Control and Cybernetics
vol. **54** (2025) No. 2
pages: 223-253
DOI: 10.2478/candc-2025-0008

A novel LAPN algorithm based path navigation approach for autonomous agents*

by

Subhradip Mukherjee

MVJ College of Engineering,
Faculty of Industrial Internet of Things, Bangalore, Karnataka, India
subhradip11@gmail.com

Abstract: This paper discusses multi-agent path navigation in unknown environments. It is a challenging task to design an effective communication-based algorithm with minimal error, which would ensure secure navigation of multi-agent paths under complex circumstances, reduce the length of the travelled path, and minimize the runtime. A leader agent path navigation (LAPN) algorithm is proposed in this paper for multi-agent communication. The obstacle avoidance mechanism is used in the first part of the algorithm. The execution time of the algorithm is influenced by the process of leader-to-follower path update, since the updating process determines how quickly the followers receive corrected trajectories. One leader and two follower agents were considered in simulation environments to establish the feasibility of the algorithm. The LAPN algorithm achieves an average percentage deviation, calculated as the relative difference from the ideal straight-line path of 2.82% in travel time and 31% in path length, showing satisfactory navigation efficiency under obstacle-constrained conditions.

Keywords: multi-agent path navigation, LAPN algorithm, obstacle avoidance mechanism, path update, simulation environments

1. Introduction

Nowadays, high-level communication between embedded devices leads to an emerging need for various operations. One of the popular research subjects in this context is multi-agent path navigation under different circumstances. The primary focus in the respective studies is on safe and secure autonomous vehicle path navigation in difficult conditions. The key challenges in autonomous

*Submitted: September 2024; Accepted: October 2025.

navigation are obstacle avoidance and efficient goal pursuit. Additionally, the agents considered should be capable of carrying out a variety of tasks while navigating complicated surroundings. These robot agents are used to carry out activities in industries like transportation, farming, mining, and rescue operations in hazardous environments, among others. To address the navigation issue, various path-planning techniques have been developed. Increasing algorithmic complexity typically results in longer computational time, and in some cases may also lead to longer paths, due to excessive detours, generated during obstacle handling.

The here proposed approach aims to enable real-time autonomous vehicular navigation. One encounters a difficult issue when it comes to safe and secure autonomous vehicle path navigation in an uncertain environment. The primary problems regarding the spatial environment are avoiding impediments and pursuing the objective. During this procedure, an intelligent algorithm can shorten the robot's run time and length of the travelled path, thereby also simplifying the execution of other, specific tasks.

2. Literature review

Within the control-oriented stream of research, Bhowmick and Panja (2019) proposed a mathematical model for a simulated multi-agent leader–follower configuration and demonstrated that the tracking errors could be minimized even under external disturbances. In a similar way, Wang and Huang (2019) formulated a disturbance observer–based control problem, assuming the leader's state to be unknown, and designed an adaptive distributed control rule to overcome this challenge. Hu, Park and Cao (2020) presented a two-layered multi-agent architecture, in which the selection of follower agents is influenced by the degree of inter-layer coupling.

In the area of autonomous navigation, Sebastian and Ben-Tzvi (2019) developed a path-planning architecture, based on the D* Lite algorithm, applied to a two-dimensional environment, for vehicular navigation. Ho et al. (2019) investigated aerial navigation and introduced a collision detection method for unmanned aerial vehicles. Yang et al. (2019) combined an ant colony optimization (ACO) approach with a turning point optimization scheme to generate collision-free and smooth trajectories in complex environments. Mukherjee, Kumar and Borah (2020a) proposed an intelligent algorithm for wheeled mobile robots that ensures reliable, obstacle-free motion in difficult terrains. Artunedo et al. (2020) implemented a localization and path-planning approach using probabilistic occupancy grids for autonomous ground vehicles. In a similar study, Mukherjee, Kumar and Borah (2020b) examined vehicle performance through a distance-function-modified BUG algorithm under various two-dimensional en-

vironments. Fitzpatrick et al. (2020) provided an overview of navigation and localization methodologies, tailored for underwater autonomous systems. Another study by Mukherjee et al. (2020) demonstrated the real-time implementation of a smart obstacle-avoidance algorithm for dynamic path planning. Chi et al. (2019) analyzed the effectiveness of a Rapidly Exploring Random Tree (RRT)-based navigation algorithm, evaluating its efficiency in terms of path length and travel time. Magnago et al. (2019) introduced a greedy prediction model to anticipate agent trajectories in dynamic environments. Feng et al. (2018) improved the line-of-sight (LOS) control for intelligent agents pursuing moving targets using Kalman predictor–based methods. Jiang, Yao and Zhou (2018) designed a genetic algorithm (GA) based real-time path planning strategy for mechanical arms with obstacle avoidance. Ning et al. (2018) further enhanced the ACO algorithm by introducing a pheromone update mechanism that accelerates convergence during route generation.

Neural network–based techniques have also been applied to robot navigation. Thus, Sun et al. (2017) utilized neural control for a bipedal robot to achieve balance and posture stability while compensating for system uncertainties. The subsequent studies (Rath et al., 2018; Kamil et al., 2017) use fuzzy logic and heuristic strategies rather than neural networks. So, Rath et al. (2018) designed a fuzzy logic–based navigation and obstacle-avoidance strategy for the NAO humanoid robot within the V-REP simulation environment. Kamil et al. (2017) demonstrated that the MLD algorithm enables robots to navigate effectively among both static and dynamic obstacles. In agricultural applications, Ilker and Topakci (2015) introduced a GPS-guided Smart Robot Navigation (SRN) technique for precision farming.

More recent research highlights the growing emphasis on intelligent learning and sensing approaches. Zhao and Ning (2022) proposed an enhanced reinforcement learning–driven navigation method for multi-robot systems. Yin et al. (2022) presented a LIDAR-assisted visual localization and mapping method, validated through real-time experiments. Shahriari and Biglarbegian (2022) developed a novel time-to-collision–based controller capable of dynamically adjusting steering to enhance robot safety. Wang et al. (2022) addressed cooperative navigation through a modular formation-control structure that minimizes overall system cost. Antipov, Kokunko and Krasnova (2022) designed a dynamic differentiator, employing piecewise linear correction to produce smoother, realizable control signals. Biundini et al. (2022) integrated multiple heuristic approaches to improve the cost-efficiency of autonomous path planning. Raja et al. (2023) combined deep reinforcement learning with a leader–follower methodology to develop an IoT-based system for aerial network formation control. Huang et al. (2023) proposed a teleoperation-based technique for collaborative object manipulation in single-leader dual-follower configurations, while Huang, Xu and Wu (2024) extended the concept to magnetic millirobots with heterogeneous mag-

netization properties. Zhang et al. (2024) investigated optimal containment in heterogeneous multi-agent systems using reinforcement learning to manage unknown agent dynamics.

In socially aware navigation research, Chandra et al. (2023) designed algorithms that enhance collision avoidance and reduce travel time within SocialMAPF contexts. Ngo et al. (2014) developed a human-aware navigation method that dynamically regulates interpersonal distance to improve motion planning and safety in human–robot interaction. Further, Chung et al. (2024) provided a unified evaluation method integrating deep reinforcement learning with multi-agent pathfinding (MAPF) technology. Wen, Liu and Li (2022) presented a hierarchical Car-Like Conflict-Based Search (CL-MAPF) method capable of generating kinematically feasible, collision-free trajectories for car-like robots.

According to the literature review, the controller of an autonomous vehicle should be intelligent enough to avoid obstacles using goal-seeking behaviour and efficient in terms of the length of the travelled path and the amount of vehicle run time. In this paper, a multi-agent system is studied, consisting of one leader and two followers operating within simulated environments. In Fig. 1, the leader-follower process is shown in a sketch.

