Karbala International Journal of Modern Science

Volume 11 | Issue 2 | Article 1

Articulated Robot Path Planning Based on Hybridization of Adaptive Dimensionality Algorithm and Grey Wolf Optimizer in Dynamic Environments

Noor Kadhim Ayoob
Department of Software, College of Information Technology, University of Babylon, Babylon, Iraq,
noor.kadhum@uobabylon.edu.iq

Ali Hadi Hasan
Department of Software, College of Information Technology, University of Babylon, Babylon, Iraq

Recommended Citation
Ayoob, Noor Kadhim and Hasan, Ali Hadi (2025) "Articulated Robot Path Planning Based on Hybridization of Adaptive Dimensionality Algorithm and Grey Wolf Optimizer in Dynamic Environments," *Karbala International Journal of Modern Science*: Vol. 11 : Iss. 2 , Article 1.
Available at: https://doi.org/10.33640/2405-609X.3398

Articulated Robot Path Planning Based on Hybridization of Adaptive Dimensionality Algorithm and Grey Wolf Optimizer in Dynamic Environments

Abstract

A new method was developed to plan a path for a robotic articulated vehicle using the Grey Wolf Optimizer (GWO) and Adaptive Dimensionality (AD). Existing studies in robotics path planning ignore the differences between robots in terms of size and flexibility and allocate a single cell to the robot regardless of the mentioned factors. Since the articulated robotic vehicle is longer than obstacles moving in the environment, this study takes into account vehicle size and flexibility in path planning by adapting the number of cells allocated to the robotic vehicle to contain the vehicle parts while performing different movements. Considering the number of fixed obstacles during the environmental analysis improves safety and reduces dangerous turns. In the moving stage, the sensing area is reduced by AD based on steering angle range, and connectivity. The GWO leaders form the local path followed by the vehicle. Simulation results showed that the proposed method finds an optimal, collision-free, and safer path. Compared to most related studies, the average path cost and the number of iterations increased by (47.88%) and (59.15%) compared to the GSO-AD, because increasing safety in the proposed method guided the vehicle through a higher-cost path and imposed more iterations. These metrics decreased by (12.83%) and (50.14%) respectively compared to the Max-Min Ant because the latter uses complex calculations to lead the robot through large free spaces. The average time of the proposed method decreased by (17.57%) and (72.94%) compared to these methods which indicates the efficiency of the proposed method.

Keywords

Robot; Path Planning; Grey wolf Optimizer; Adaptive Dimensionality; Dynamic Environment; Articulated Vehicle

Creative Commons License

RESEARCH PAPER

Articulated Robot Path Planning Based on Hybridization of Adaptive Dimensionality Algorithm and Grey Wolf Optimizer in Dynamic Environments

Noor K. Ayoob*, Ali H. Hasan

Department of Software, College of Information Technology, University of Babylon, Babylon, Iraq

Abstract

A new method was developed to plan a path for a robotic articulated vehicle using the Grey Wolf Optimizer (GWO) and Adaptive Dimensionality (AD). Existing studies in robotics path planning ignore the differences between robots in terms of size and flexibility and allocate a single cell to the robot regardless of the mentioned factors. Since the articulated robotic vehicle is longer than obstacles moving in the environment, this study takes into account vehicle size and flexibility in path planning by adapting the number of cells allocated to the robotic vehicle to contain the vehicle parts while performing different movements. Considering the number of fixed obstacles during the environmental analysis improves safety and reduces dangerous turns. In the moving stage, the sensing area is reduced by AD based on steering angle range, and connectivity. The GWO leaders form the local path followed by the vehicle. Simulation results showed that the proposed method finds an optimal, collision-free, and safer path. Compared to most related studies, the average path cost and the number of iterations increased by (47.88 %) and (59.15 %) compared to the GSO-AD, because increasing safety in the proposed method guided the vehicle through a higher-cost path and imposed more iterations. These metrics decreased by (12.83 %) and (50.14 %) respectively compared to the Max-Min Ant because the latter uses complex calculations to lead the robot through large free spaces. The average time of the proposed method decreased by (17.57 %) and (72.94 %) compared to these methods which indicates the efficiency of the proposed method.

Keywords: Robot, Path planning, Grey Wolf Optimizer, Adaptive dimensionality, Dynamic environment, Articulated vehicle

1. Introduction

The primary task of path planning for a mobile robot is to find a path through which the robot can navigate from a starting point to a target point safely, smoothly, and with minimal energy consumption [1]. There are many techniques for route planning, including [2] basic methods (cell decomposition, road map, etc), heuristic (A*, D*, etc.), and smart techniques (neural nets, swarms, etc), each of which has its pros and cons. The choice of planning approach depends on the robot's aptitudes, the nature of the environment, and the requirements of the application [3].

Swarm intelligence is a class of nature-inspired algorithms that imitate the social organization of species in nature through the interactions of individuals with each other and with the environment [4]. These algorithms propose a set of solutions that are organized into communities called swarms. The solutions evolve during the execution of the algorithm and thus the swarm shows high scalability, flexibility, and adaptability to environmental changes. All swarm algorithms start with an initial swarm, define a function for solution evaluation, update solutions (moving individuals), terminate at a certain condition, and return the best-discovered solution [5]. Swarm intelligence is an effective approach to optimize solutions to many well-known problems in computer science, for example, traveling salesman [6], data clustering and analysis [7−9], feature selection [10], and path planning [11].

Received 1 November 2024; revised 7 February 2025; accepted 9 February 2025.
Available online 11 March 2025

* Corresponding author.
E-mail address: noor.kadhum@uobabylon.edu.iq (N.K. Ayoob).

https://doi.org/10.33640/2405-609X.3398

The adaptive dimensionality approach aims to speed up the planning process in a dynamic environment by planning in full dimensions if a risk of clashing may occur and planning in low dimensions elsewhere [12]. The advantage of AD is decreasing time and complexity by reducing the number of neighbors to be processed [13].

In 2020, Guanghui Xu et al. [14] showed that the traditional version of the firefly swarm did not succeed in overcoming the obstacles. The algorithm was improved by formulating proposed equations to update the step size and absorption coefficient. These improvements made the algorithm more efficient in finding a shorter path with fewer iterations. Jianzhang and Zhihao Zhang [15] used a hybridization of the simulated annealing and PSO to reduce falling into the local optima by adding the replacing probability for each particle. They proposed a new equation for updating the speed. The mutation concept was borrowed from the genetic algorithm to enhance the planner. The results showed that the cost function is minimized significantly compared to the classical PSO with the same complexities. Qasim R. et al. [12] presented the AD principle with the Glowworm algorithm for pathfinding in dynamic workspaces. The cells of three levels are filtered by AD which selects cells with luciferin less than that of the current cell. By comparing the results with other studies, it was found that the proposed planner took less time and fewer cycles. Xuezhen Cheng et al. [16] proposed a method that combines PSO and GWO for efficient path planning. The PSO algorithm finds the best path and GWO ranks particles to choose the three fittest particles to lead others. The method relied on chaos to generate the initial population to get rid of the local end by replacing a random particle. According to the results, the method was able to find an optimal path in a shorter time compared to the traditional methods. Fish swarm algorithms with Bezier curves [17] were used to obtain a smooth path. The fish algorithm has been improved by examining 16 and 24 neighbors. A dynamic feedback horizon and adaptive step size were suggested to overcome the poor performance in the advanced stages. Dijkstra algorithm was also used to determine the step size. It was noted that the method produced smooth, continuous paths with an improvement in the performance of the fish flock algorithm. Oussama H. et al. [18] proposed hunting a moving target by multi-robots using GWO and potential field (PF). Each robot represents a wolf moving in the potential field. PF was used to plan the path of the wolf (robot). The method found the target with a safe and optimal path. Researchers in [19] used the ant algorithm for multi-robot planning where each robot has a priority, and the paths are planned in descending order of priority, then a collision-free path is found for each robot. They represented the environment in the form of a 2.5 map consisting of two layers to store ground and obstacle data respectively. The results showed that the proposed method has remarkable improvements in terms of energy saving. Zhen Yang et al. [20] proposed using alternating jump point search A* algorithm with two types of distance (Euclidean and Chebyshev) in global planning and DWA with an improved evaluation function method for local planning. The results showed a higher possibility of decreasing the number of neighbors and the execution time. In [21], an improved min−max ant algorithm is used to guide the robot away from obstacles by computing a new parameter called clean in the step of map analysis and using it to modify the original probability formula. The method was able to detect tight tunnels during analysis. As a result, the robot became safer by directing it towards wider areas. in [22], to obtain near-optimal paths in 3D without collisions, an improved RRT using three versions of GWO to obtain the benefits of both methods while getting rid of the drawbacks of each. The idea of these versions is that updating the wolf's position does not depend on the leaders only. In the extended version, the nth wolf updates its position based on n-3 preceding wolves, while the wolf depends on all n-1 preceding wolves to receive the knowledge in the incremental approach. The method eliminates the need for prior knowledge maps by making efficient use of the UAV's memory and does not suggest trajectories with a limited number of locations but rather constructs them dynamically. The results showed that the hybrid based on the extended version of the gray wolf performed better in large environments, while the hybrid based on the enhanced version succeeded in small to medium-sized environments. The standard gray wolf achieved a balanced performance between the two methods. In [23], a new method for path planning called Dhouib-Matrix-SPP was proposed. It depends on representing the environment as a grid that is converted into a graph where each cell represents a node and the edges between the nodes represent the possible movements between the neighboring cells, assuming that the robot can move in eight directions: up, down, right, left, and four diagonal directions. The graph can be easily represented as a matrix where the elements represent the distances between the nodes. At each step, the node closest to the goal is chosen while avoiding obstacle nodes. The method was tested on environments of various sizes and distributions of obstacles. The time complexity of the algorithm is O (9 × n), which makes it faster than algorithms such as Dijkstra or A*. By comparing the algorithm with 12 other methods, the study concluded that the proposed method was fast, flexible, and accurate, but the experiment was limited to applying the method in static environments only. A summary of these studies, including the main contributions and limitations, is shown in Table 1.

This paper aims to develop a method for finding a path for an articulated robotic vehicle taking into account the space required to implement the robotic vehicle movement i.e. the planning algorithm does not lead the robot to place.

