IJRCS

International Journal of Robotics and Control Systems
Vol. 5, No. 3, 2025, pp. 1866-1885
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

# Autonomous Mobile Robots Path Planning with Integrative Edge Cloud-Based Ant Colony Optimization

Siti Nur Lyana Karmila Nor Azmi [a,1,*], Nur Ilyana Anwar Apandi [a,2,*], Majid Rafique [a,3], Nor Aishah Muhammad [b,4]

[a] Faculty of Technology and Electrical Engineering, Universiti Teknikal Malaysia Melaka, Hang Tuah Jaya, 76100 Durian Tunggal, Melaka, Malaysia
[b] Telecommunication Software and System, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, Johor, Malaysia
[1] lyanakamila.msc@gmail.com; [2] ilyana@utem.edu.my; [3] majidrafique777@gmail.com; [4] noraishahm@utm.my
* Corresponding Author

ARTICLE INFO

**Article history**
Received April 21, 2025
Revised July 01, 2025
Accepted August 13, 2025

**Keywords**
Autonomous Mobile Robot;
Ant Colony Optimization;
Dynamic Environment;
Edge Cloud-Based;
Path Planning

ABSTRACT

In recent years, Automated Mobile Robots (AMRs) have gained significant attention in industry and research applications, requiring efficient path-planning algorithms to optimize task performance. While widely adopted, conventional Ant Colony Optimization (ACO) algorithms suffer from low convergence rates and delays in task execution, particularly in dynamic environments due to insufficient exploration of this context. However, traditional Ant Colony Optimization (ACO) algorithms, widely used for AMR path planning, exhibit limitations such as low convergence rates and redundant recalculations, particularly in environments with frequently changing obstacles. To address these challenges, this study proposes an Integrative Edge Cloud-Based Ant Colony Optimization (IECACO) algorithm. IECACO incorporates a novel path retrieval mechanism and edge cloud computing infrastructure to minimize redundant path computation and improve convergence efficiency. The proposed algorithm is tested within a simulated 2D occupancy grid environment using both a 4×4 map for controlled experiments and a 20×20 map for comparative evaluation against a prior Improved ACO (IACO) study. Experimental simulation results, based on 50 independent runs in settings, demonstrate that IECACO achieves at least 4.76% reduction compared to traditional ACO. Based on the observation of 10 independent runs between IECACO and IACO, IECACO leading a significant reduction in both static and dynamic settings. Although this study is conducted in a simulated environment, the findings lay a foundation for future real-world implementations.



## 1. Introduction

Autonomous Mobile Robots (AMRs) represent a pivotal advancement in modern automation, enabling intelligent and autonomous navigation within dynamic and unstructured environments. Unlike Automated Guided Vehicles (AGVs), which require predefined tracks or markers, AMRs integrate advanced perception systems, localization, and intelligent decision-making capabilities to interpret and respond to real-time changes [1]-[8]. These robots are increasingly deployed across

industrial sectors, particularly in logistics and smart factories, to support flexible and efficient material handling. However, their successful operation relies heavily on responsive decision-making and robust navigation strategies, particularly under time-sensitive and uncertain conditions [1], [9]-[11].

A core component of AMR navigation is path planning, which involves determining an optimal trajectory from a start to a goal point while avoiding obstacles. While path planning in static environments where obstacle locations remain fixed that have been extensively explored using classical methods such as A* [12]-[16] and Dijkstra [17]-[20], these approaches often fall short in dynamic scenarios where the environment changes over time. In dynamic environments, the unpredictable nature of obstacles requires not only reactivity but also computational efficiency to ensure real-time operability. Many past studies have focused solely on static path planning [9]-[20], thereby limiting the applicability of these approaches in real-world settings where dynamic conditions prevail.