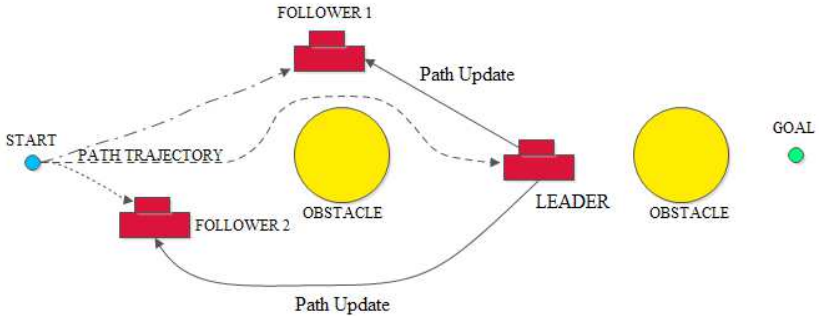


Figure 1. Leader-follower system

The proposed approach consists in the path navigation and obstacle avoidance technique for the agents. This procedure can be applied in real-time vehicular transportation using sensors such as LIDAR or ultrasonic modules, based on task-specific requirements. The simulation environments with start and goal positions are established so as to test the feasibility of the here outlined leader-agent-path navigation (LAPN) algorithm. In this approach, a leader agent will follow the goal from the start position with the obstacle avoidance mechanism. After the leader starts, one agent will follow the leader with a few seconds time gap. Other agents will then follow the leader with controlled time gaps (ranging

from 0.5 to 10 seconds) between subsequent agent activations. The leader will update the secure and collision-free path information, supplied to the followers. The outcome of this approach can be seen in the case of the last follower, which has a better path length and travel time compared to the previous agents in this system. The success rate and accuracy of the system have been shown in the result section.

3. Methods

The tracking scheme of the leader-follower system is designed here with three agents. The tasks of the agents are different, depending on their roles. The leader agent's primary job is to seek, in a dynamical setting, safe and collision-free trajectory. Follower agents get a sensor-based update about the secure nodes from the leader agent and try to determine the path trajectory of the lead agent. A node in this context is a specific coordinate point (x, y) in the 2D environment, used for navigation and obstacle avoidance. The distinguished types of nodes are as follows: Start Node (S): Initial position of the agent, Goal Node (G): Target location, Detection Node (A): Location where an obstacle is first detected, and Escape Node (E): Position beyond the obstacle where line-of-sight to the goal is restored.

The leader agent follows the angle of tracking to the goal, which is α_{track}. Here, tracking angle to the goal is introduced as,

$$\alpha_{track} = \tan^{-1} \frac{G_Y - S_Y}{G_X - S_X}, \tag{1}$$

where S_X is the X co-ordinate of start position (S);

S_Y = Y co-ordinate of start position (S);

G_X = X co-ordinate of goal position (G);

G_Y = Y co-ordinate of goal position (G).

The distance (d) between goal and start position is considered is considered to be the Euclidean distance; the straight-line distance between start and goal positions. Hence,

$$d = \sqrt{(G_Y - S_Y)^2 + (G_X - S_X)^2}. \tag{2}$$

When an obstacle is detected by the sensor system of the leader agent in the traveled path, then the obstacle avoidance mechanism is activated. The leader agent starts moving to any side of the obstacle border with the minimum avoidance distance. It continues moving until it reaches the escape node, from which it can sense the obstacle-free path. The relationship between the safe

distance and the escape node can be expressed as

$$EE' = \|Agent(x, y) - Ob(x, y)\| \tag{3}$$

Here, E: Escape node (safe point beyond obstacle) and $E\prime$: Closest border point of the obstacle from the agent's perspective. The operator $\|\cdot\|$ represents the Euclidean distance norm. Formula (3) states that the safe distance between the agent and the obstacle is measured as the Euclidean distance between the agent's current position and the obstacle border. $Agent\,(x, y)$ is the current position of the agent and $Ob\,(x, y)$ is the position of the obstacle. This defines how the leader detects the escape node — the point, from which a clear path to the goal becomes visible. One strong ranging sensor is needed for that purpose. The leader agent will move with a steering angle $\beta_{steering}$. The steering angle is formulated as,

$$\beta_{steering} = \cos^{-1} \frac{d_{obs}}{k^j}, \tag{4}$$

where d_{obs} = the minimum distance between the agent and the obstacle; and k^j = the minimum distance between the A node and E node after j^{th} iteration.

This steering angle dictates the avoidance manoeuver, intended for reaching the escape node. The escape node ensures that the line-of-sight (LOS) to the goal is obstacle-free.

Now, the A node is the obstacle detection node and the E node is the best escape node for the leader agent. The path from the E node to the goal will be always a straight line, because the distance between the E node and the goal is calculated as in formula (2). If the E node is not detected, then the algorithm stops. Figure 2 illustrates the navigation and obstacle avoidance strategy. The leader agent moves from the start node towards the goal, continuously checking for obstacles. Upon detecting an obstacle at node A, it searches for an escape node E along the obstacle boundary. From node E, the agent resumes straight-line motion to the goal. Node A is the point where a potential collision is first detected when an obstacle is sensed.

In Fig. 2, the path trajectory of the lead agent has been shown. The agent starts moving from the obstacle detection node with continuous obstacle detection through the calculation of distance from the current position to the goal. The idea behind the proposed design is that the agent's i desired body rate is directly related to the target's j LOS rate; that is:

$$s_{i_d} = [P]_i s_{d_{ij}}, \tag{5}$$

where:

LOS rate is the rate of change of the line-of-sight angle between agent i and target j.

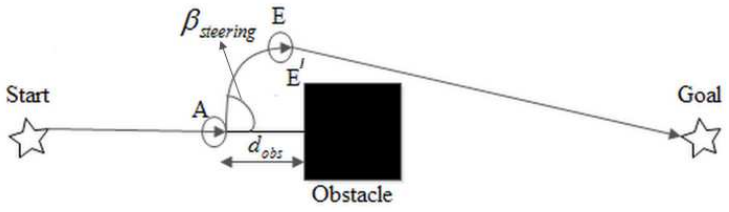


Figure 2. Path navigation and obstacle avoidance mechanism for the leader agent

Body rate is angular velocity command in the agent's body frame.

$s_{i_d} = [A_{i_d}\ B_{i_d}\ C_{i_d}]^T$ is the agent i's required body rotation vector in the fixed coordinate system.

$[P]_i = diag(P_{1_i}\ P_{2_i}\ P_{3_i})$ are corresponding demand channels' navigation constants. Navigation constants P_{1_i}, P_{2_i}, P_{3_i} are tuning parameters that scale the responsiveness of the agent in roll, pitch, and yaw channels, respectively. They determine how aggressively the agent reacts to LOS changes. Higher values mean more aggressive steering; lower values mean smoother but slower responses.

$s_{d_{ij}}$ = the sight line rotation vector.

The principle of Proportional Navigation (PN) states that the commanded body rate of the pursuer (agent) is proportional to the LOS rate to the target. If the LOS reflects fast rotation, the agent must rotate proportionally to keep tracking the goal. The required acceleration for vehicle / agent i is provided by:

$$h_{i_d} = s_{i_d} \times u_i = [P]_i s_{d_{ij}} \times u_i. \tag{6}$$

Here, s is the rotation vector, h is the acceleration, u_i represent a unit or basis vector, and α_{track} is the angle of tracking. Since guidance commands are executed in the body coordinate frame of the agent, the acceleration vector from Eq. (6) must be transformed from the inertial frame to the body frame using the transformation matrix $[W_f^b]$, thus:

$$h_{i_d}^b = [W_f^b]_i \left([P]_i\, s_{d_{ij}} \times u_i\right). \tag{7}$$

Assuming that in response to the steering commands, there is no longitudinal acceleration, we get:

$$h_{x_{i_d}}^b = \frac{\delta S_i - \delta C_i}{p_i} = 0. \tag{8}$$

Equation 8 indicates no longitudinal acceleration in the x-direction (body frame wise). This is an assumption, which is typical in planar PN guidance—focusing only on lateral control. The foundation of this approach is the idea that the agent's (i) required lateral acceleration is proportional to the acceleration normal to the LOS acceleration to target (j), which is brought on by the LOS rotation. Now, the LOS acceleration is given by:

$$h_{n_{ij}} = s_{d_{ij}} \times u_{ij} \tag{9}$$
$$h_{i_d} = [M]_i h_{n_{ij}} = [P]_i s_{d_{ij}} \times u_{ij}.$$

By transforming to body axis, we get:

$$h^b_{i_d} = [W^b_f]_i \left([P]_i \, s_{d_{ij}} \times u_{ij} \right) \tag{10}$$

Again, we transform guidance acceleration from fixed to body frame, but this time using the actual LOS vector. Assuming once more that there is no longitudinal acceleration in response to the guidance instructions, we get:

$$h^b_{x_{i_d}} = \frac{\delta W_i - \delta C_i}{p_i} = 0 \tag{11}$$

where $h_{n_{ij}}$is the normal LOS acceleration.

