ **and not** directPath(*current_point*, p_j, obstacles) **do**
      $j \leftarrow j - 1$
    end while
    if $j \geq i$ **then**
      $i \leftarrow j$
      subgoal p_i ← segmented_path[i]
      prepare DWA with subgoal p_i
    end if
  end while
end for

3. Experiments and results

Experiments were conducted on a 144 Hz FHD IPS Intel i7, 8-Core, 8 GB RAM, RTX 3050 system running Ubuntu 20.04. A comprehensive set of five environments with varying complexity was created to investigate three factors: time, cost, and path length in global path planning. On the other hand, the proposed system is implemented using the Robot Operating System (ROS), which provides an integrated environment for robot simulation and a flexible framework for developing robotic algorithms and software. The start and end points, along with a set of static obstacles, are input into the environment of the proposed algorithm, which is encapsulated within a dedicated ROS node. A global trajectory is then computed, consisting of a series of waypoints that are transmitted to the robot as references for trajectory optimization and smoothing. This process also accounts for dynamic obstacles, utilizing an algorithm integrated into the robot navigation toolkit in ROS. The robot simulation system computes dynamically safe velocities based on real-time sensor data. Three-dimensional simulations are conducted to monitor the robot's behavior and to evaluate the robustness of the proposed method in integrated global–local path planning.

3.1. Experiments-1 global path planning

In this experiment, multiple environments of size **1500 × 1500 cm** were created, each containing a set of static obstacles. The obstacle-to-space ratio ranged from **15**% to **35**% in five environments in Fig. 8 and from **45**% to **55**% in two additional environments, as illustrated in Fig. 9. These configurations were designed to evaluate the algorithm's efficiency in crowded environments with narrow passages. The red and green circles in the figure represent the start and goal points, respectively. Figs. 8(a–e) and 9(a,b) present the paths generated by the GOA algorithm, whereas Figs. 8(f–j) and 9(c,d) illustrate the corresponding results of the BiGOA algorithm. Each path comprises three segments: the first extending from the start point, the second from the goal point, and the third representing the connection between the two preceding segments. The experimental results demonstrate that BiGOA significantly reduces path randomness and yields a near-optimal trajectory. Because GOA relies on probabilistic search to locate optimal positions, the performance of the proposed method was validated by averaging the outcomes of 1000 independent executions and comparing them with those obtained from the original GOA algorithm. Fig. 8 was selected for experimentation due to its diverse environments, extensive area, and the presence of numerous obstacles. These characteristics provide suitable conditions for the grasshopper population to disperse and generate diverse movement paths, as detailed in the following factors:

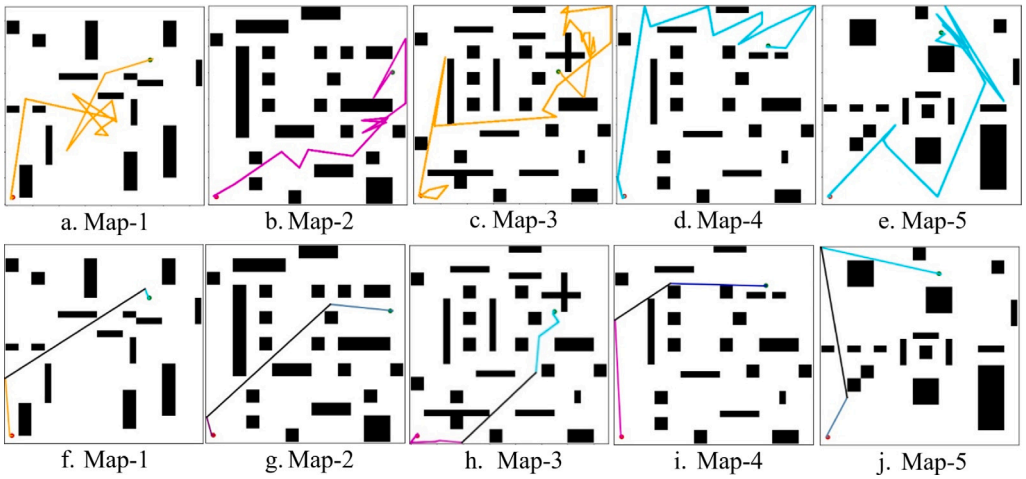

Fig. 8. Global paths experiments of (a–e) GOA and (f–j) BiGOA: A comparative analysis.