A critical requirement for AMRs is the ability to make fast and accurate decisions in uncertain and dynamically changing environments. In this study, edge cloud architecture is capable to enables AMRs to offload computation tasks such as path planning and sensor data processing to edge nodes while maintaining seamless communication with the cloud services. This system improves decision-making efficiency, reduces latency, and improves overall system responsiveness, which are important for time-sensitive path planning in dynamic settings. By implementing edge cloud computing, AMRs can achieve a balance between immediate responsiveness and large-scale data analysis, with improved adaptability in changing environments. Although prior studies [12], [18] have demonstrated the general advantages of edge computing in robotic systems, its full potential remains underexplored particularly in its capacity to enable concurrent execution of multiple computational processes, such as parallel path planning and retrieval mechanisms, which could alleviate delays in dynamic navigation tasks.

While previous studies [10], [21]-[24] may have explored the use of edge computing to accelerate path planning through distributed computation and reduced latency, these approaches typically focus on improving raw processing speed [10], [11] or supporting hybrid algorithm implementations [24], [25]. However, they often lack a mechanism to retain and reuse prior solutions once a path has been computed, resulting in redundant calculations when facing recurring environmental patterns. The Improved ACO (IACO) for wireless robot path planning utilizes an edge computing framework and a multi-step planning approach to improve solution quality and reduce reliance on single-step searches [21]. Although this method introduces inflection points for smoother navigation and uses edge nodes to handle computational load, it still requires repeated recalculations when obstacle conditions change, which may increase execution time in highly dynamic settings. The path optimization techniques combining ACO with reinforcement learning [26]-[28] improve convergence speed and adaptability, but the lack of a retrieval mechanism results in inefficiencies when facing recurring environmental patterns. The fast two-stage ACO algorithm [29]-[36], which adds a preliminary scent-based exploration phase, accelerates the search process but does not address reusability of prior paths, leading to redundant exploration in evolving environments. In contrast, the proposed approach introduces a novel integration of a global path retrieval mechanism within an edge cloud infrastructure, allowing previously computed optimal paths to be stored and reused efficiently. This not only accelerates decision-making but also enhances adaptability in dynamic environments by reducing the need for path planning recalculation

Among bio-inspired optimization methods, Ant Colony Optimization (ACO) has shown notable success in tackling path planning problems by simulating the foraging behavior of ants and utilizing pheromone trails to guide solution discovery [25], [37], [38]. Despite its adaptability, traditional ACO suffers from slow convergence, high computational cost, and excessive redundancy due to repeated path recalculations, especially when applied to dynamically changing environments [25], [33], [37], [39]. While hybrid approaches, such as ACO with Genetic Algorithm (GA) [36], [39], [40] or Particle Swarm Optimization (PSO) (41–43), have been proposed to address some of these issues, most still

rely on complete re-planning in each iteration, resulting in inefficiencies in real-time robotic navigation [41]-[44].

To overcome the redundant computation observed in traditional ACO methods, path retrieval mechanisms have been introduced to reuse previously computed optimal paths [9], [11], [44]. However, as an example in [10], [11], such retrieval strategies are implemented in isolation and are not fully integrated into broader edge-cloud architectures. Moreover, many of these methods do not allow real-time querying or updating of stored paths, limiting their effectiveness in highly dynamic environments where obstacle positions change frequently.

Motivated by the limitations mentioned above, this study proposes an improved ACO algorithm integrated with edge cloud computing and a novel path retrieval mechanism which is also known as IECACO (Integrative Edge Cloud-based ACO). In this study, the proposed algorithm is simulated within a 2D occupancy grid map environment, where AMRs must navigate through static and dynamic scenarios. This setup reflects realistic spatial constraints and challenges encountered in real-world autonomous navigation. The key contributions of this research are as follows:

1. An improved ACO algorithm with path retrieval capabilities which allows reuse of successful past paths and reducing redundant computations. Unlike prior multi-step or hybrid ACO approaches [21], [45], [46] which typically replan paths at each iteration or rely solely on pheromone memory. Instead, this study approach introduces a global path storage component capable of retrieving previously computed optimal solutions. This retrieval system, executed concurrently via edge cloud infrastructure, enables selective reuse of validated paths when encountering recurring obstacle configurations.
2. Edge cloud-based execution, enabling concurrent computation of ACO and retrieval processes for lower latency and improved responsiveness. Despite hybrid ACO variants, low convergence and redundant computations persist in dynamic environments, exacerbated by underutilized edge-cloud parallelism
3. The improved ACO of the proposed IECACO algorithm in both static and dynamic environments, using a grid-based simulation model with variable obstacle conditions to benchmark against conventional ACO and recent improved ACO models such as [21].