This reiterates the absence of longitudinal control, emphasizing that only lateral steering adjustments are applied for trajectory correction Lastly, the APN guidance law is a variant of the PN guidance law that takes the target acceleration into account, and it can be implemented as follows:

$$h^b_{i_d} = \left[PNGL \left(P, s_{d_{ij}} \right) \right] + [W^b_f]_i \left(\left[\bar{P} \right]_{g_j} \right) \tag{12}$$

where $\left[\bar{P} \right]$ is the target acceleration navigation constant, and $PNGL$ is the proportional navigation guidance law, given in formulae (5) through (11).

The final LAPN guidance formulation combines the Proportional Navigation Guidance Law output with dynamic obstacle-avoidance control inputs. The leader agent tracks the goal using PN-based angular rate tracking. It uses real-time LOS rotation and obstacle detection to update its steering angle. When an obstacle is detected, the escape node search is activated, aiming at a minimal avoidance radius. Use is made of the steering angle β to move tangentially around the obstacle, and reengage the PN guidance after reaching the escape node.

When the best escape node to the goal is identified by the leader agent after j^{th} iteration, then the agent updates the node information to the follower agents with the help of wireless communication sensors like XBee module. The pseudo code for the LAPN algorithm is given in Table 1.

Table 1. Pseudocode of the LAPN algorithm

Algorithm: LAPN Algorithm
1. **Input:** Initialize the start position, the goal position, path of the agent, initialize the number of obstacles, initialize the maximum number of iterations, and initialize other major parameters.
2. **Output:** Safe Path Trajectory
3. **BEGIN ALGORITHM**
4. The movement and path find with the help of equation 1 and 2 5. For j number of iterations
6. Calculate the safe distance from start to goal 7. If (obstacle >0) 8. Calculate the escape node 9. Calculate the steering angle 10. Update the escape node position to the followers 11. Update the best path to the followers 12. Calculate the safe distance from updated position to goal 13. Calculate the updated steering angle 14. End If 15. Set safe path trajectory to goal 16. Update to the followers 17. End For 18. Follow the best updated path to goal
19. **END ALGORITHM**

The active follower agents will move to the updated escape node from their current position. In this work, the first follower agent starts its journey after a time gap t_{gap} after the leader agent starts and before the leader agent has reached the obstacle detection node. The first follower agent starts moving in the goal direction. After getting the escape node information from the leader agent, the first follower agent starts moving from the current node (B) to the escape node. Using a GPS sensor, the follower agents can track the escape node. The path trajectory for the first follower is shown in Fig. 3.

The first agent will travel from the start position to the B node, then from the B node to the E node, and from the E node to the goal position. Then, its travel times are calculated from (B-S) mod3, (E-B) mod3, and (G-E) mod3. It can be seen that none of the node-to-node travel times are evaluated using the same reference interval; therefore, the addition of these travel times yields $(G + t_{gap})$. The second agent will start with a time gap t'_{gap} after the leader agent starts and after the leader agent has passed the escape node. If t'_{gap} is twice greater than t_{gap}, then the second agent will be updated by the leader about the best escape node from the beginning. So, it will follow a straight line from the start position to the G node and from the E node to the goal position. Therefore, the total travel time for the second agent will be measured by $(G + t'_{gap})$ mod3. The path trajectory for the second agent is given in Fig. 4.

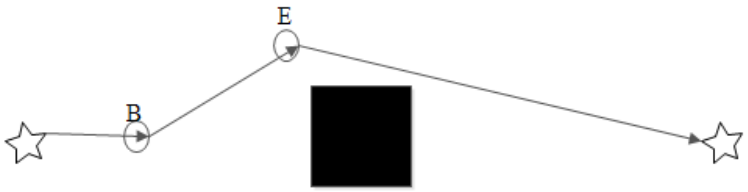


Figure 3. Path trajectory for the first follower agent

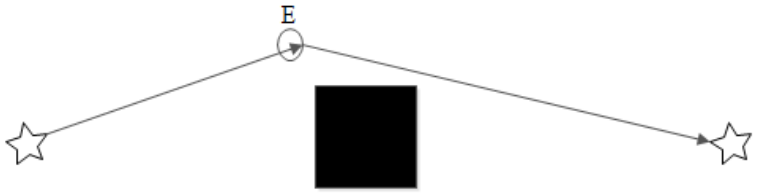


Figure 4. Path trajectory for the second follower agent

It is obvious that the travel time and travelled path length for the last agent are lesser than for the other agents. The time gap is the factor that influences the path length and travel time for the follower agents. The flowchart of the LAPN algorithm, outlined here, is given in Fig. 5.

4. Results and discussion

The simulation results of the proposed algorithm are presented in this section. The entire simulation of the proposed algorithm was done by MATLAB software.

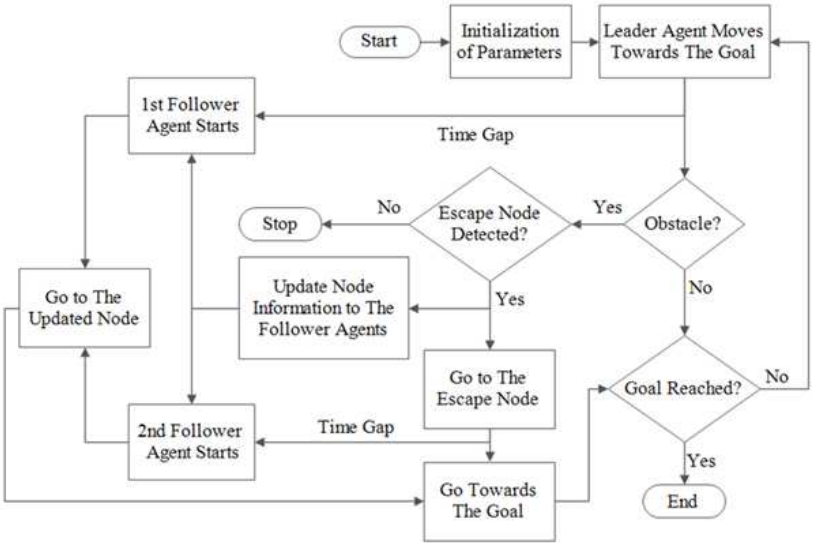


Figure 5. Flowchart of LAPN algorithm

The exemplary path trajectories of the leader agent in multiple environments are shown in Fig. 6. Then, Table 2 displays the outcomes of the leader agent in various environments.

Excessive number of iterations is avoided in order to maintain real-time responsiveness and reduce communication delays. The agents' starting speed was limited to fewer than 10 meters per second. The algorithm was tested in five different environments, as shown further on in Fig. 7. Figure 6 illustrates eight leader path examples. The follower agents' time gap is adjusted based on the leader's movements. The initial parameter values for the LAPN algorithm are shown in Table 3.

The environments were designed with multiple obstacles and with their different coordinate values. For all kinds of situations, a grid of 300×300 units was utilized. Table 4 lists the environments that are taken into account when testing the algorithm.