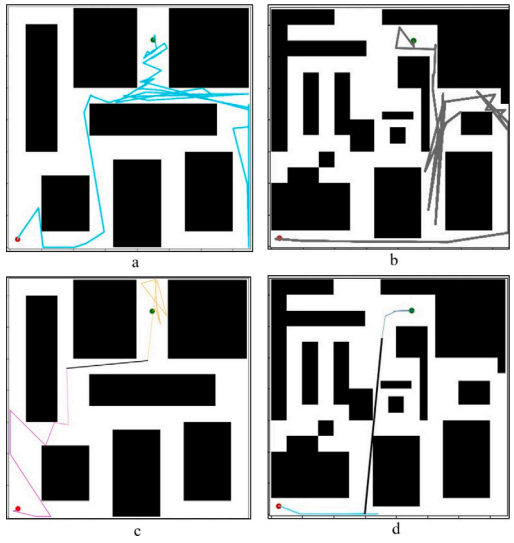

Fig. 9. Global paths experiments of (a–b) GOA and (c–d) BiGOA in crowded and complex environments: A comparative analysis.

- *Path length:*
  The path length has shown the highest values in Map-2 and Map-3 for both algorithms, GOA and BiGOA, as shown in Fig. 10. Alternatively, the lowest values were obtained in Map-5. Table 2 presents the detailed results for all environments.
  By applying the enhancement percentage (EP) formula [37], as shown in Eq. (4), to the results in Table 2,

$$EP = \left(1 - \frac{\text{BiGOA}}{\text{GOA}}\right) \times 100\% \tag{4}$$

  The enhancement percentages were found to range from 86.05% to 92.39%. This confirms a significant reduction in path length achieved by the proposed BiGOA method compared to the original GOA algorithm for all environments.
- *Cost:*
  In the context of path planning, the cost in the grasshopper algorithm represents the number of locations a grasshopper can jump into. Experiments have shown that the cost is less in low-complexity environments for both GOA and BiGOA, as shown in Fig. 11. The enhancement percentage (EP) was calculated for the cost values presented in Table 3. The proposed BiGOA approach demonstrated notable improvements in the more crowded maps, with enhancement percentages ranging from 19.05% to 42.79%. An exception was observed in Map-4, where a slight increase in cost (−2.38%) occurred. These findings confirm the overall effectiveness of the BiGOA algorithm in reducing cost values compared to the original GOA method.
- *Time:*

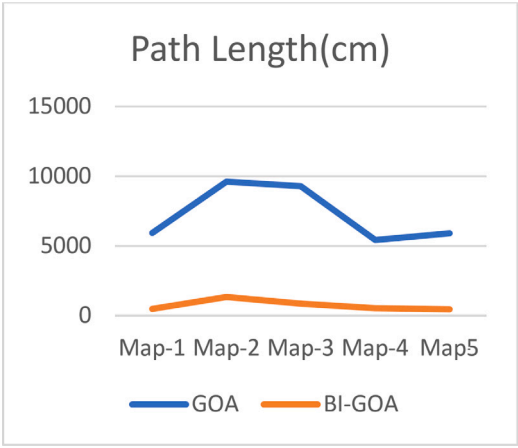

Fig. 10. Path length of (GOA, BiGOA).

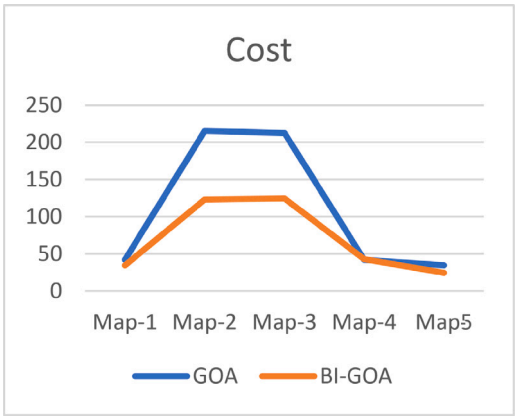

Fig. 11. Cost of (GOA, BiGOA).