That is too small for the robot body to pass through. The main contributions of this study are:

1) Propose a hybridization of the AD algorithm with the multi-objective function GWO to plan a safer path for the articulated robotic vehicle. AD algorithm reduces sensing space and determines the candidate cells in proportion to the size of the robot. The multi-objective GWO decides the best three candidates considering the number of dynamic obstacles around each candidate as a penalty to protect the two parts from colliding with an obstacle while executing complex movements. Two reasons can serve as motivation for using GWO in the proposed planning method. First, the simple mathematical model and the small number of adaptable parameters make this algorithm an effective choice for a complex system such as robot path planning. GWO planner reduces the computations and time consumption compared to other swarms such as ABC or ACO. Second, the inspiring feature of being a class-based community algorithm. GWO introduces three leaders (solutions) which are perfectly consistent with the proposed algorithm. The sensing area consists of three layers, and GWO has three leaders, the proposed idea is to assign a leader in each level to guide the robot from its current position to the first level, then the second level, and then the third level.
2) Considering the size of the robot as an influential factor in path planning, in contrast to previous studies that assumed that the robot is an object that occupies only one cell to simplify the planning process, ignoring the fact that robots differ in size and flexibility, and therefore the number of cells occupied by the robot varies according to the size of the robot and the flexibility of its body. Long vehicles cannot be equal to small ones, and the presence or absence of a joint affects the planning algorithm decisions during movement. In this study, we propose to use the multi-cell principle so that the number of cells occupied by a robotic vehicle varies based on the alignment of the tractor and trailer. Two cells are needed for the robot if the tractor and trailer are aligned horizontally or vertically, which is the least number of cells occupied by the articulated robotic vehicle, four cells if they are placed diagonally, and three cells in the turning due to the joint that facilitates rotation to the maximum of 45∘ to the right and left.
3) Improving the analysis process by taking into account the number of fixed obstacles around the cell to evaluate the cell in terms of cost and safety, the fewer the obstacles, the lower the cost. In this way, the analyzer ensures finding a path away from fixed obstacles.

The paper is organized as follows: In Section 2, the theoretical background of the grey wolf algorithm is viewed. The proposed method is described in detail in Section 3. The results and comparison are documented in Section 4. Finally, conclusions and future works are stated in Section 5.

2. Basics of Grey Wolf Optimizer

GWO was designed to mimic the hunting style of a grey wolf population [24]. The pack is subject to the directions of three types of leaders: the supreme leader (alpha), the assistant leader (beta), and the protector leaders (delta), the remaining wolves (omega) are ordinary individuals who maintain diversity, and accordingly, the wolf community is organized into a hierarchy of four classes as shown in Fig. 1. The alpha leader has a better perception of the target (prey), so it represents the best solution followed by the beta and delta leaders, who represent the second and third-best solutions, respectively. The omega individuals update their positions relative to the leaders' positions [25]. The GWO algorithm begins by creating a random swarm and evaluating each wolf (solution) to choose the best three wolves: alpha, beta, and delta [26]. The algorithm develops solutions through a set of cycles where the wolf's position is updated according to the leaders' positions using the equations [27]:

$$X1 = X\_delta - A[1] \times Abs(C[1] \times X\_delta - X) \tag{1}$$

$$X2 = X\_beta - A[2] \times Abs(C[2] \times X\_beta - X) \tag{2}$$

$$X3 = X\_alpha - A[3] \times Abs(C[3] \times X\_alpha - X) \tag{3}$$

Table 1. *The summary of the related studies.*

Year	Ref No.	Methods	Max no. of robots	Map size	Main contributions	Limitations
2020	[14]	Firefly	1	20*20	Adaptive strategies to update key parameters: α and γ.	Tested on static environments only and representing the robot as a point without considering the effect of size.
2021	[15]	Simulated annealing and PSO	1	–	Solving premature convergence and falling into local solutions, by introducing simulated annealing probabilities and mutation.	Tested on static environments with polygon obstacles only and representing the robot as a moving point without considering the effect of size.
	[12]	AD and Glowworm	1	10*10 50*50	Improving the planning in high-dimensional spaces by reducing the number of cells considered.	The robot is represented as a point without taking into account the effect of robot size on path planning.
	[16]	PSO and Grey wolf	1	20*20 30*30	PSO improving global exploration, and GWO improving local exploitation.	Tested on static environments only. The robot is represented as a point without taking into account the effect of robot size on path planning.
	[22]	RRT and Grey wolf	1	50*50*50 100*100*100 150*150*150	Optimize RRT paths using the gray wolf to find obstacle-free paths by specifying the length and direction of each move.	Treating robot (UAV) as a moving point in the 3D space without considering the effect of occupied space.
2022	[17]	Fish swarm, Dijkstra	1	10*10 20*20	Taking 24 neighbors into account to expand movement options and improve planning accuracy by using adaptive stepping.	Tested on static environments only. The robot is represented as a point without taking into account the effect of robot size on path planning.
	[18]	Wolf swarm and potential field	4	20*20	Improving collaboration of robots for dynamic target detection using potential fields for planning and avoiding static and moving obstacles.	The robot is represented as a point without taking into account the effect of robot size on path planning.
	[19]	Priority-free ant colony	10	10*10 20*20	Use the scheduling concept to start planning with the highest priority robots, taking into account the friction factors to get paths with the least energy consumption.	The robot is represented as a point without taking into account the effect of robot size on path planning. The behavior of the method in the presence of dynamic obstacles has not been studied.
	[20]	BAJPSA* and DWA	3	30*30 100*100	Generating a global path through a two-way A*. Local planning is done by improved DWA with a new evaluation function.	The robot is represented as a point in a cell without taking into account the effect of robot size on path planning.
2023	[21]	Max-min	1	10*10 50*50	Increase safety by guiding the robot towards the wide open spaces.	The robot is represented as a point without taking into account the effect of robot size on path planning.
	[23]	Dhouib-matrix-SPP	1	20*20 30*30 40*40	Find the optimal path in static grid maps faster and with less complexity than other methods.	The robot is represented as a point without taking into account the effect of robot size on path planning. The behavior of the method in the presence of dynamic obstacles has not been studied.

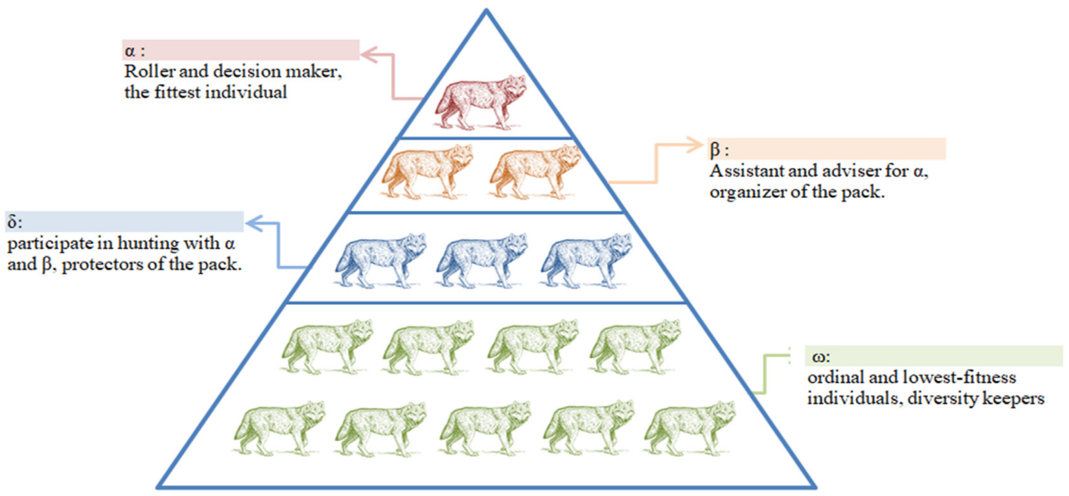

α :
Roller and decision maker, the fittest individual
β :
Assistant and adviser for α, organizer of the pack.
δ:
participate in hunting with α and β, protectors of the pack.
ω:
ordinal and lowest-fitness individuals, diversity keepers

Fig. 1. The hierarchy of grey wolves' population according to fitness and responsibilities.

$$X\_new = (X1 + X2 + X3)/3 \quad (4)$$

X_delta, X_beta, and X_alpha are the positions of leaders delta, beta, and alpha respectively. X_ new is the updated position of the wolf. X1, X2, and X3 are the positions of the wolf according to the leaders. Exploration and exploitation are controlled by vectors A and C which are calculated using the following equations [28]:

$$a = 2 - 2 \times (it \ / \ M) \quad (5)$$

$$A = a \times 2 \times rand_1 - a \quad (6)$$

$$C = 2 \times rand_2 \quad (7)$$

Where (it) is the number of current iterations, (M) represents the maximum number of iterations. $Rand_1$ and $rand_2$ are randomly generated values in the range (0,1). After a predetermined number of iterations or meeting stopping criteria, GWO is terminated by presenting the position of alpha leader as the best solution discovered.

Over time, researchers have developed the algorithm by improving the strategy for updating wolf locations [29], improving the generation of the initial population, calculating control coefficients, and even integrating with other swarm algorithms [30]. GWO is an effective solver for various problems such as robot path planning [33], finding an optimal path in a wireless network [31], selecting features [32], and other applications.

3. The proposed method

In this study, a method is proposed for planning the path of an articulated robotic vehicle based on the stages shown in Fig. 2.

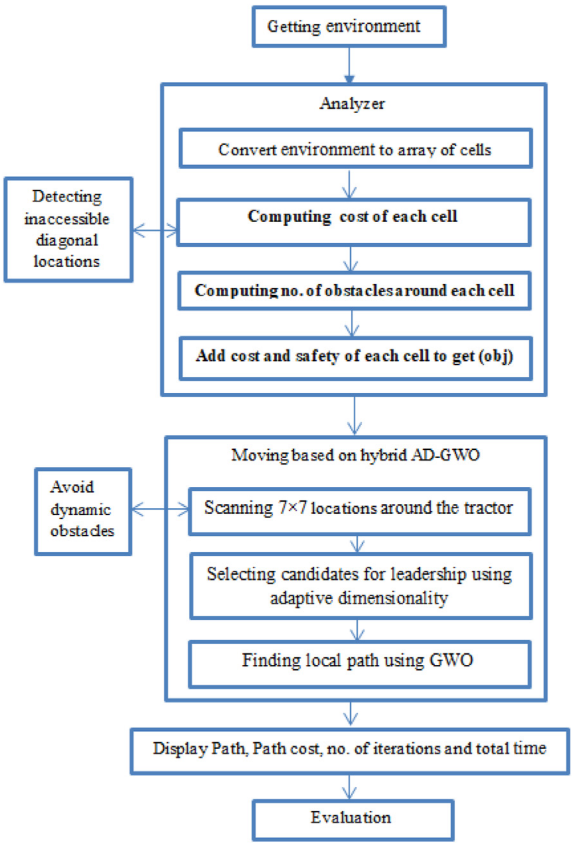

Getting environment
Analyzer
Convert environment to array of cells
Computing cost of each cell
Computing no. of obstacles around each cell
Add cost and safety of each cell to get (obj)
Detecting inaccessible diagonal locations
Moving based on hybrid AD-GWO
Scanning 7×7 locations around the tractor
Selecting candidates for leadership using adaptive dimensionality
Finding local path using GWO
Avoid dynamic obstacles
Display Path, Path cost, no. of iterations and total time
Evaluation

Fig. 2. The stages of the proposed method.

The environment is imported and analyzed then the moving stage takes advantage of the analysis information to move the robot using hybrid AD- GWO and finally, the method is evaluated in terms of path cost, total time, and number of iterations.