## 2. Methodology

### 2.1. Workspace Model

The workspace for the AMR in this study is structured as a 2-dimensional Occupancy Grid Map, which serves as a discrete representation of the operating environment. The grid-based model enables the AMR to autonomously navigate through the workspace while avoiding obstacles and optimizing its path toward a predefined goal. Each grid in the workspace is sequentially numbered from left to right and top to bottom, forming a structured coordinate indexing system.

The AMR is programmed to navigate forward in 2 distinct environment which is static and dynamic environment. In the static environment, the configuration of the grid remains unchanged throughout the simulation. Specifically, the walls cells and free cells are predefined. This setup provides a controlled environment to benchmark the path planning algorithm without the influence of variability. In contrast, the dynamic environment introduces variability by modifying the positions of wall cells and free cells in every execution or run. For each run, a new configuration is generated where wall cells or obstacle cells are randomly reallocated within the free cells. The dynamic environment is introduced to validate the adaptability of the proposed algorithm, ensuring it can respond effectively to changing obstacle positions across different runs of the system.

The occupancy grid framework facilitates efficient path computation and real-time obstacle avoidance by classifying each cell in the grid as either free or occupied. This binary representation of the environment as illustrated in Fig. 1, enables the AMR to make localized and informed decisions

about movement at each step of its navigation process. The computational details of the static and dynamic environments are further elaborated in Subsections 2.1.1. and Subsections 2.1.2., respectively.

### 2.1.1. Static Environment Setup

To construct a static environment 2-D occupancy grid map in this study, we let N represents the total number of grid cells per column in a square occupancy N × N grid map and s is the sequence serial index for a specific cell, given by.

$$s = 1,2,\dots,N,N+1,\dots,2N,2N+1,\dots 3N,\dots N^2 \quad (1)$$

To enable the path computation on a 2D plane, let x and y represent the column and row coordinates of the grid cell, respectively. Thus, the serial index is converted into its corresponding Cartesian coordinate $(x,y)^s$ given by.

$$(x,y) = \left(mod\left(\frac{s}{N}\right) - 0.5, N - ceil\left(\frac{s}{N}\right) + 0.5\right) \quad (2)$$

where the modulo function, $mod(.)$ represents the remainder of the division $\frac{s}{N}$, indicating the column position of the cell and the ceiling function, $ceil(.)$ rounds up to the nearest integer used to determine the row position, as shown in Fig. 1. Fig. 1 (a) presents a specific example using a 4 × 4 grid, while Fig. 1 (b) generalizes this structure for any grid size, highlighting how serial indices increment row by row from bottom to top.

| | | | |
|---|---|---|---|
| 13 | 14 | 15 | 16 |
| 9 | 10 | 11 | 12 |
| 5 | 6 | 7 | 8 |
| 1 | 2 | 3 | 4 |

(a)

| | | | |
|---|---|---|---|
| 3N+1 | ... | ... | $N^2$ |
| 2N+1 | ... | ... | 3N |
| N+1 | ... | ... | 2N |
| 1 | ... | ... | N |

(b)

**Fig. 1.** The mapping of grid index, $s$ (a) general and (b) in 4 × 4 grid map

### 2.1.2. Dynamic Environment Setup

For this study, the simulation is conducted under two different dynamic environment setups. Environment 1 is a $4 \times 4$ grid map which the dynamic version of the static setup as shown in Subsection 2.1.1. This setup is to observe the algorithmic behavior in a simplified scenario. Followed by the Environment 2, a $20 \times 20$ grid map as shown in Fig. 2, which is configured to match the setup used in [21] and this setup is equivalent to a $20m^2$ warehouse, with addition of two obstacle points $Obs_p$ within the free cells that dynamically change its location per execution. This configuration enables a direct performance comparison of the proposed algorithm, IACO [21] and ACO under consistent conditions that will be further explained in Section 4.