Table 3
Experimental results comparison of cost (GOA and BiGOA).

	Map-1	Map-2	Map-3	Map-4	Map-5
GOA	42	215	213	42	34
BiGOA	34	123	124	43	24
EP (%)	19.05	42.79	41.78	−2.38	29.41

Table 4
Experimental results comparison of time (GOA and BiGOA).

	Map-1	Map-2	Map-3	Map-4	Map-5
GOA	0.21	1.22	1.29	0.28	0.20
BiGOA	0.03	0.18	0.15	0.06	0.03
EP (%)	85.71	85.25	88.37	78.57	85.00

Fig. 12 illustrates the impact of map complexity on convergence time. The time required to generate the final path using the original GOA algorithm shows noticeable improvement as complexity increases; however, the proposed BiGOA method demonstrates even faster performance across all scenarios. As presented in Table 4, the BiGOA approach significantly reduced arrival time in all tested maps, achieving enhancement percentages (EP) ranging from 78.57% to 88.37%.

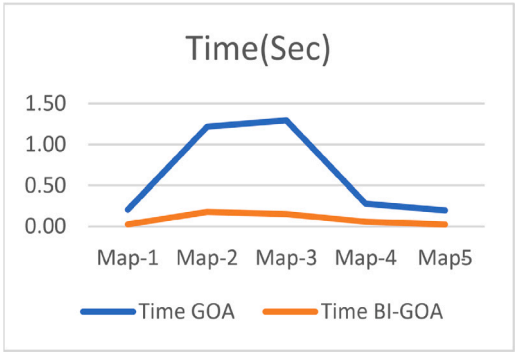

Fig. 12. Time of (GOA, BiGOA).

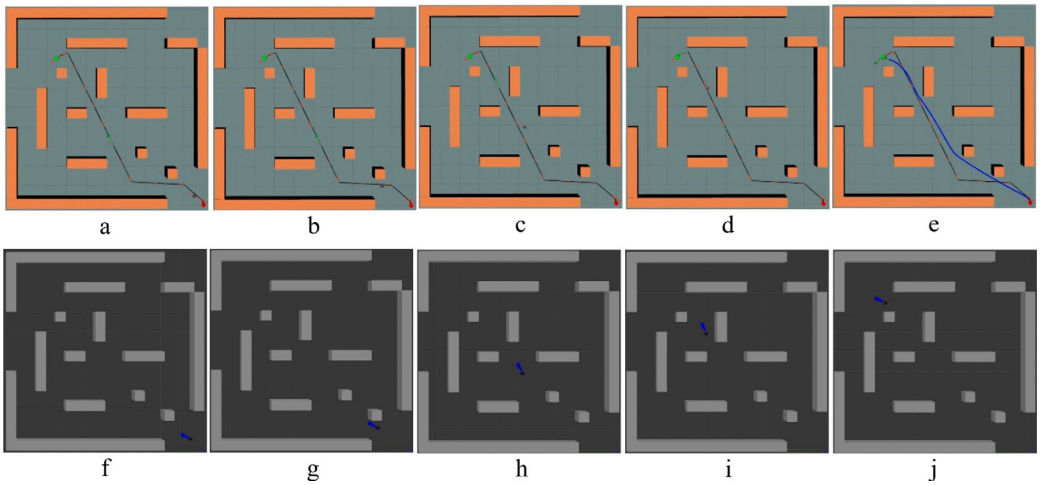

Fig. 13. Screen snapshots of optimal path simulation in a static map. (a–e) RViz view, (f–j) Gazebo view.

3.2. Experiments-2 local path planning

In this experiment, the path obtained from the BiGOA algorithm is shortened and smoothed. The experiment is simulated using the TurtleBot3 (Burger) robot within a 2D environment measuring 1000 × 1000 cm and 3D obstacles. The environment was loaded into Rviz, a visualization tool, as Fig. 13(a–e), and Gazebo, a simulation environment, as Fig. 13(f–j). ROS uses these graphical interfaces to track the path and represent the environment and its obstacles.