In Environment 1, as specified in Table 4, the agents, functioning according to the LAPN algorithm, performed satisfactorily. The leading agent always reaches the target, while the follower agents sometimes malfunction due to update errors. The path trajectories of the agents, functioning according to the LAPN algorithm in Environment 1, are shown in Fig. 7a. The time it takes for each agent to reach the goal position depends on the number of obstacles present

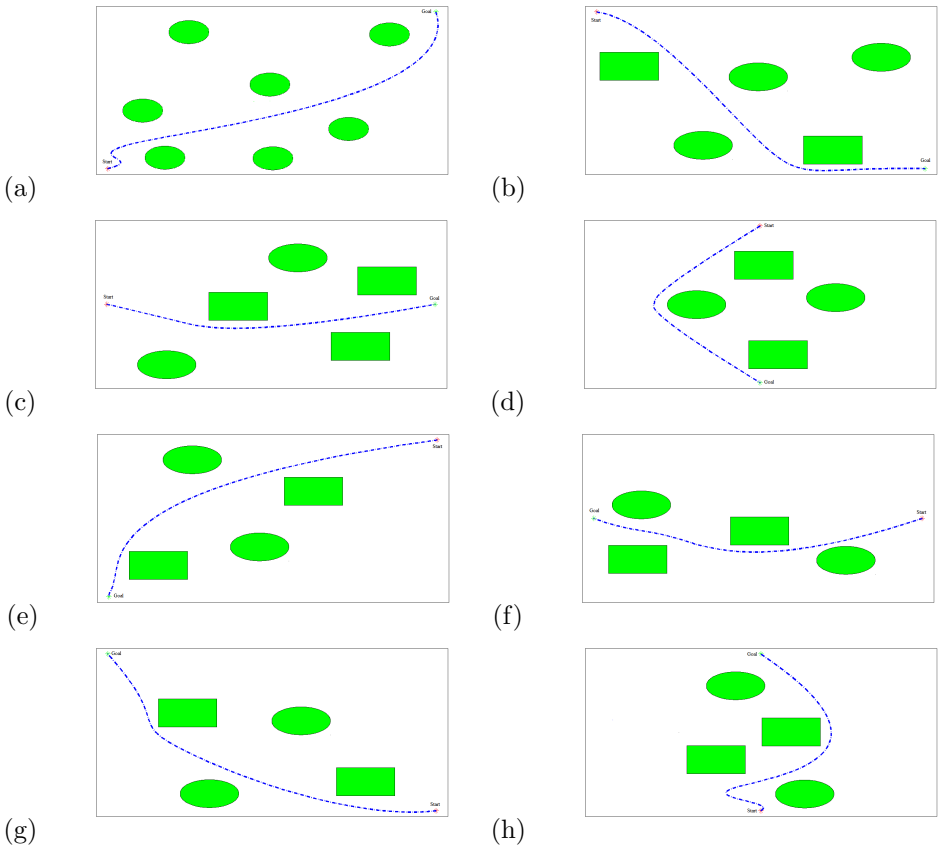


Figure 6. Navigation paths of the proposed algorithm in different simulation environments

Table 2. The performance measures of the leader agent in various environments from Fig. 6

Environments	**Path length (units)**	**Travel time (seconds)**
a	403.302	19.245
b	283.255	11.258
c	403.761	19.417
d	291.207	12.826
e	402.414	18.860
f	283.642	11.424
g	402.225	18.650
h	288.425	12.273

Table 3. Parameter values for the LAPN algorithm simulations

Parameter	**Value**
No. of iterations	≤ 100
agent speed	0 –10 meters/second
d_{obs}	0.1 – 2 meters
t_{gap}	0.5 – 10 seconds
t'_{gap}	1 – 20 seconds

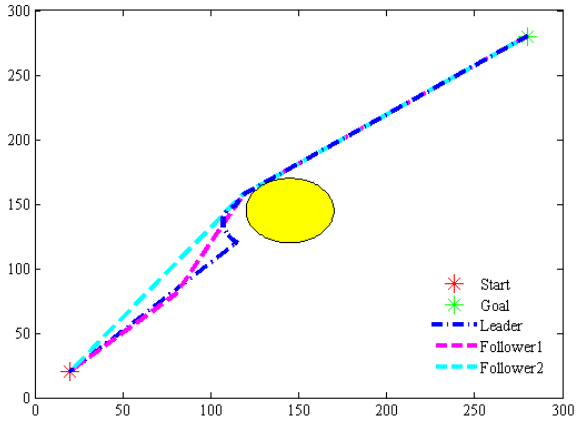
Table 4. Different environments assumed for simulation with results shown further on

Environments	**Start (Units)**	**Goal (Units)**	**Obstacle**
1	(20, 20)	(280, 280)	1
2	(20, 20)	(280, 280)	3
3	(20, 150)	(280, 150)	2
4	(20, 280)	(280, 20)	2
5	(20, 280)	(280, 20)	3

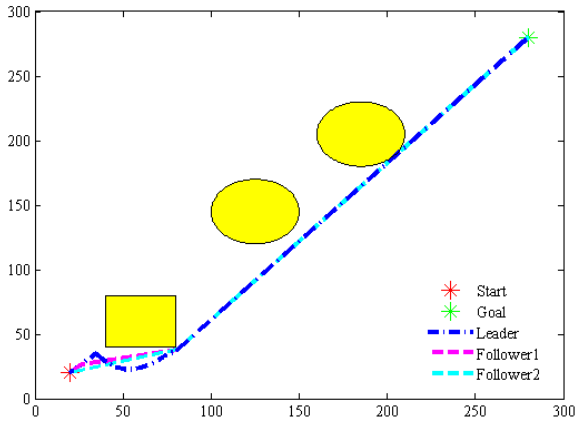
in the environment. There are more impediments in Environment 2 than in Environment 1, and that is why the agents' trip time has nearly doubled. Figure 7b shows how well the algorithm performs in Environment 2. The path lengths of the agents depend, of course, on the placement of the obstacles. That is the reason for the slightly increased path length of agents in Environment 3. The path trajectories of the agents in the Environment 3 are shown in Fig. 7c. Square and otherwise rectangular obstacles were used as random test objects to evaluate adaptability. The path trajectory of the leader has the best escape point dependent on the shape of the obstacle. In Fig. 7d, the path trajectories for Environment 4 are shown. It is difficult for the leader to locate the escape route when the barriers are arranged in a complicated way. Additionally, updating the follower agents takes a longer time. Therefore, in more complex environments with multiple obstacles for this kind of environment, both the journey time and the path length increases. The travel trajectories for Environment 5 are presented in Fig. 7e.

In every scenario, follower 2 has the shortest path length and travel time in comparison to other agents. These agents are best suited for a smart traffic system using the recommended approach. Depending on the environment, each agent starts at a different moment. Even though the simulation process has multiple time gaps, only suitable time gaps are taken into account while executing the technique. Because the calculation process slows down as the number of barriers increases, the maximum number of obstacles in different circumstances is three. The obstacle locations are specified, along with their sizes. The data, concerning the optimal performance of the agents in different scenarios is shown in Table 5.