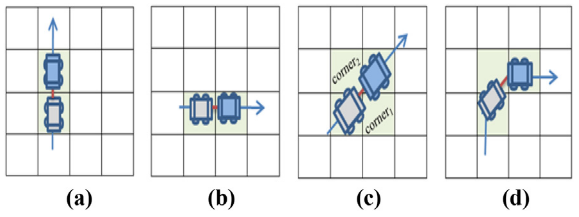
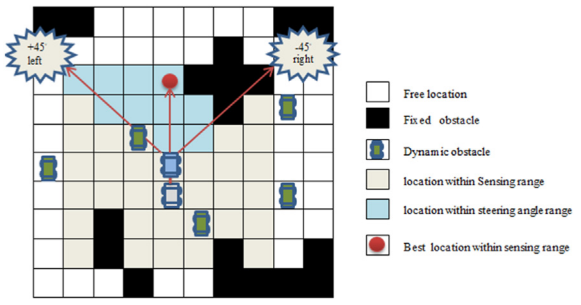


Fig. 3. Space requirements according to possible movements. (a) Vertical alignment (b) horizontal alignment (c) diagonal alignment (d) turning.



+45 left
-45 right
Free location
Fixed obstacle
Dynamic obstacle
location within Sensing range
location within steering angle range
Best location within sensing range

Fig. 4. Sensing range and the cells selected according to the steering angle.

3.1. Robot characteristics

The articulated robotic vehicle consists of two parts: a tractor and a trailer with a joint connecting them. The robot's capabilities and characteristics are presented in Table 2.

Fig. 3 shows the multi-cell representation proposed in this paper, which depends on the type of movement. The articulated vehicle takes at least two positions when both the tractor and the trailer are horizontal or vertical. When the tractor is diagonal to the trailer, i.e. straight diagonal movement, this requires reserving 4 cells for the robot. We need three cells to implement the turn, as one free corner is enough to implement this turn due to the presence of the joint. This explains the high possibility of this type of vehicle to turn around an obstacle if necessary.

As shown in Fig. 4, the angle between the tractor and the best cell within the sensing range is calculated. All cells whose angle to the tractor falls within the range (angle-45, angle+45) are chosen and reduced based on back pointers and connectivity as seen later in Section 3.3.

Table 2. Robot characteristics.

Characteristic	Value	Explanation
Tractor	–	The leader part
Trailer	–	The follower part
No. of cells occupied by the robot based on the movement type	2	Horizontal/vertical alignment as in Figs 3-a and 2-b
	3	Turning as in Fig. 3-d
	4	Diagonal alignment as in Fig. 3-c.
Range of sensor	Three layers	Scanning 7 × 7 of cells centered on the tractor as in Fig. 4.
Steering angle range	Angle ± 45°	*"Angle"*: angle from tractor to best cell in sensing range −45: right, +45: left as in Fig. 4.
Articulation angle range	± 45°	The angle between the two parts.

3.2. The analysis stage

The purpose of this stage is to gather data about each cell using D* style. The analyzer discovers the environment by expanding cells from the goal back to the starting cell. A queue is used to store the cells resulting from the expansion, arranged in ascending order according to their cost. Each cell pulled from the queue is expanded to discover new cells that can be accessed through that cell. Besides the coordinates (X, Y), the information recorded for each cell is shown in Table 3.

In this paper, the performance of the analyzer is improved by the following:

1) Detection of non-executable movement, by closing inaccessible diagonal cells because both corners are blocked.
2) Keep the robot away from a fixed obstacle to increase safety degree by counting the number of obstacles around each cell.
3) Reducing dangerous turns i.e. turns around an obstacle as a result of staying away from obstacles (explained in 2).

Table 3. Cell information gathered by the analyzer.

Info	Meaning	Value/Calculation
Status	What this cell represents.	Free, fixed obstacle, tractor, trailer, goal
Back	The reference cell that, upon expansion, led to the cell.	Cell. Back = Current cell (cell to be expanded).
Safety	The number of fixed obstacles around the cell.	Min = 0 (Safe from all directions (Max = 8 (inaccessible/blocked from all directions)
Cost	Cost of the cell.	Current. Cost + Euclidean distance (cell, Current)
OBJ	Cell evaluation from the analyzer's point of view.	Current. Cost + Current. Safety.
Tag	Cell state during analysis	*"New"*: not expanded yet, *"open"*: waiting for expansion in the queue, *"closed"*: already expanded.

The steps of the analyzer are explained in Algorithm 1. The information of all cells is stored in the array "*Cells*". Bold lines represent the steps that led to the improvements mentioned above.

Algorithm 1: The Analyzer

```
Input :
  Start and Goal coordinates.
Output:
  Cells: array to store the information of each cell (Table 3).
Local Variables definition :
  Done: Boolean
  Current: cell  // cell to be analyzed currently.
  Queue: queue of cells. //  keeping the result of expansion
  neighbor: cell. // an adjacent cell to the current.

1. Begin
2.   Done ← true.
3.   Goal.Cost ←  0.
4.   insert(Queue, Goal).
5.   do
6.     begin
7.       Current← Queue. De-queue()
8.       current. Tag ← "CLOSE".
9.       If (Current. Status == "Fixed_obstacle")
10.          Begin
11.             Done ← false.
12.             Go to step 26.
13.         End
14.       current. Safety ← 0.
15.       Find eight neighbors of the current.
16.       For each neighbor do
17.         Begin
            //Algorithm2
18.          Call Cost Computing (neighbor, current, Queue).
19.          If (neighbor is diagonal and both corners are "Fixed_obstacle")
20.          Then neighbor. Status ← " Fixed_obstacle ".
21.          If (neighbor is "OBSTACLE") Then
22.             current. Safety ← current. Safety + 1.
23.       End
24.     current. Obj ← current. Cost + current. Safety.
25.     Sort (Queue).
26. While (Start not discovered or Done is true).
27. If (Done ==false) Then
28.    Write  ("No path, the start not discovered");
29. Else
30.     Write  ("Done, there is a path between Start and Goal");
31. End
32. End
```

Algorithm 1 begins the analysis from the goal to the start. For each cell analyzed, the eight adjacent neighbors are found. The cost of each neighboring cell is calculated according to the steps of Algorithm 2. According to Equation 8, the cost (accumulated distance) is the summation of the current cost and the Euclidean distance between the cell and its current. The Euclidean distance to the horizontal/vertical neighbor is (1), and the distance of the diagonal neighbor is (1.4). The cost of the obstacle is 10,000 (a very high cost).

$$\text{Cell.cost} = \begin{cases} \text{Current. Cost} + 1.4 & \text{For diagonal cell} \\ \text{Current. Cost} + 1 & \text{For vert./hor. cell} \\ 10000 & \text{For obstacle cell} \end{cases} \tag{8}$$

Algorithm 2: Cost Computing

```
Input :
Cell: adjacent neighbor to Back
Current: reference cell (cell to be analyzed currently).
Queue: keeping the result of expansion.
Output: values of Cost, and Back of Cell

1. Begin
2.   If Cell. Tag = "New" then
3.     If  Cell. Status = "Fixed_obstacle"
4.       Cell. Cost ←  10,000.
5.     Else
6.       If (Cell is diagonal neighbor to current) Then
           // Cost of free diagonal neighbor cell.
7.         Cell. Cost ←  Current. Cost + 1.4
8.       Else  // Cost of horizontal/vertical neighbor cell.
9.           Cell. Cost ←  Current. Cost + 1
10.        End
11.      Cell. Back ←  Current
12.      Insert(Queue, Cell).
13.   End
14.  End
15. End
```

As shown in line 19 in Algorithm 1, diagonal neighbors with obstacle corners are detected and their status is changed to a permanent obstacle of cost equal to 10,000 because they are unreachable. For each obstacle neighbor encountered, the safety value is increased by one to indicate that the safety level of the current node has decreased (lines 21 and 22). After working through all the neighbors, the analyzer evaluation of the current cell is calculated by summing the cost and the safety value and storing the result in current. Obj (line 24) which is very important because the GWO equations are implemented using these values later in the movement phase.

3.3. Moving stage

This stage consists of a set of iterations that end with reaching the goal. The main steps of each iteration are:

1) Sensing: checks all cells in the three layers surrounding the tractor to get only free cells and avoid the cells containing dynamic and fixed obstacles.
2) Adaptive dimensionality: this step aims to find the best cells to be candidates for GWO leaders based on steering angle range and connectivity. AD is started by determining the best cell reached by the current tractor through the back pointers and calculating the angle between them using Equation 9. The angle between all scanned cells and the tractor is also calculated using Equation 9:

$$\text{angle} = \tan^{-1}(y - \text{Tractor}_x, x - \text{Tractor}_y) \times (180 / \text{PI}) \qquad (9)$$

Where (Tractor_x, Tractor_y) is the coordinate of the tractor, (x, y) is the coordinate of the best cell or a cell within the sensed area). The valid candidates for leaders are the cells that fall within the range (angle to best ± 45°). The candidates are further reduced according to the connectivity to cells in the previous layer. The resulting candidates are organized according to their layer. If the cell is connected to all cells in the previous layer, it is considered a valid candidate. AD steps are explained in detail in Algorithm 3 and Fig. 5.

3) Finding local path by GWO: this step aims to get the local path from the tractor to the outer layer. The leader of each layer is determined by choosing the cells with the least (Obj) values in each layer which represent the best cell in terms of safety and cost from the analyzer's point of view. The control parameters A and C are calculated using Equations 6 and 7. The GWO computing is applied to each candidate cell using Equations 1–4 to get the first part of cell fitness. Table 4 shows the values that are substituted into these equations and how to calculate them where the "parameter" column refers to the coefficients mentioned in these equations, while the "value" column describes the values that are substituted into the coefficients. The final fitness value of each candidate (Candidate[i]. fitness) is calculated by adding the number of dynamic obstacles as a penalty. The algorithm updates the leadership by selecting minimum-fitness cells in each layer to lead the robot to delta, beta, and alpha respectively. The total path cost is modified by adding the cost of leaders. Fig. 6 explains this step.
4) Displaying local path: the movement in each iteration is displayed on environments.

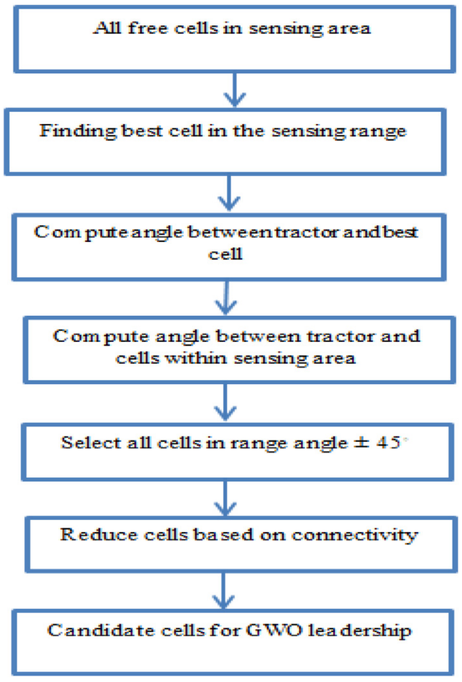

All free cells in sensing area
Finding best cell in the sensing range
Compute angle between tractor and best cell
Compute angle between tractor and cells within sensing area
Select all cells in range angle ± 45°
Reduce cells based on connectivity
Candidate cells for GWO leadership

Fig. 5. Steps of adaptive dimensionality.