- *Optimal Path:*
  Most of the robots within ROS operate using the waypoint and DWA algorithm to track the path. The global path of BiGOA contains hinge points due to its working mechanism. The path should be divided into more points by segmenting the global path. During its navigation, the robot meticulously examines all the segment points with the goal. If there is an available path without intersecting with an obstacle, the robot goes towards it. The positions taken by the robot are archived when the robot reaches the goal, and these positions are planned to show the optimal path, as shown in Fig. 14(a, b).
  In the end, we conducted comparative experiments on the robot's movement on the paths. We knew previously from the randomness of the grasshoppers that the GOA path was unsuitable for the robot's movement because it had sharp turns, causing it to make many stops and leading to delays in the overall time. Therefore, the global path BiGOA and the optimal path were compared when the robot's moving velocity was fixed on both paths. We observe in Table 5 ten times of execution that may prove, every time the superiority of the optimal path reaching the global goal.
- *Real-Time:*
  The path is planned locally when dynamic obstacles appear in front of the robot. The robot detects dynamic obstacles using the Lidar sensor. Three cylindrical obstacles have been added to the path. The DWA algorithm, known for its adaptability, is

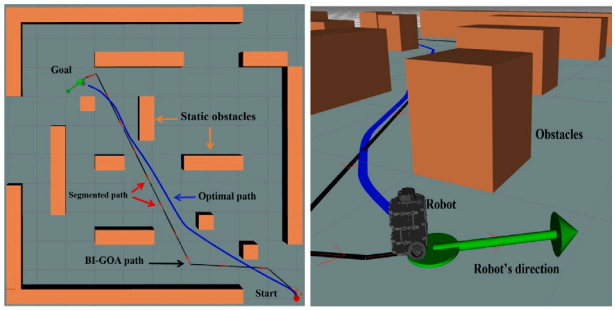

Fig. 14. Screen snapshots of the optimal path (a) Optimal path, (b) TurtleBot3 (Burger).

Table 5
Experimental arrival time comparison based on robot velocity (10 cm/s) between BiGOA and the optimal path.

BiGOA time (s)	Optimal path time (s)
92	64
106	80
152	73
92	67
201	75
89	66
69	72
73	66
83	66
67	65

Table 6
Quantitative comparison of advanced path-planning algorithms.

Algorithm	Type/Year	Path length (cm)	Planning time (s)	Computational cost	Arrival time (s)
Improved A* [22]	Heuristic Search (2025)	1880	0.52	245	152
Hybrid A* + Improved APF [2]	Global–Local Hybrid (2023)	1640	0.63	231	138
Bidirectional Adaptive RRT [17]	Sampling-Based (2023)	1480	0.75	218	125
ACO–GWO Hybrid [6]	Swarm Optimization (2024)	1315	0.47	192	111
Bi-population PSO [18]	Swarm Optimization (2024)	1098	0.27	148	89
Proposed BiGOA-DWA	**Swarm + Local Hybrid (2025)**	**853**	**0.15**	**124**	**69**

called to avoid colliding with the obstacle and find the closest available point. All points on the global path are considered sub-goals when the path has intersected. The robot moves away from the obstacle and towards the available sub-goals. The local path is planned in this experiment, and then all the positions the robot passes are archived to be planned when the robot reaches the global goal, as shown in Fig. 15.

3.3. A comparative study of advanced path-planning algorithms

The final part of the experimental study validates the effectiveness of the proposed BiGOA-DWA algorithm through a comparative analysis with several recently published advanced path-planning approaches. The evaluation encompassed the Improved A* algorithm, the Hybrid A* + Improved APF algorithm, the Bidirectional Adaptive RRT algorithm, the Hybrid ACO–GWO algorithm, and the Bi-population PSO algorithm. All methods were implemented under identical simulation conditions to ensure a fair comparison. The performance metrics considered included path length, planning time, computational cost, and robot arrival time. The results, summarized in Table 6, clearly indicate that the proposed BiGOA-DWA outperforms all benchmark algorithms by achieving the shortest path, lowest computational cost, and minimum arrival time, thereby confirming its superior efficiency and robustness in complex navigation scenarios.