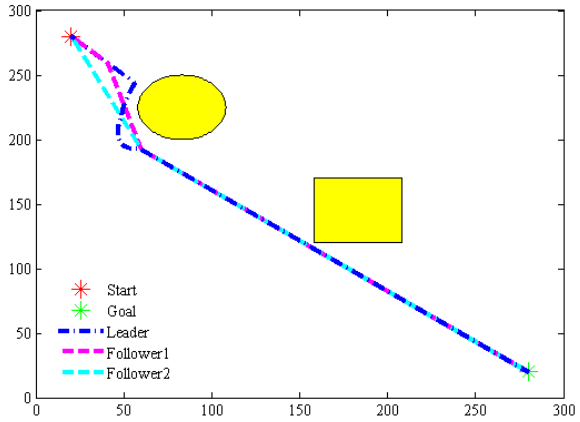
The best results are chosen after ten independent simulation runs in each environment with different time gaps. After successful simulations, the best values of time gaps are used in the algorithm. In general, for the first follower the time difference is between 500 milliseconds and 10 seconds, whereas the second follower's time gap is between 1 and 20 seconds. In the majority of cases, the second follower's time gaps are double of that of the first follower. The success rates attained in every instance are satisfactory. It will be quite challenging to set up the communication process between agents because of this communication delay during node updates, and the associated updating error. The path and the escape node that the agents create and follow are accurate, in terms of similarity of theoretical and actual paths in every setting. In the majority of situations, the escape node location and path length were also satisfactorily accurate, in the same sense as before. The accuracy, expressed in percent, for each example is calculated, and the mean of the ten cases in each setting is shown. To display the algorithm's accuracy and success rate, Table 6 is presented.


Start
Goal
Leader
Follower1
Follower2

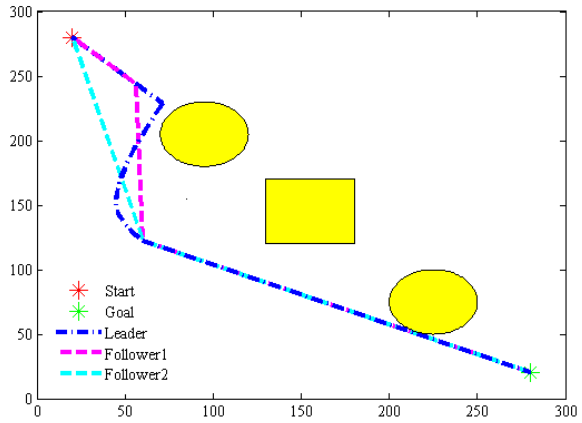
(a)


Start
Goal
Leader
Follower1
Follower2

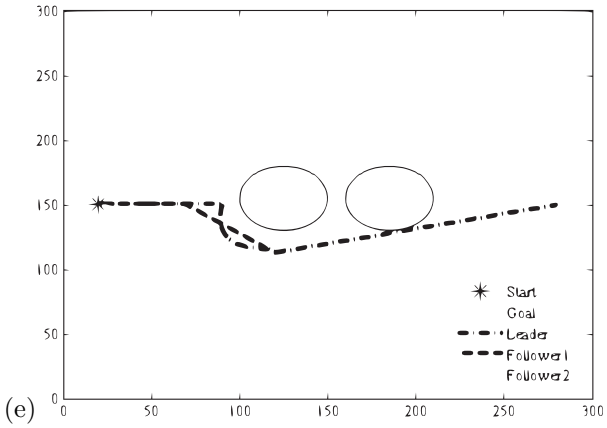
(b)


300
250
200
150
100
50
0
0
50
100
150
200
250
300
Start
Goal
Leader
Follower1
Follower2

(c)


300
250
200
150
100
50
0
0
50
100
150
200
250
300
Start
Goal
Leader
Follower1
Follower2

(d)



(e)

Figure 7. Navigation paths of the proposed algorithm with three agents in (a) Environment 1, (b) Environment 2, (c) Environment 3, (d) Environment 4, and (e) Environment 5

Table 5. Best results of different agents in terms of path length and travel time

Environment	**Agent**	**Path length (units)**	**Travel time (seconds)**
1	Leader	382.295	16.723
	Follower1	379.133	13.636
	Follower2	377.172	9.152
2	Leader	380.143	27.254
	Follower1	378.652	22.287
	Follower2	377.571	15.168
3	Leader	275.178	11.662
	Follower1	274.229	9.026
	Follower2	271.427	5.059
4	Leader	381.075	29.363
	Follower1	379.492	23.649
	Follower2	377.822	16.104
5	Leader	418.383	32.312
	Follower1	415.472	26.491
	Follower2	405.480	17.842

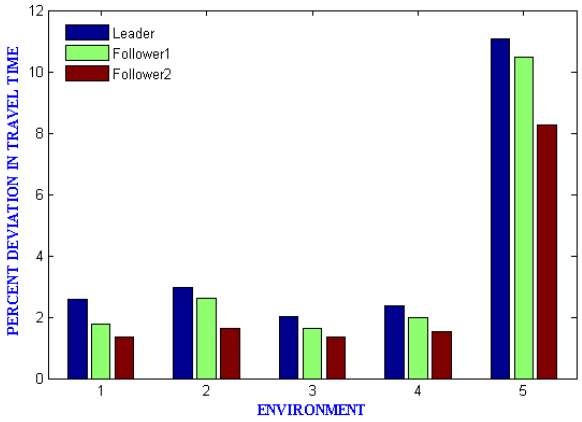
In the above description, "success" refers to simulation runs, in which all agents reached the goal position without collision or communication breakdown, while "failure" means that at least one agent could not reach the goal due to a path deadlock, obstacle entrapment, or update delay, see also below.

Table 6. Success rate and accuracy percentage for different environments

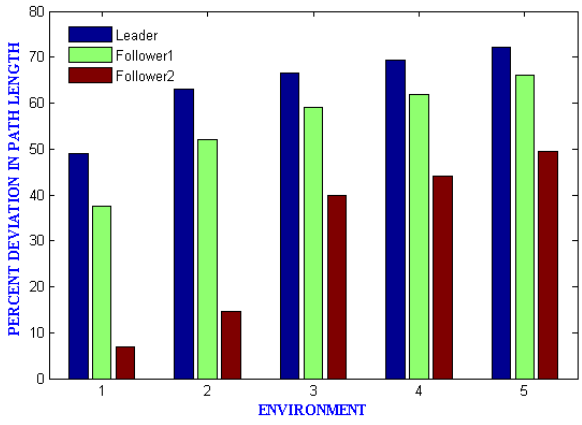
Environment	**No. of simulations**	**No. of successful simulations**	**Approx. accuracy (%)**
1	10	7	58
2	10	6	52
3	10	8	61
4	10	7	55
5	10	5	46

So, while in Table 5, the best path lengths and travel times of different agents are shown in each environment with varying gaps of time, in Table 6, successful simulation means that all agents (leader and followers) successfully reached their goal. Overall, 66% of the total simulations were successful, representing the average success rate computed across all five simulated environments. Accuracy percentage was calculated as (1 – (travelled path length – actual path length) / actual path length) * 100%. (The "actual path length" is the theoretical simplest route length.) The total accuracy percentage is higher than 54%. This is the average of all accuracy percentages in five environments. In each setting, the agents' percentage variation in terms of path length and travel time is computed. The travel time and path length were measured under uninterrupted simulation conditions, ensuring that no external interference or data loss affected the results. After that, the measured values in each example are used to calculate the percentage deviation. Figure 8 shows a graph with deviation values in % for journey time and for path length.

Follower 2 has very clearly the lowest deviation when compared to other agents. In Environment 1, follower 2 has 1.36% deviation in time, which is by 0.40% and 1.22% lower than follower 1 and leader agent, respectively. In Environments 2, 3, 4, and 5, follower 2 has 1.63%, 1.36%, 1.54%, and 8.25% deviation in time, respectively, this being by 1%, 0.28%, 0.43%, and 2.21% lower than for follower 1 and by 1.34%, 0.67%, 0.84% and 2.83% lower than for the leader agent. In successive five environments, the follower 2 has 6.90%, 14.60%, 39.87%, 44.11%, and 49.55% of deviation in path length, this being by 30.57%, 37.53%, 19.20%, 17.83%, and 16.47%, respectively, less than for follower 1, and



a



b

Figure 8. Percentage deviation in (a) travel time and (b) path length of the agents in different environments

by 42.15%, 48.35%, 26.66%, 25.23% and 22.59%, respectively, less than for the leader agent.