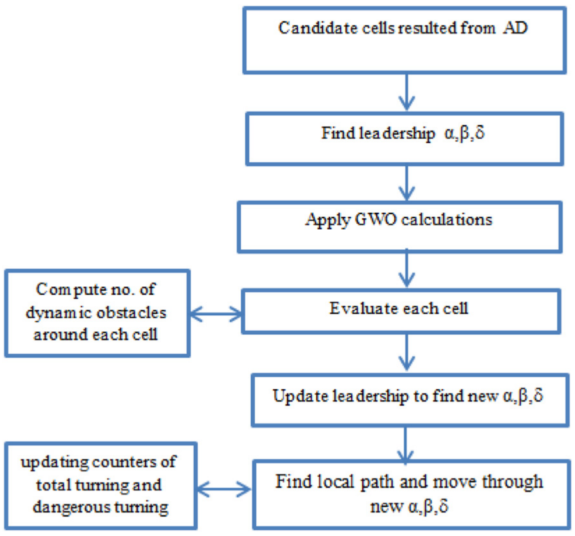

Candidate cells resulted from AD
Find leadership α,β,δ
Apply GWO calculations
Compute no. of dynamic obstacles around each cell
Evaluate each cell
Update leadership to find new α,β,δ
updating counters of total turning and dangerous turning
Find local path and move through new α,β,δ

Fig. 6. Finding local path using GWO.

Table 4. Values substituted into Equations 1–4.

Parameter	Value/calculation	Role
X_delta	Leader [3].obj	Leader of level 1
X_beta	Leader [2].obj	Leader of level 2
X_alpha	Leader [1].obj	Leader of level 3
A, C	Calculated by Equations 6 and 7	Control parameters
X	Candidate[i]. Obj	Candidate (wolf)
Note	Obj is calculated in the analysis stage by adding the cost and safety of the cell as explained in Table 3.	

Finally, after reaching the goal, the path, path cost, total time consumed, and no. of iterations are displayed to evaluate the performance. Details of these steps are documented in Algorithm 4. Fig. 7 shows the hybridization between AD and GWO at one iteration.

Algorithm 3: AD

```
Input:
 NCells: data of all free neighboring cells in the sensing area.
 cx, cy: current coordinates of tractor
Output: Candidates: list of valid candidates for GWO leaders.
Local Variables definition :
si,sj: int
x,y: [] int
index: int
Angle: double                // angle between best cell and tractor
 Theta: []double             // angles between neighboring cells and tractor
From, to double              // determine limits of allowable angle range
Taken: list of cell          // cells within the angle range
Previous: list of cells      // accepted cells in the previous layer
Temp: list of cell           // saving candidate cells in the current layer

1. Begin
2.   si ← cx, sj ← cy // start from current
3.   index← 0;
 //Get the best-connected cell in sensing range using the back pointer
4. for i ← 1 to 3
5.    begin
6.      if (NCells [si, sj]. Back != null)
7.          begin
            // get coordinates of back cell and save them in x[] and y[]
8.            x[i] ← NCells [si, sj].Back. x;
9.            y[i] ← NCells [si, sj].Back. y;
10.           si ← x[i];
11.           sj ← y[i];
12.           index ← index + 1
13.           if (si = goal. x and sj = goal. y) Then break;
14.         end
15.         else break;
16.   End
 //Start reduction based on allowable angle range
// Calculate and normalize the angle between the best cell and Tractor
17. angle ← $\tan^{-1}$(y[index] - cy, x[index] - cx) * (180 / PI);
18. angle ← (angle + 360) % 360;
// Calculate and normalize steering angle range to accept candidates
19. from ← angle - 45;
20. to ← angle + 45;
21. from ← (from + 360) % 360;
22. to ← (to + 360) % 360;
23. for each Cell in NCells
// Calculate and normalize the angle between the Tractor and the cell
24. begin
25.      theta[i] ← $\tan^{-1}$ (Cell.y- cy, Cell.x - cx) * (180 / PI);
26.       theta[i] ← (theta[i] + 360) % 360;
27.       if (theta[i] in the range [from, to])Then
28.           add Cell to the Taken list
29.    end
// strat reduction based on cell connectivity
30. Previous ← Tractor
31. For i ← 1 to 3
32.   temp ← null
33.   for each Taken [i]
34.      if Taken [i] connected to the Previous then
35.        add Taken [i] to the Candidates list and temp list.
36.    end
37.   Previous ← temp
38. End
39. Return Candidates.
40. End
```

Algorithm 4 Moving

```
Input:
Cells: array of cell information (as in Table 3) produced by the analyzer.
Tractor, trailer, and Goal.
Output: path, pathCost, time, and iterations.
Local Variables definition :
Iteration: int            // counter of iteration
Max_ iterations: int      // max limit of iterations
N_DOB: int                // number of the dynamic obstacle around the cell
Dturns: int               // number of dangerous turning
Reached: Boolean          // flag to check reaching the goal
NCells: list of cells     //data of free neighboring cells.
Leaders: list of cells    //all information of leaders α,β, and δ
A, C:[] double, a: double // GWO parameters
Last_move: string         // type of the last move ( diagonal or straight)
Current_move: string      // type of current move ( diagonal or straight)

1. Begin
2.   Max_ iterations ← 2000, iteration ← 0, Dturns ← 0
3.   Reached ← false,
//Place the dynamic obstacles in random positions.
4.   InitializeDobstacles()
5.   While (iteration ≠ Max_ iterations and Reached= false)
6.     begin
7.        UpdateDobPositions ()
8.        If (Tractor. x ≠Trailer. x and Tractor. y ≠Trailer. y) then
9.           Last_move=" diagonal"
10.        Else
11.           Last_move=" straight"
12.      End
// Sensing
13.   Find all free neighbors in the three layers (NCells).
// Adaptive dimensionality
14.   Call AD (NCells, Tractor. x, Tractor. y)
// Discovering local path by GWO
15.   Determine the leader of each layer based on the Obj value.
// Evaluating candidates using GWO
16.   Compute a, A, and C using Equations 5, 6, and 7 respectively.
17.   for each Candidate [i] do
18.     begin
19.       sum ← 0
20.       for each leader [j]
21.          begin
//Apply Equations 1, 2, 3, and 4.
22.     X[j]= leader [j]. Obj - A[j] *Abs(C[j] * leader [j]. Obj –
        Candidate[i]. Obj);
23.     sum ← sum + X[j]
24.   End
25.   Xnew= sum/3;
26.   Compute no. of dynamic obstacles (N_DOB) around the Tractor.
27.   Candidate[i]. fitness = Xnew + N_DOB
28. end
29. Determine the new Leader of each layer based on the fitness and
connectivity to the previous leader.
30.   For each Leader [i]
31.    Begin
32.      OldTractor← Tractor
33.      Tractor ← Leader [i]
34.      Trailer ← OldTractor
35.      If (Tractor. x ≠Trailer. x and Tractor. y ≠Trailer. y) then
36.         Current_move= "diagonal"
//Counting total turning
37.      if (last_move != current_move ) then turning ← turning + 1
// Counting dangerous turning Dturn
38.      if (Corner1. Status= Fixed_obstacle) or
             (Corner1. Status= Dynamic_obstacle) or
             (Corner2. Status= Fixed_obstacle) or
      (Corner2. Status= Dynamic_obstacle) then
39.      Dturns ← Dturns + 1
40. Else Current_move= "straight"
41. Path. Add (Leaders [i])
42. PathCost← PathCost+ Leaders [i]. Cost
43. iteration ← iteration + 1
44. If (Tractor. Status =" Goal")
45. Reached ← true
46. Displaying local path on Environment
47. End
48. Display Path
49. Display PathCost
50. Display iteration
51. End
```

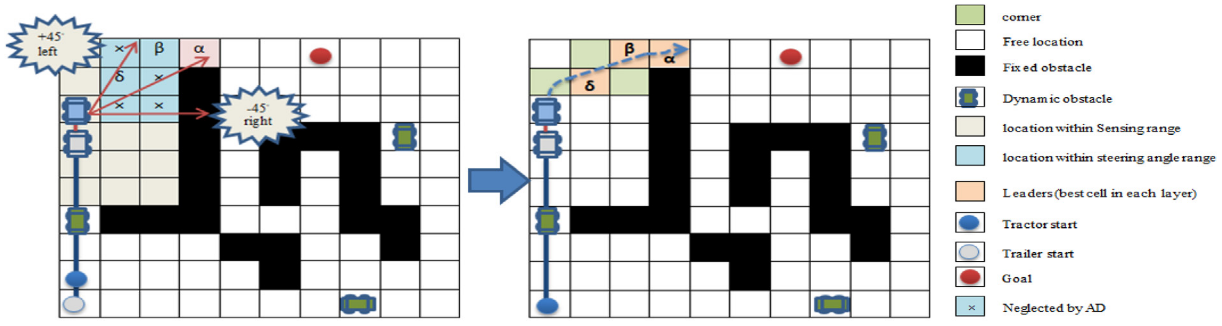


Fig. 7. Hybrid AD and GWO.

4. Simulation results

In this section, practical details about the proposed system are documented, such as the environments used for implementation and criteria for evaluating the performance of the proposed method. The results are also presented and comparisons are made to show the improvement in the analysis phase, in addition to comparisons of the results of the proposed method with those presented in [12,21].

4.1. The experimental environments

Six environments with a dimension of 50 × 50 pixels were used in the experiments as shown in Fig. 8. These environments contain large areas of static, complex-shaped obstacles with many curves and narrow passages. The complexity of the environments increases by adding 3 to 18 dynamic obstacles.

4.2. Evaluation measurements

The performance measurements used in each test are:

- Total time from starting analysis to reaching goal in moving phase (in seconds and milliseconds).
- The number of iterations.
- Path cost (length)to reach the goal.

The values of these metrics are affected by the path length and the number of dynamic objects in the robot's path.