4. Conclusion

This paper introduces a novel method for global and local path planning for mobile robots based on the swarm intelligent algorithm GOA for planning the global path, which is enhanced by the bi-population technique BiGOA. We conducted experiments on five static environments of varying complexity to analyze and study different factors such as path length, cost, and time. Our proposed method achieved superiority in all of the factors mentioned above, in addition to improving the path and making it nearly

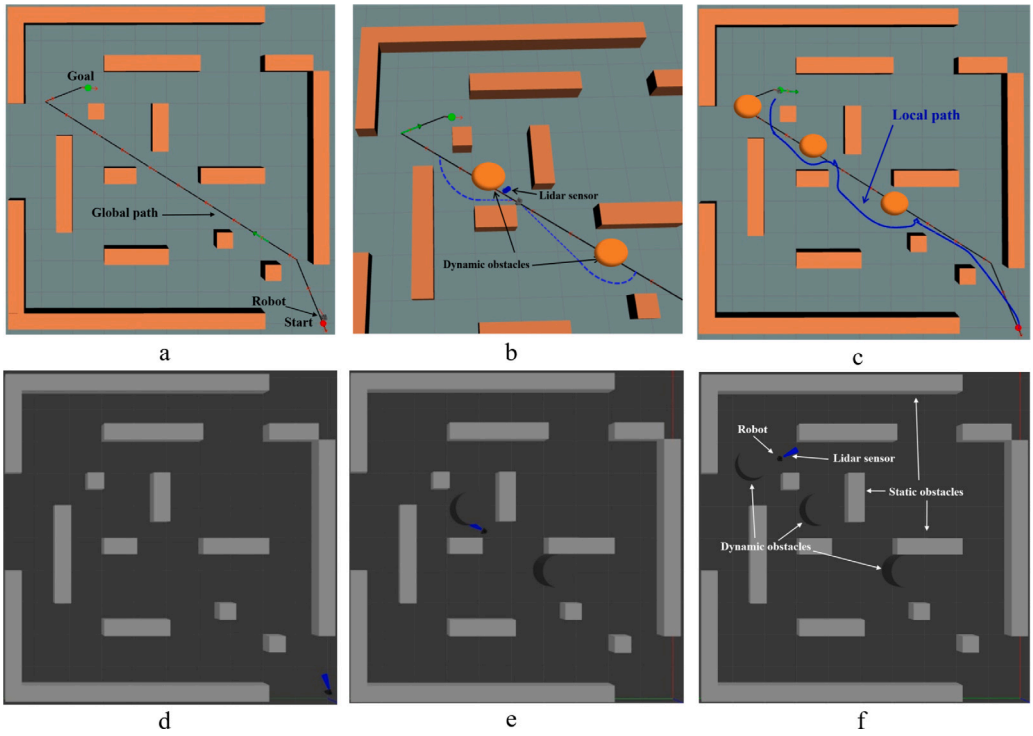

Fig. 15. Screen snapshots of local planning simulation in a dynamic map. (a–c) RViz view, (d–f) Gazebo view.

optimal. We presented path optimization using the waypoints method and DWA algorithm on the global path BiGOA. By using Rviz, Gazebo, and TurtleBot3, we simulated experiments. We presented an improved waypoint algorithm that ensures a smooth, shortened, and continuous path during the robot's movement by creating a state machine controller. This work was compared to the robot's arrival time on the proposed paths (BiGOA, Optimal Path) at a fixed velocity. The robot saved time and effort in reaching the goal, achieving superior performance. At 10 cm/s velocities, an average of 102 and 69 s was obtained for each BiGOA and optimal path, respectively. On the other hand, to confront dynamic obstacles, it is necessary to plan the path locally. The experiments demonstrated the robot's ability to overcome dynamic obstacles by maneuvering among the obstacles and returning to the global target. In future work, we aim to conduct a comparative study involving other modern swarm intelligence algorithms of the same category, such as the Artificial Bee Colony (ABC) and the Butterfly Optimization Algorithm (BOA), and to propose tailored enhancements for each in the context of robot path planning.