To model Environment 1, which implement the dynamic changes in the placements of the obstacle, a randomized assignment function is used to reconfigure wall cells at each simulation run which denoted as $r$. Let $G$ be an $N \times N$ occupancy grid, where each grid is indexed by its 2D coordinates $(x,y)$. The possible values of $g(x,y)$ are define as $g(x,y) \in \{0,1,2,3,4\}$ and as detailed in Table 1. The set of wall cells for given run $r$, both static and dynamic, is defined as (3).

**Table 1.** Cell unit representation

| $g(x,y)$ | Cell Representation |
|---|---|
| $g(x,y)=0$ | Free |
| $g(x,y)=1$ | Barrier/wall |
| $g(x,y)=2$ | Start point |
| $g(x,y)=3$ | Goal point |
| $g(x,y)=4$ | Dynamic obstacle |

The set of wall cells for given run $r$, both static and dynamic, is defined as

$$W_r = \{(x,y) \mid (x,y) \in G, g(x,y) = 1\} \quad (3)$$

Let $W_r$ denotes the set of coordinates within the grid where wall cell is $g(x,y)=1$ are assigned for the current simulation run. The subscript $t$ in $g(x,y)$ captures the time-varying nature of the environment, allowing the simulation to reflect real-time changes.

To model Environment 2, the $20 \times 20$ grid occupancy map as shown in Fig. 2, (4) presents a modified formulation of (4), which is then derived as.

$$(x,y) = mod(s-1,N)+1, N - ceil\left(\frac{s-1}{N}\right)+1 \quad (4)$$

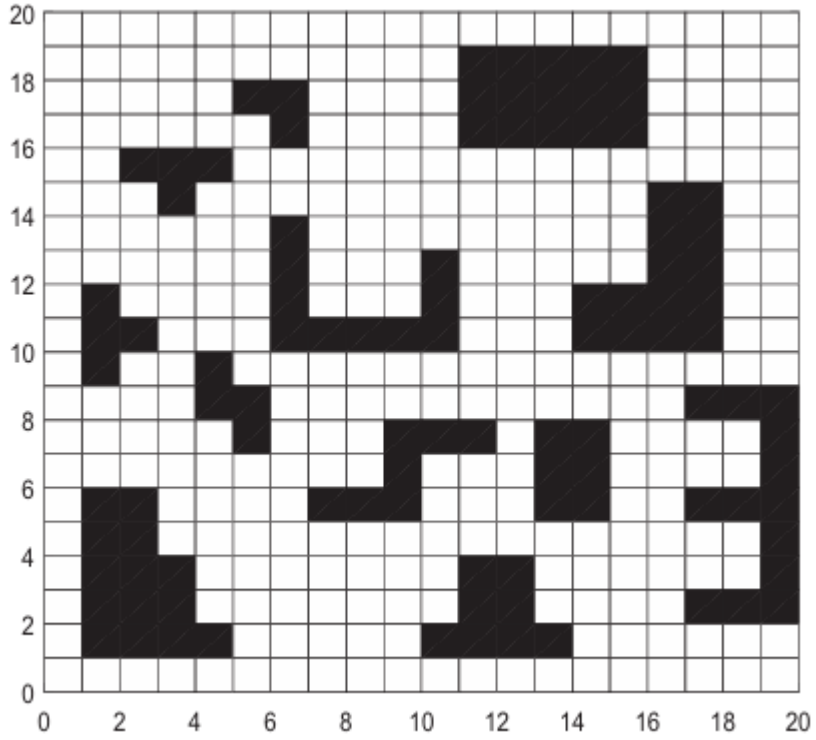


**Fig. 2.** The $20 \times 20$ occupancy grid model for environment 2

## 2.2. Ant Colony Optimization Algorithm

Many existing research on ACO algorithms for path planning in AMR has focused on the natural foraging behavior of ants. Ants deposit pheromones, which serve as a communication mechanism to reinforce the shortest path between their nest and a food source. Conventional ACO studies have primarily focused on static environments, limiting their applicability in real-world dynamic environment. In dynamic conditions, obstacles often appear and disappear, requiring the algorithm to adapt. However, the fundamental selection mechanism remains crucial in path optimization, as it depends on pheromone concentration and evaporation rate [9], [25], [37], [38], [45], [47]-[49].