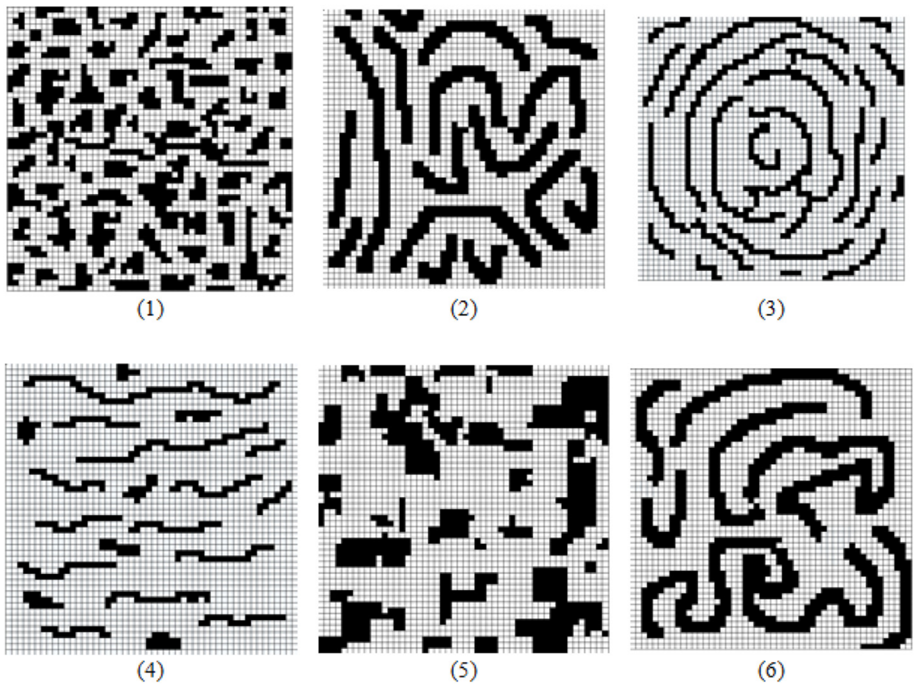


Fig. 8. The environments represented as grid maps.

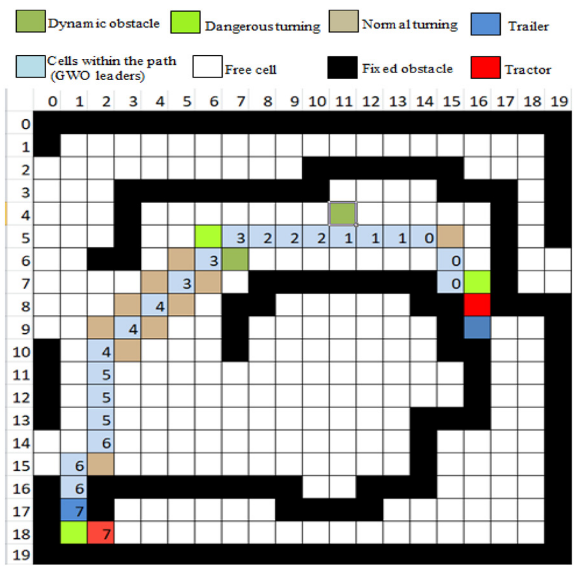


Fig. 9. A case study to implement the proposed method on an environment 20 20 grid map.*

4.3. A case study of the proposed method

In this section, a small-sized environment (20 × 20) is used to illustrate how the proposed method plans the path of the robotic vehicle while performing movements and turns and how dynamic obstacles are avoided.

The analyzer converts the environment into a 20 × 20 matrix of nodes where each cell node contains the information shown in Table 3.

The vehicle is located at (16, 8) and (16,9) as the starting point to reach the goal (2,18) through eight iterations.

In the first iteration, the sensing area contains 30 free cells, which is a large number of cells that leads to calculating a large number of possible local paths, most of these paths are not feasible because they do not match the steering angle range condition or the connectivity condition or violate both of them, so the cells that do not match these two conditions are ignored by AD. AD reduces the number of free cells in the sensing area to only 11 cells according to the steering angle, then it selects only 4 cells that match the connectivity condition. For each candidate cell, the equations of GWO are applied to the OBJ values of the candidate, and the result is summed to the number of dynamic obstacles around each candidate to find the multi-objective fitness value. The GWO chooses the fittest candidates to be the leaders (cells of the local path). In the following iterations, the same scenario is repeated. The number of cells resulting from AD differs from one iteration to another according to the available free space and the distribution of obstacles in the sensing area for that iteration.

Table 5 shows the details of the eight iterations including the vehicle coordinates, the total number of sensed cells, the number of cells selected by AD based on angle range and connectivity, the application of the GWO calculations, and the selected leaders.

The resulting path is shown in Fig. 9. The blue cells represent the cells within the path (the leaders) while the Numbers in these cells indicate the number of iterations at which they became part of the path.

The complete path of the vehicle to reach the goal consists of the following locations: (15,7)→ (156)→ (14,5)→ (13,5)→ (12,5)→ (11,6)→ (10,5)→ (9,5)→ (8,5)→ (7,5)→ (6,6)→ (5,7)→ (4,8)→ (3,9)→ (2,10)→ (2,11)→ (2,12)→ (2,13)→ (2,14)→ (1,15)→ (1,16)→ (1,17)→ (2,18) (the goal).

The important thing is the number of cells reserved for the vehicle at each step. In iterations 1, 2, and 5, the vehicle moves in a straight line so the robot only needs two cells, one for the tractor and the other for the trailer. In turning cases, the algorithm verifies the availability of a free corner cell in addition to two cells for the tractor and trailer. As shown in Fig. 9, the path contains five turns three of which the vehicle goes around an obstacle.

Table 6 documents the details of all turns on the path shown in Fig. 9 in terms of the coordinates of the tractor, trailer, and corner, turn type: normal or dangerous (turns around obstacle).

In the third and fourth iterations, the vehicle moves in a straight diagonal line, and in this type of movement, it is necessary to have free corners to execute this movement. The details of these movements are shown in Table 7.

4.4. Results of improved safety in analysis

To demonstrate the effect of the improvements made to the analyzer, the performance before and after the improvements were compared and documented in Table 8 in terms of number of turns and dangerous turns (Dturns), time, path cost, and number of iterations. The percentage change between the metrics of the two versions was calculated using the following formula:

$$\text{percentage change} = \frac{Av_{\text{after}} - AV_{\text{before}}}{AV_{\text{before}}} \times 100\ \% \tag{10}$$

From Table 8, Although the average total number of turns has increased, it is clear that the number of dangerous turns (Dturns) was significantly reduced

Table 5. Detailed Results of the hybridization between AD and GWO in the eight iterations to find the path in Fig. 9.

i	Tractor	Trailer	Adaptive dimensionality				Multi-objective Grey wolf				
			All candidates	Reducing the No. of candidate locations based on		Coordinates of final candidate locations	Obj (input)	GWO value	ND	Multi-objective Fitness	Coordinate of Leaders (local path)
				Angle	Connectivity						
0	(16,8)	(16,9)	30	11	4	(15,7)	40.2	11.66	0	35.09	δ: (15,7)
						(16,7)	41.6	13.72	0	37.144	β: (15,6)
						(15,6)	36.2	9.26	0	37.224	α: (14,5)
						(14,5)	33.8	8.2	0	25.704	
1	(14,5)	(15,6)	28	11	3	(13,5)	32.8	38.50	0	38.50	δ: (13,5)
						(12,5)	31.8	37.56	1	38.56	β: (12,5)
						(11,5)	30.8	36.63	1	37.63	α: (11,5)
2	(11,5)	(12,5)	30	10	3	(10,5)	29.8	21.24	0	21.24	δ: (10,5)
						(9,5)	28.8	20.17	0	20.17	β: (9,5)
						(8,5)	27.8	19.11	1	20.11	α: (8,5)
3	(8,5)	(9,5)	32	10	3	(5,7)	24	54.28	0	54.28	δ: (7,5)
						(6,6)	25.4	51.86	0	51.86	β: (6,6)
						(7,5)	26.8	49.43	1	50.43	α: (5,7)
4	(5,7)	(6,6)	38	12	4	(4,8)	22.6	16.37	0	16.37	δ: (4,8)
						(3,9)	21.2	15.06	0	15.06	β: (3,9)
						(2,10)	19.8	13.76	0	13.76	α: (2,10)
						(3,10)	20.2	14.13	0	14.13	
5	(2,10)	(3,9)	37	12	7	(1,11)	22.2	21.22	0	21.22	δ: (2,11)
						(2,11)	18.8	18.04	0	18.04	β: (2,12)
						(3,11)	19.2	18.42	0	18.42	α: (2,13)
						(2,12)	17.8	17.11	0	17.11	
						(1,13)	19.2	18.42	0	18.42	
						(2,13)	16.8	16.18	0	16.18	
						(3,13)	17.2	16.55	0	16.55	
6	(2,3)	(2,12)	32	7	4	(2,14)	15.8	33.03	0	33.03	α: (2,14)
						(1,15)	14.4	31.53	0	31.53	β: (1,15)
						(2,15)	14.8	31.96	0	31.96	α: (1,16)
						(1,16)	11.4	28.33	0	28.33	
7	(1,16)	(1,15)	21	7	2	(1,17)	6.4	5.22	0	5.22	δ: (1,17)
						(2,18)	0	−3.71	0	−3.71	β: (2,18)

Table 6. Details of the turns in Fig. 9.

Iteration	Tractor	Trailer	Corner	Type
0	(15,7)	(16,8)	(16,7)	Dangerous
0	(14,5)	(15,6)	(15,5)	Normal
3	(6,6)	(7,7)	(6,5)	Dangerous
6	(1,15)	(2,14)	(2,15)	Normal
7	(2,18)	(1,17)	(1,18)	Dangerous
Note	All these turns require 3 free cells to be executed.			

Table 7. Diagonal movements that need four cells in Fig. 9.

Iteration	Tractor	Trailer	Corner 1	Corner 2
3	(5,7)	(6,6)	(5,6)	(6,7)
4	(4,8)	(5,7)	(4,7)	(5,8)
4	(3,9)	(4,8)	(3,8)	(4,9)
4	(2,10)	(3,9)	(2,9)	(3,10)

by (70.73 %), as the average number of these turns was (3.17) for the improved analyzer compared to (10.83) for the analyzer before enhancement. The average time is reduced by (91.52 %) for the benefit of the improved approach. It is also noted that the number of iterations was close for the two approaches, the average number of iterations for the improved analysis was increased by (5.92 %). According to Equation 10, the path cost is increased by 7.77 % in the updated analyzer, because the old version of the analyzer gives priority to the accumulated distance at the expense of safety, therefore, the robot passes adjacent to the obstacles since those cells have lower distances. In the updated analyzer, the distance and safety are balanced, so the path cost may increase to provide more safety.

Fig. 10 clarifies a sample of the effect of the improvement on the resulting path for an experiment conducted on Environment 3. It is noticeable that the path resulting from the improved analyzer moved away from fixed obstacles, and thus the number of dangerous turns (marked in green) was reduced from 14 to only 5. However, the improved analyzer needed more iterations with an increase in cost due to passing through high-cost cells compared to those adjacent to the obstacle used by the old version (before enhancement). The numeric details of this experiment are shown in the row devoted to Environment 3 in Table 8.

Table 8. Result comparison before and after enhancement of the analysis stage.