Declaration of competing interest

The authors have no relevant financial or non-financial interests to disclose.

We confirm that this work is original and has not been published elsewhere, nor is it currently under consideration for publication elsewhere.

Acknowledgment

The corresponding author wish to thank the support of the Hungarian Research Fund (OTKA K143595). To further validate the proposed methodology and enhance the transparency of the experimental procedures, the authors have produced a detailed video demonstration. This video visually presents the simulation environment, the execution of the path planning algorithms, and the corresponding results discussed in the paper. The demonstration can be accessed at: VideoDemonstration.

Data availability

No data was used for the research described in the article.

References

[1] Qin H, Shao S, Wang T, Yu X, Jiang Y, Cao Z. Review of autonomous path planning algorithms for mobile robots. Drones 2023;7(3):211.
[2] Yao J. Path planning algorithm of indoor mobile robot based on ROS system. In: Proc. IEEE int. conf. image process. comput. appl.. ICIPCA, 2023, p. 523–9.
[3] Ntakolia C, Moustakidis S, Siouras A. Autonomous path planning with obstacle avoidance for smart assistive systems. Expert Syst Appl 2023;213:119049.
[4] Al-Ansarry S, Al-Darraji S, Honi DG. An efficient path planning in uncertainty environments using dynamic grid-based and potential field methods. Iraqi J Electr Electron Eng 2023;19(2).
[5] Xu Y, Li Q, Xu X, Yang J, Chen Y. Research progress of nature-inspired metaheuristic algorithms in mobile robot path planning. Electronics 2023;12(15):3263.
[6] Widians JA, Wardoyo R, Hartati S. A hybrid ant colony and grey wolf optimization algorithm for exploitation-exploration balance. Emerg Sci J 2024;8(4):1642–54. http://dx.doi.org/10.28991/ESJ-2024-08-04-023.
[7] Saremi S, Mirjalili S, Lewis A. Grasshopper optimisation algorithm: Theory and application. Adv Eng Softw 2017;105:30–47.
[8] Patel B, Dubey V, Sharma N. Mobile robot navigational planning using grasshopper algorithm. In: Proc. OITS int. conf. inf. technol.. OCIT, 2023, p. 944–9.
[9] Shankar K, Elhoseny M, Chelvi ED, Lakshmanaprabu SK, Wu W. An efficient optimal key based chaos function for medical image security. IEEE Access 2018;6:77145–54.
[10] Bukhari SMS, Zafar MH, Moosavi SKR, Mansoor M, Sanfilippo F. An integrated stacked convolutional neural network and the levy flight-based grasshopper optimization algorithm for predicting heart disease. Healthc Anal 2025;7:100374.
[11] Rasinac M, Bjelic M, Petrovic A, Ivanovic M, Pajovic S. Optimization of GMA welding parameters using the grasshopper optimization algorithm, faculty of mechanical and civil engineering in Kraljevo. 2023.
[12] Elmi Z, Efe MÖ. Multi-objective grasshopper optimization algorithm for robot path planning in static environments. In: Proc. IEEE int. conf. ind. technol.. ICIT, 2018, p. 244–9.
[13] Elmi Z, Efe MÖ. Online path planning of mobile robot using grasshopper algorithm in a dynamic and unknown environment. J Exp Theor Artif Intell 2021;33(3):467–85.
[14] Shareef A, Al-Darraji S. Grasshopper optimization algorithm based path planning for autonomous mobile robot. Bull Electr Eng Inform 2022;11(6):3551–61.
[15] Zhao M, Lu H, Yang S, Guo Y, Guo F. A fast robot path planning algorithm based on bidirectional associative learning. Comput Ind Eng 2021;155:107173.
[16] Ge Q, et al. Improved bidirectional RRT* path planning method for smart vehicle. Math Probl Eng 2021;2021(1):6669728.