The pheromone updating mechanism is central to the optimization process of ACO. Let the terms $\Delta\tau_{ij}$ is the amount of newly deposited pheromone which can be written as.

$$\Delta\tau_{ij} = \begin{cases} \frac{Q}{L_k} & \text{, if ant k travels on edge } (i,j) \\ 0 & \text{, otherwise} \end{cases} \quad (5)$$

where $Q$ is a constant representing the total amount of pheromone deposited per iteration and $L_k$ is the length of the path constructed by the ant $k$. When an ant moves from node $i$ to node $j$, it deposits a pheromone trail $\tau_{ij}$ [37], [50]. The amount of pheromone is updated given by.

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij} \quad (6)$$

where the $\rho$ is the pheromone evaporation rate is constrained within the range $(0 < \rho < 1)$ to prevent the unlimited accumulation of pheromone [51]. Based on (6), the probability $P_{ij}^{k}(t)$ that ant $k$ will move from node $i$ to node $j$ at time is determined by the pheromone intensity $\tau_{ij}(t)$ and the heuristic desirability $\eta_{ij}$ of the edge is given as

$$P_{ij}^{k}(t) = \begin{cases} \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{\iota\epsilon N_i}[\tau_{ij}(t)]^{\alpha}[\eta_{i\iota}]^{\beta}} & \text{if } j\epsilon N_i \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

where $\eta_{ij}$ is the heuristic desirability of edge $(i,j)$, usually defined as the inverse of the Euclidean distance. Meanwhile, the terms $\alpha$ and $\beta$ are the parameters that control the relative influence of pheromones and heuristic information, respectively, and $N_i$ is the set of neighboring nodes. The analysis of the traditional ACO algorithm highlights its fundamental principle where the pheromone level in a path is inversely linked to its length. Specifically, shorter paths accumulate higher pheromones, increasing their probability of selection in subsequent iterations.

## 3. Integrative Edge Cloud-ACO (IEACO) Algorithm

### 3.1. System Architecture

This study proposes the Integrative Edge Cloud-ACO (IEACO) Algorithm, which is designed to optimize AMR path planning systems, as illustrated in Fig. 3. IECACO introduces a novel approach to path planning by implementing the retrieval influence factor, $R_{ij}(t)$, which distinguishes between previously stored paths and newly computed routes. This feature reduces redundant computations and improves calculation in (7). IECACO uses dynamic evaluation and updating pheromone trails along with a path retrieval mechanism that interacts with global path storage, $R$ in the edge cloud. It determines if an optimal stored path can be reused when workspace environment changes, minimizing computational costs. If no suitable path exists, a new one is computed and stored for future reference.

Based on Fig. 3, following the initialization phase, the ant iteration phase is executed. This phase applies the ACO algorithm to explore the environment and construct potential paths from $S_p$ to $G_p$. The decision-making process during this phase relies on the pheromone update rule and heuristic function, which determine the probability of selecting a specific path. If path retrieval is needed, the system transitions to the path retrieval phase, which interacts with $R$. This module stores previously computed paths, allowing for rapid retrieval and reducing redundant path recalculation. If a precomputed path is available, it is retrieved and followed. Otherwise, the algorithm recalculates an alternative trajectory while updating the pheromone levels accordingly. Once the optimal path is determined, it is stored as $P_{best}$, representing the best solution derived from the ACO-based iterative process. This finalized path is then executed by the AMR. The feedback loops to the $R$ further refine

the system's learning capability by continuously updating stored paths, improving future path selection.