Environment no.	Analyzer	Tractor	Trailer	Corner1	Corner2	Goal	No. of turns	No. of Dturns	Time	No. of iterations	Total Cost
1	Before	(18,38)	(17,38)	–	–	(41,30)	8	8	1:18	15	395.8
	After			–	–		7	4	0:99	14	399.4
2	Before	(23,39)	(24,39)	–	–	(23,17)	10	9	1:13	23	1042.8
	After	(23,38)		(23,39)	(24,38)		14	4	1:16	24	1078
3	Before	(26,36)	(25,35)	(25,36)	(26,35)	(23,24)	14	14	1:12	26	1443.4
	After	(26,35)		–	–		13	5	1:11	29	1567.4
4	Before	(33,5)	(32,4)	(32,5)	–	(30,40)	8	8	1:12	25	1346.4
	After	(33,5)		(32,5)	–		9	2	1:12	27	1475.8
5	Before	(44,12)	(45,11)	(44,11)	(45,12)	(10,33)	4	4	1:11	19	741.9
	After	(44,12)		(44,11)	(45,12)		10	3	1:12	19	792
6	Before	(43,18)	(43,19)	–	–	(19,24)	22	22	1:15	44	4221.8
	After	(44,18)		–	–		19	1	1:16	48	4593.6
					Average	Before	11	10.83	1:14	25.33	1532.02
						After	12	3.17	0:86	26.83	1651.03
					Percentage change		9.09 %	70.73 %	91.52 %	5.92 %	7.77 %

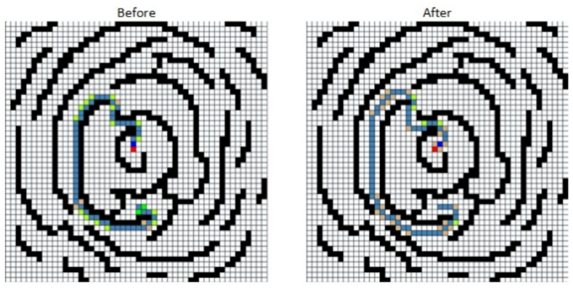


Fig. 10. Sample of resulting paths for the experiment of Environment 3 before and after analyzer enhancement.

4.5. Performance of the proposed method

Further experiments to verify the performance of the method in the presence of dynamic obstacles were conducted on the environments in Fig. 8. The number of moving obstacles ranges from 3 to 18. The details of these experiences are documented in Table 9.

For each environment, six experiments were conducted, in each experiment, the coordinates of the starting (tractor, trailer, and corners) and the goal were documented, as well as the number of dynamic obstacles (Dobs), in addition to the outputs in terms of the path cost, total time, number of iterations, and the number of total turns and dangerous turns. Fig. 11 shows the resulting path for a selected experiment for each environment which explains that the proposed method leads the robotic vehicle to the goal via a smooth and safe path with the ability to avoid a different number of obstacles successfully.

4.6. Comparison results

The results of the proposed method are compared with the results in [12,21] using the same environments. Table 10 summarizes the properties of planners in these approaches.

The trailer in this paper is equivalent to the start cell in [12,21]. Three experiments were implemented for each environment using the same coordinates and the results were documented in Table 11.

Fig. 12 shows the difference in the path shape resulting from the implementation of the methods to be compared. The first experiment on Environment 3 was chosen as an example. All the details of this experiment are documented in Table 11. In the proposed study, the articulated vehicle in this experiment started at locations (25,23) and (24,24), while the robot was located at (24,24) in the other two studies to be compared.

The goal in the three experiments is located at (35,38). The path shape in the proposed method was smooth, far from the obstacles, and slightly longer compared to the path resulting in [12] which was shorter but less safe because it was close to the obstacles.

Compared to [21], the resulting path was shorter since the articulated vehicle can pass through narrow passages, in contrast [21], provided more safety by closing such passages, consequently, the robot in [21] took a longer path to reach the destination. In conclusion, the planning approach proposed in this paper produces a path that balances path length and safety compared to other works.

Table 12 shows the average values of the path cost, total time, and number of iterations for the experiments recorded in Table 11.

As shown in Table 12, the method in [12] had the least average cost because it aimed to find the shortest path without considering safety. The method [21] had the highest cost due to giving higher priority to navigating in a safe wide area. Our proposed method had a reasonable cost compared

*Table 9. Results of the proposed method on 50*50 environments in terms of cost, number of iterations, and time .*

Environment no.	Start				Goal	# of Dobs	Total Time	# of Iterations	Path cost	# of Total turns	# of Dturns
	Tractor	Trailer	Corner1	Corner2							
1	(18,38)	(17,38)	–	–	(41,30)	3	1:10	14	399.4	9	5
	(18,4)	(17,5)	(17,4)	(18,5)	(32,36)	6	1:15	28	1601.8	14	8
	(33,48)	(32,49)	–	(33,49)	(38,17)	9	0:91	21	813	10	5
	(4,40)	(3,39)	–	(4,39)	(41,4)	12	1:14	34	2343.2	19	12
	(25,46)	(46,47)	–	(46,46)	(8,13)	15	1:12	34	2327.6	17	8
	(19,35)	(18,34)	(18,35)	–	(48,31)	18	1:11	17	518.6	8	3
2	(44,31)	(45,32)	(44,32)	(45,31)	(21,28)	3	1:16	23	1076.8	18	8
	(23,38)	(24,39)	(23,39)	(24,38)	(23,17)	6	1:14	24	1078.8	10	2
	(40,45)	(41,46)	(40,46)	(41,45)	(32,23)	9	1:12	19	746	12	4
	(33,30)	(34,31)	(33,31)	(34,30)	(32,8)	12	1:15	27	1344	9	2
	(24,38)	(25,39)	(24,39)	(25,38)	(33,17)	15	2:22	35	2418	20	9
	(43,32)	(42,33)	(42,32)	(43,33)	(10,7)	18	2:21	32	2043.8	13	3
3	(34,29)	(34,30)			(33,9)	3	1:13	29	1677.8	17	4
	(18,33)	(17,34)	(17,33)	(18,34)	(33,30)	6	0:92	21	760.6	11	2
	(26,35)	(25,35)	–	–	(23,24)	9	1:13	29	1567	15	9
	(46,31)	(45,32)	(45,31)	(46,32)	(13,7)	12	0:96	24	1246.6	14	6
	(32,17)	(33,18)	(33,17)	(32,18)	(1,48)	15	0:99	37	2840.8	17	7
	(43,19)	(43,18)	–	–	(28,42)	18	0:82	16	518.2	4	1
4	(36,45)	(35,44)	(36,46)	(35,45)	(2,13)	3	0:94	24	1251.4	12	6
	(19,33)	(18,32)	(18,33)	–	(40,5)	6	1:10	26	1376.2	10	1
	(30,23)	(31,22)	(30,22)	(31,23)	(21,40)	9	0:88	18	566	7	1
	(33,5)	(32,4)	(32,5)	–	(30,40)	12	1:12	27	1475.8	9	3
	(49,47)	(49,48)			(4,9)	15	1:12	28	1723.8	8	3
	(6,8)	(6,7)	–	–	(17,32)	18	1:10	17	529.2	8	1
5	(20,49)	(19,49)	–	–	(29,1)	3	1:12	26	1391.6	9	3
	(48,13)	(47,12)	(47,13)	(48,12)	(11,41)	6	0:99	27	1442.2	9	0
	(10,34)	(10,33)	–	–	(25,11)	9	0:99	16	441.6	7	3
	(3,22)	(2,22)	–	–	(12,44)	12	0:96	18	565.4	8	3
	(44,12)	(45,11)	(44,11)	(45,12)	(10,33)	15	1:12	20	813.20	7	1
	(38,42)	(39,41)	(38,41)	(39,42)	(0,9)	18	1:13	23	1202.2	9	3
6	(24,42)	(23,41)	(23,42)	(24,41)	(40,19)	3	0:99	13	348.8	5	0
	(19,35)	(18,34)	(18,35)	(19,34)	(40,42)	6	1:10	34	2178.8	13	2
	(7,39)	(8,38)	–	(8,39)	(39,19)	9	1:12	34	2460.8	10	1
	(5,25)	(5,26)	–	–	(12,42)	12	0:92	17	495.8	8	1
	(18,22)	(18,23)	–	–	(18,33)	15	1:16	54	5774.6	16	0
	(43,18)	(43,19)	–	–	(19,24)	18	1:10	49	4671.2	16	2

to the two methods because it balances path length and safety.

The total cost in the proposed method is *increased* by 69.92 %, 38.59 %, 54.85 %, 31.52 %, 45.78 %, and 46.62 % for the six environments compared with that of [12]. Compared with [21], the cost of the proposed method *decreased* by 17.17 %, 11.53 %, 34.28 %, 2.19 %, 7.57 %, and 4.22 % for the six environments.

It is clear that the proposed method is the best in terms of time then method [12] and lastly method [21]. The superiority of our method in terms of time is due to the efficiency of AD in reducing the number of cells being processed. The average time in the proposed method *decreased* for the six experiments by 49.58 %, 49.38 %, 1.07 %, 1.36 %, 1.08 %, and 2.97 % compared to [12] and also *decreased* by 79.74 %, 74.97 %, 74.96 %, 66.83 %, 66.46 %, and 74.69 % comparing with method [21].

In terms of the number of iterations, the method in [12] is the best, followed by the proposed method with a slight difference, then the method in [21] comes in third place, which consumes a large number of iterations as a result of the complex search for narrow tunnels. The number of iterations in the proposed methods significantly *increased* by 72.61 %, 57.14 %, 63.80 %, 52.38 %, 52.94 %, and 56.04 % compared to [12] and significantly *decreased* by 53.04 %, 49.23 %,55.61 %, 47.28 %, 46.94 %, and 48.73 % comparing to that of [21].