We have compared the final results of the proposed algorithm with those, produced by OAIA (Mukherjee, Kumar and Borah, 2020a), DFMB (Mukherjee, Kumar and Borah, 2020b), SAWOA (Mukherjee et al. 2020), MLD (Kamil et al., 2017), and SRN (Unal and Topakci, 2015) approaches. The results of this comparison for Environment 1 are presented in Table 7.

Table 7. Comparison of results for Environment 1

Algorithm	**Path length**	**Travel time**	**Success rate (%)**	**Accuracy (%)**
Proposed LAPN (Follower 2)*	377.172	9.152	70	58
OAIA	379.344	10.106	64	52
DFMB	378.753	9.620	68	55
SAWOA	379.741	10.493	58	50
MLD	381.548	11.850	55	48
SRN	382.663	11.245	52	46

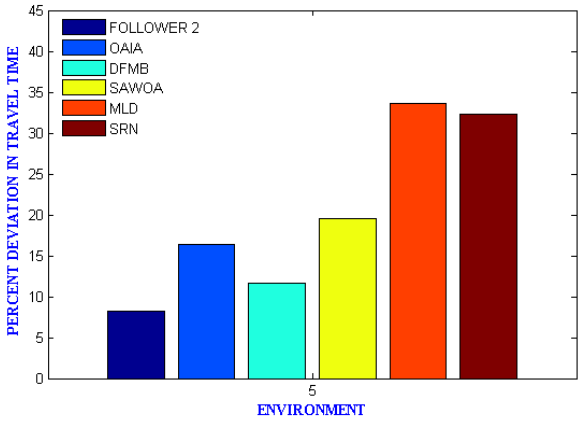
* "Follower 2" represents the best performing agent when using the LAPN algorithm

The values of deviations for different algorithms have also been calculated in the case of Environment 1. The effect of comparison with other algorithms in this respect is shown in Fig. 9.

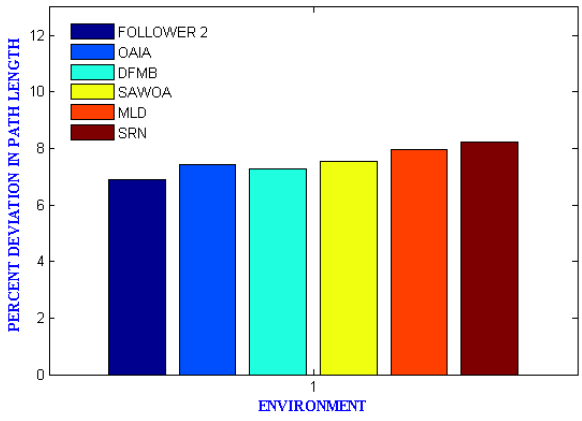
The results for Environment 1 show that the proposed algorithm has by 9.31%, 4.80%, 12.61%, 22.46%, and 18.36% lower deviation in travel time and by 0.53%, 0.39%, 0.63%, 1.06%, and 1.33% lower deviation in path length, compared, respectively, to OAIA, DFMB, SAWOA, MLD and SRN approaches.

Analogous results to those from Table 7, but obtained for Environment 2, are presented in Table 8.

Similarly, deviation values for different algorithms in the experiment with Environment 2 were obtained and are shown in Fig. 10 and then in Table 9. It can be seen that the proposed algorithm has, respectively, by 6.46%, 2.96%, 8.87%, 21.58%, and 15.66% lower deviation in travel time and by 0.51%, 0.36%, 0.59%, 1.15%, and 1.34% lower deviation in path length when compared to OAIA, DFMB, SAWOA, MLD and SRN approaches.



a



b

Figure 9. Comparison of deviation values for (a) travel time and (b) path length, resulting from the use of different algorithms in Environment 1

Table 8. Comparison of results for Environment 2

Algorithm	**Path length**	**Travel time**	**Success rate (%)**	**Accuracy (%)**
Proposed LAPN (Follower 2)*	377.571	15.168	60	52
OAIA	379.847	16.235	56	48
DFMB	379.164	15.639	59	51
SAWOA	380.221	16.672	49	46
MLD	382.752	19.430	46	45
SRN	383.605	18.040	44	41

* See Table 7 for explanation

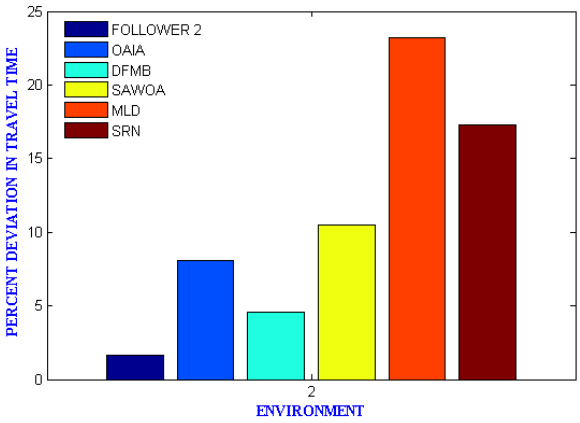
Results, analogous to those shown before for Environment 1 and 2, obtained for various considered algorithms in conditions of Environment 3, are now presented in Table 9 and Fig. 11. It is easy to see that also in this case the here proposed algorithm presents an advantage over the other ones in terms of all the criteria taken into account in the study.

Table 9. Comparison of results for different algorithms applied in conditions of Environment 3

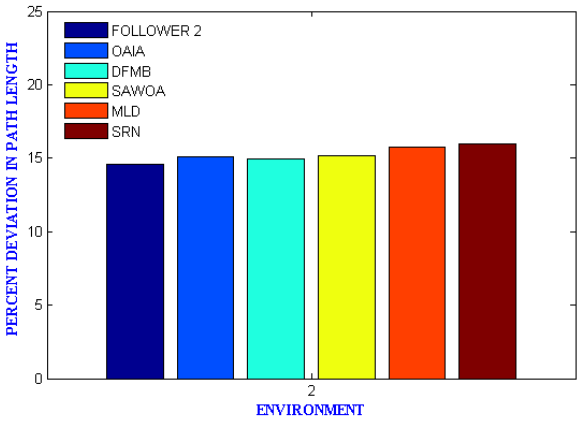
Algorithm	**Path length**	**Travel time**	**Success rate (%)**	**Accuracy (%)**
Proposed LAPN (Follower 2)*	271.427	5.059	80	61
OAIA	272.652	7.403	71	58
DFMB	272.021	6.610	74	60
SAWOA	272.838	8.397	63	54
MLD	275.417	10.541	57	51
SRN	276.502	9.225	54	47

* See Table 7 for explanation

The results show that the proposed algorithm has by 31.23%, 23.14%, 39.21%, 51.30%, and 44.54% lower deviation in travel time and by 0.27%, 0.13%, 0.31%,



(a)



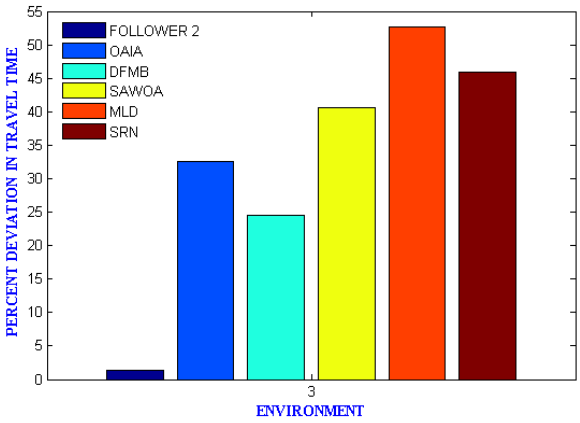
(b)

Figure 10. Comparison of deviation values with respect to (a) travel time and (b) path length for the different algorithms applied to Environment 2

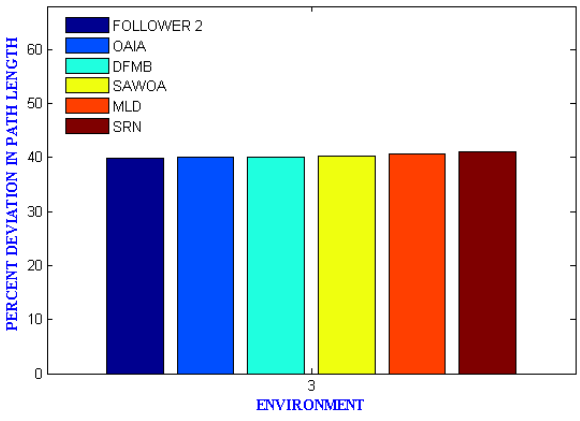
0.87%, and 1.10% lower deviation in path length, when compared, respectively, to OAIA, DFMB, SAWOA, MLD and SRN algorithms.