[17] Al-Ansarry S, et al. Bi-directional adaptive probabilistic method with a triangular segmented interpolation for robot path planning in complex dynamic-environments. IEEE Access 2023;11:87747–59.
[18] Tao B, Kim J-H. Mobile robot path planning based on bi-population particle swarm optimization with random perturbation strategy. J King Saud Univ - Comput Inf Sci 2024;36(2):101974.
[19] Gutiérrez R, et al. A waypoint tracking controller for autonomous road vehicles using ROS framework. Sensors 2020;20(14):4062.
[20] Hrabar S. An evaluation of stereo and laser-based range sensing for rotorcraft unmanned aerial vehicle obstacle avoidance. J Field Robotics 2012;29(2):215–39.
[21] Fox D, Burgard W, Thrun S. The dynamic window approach to collision avoidance. IEEE Robot Autom Mag 2002;4(1):23–33.
[22] Li C, Yao L, Mi C. Fusion algorithm based on improved A* and DWA for USV path planning. J Mar Sci Appl 2025;24(1):224–37.
[23] Zhong X, Tian J, Hu H, Peng X. Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment. J Intell Robot Syst 2020;99(1):65–77.
[24] Gong X, Gao Y, Wang F, Zhu D, Zhao W, Wang F, Liu Y. A local path planning algorithm for robots based on improved DWA. Electronics 2024;13(15):2965.
[25] Yan L, Qi L, Feiran K, Guang C, Xinbo C. A study of improved global path planning algorithm for parking robot based on ROS. In: Proc. CAA int. conf. veh. control intell.. CVCI, 2020, p. 607–12.
[26] Liu Z, Zhang N, Chen Y, Chi X. Design of local path planning algorithm for mobile robot. In: Proc. int. conf. electr. mech. comput. eng.. ICEMCE, 2023, p. 1046–9.
[27] Shareef A, Al-Darraji S. Dynamic multi-threaded path planning based on grasshopper optimization algorithm. In: Proc. Iraqi int. conf. commun. inf. technol.. IICCIT, 2022, p. 159–64.
[28] Srinivasan MV, et al. Robot navigation inspired by principles of insect vision. Robot Auton Syst 1999;26(2–3):203–16.
[29] Jiang Y, Pang X, Zhang Z, Jing H, Wei L, Su J. Integrating intelligent sensors for safe UAV distribution: Design and evaluation of ranging system. HighTech Innov J 2024;5(3). http://dx.doi.org/10.28991/HIJ-2024-05-03-04.
[30] Puentes K, Morales L, Pozo-Espin DF, Moya V. Enhancing control systems with neural network-based intelligent controllers. Emerg Sci J 2024;8(4). http://dx.doi.org/10.28991/ESJ-2024-08-04-01.
[31] Katona K, Neamah HA, Korondi P. Obstacle avoidance and path planning methods for autonomous navigation of mobile robot. Sensors 2024;24(11):3573.
[32] Al-Kaff A, et al. Obstacle detection and avoidance system based on monocular camera and size expansion algorithm for UAVs. Sensors 2017;17(5):1061.
[33] Gilliam B, Sahai Q, Chandrasekaran B. Path planning and mapping of an autonomous agricultural robot using robot operating system (ROS) and Gazebo. In: Proc. int. conf. comput. autom. eng.. ICCAE, 2023, p. 528–33.
[34] Gao D, et al. A dynamic obstacle avoidance method for unmanned surface vehicle under the international regulations for preventing collisions at sea. J Mar Sci Eng 2022;10(7):901.
[35] Yu M, Luo Q, Wang H, Lai Y. Electric logistics vehicle path planning based on the fusion of the improved A-star algorithm and dynamic window approach. World Electr Veh J 2023;14(8):213.
[36] Shahab F, et al. Leader-follower formation of a car-like robot using ROS and trajectory tracking. Int Conf Soc Robotics 2024;355–65.
[37] Hasan MW, Abbas NH. An adaptive nonlinear PID design for 6-DOF underwater robotic vehicle. Adv Electr Electron Eng 2022;20(2):193.