To summarize the results of the comparison above, in the proposed method, there is an *increase* in the cost by 47.88 % compared to that of [12] but it *decreased* by 12.83 % compared to that of [21]. The number of iterations *increased* by 59.15 % compared to that of [12] but it *decreased* by 50.14 % compared to the method adopted in [21]. The proposed method is the least time-consuming to find the path. It was noted that the time *decreased* by 17.57 % and 72.94 % compared to that of [12,21]. The reason for the balanced performance between

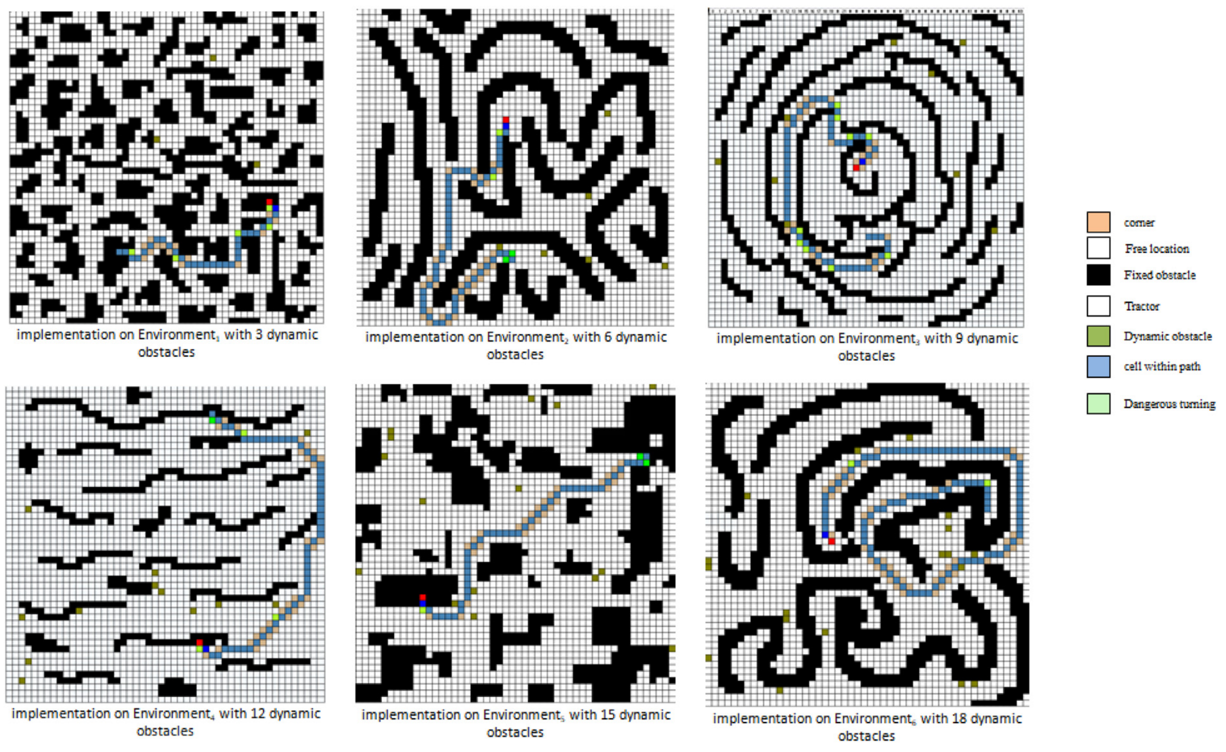

implementation on Environment1 with 3 dynamic obstacles
implementation on Environment2 with 6 dynamic obstacles
implementation on Environment3 with 9 dynamic obstacles
implementation on Environment4 with 12 dynamic obstacles
implementation on Environment5 with 15 dynamic obstacles
implementation on Environment6 with 18 dynamic obstacles
corner
Free location
Fixed obstacle
Tractor
Dynamic obstacle
cell within path
Dangerous turning

Fig. 11. Paths discovered by the proposed method in dynamic environments for selected experiments from Table 9.

Table 10. Summary of the methods to be compared.

Property	Proposed method	[12]	[21]
Robot type	Articulated robotic vehicle	Traditional moving object	Traditional moving object
No. of cells	2 to 4	1	1
No. of sensed layers	3	3	1
Reduction approach	AD	AD	–
Reducing condition	Steering angle range, connectivity, and back pointers	The Luciferin of the candidate is lower than that of the current cell	–
Swarm method	Grey wolf	Glow warm	Min-max ant colony
Safety Approach	Calculating no. of obstacles as a penalty	–	Closing narrow tunnel
Processor	Intel Core i5	Intel Core i7	Intel Core i7
Software	C# 2022	C# 2017	C# 2019

the two methods in terms of cost and number of iterations is that the method takes into account obtaining a short path with a reasonable degree of safety, while the other methods pay biased attention to a certain criterion. The method in [12] cared about the length of the path, thus guiding the robot adjacent to obstacles, so it achieves a lower cost and takes fewer iterations. In complete contrast, the method in [21] paid utmost attention to safety in an attempt to guide the robot to navigate in wide open spaces, thus increasing the length of the path, which means the need for more cycles and cost.

4.7. Limitations

In this paper, the results were obtained by simulation. It should be noted that some limitations may affect the accuracy of these results:

1) In the experiments reported in this paper, the number of dynamic obstacles ranged from 3 to 18 in environments of size 20 × 20 and 50 × 50. Increasing the number of dynamic obstacles in such a limited-size environment may consume more time and the number of iterations needed to reach the goal.

Table 11. Comparison of the proposed method [12,21], in terms of cost, no. of iterations, and time.

Environment no.	Method	Start		Goal	Total cost	No. of iterations	Time
		Tractor	Trailer				
1	[12]	–	0,42	6,2	3254.2	30	2:53
		–	3,2	2,48	3190.2	29	2:34
		–	45,6	19,41	1139	17	0:84
	[21]		0,42	6,2	6400.4	103	5:12
			3,2	2,48	6124.4	103	5:92
			45,6	19,41	3031.8	73	4.17
	Proposed method	1,41	0,42	6,2	5357.2	49	1:19
		4,3	3,2	2,48	6060	54	1:19
		45,1	45,6	19,41	1468.2	28	1:13
2	[12]	–	14,48	33,16	1235.8	18	1:39
		–	8,0	48,46	1467	21	2:05
		–	48,24	1,24	2005.4	24	1:92
	[21]		14,48	33,16	1628.2	53	3:74
			8,0	48,46	2295	62	5.56
			48,24	1,24	3452	80	4:37
	Proposed method	13,47	14,48	33,16	1562.6	29	1:14
		8,1	8,0	48,46	2217.8	33	1:15
		47,24	48,24	1,24	2744.6	37	1:16
3	[12]	–	24,24	35,38	1035	18	1:34
		–	33,18	25,35	1124	19	1:45
		–	42,3	0,37	1420.8	19	0:53
	[21]		24,24	35,38	2988.2	73	4:28
			33,18	25,35	1685.8	54	3:99
			42,3	0,37	3761.2	80	4:71
	Proposed method	25,23	24,24	35,38	1928.8	31	1:14
		32,14	33,18	25,35	1633.6	29	1:14
		41,4	42,3	0,37	1981.2	31	1:12
4	[12]	–	0,0	49,49	1931	23	0:99
		–	39,2	15,37	1259.4	18	0:94
		–	0,49	49,0	1769.8	22	1:09
	[21]		0,0	49,49	2442.8	64	3:69
			39,2	15,37	1796	54	3:57
			0,49	49,0	2431.2	64	3:38
	Proposed method	1,1	0,0	49,49	2560.4	35	1:12
		40,2	39,2	15,37	1651.2	28	1:13
		1,48	0,49	49,0	2312	33	1:14
5	[12]	–	41,1	2,3	839	17	1:86
		–	3,6	47,14	1400	21	1:16
		–	25,22	47,14	594.2	13	0:25
	[21]		41,1	2,3	1597.6	51	3:38
			3,6	47,14	2073.2	59	3:66
			25,22	47,14	797.8	37	2:00
	Proposed method	40,2	41,1	2,3	1263.2	24	1:12
		4,7	3,6	47,14	2112	34	1:13
		26,23	25,22	47,14	754.8	20	1:10
6	[12]	–	33,36	22,12	875.2	16	0:40
		–	1,48	49,1	2044.2	25	1:62
		–	22,13	27,17	1641.4	21	1:91
	[21]		33,36	22,12	1405	51	2:84
			1,48	49,1	3069.2	71	4.10
			22,13	27,17	2507.6	67	4:34
	Proposed method	34,35	33,36	22,12	1267.8	26	1:12
		1,47	1,48	49,1	3066	38	1:15
		23,12	22,13	27,17	2353.4	33	1:13

2) The sensing area is limited to three layers which is consistent with the hierarchical nature of the GWO pack which finds three leaders (solutions), and therefore, a leader is appointed at each layer to guide the robotic vehicle through the three layers.

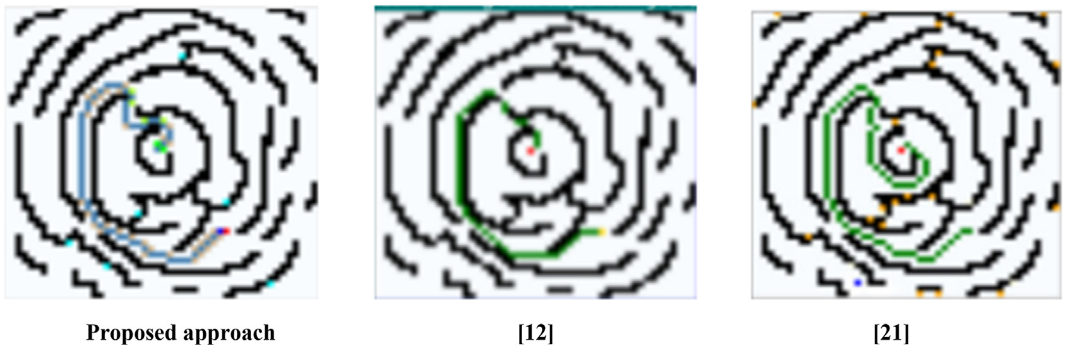


Fig. 12. Comparison between the paths obtained from the three planers for environment 3.

Table 12. The average of cost, time, and no. of iterations for the experiments in Table 11.

Measure	Method	Environment number					
		1	2	3	4	5	6
Cost	[12]	2527.8	1569.4	1193.3	1653.4	944.4	1520.3
	[21]	5185.5	2458.4	2811.7	2223.3	1489.5	2327.3
	Proposed	4295.13	2175	1847.867	2174.5	1376.7	2229.1
Time	[12]	2:17	2:05	1:24	1:27	1:22	1:44
	[21]	5:20	4:55	4:46	3:54	3:14	4:02
	Proposed	1:17	1:15	1:13	1:13	1:11	1:13
No. of iterations	[12]	25.3	21	18.7	21	17	20.7
	[21]	93	65	69	60.7	49	63
	Proposed	43.67	33	30.63	32	26	32.3

5. Conclusions and future works

In this study, path planning is developed for an articulated robotic vehicle navigating in a dynamic environment using hybrid AD-GWO. The results showed that the proposed method succeeds in keeping the vehicle away from fixed obstacles and avoiding moving objects in less time with a reasonable number of iterations and total cost to enhance the degree of safety. From this work we can conclude:

1) Using an adaptive number of cells to represent the robot during its different movements is a more efficient and realistic representation because it simulates the size of the robot and its capabilities (such as the presence of a joint) unlike the single-cell representation that treats robots equally without giving importance to these individual differences.
2) Improving analysis by the calculation of the number of fixed obstacles in the analysis phase resulted in a decrease in dangerous turns and average time by (70.73 %) and (91.52 %) respectively with an increase in the average number of iterations and path cost by (5.92 %) and (7.77 %) respectively.
3) Using AD to reduce the sensing area by selecting an adaptive number of cells that match the size and movement constraints of the articulated vehicle saves time and reduces the computational complexity spent on processing cells that cannot be reached due to the presence of dynamic obstacles or articulated constraints.
4) Using GWO to lead the robotic vehicle is an effective choice because the hierarchy of the GWO community fits the three layers of the sensing area. The GWO leads the vehicle from the current location through the three layers of sensing by assigning a leader at each layer. The layer leader is the best-evaluated cell according to multi-objective Grey Wolf's calculations.
5) Increasing safety generally leads to an increase in the cost and number of iterations due to moving through higher-cost but safer cells. For this reason, there was an increase in the path cost and number of iterations by (47.88 %) and (59.15 %) compared to the results of [12], while the cost and the number of iterations decreased by (12.83 %) and (50.14 %) compared to [21] because the latter increases safety in a way that leads to a longer path and thus more costly.
6) The average total time in the proposed method decreased by (17.57 %) and (72.94 %) compared to [12,21] respectively, which indicates the efficiency of the proposed method despite the complexities imposed by the size of the robot

and the number of cells needed to represent the robot compared to these methods that assume that the robot needs one free cell for each step, ignoring its size and flexibility. For future works, we suggest the following improvements:

1) The method can be adapted to work on double-trailer robotic vehicles.
2) The method can also be developed to plan paths of multiple robotic articulated vehicles.
3) The vehicle speed should be taken into consideration in the planning process, especially when executing a turn. The speed should be calculated based on the turning angle, i.e. the speed must be reduced when the turning angle is large so that the robotic vehicle does not roll over.