Analogous results of comparison are next presented in Table 10 and Fig. 12 for Environment 4.

The here proposed algorithm has by 4.29%, 0.98%, 7.28%, 20.43%, and 16.06% lower deviation in travel time and by 0.34%, 0.21%, 0.38%, 0.83% and 0.94% lower deviation in path length, when compared, respectively, to OAIA, DFMB, SAWOA, MLD and SRN algorithms, when applied in Environment 4.

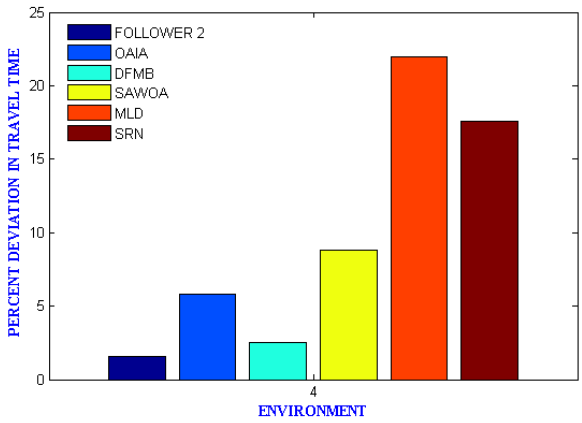


(a)

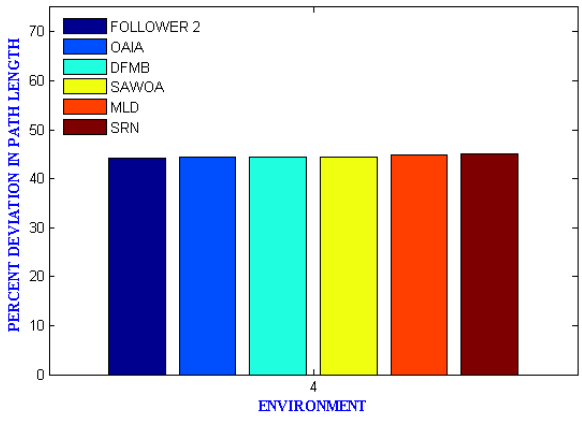


(b)

Figure 11. Comparison of deviation values for different algorithms in terms of (a) travel time and (b) path length in Environment 3



(a)



(b)

Figure 12. Comparison of deviation values in terms of (a) travel time and (b) path length for different algorithms functioning in Environment 4

Table 10. Comparison of results from different algorithms for Environment 4

Algorithm	**Path length**	**Travel time**	**Success rate (%)**	**Accuracy (%)**
Proposed LAPN (Follower 2)*	377.822	16.104	70	55
OAIA	380.193	16.832	63	51
DFMB	379.250	16.260	66	54
SAWOA	380.452	17.385	55	47
MLD	383.578	20.314	52	43
SRN	384.317	19.236	48	39

* See Table 7 for explanation

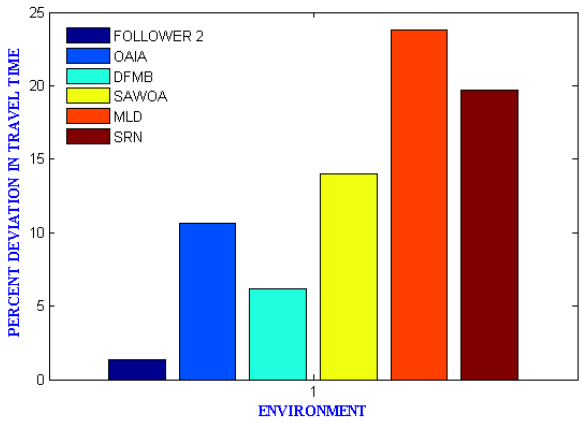
Finally, the results from a similar comparison of the selected algorithms, performed for the conditions of Environment 5, are shown in Table 11 and in Fig. 13.

It can be easily seen that the proposed algorithm is better than the other ones, with 8.13%, 3.35%, 11.32%, 25.32%, and 24.06% lower deviation in travel time and 0.40%, 0.34%, 0.48%, 0.77%, and 0.83% lower deviation in path length, when compared to OAIA, DFMB, SAWOA, MLD and SRN algorithms, functioning in Environment 5.

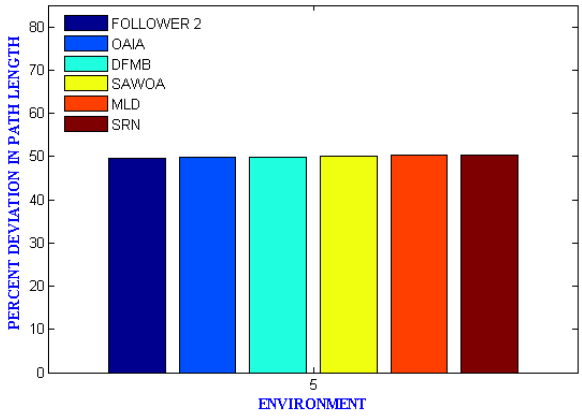
Thus, the simulation results, here illustrated, demonstrate that the LAPN algorithm has worked efficiently in all the given environments with satisfactory success rate and accuracy.

5. Conclusions

The proposed LAPN algorithm has shown its ability to perform successfully in different, relatively challenging environments. Multi-agent communication can be established with the help of this algorithm. The path updating mechanism plays a dominant role, as it takes the major part of time in the computation process. In the simulation results, we can see that the movement of the last agent has taken less time and it takes place over a shorter path length compared to the previous agents. Most of the simulation was error-free, with consistent and numerically precise results across all test environments. The comparative studies show that the proposed algorithm has on average by 17.34% lower deviation in travel time and on average by 0.64% lower deviation in path length. On the other hand, the average success rate of the proposed algorithm is by 10.84%



(a)



(b)

Figure 13. Comparison of deviation values in terms of (a) travel time and (b) path length for different algorithms applied in conditions of Environment 5

Table 11. Comparison of results for different algorithms applied in conditions of Environment 5

Algorithm	**Path length**	**Travel time**	**Success rate (%)**	**Accuracy (%)**
Proposed LAPN (Follower 2)*	405.480	17.842	50	46
OAIA	408.715	19.577	50	45
DFMB	408.259	18.520	50	45
SAWOA	409.440	20.355	47	43
MLD	411.832	24.643	40	39
SRN	412.271	24.185	38	36

* See Table 7 for explanation

higher than for the existing algorithms. The average accuracy of this algorithm is by 6.56% higher than that of the existing algorithms compared.

The future work on the proposed method will introduce dynamic object tracking with bounded network coverage. For agent route optimization, a hybrid meta-heuristic procedure is also a better option to reduce the computational cost associated with fitness function evaluation. The proposed algorithm can be implemented with stable communication between real-time smart vehicles. The proposed approach can be also adapted to solve multi-agent routing problems inspired by the Traveling Salesman Problem.

References

Antipov, A., Kokunko, J. and Krasnova, S. (2022) Dynamic models design for processing motion reference signals for mobile robots. *J. Intell. Robot Syst.* 105:77.

Artunedo, A., Villagra, J., Godoy, J. and Castillo, M. D. (2020) Motion Planning Approach Considering Localization Uncertainty. *IEEE Trans. Veh. Technol.* **69**(6):5983–5994.