Funding

No funds.

Ethical statement

This study does not involve any human participants or animal experiments requiring ethical approval.

Conflict of interest

The authors declare that they have no conflict of interest that could influence this work, and they all agree to publish this paper under ethical academic.

Acknowledgments

We would like to express our deep gratitude to the Information Technology College, University of Babylon for their invaluable guidance and unwavering support for this research.

References

[1] A.N. Atiyah, N. Adzhar, N.I. Jaini, A review on path planning and obstacle avoidance algorithms for autonomous mobile robots, J. Robotics 2022 (538220) (2022) 1−14, https://doi.org/10.1155/2022/2538220.

[2] S.K. Sahoo, B.B. Choudhury, A review of methodologies for path planning and optimization of mobile robots, J. Proc. Manag. New Technol. 11 (1) (2023) 122−140, https://doi.org/10.5937/jouproman2301122S.

[3] M. Panda, B. Das, B. Subudhi, B.B. Pati, A comprehensive review of path planning algorithms for autonomous underwater vehicles, Int. J. Autom. Comput. 17 (3) (2020) 321−352, https://doi.org/10.1007/s11633-019-1204-9.

[4] C.O. Yinka-Banjo, U. Agwogie, Mobile robot path planning in an obstacle-free static environment using multiple optimization algorithms, Niger. J. Technol. Dev. 17 (3) (2020) 165−173, https://doi.org/10.4314/njtd.v17i3.3.

[5] L. Brezočnik, I. Fister, V. Podgorelec, Swarm intelligence algorithms for feature selection: a review, Appl. Sci. 8 (9) (2018) 1−31, https://doi.org/10.3390/app8091521, 1521.

[6] F.S. Gharehchopogh, B. Abdollahzadeh, B. Arasteh, An improved farmland fertility algorithm with hyper-heuristic approach for solving travelling salesman problem, Comput. Model. Eng. Sci. 135 (3) (2023) 1981−2006, https://doi.org/10.32604/cmes.2023.024172.

[7] F.S. Gharehchopogh, A.A. Khargoush, A chaotic-based interactive autodidactic school algorithm for data clustering problems and its application on COVID-19 disease detection, Symmetry 15 (4) (2023) 1−26, 894.

[8] F.S. Gharehchopogh, B. Abdollahzadeh, N. Khodadadi, S. Mirjalili, A hybrid African vulture optimization algorithm and harmony search: algorithm and application in clustering, in: A. Biswas, C.B. Kalayci, S. Mirjalili, eds., Advances in Swarm Intelligence, Springer, Cham. 2023, pp. p241−p254, https://doi.org/10.1007/978-3-031-09835-213.

[9] F.S. Gharehchopogh, S. Mirjalili, G. Işõk, B. Arasteh, A new hybrid whale optimization algorithm and golden jackal optimization for data clustering, in: S. Mirjalili, eds., Handbook of Whale Optimization Algorithm, Academic Press. 2024, pp. 533−546, https://doi.org/10.1016/B978-0-32-395365-8.00044-0.

[10] S.S. Rezk, K.S. Selim, Metaheuristic-based ensemble learning: an extensive review of methods and applications, Neural Comput. Appl. 36 (29) (2024) 17931−17959, https://doi.org/10.1007/s00521-024-10203-4.

[11] A.H. Hasan, A.M. Mosa, Multi-robot path planning based on max-min ant colony optimization and D* algorithms in a dynamic environment, in: 2018 International Conference on Advanced Science and Engineering (ICOASE), 2018, pp. 110−115, https://doi.org/10.1109/ICOASE.2018.8548805.0252.

[12] Q.R. Mahmood, A.H. Hasan, H.K. Khafaji, Robot path planning based on hybrid adaptive dimensionality representation with glowworm swarm optimization, in: 2021 4th International Iraqi Conference on Engineering Technology and Their Applications (IICETA), 2021, pp. 241−246, https://doi.org/10.1109/IICETA51758.2021.9717626.

[13] A. Vemula, K. Muelling, J. Oh, Path planning in dynamic environments with adaptive dimensionality, Proceed. Ninth Intern. Sympos. Combinat. Search 7 (1) (2016) 107−115, https://doi.org/10.1609/socs.v7i1.18386.

[14] G. Xu, T. Zhang, Q. Lai, J. Pan, B. Fu, X. Zhao, A new path planning method of mobile robot based on adaptive dynamic firefly algorithm, Mod. Phys. Lett. B 34 (29) (2020) 1−18, https://doi.org/10.1142/S0217984920503224, 2050322.

[15] J. Lu, Z. Zhang, An improved simulated annealing particle swarm optimization algorithm for path planning of mobile robots using mutation particles, Wireless Commun. Mobile Comput. 2021 (2374712) (2021) 1−12, https://doi.org/10.1155/2021/2374712.

[16] X. Cheng, J. Li, C. Zheng, J. Zhang, M. Zhao, An improved PSO-GWO algorithm with chaos and adaptive inertial weight for robot path planning, Front Neurorobot. 15 (770361) (2021) 1−10, https://doi.org/10.3389/fnbot.2021.770361.

[17] F. Li, Y. Du, K. Jia, Path planning and smoothing of mobile robot based on improved artificial fish swarm algorithm, Scient. Reports 12 (659) (2022) 1−16, https://doi.org/10.1038/s41598-021-04506-y.

[18] O. Hamed, M. Hamlich, M. Ennaji, Hunting strategy for multi-robot based on wolf swarm algorithm and artificial potential field, Indonesian J. Electric. Eng. Comput. Sci. 25 (1) (2022) 159−171, https://doi.org/10.11591/ijeecs.v25.i1.pp159-171.

[19] P. Li, L. Yang, Conflict-free and energy-efficient path planning for multi-robots based on priority free ant colony optimization, Math Biosci. Eng. 20 (2) (2022) 3528−3565, https://doi.org/10.3934/mbe.2023165.

[20] Z. Yang, J. Li, L. Yang, Q. Wang, P. Li, G. Xia, Path planning and collision avoidance methods for distributed multi-robot systems in complex dynamic environments, Math Biosci. Eng 20 (1) (2022) 145−178, https://doi.org/10.3934/mbe.2023008.

[21] A.H. Hasan, H.S. Abdullah, M.A. Saleh, Finding robotic wide free space path in dynamic environment by improving MAX−MIN ant system algorithm, Intern. J. Interact. Mobile Technol. 17 (13) (2023) 59−78, https://doi.org/10.3991/ijim.v17i13.41513. .

[22] F. Kiani, A. Seyyedabbasi, R. Aliyev, M.U. Gulle, H. Basyildiz, M.A. Shah, Adapted-RRT: novel hybrid method to solve three-dimensional path planning problem using sampling and metaheuristic-based algorithms, Neural Comput. Appl. 33 (22) (2021) 15569−15599, https://doi.org/10.1007/s00521-021-06179-0.

[23] S. Dhouib, Shortest path planning via the rapid Dhouib-Matrix-SPP (DM-SPP) method for the autonomous mobile robot, Results Cont. Optimiz. 13 (100299) (2023) 1−12, https://doi.org/10.1016/j.rico.2023.100299.

[24] T.F. Abaas, A.H. Shabeeb, Safe and optimum navigation of wheeled mobile robot using grey wolf optimization algorithm, IOP Conf. Ser.: Mater. Sci. Eng 928 (022006) (2020) 1−12, https://doi.org/10.1088/1757-899X/928/2/022006.

[25] Y. Ou, P. Yin, L. Mo, An improved gray wolf optimizer and its application in robot path planning, Biomimetics 8 (1) (2023) 1−34, https://doi.org/10.3390/biomimetics8010084, 84.

[26] M.H. Hashem, H.S. Abdullah, k.I. Ghathwan, Gray wolf optimization algorithm: a survey, Iraqi J. Sci. 64 (11) (2023) 5964−5984, https://doi.org/10.24996/ijs.2023.64.11.40.

[27] Y. Luo, Q. Qin, Z. Hu, Y. Zhang, Path planning for unmanned delivery robots based on EWB-GWO algorithm, Sensors 23 (4) (2023) 1−36, https://doi.org/10.3390/s23041867, 1867.

[28] J. Chai, L. Zhang, T. Liu, The development and application research of grey wolf optimization algorithm, Electron. Commun. Networks 381 (2024) 683−690, https://doi.org/10.3233/FAIA231253.

[29] A. Seyyedabbasi, F. Kiani, I-GWO and Ex-GWO: improved algorithms of the Grey Wolf Optimizer to solve global optimization problems, Eng. Comput. 37 (1) (2021) 509−532.

[30] Y. Qiu, X. Yang, S. Chen, An improved gray wolf optimization algorithm solving to functional optimization and engineering design problems, Sci. Rep. 14 (14190) (2024) 1−24, https://doi.org/10.1038/s41598-024-64526-2.

[31] A. Seyyedabbasi, F. Kiani, T. Allahviranloo, U. Fernandez-Gamiz, S. Noeiaghdam, Optimal data transmission and pathfinding for WSN and decentralized IoT systems using I-GWO and ex-GWO algorithms, Alex. Eng. J. 63 (2023) 339−357, https://doi.org/10.1016/j.aej.2022.08.009.

[32] B. Arasteh, B. Aghaei, B. Farzad, K. Arasteh, F. Kiani, M.T. Afshar, Detecting SQL injection attacks by binary gray wolf optimizer and machine learning algorithms, Neural Comput. Appl. 36 (12) (2024) 6771−6792, https://doi.org/10.1007/s00521-024-09429-z.

[33] F. Kiani, A. Seyyedabbasi, R. Aliyev, M.A. Shah, M.U. Gulle, 3D Path Planning Method for Multi-UAVs Inspired by Grey Wolf Algorithms, J. Internet Technol. 22 (4) (2021) 743−755, https://doi.org/10.53106/160792642021072204003.