Bhowmick, S. and Panja, S. (2019) Leader–Follower Bipartite Consensus of Linear Multiagent Systems Over a Signed Directed Graph. *IEEE Trans. Circuits Syst. II Express Briefs*, **66**(8):1436–1440.

Biundini, I. Z., Melo, A. G., Coelho, F. O., Honório, L. M., Marcato, A. L. M. and Pinto, M. F. (2022) Experimentation and simula-

tion with autonomous coverage path planning for UAVs. *J. Intell. Robot Syst.* 105:46.

Chandra, R., Maligi, R., Anantula, A. and Biswas, J. (2023) SocialMapf: Optimal and Efficient Multi-Agent Path Finding With Strategic Agents for Social Navigation. *IEEE Robot. Autom. Lett.* **8**(6):3214–3221. doi:10.1109/LRA.2023.3265169.

Chi, W., Wang, C., Wang, J. and Meng, M. Q-H. (2019) Risk-DTRRT-Based Optimal Motion Planning Algorithm for Mobile Robots. *IEEE Trans. Autom. Sci. Eng.* **16**(3):1271–1288.

Chung, J., Fayyad, J., Younes, Y. A. et al. (2024) Learning team-based navigation: a review of deep reinforcement learning techniques for multi-agent pathfinding. *Artif. Intell. Rev.* 57:41. doi:10.1007/s10462-023-10670-6.

Feng, S., Zhang, G., Dong, Y., Zhang, X. and Wang, P. (2018) Improved line of sight robot tracking toward a moving target. *Syst. Sci. Control Eng.* **6**(3):227–234.

Fitzpatrick, M., Reis, G. M., Anderson, J., Bobadilla, L., Al Sabban, W. and Smith, R. N. (2020) Development of environmental niche models for use in underwater vehicle navigation. *IET Cyber-Syst. Robot.* **2**(2):67–77.

Ho, F., Geraldes, R., Gonçalves, A., Cavazza, M. and Prendinger, H. (2019) Improved Conflict Detection and Resolution for Service UAVs in Shared Airspace. *IEEE Trans. Veh. Technol.* **68**(2):1231–1242.

Hu, A., Park, J. H. and Cao, J. (2020) Node-to-node bipartite consensus of multi-agent systems with disturbances. *IET Control Theory Appl.* **14**(13):1692–1699.

Huang, C., Xu, T. and Wu, X. (2024) Leader–Follower formation Control of magnetically actuated millirobots for automatic navigation. *IEEE/ASME Trans. Mech.* **29**(2):1272–1282.

Huang, D., Yang, C., Li, M., Huang, H. and Li, Y. (2023) Motion regulation solutions to holding and moving an object for Single-Leader-Dual-Follower teleoperation. *IEEE Trans. Ind. Inf.* **19**(10):10170–10181.

Ilker, U. and Topakci, M. (2015) Design of a remote-controlled and GPS-guided autonomous robot for precision farming. *Int. J. Adv. Robot. Syst.* **12**(12):1–10.

Jiang, A., Yao, X. and Zhou, J. (2018) Research on path planning of real-time obstacle avoidance of mechanical arm based on genetic algorithm. *J. Eng.* **2018**(16):1579–1586.

Kamil, F., Hong, T. S., Khaksar, W., Moghrabiah, M. Y., Zulkifli, N. and Ahmad, S. A. (2017) New robot navigation algorithm for arbitrary unknown dynamic environments based on future prediction and priority behavior. *Expert Syst. Appl.* **86**(15):274–291.

Magnago, V., Palopoli, L., Passerone, R., Fontanelli, D. and Macii, D. (2019) Effective Landmark Placement for Robot Indoor Localization with Position Uncertainty Constraints. *IEEE Trans. Instrum. Meas.* **68**(11):4443–4455..

Mukherjee, S., Kumar, R. and Borah, S. (2020a) Obstacle-avoiding intelligent algorithm for quad wheel robot path navigation. *Int. J. Intell. Unmanned Syst.* 9(1):29–41.

Mukherjee, S., Kumar, R. and Borah, S. (2020b) Computational Study of DFMB Algorithm for Unmanned Vehicle in Static 2D Environment. In: *2020 IEEE Int. Conf. Comput. Performance Evaluation (ComPE)*, 126–131.

Mukherjee, S., Kumar, R., Borah, S. and Bhattacharjee, R. (2020) An Enhanced Experimental Study of GPS Based Path Tracking Nonholonomic Robot with SAWOA Algorithm. In: *2020 IEEE Int. Conf. Comput. Sci., Eng. Appl. (ICCSEA)*, 1–5.

Ngo, H. Q. T., Le, V. N., Thien, V. D. N., Nguyen, T. P. and Nguyen, H. (2024) Develop the socially human-aware navigation system using dynamic window approach and optimize cost function for autonomous medical robot. *Adv. Mech. Eng.* **12**(12). doi:10.1177/16878140209794 30.

Ning, J., Zhang, Q., Zhang, C. and Zhang, B. (2018) A best-path-updating information-guided ant colony optimization algorithm. *Inf. Sci.* 433-434:142–162.

Raja, G., Essaky, S., Ganapathisubramaniyan, A. and Baskar, Y. (2023) Nexus of deep reinforcement learning and Leader–Follower approach for AIOT enabled aerial networks. *IEEE Trans. Ind. Inf.* **19**(8): 9165–9172.

Rath, A. K., Parhi, D. R., Das, H. C., Muni, M. K. and Kumar, P. B. (2018) Analysis and use of fuzzy intelligent technique for navigation of humanoid robot in obstacle prone zone. *Def. Technol.* **14**(6):677–682.

Sebastian, B. and Ben-Tzvi, P. (2019) Physics Based Path Planning for Autonomous Tracked Vehicle in Challenging Terrain. *J. Intell. Robot. Syst.* 95:511–526.

Shahriari, M. and Biglarbegian, M. (2022) Toward Safer Navigation of Heterogeneous Mobile Robots in Distributed Scheme: A Novel Time-to-Collision-Based Method. *IEEE Trans. Cybern.* 52:9302–9315.

Sun, C., He, W., Ge, W. and Chang, C. (2017) Adaptive Neural Network Control of Biped Robots. *IEEE Trans. Syst. Man Cybern. Syst.* **47**(2):315–326.

Wang, S. and Huang, J. (2019) Adaptive Leader-Following Consensus for Multiple Euler–Lagrange Systems With an Uncertain Leader System. *IE EE Trans. Neural Netw. Learn. Syst.* **30**(7):2188–2196.

Wang, X., Liu, W., Wu, Q. and Li, S. (2022) A modular optimal formation

control scheme of multi-agent systems with application to multiple mobile robots. *IEEE Trans. Ind. Electron.* 69:9331–9341.

Wen, L., Liu, Y. and Li, H. (2022) CL-MAPF: Multi-agent path finding for car-like robots with kinematic and spatiotemporal constraints. *Robot. Auton. Syst.* 150:103997.

Yang, H., Qi, J., Miao, Y., Sun, H. and Li, J. (2019) A New Robot Navigation Algorithm Based on a Double-Layer Ant Algorithm and Trajectory Optimization. *IEEE Trans. Ind. Electron.* **66**(11):8557–8566.

Yin, J., Luo, D., Yan, F. and Zhuang, Y. (2022) A novel lidar-assisted monocular visual SLAM framework for mobile robots in outdoor environments. *IEEE Trans. Instrum. Meas.* 71:1–11.

Zhang, H., Zhao, W., Xie. X. and Yue, D. (2024) Dynamic Leader–Follower output containment control of heterogeneous multiagent systems using reinforcement learning. *IEEE Trans. Syst. Man Cybern. Syst.* **2024**:1–10.

Zhao, K. and Ning, L. (2022) Hybrid navigation method for multiple robots facing dynamic obstacles. *Tsinghua Sci. Technol.* 27:894–901.