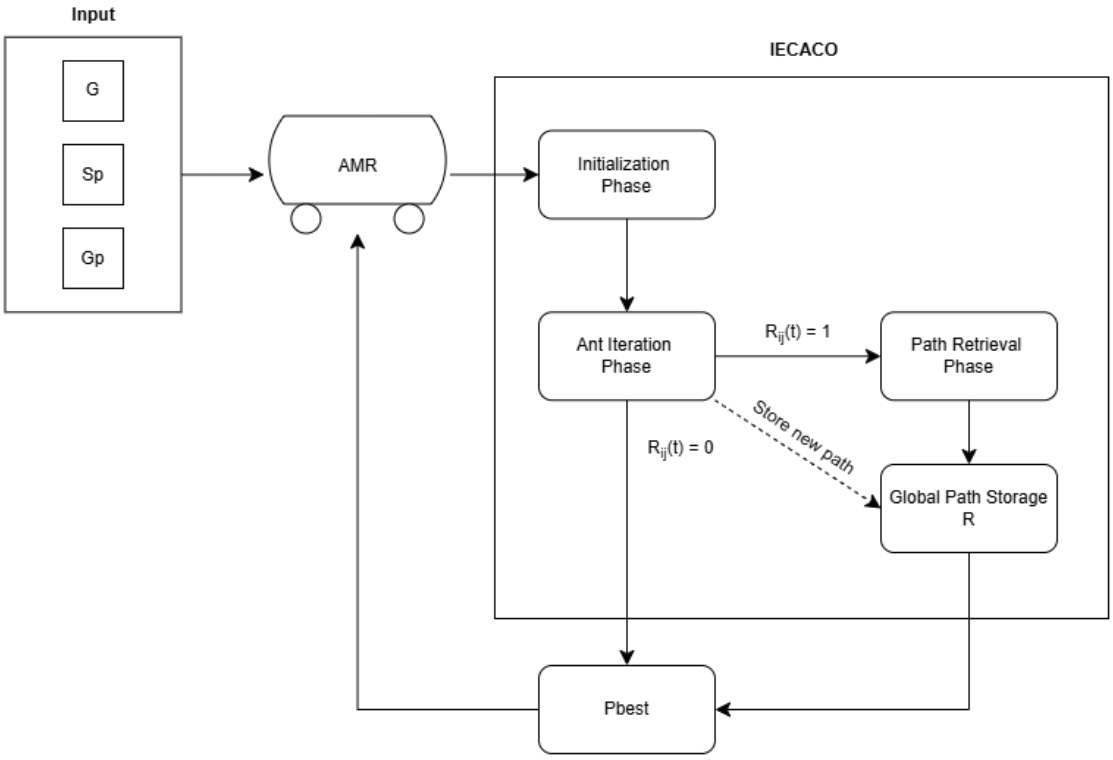


**Fig. 3.** The overall structure of IEACACO in AMR path planning system

The edge cloud framework in the proposed IECACO algorithm plays an important role in accelerating computational performance and enabling real-time decision-making. Specifically, the path retrieval mechanism and global path storage are offloaded to the edge cloud, allowing these processes to execute concurrently with ant iteration and pheromone update tasks running on the AMR's local processing unit. This architecture supports parallel computation, wherein the edge cloud handles precomputed path searches, storage updates, and retrieval influence calculations without interrupting or delaying the navigation of the AMR. As a result, latency is minimized, and the AMR avoids the need to reprocess path planning from scratch, particularly in scenarios where similar obstacle conditions have been encountered before. This reduces redundant computation and leads to faster convergence toward optimal paths.

To meet the key contributions (i) of this study, we have decided to include the path retrieval process into the ACO framework. The path retrieval process involves the mechanisms by which an AMR identifies, accesses, and follows predetermined or dynamically generated paths within its operational environment. This process is crucial for ensuring that robots can navigate efficiently, avoid obstacles, and adapt to the time-varying nature of the environment, where obstacle configurations are updated with each simulation run. In dynamic environments, integrating path retrieval mechanism becomes essential. Such a mechanism allows the AMR to implement stored optimal paths for common routes, reducing the need for real-time computation. Recent improvements have highlighted the importance of effective path retrieval in AMRs. For instance, [52]-[54] emphasizes the need for adaptive path planning and obstacle avoidance methods to enhance autonomy in mobile robots.

### 3.2. The Initialization Phase

The Initialization Phase is the first step of the IECACO ensuring that all parameters are appropriately configured before advancing to the Ant Iteration Phase. This phase is important to define the grid-based workspace, pheromone levels, and the initial conditions.

The Initialization Phase as the flowchart shown in Fig. 4, involves defining the grid map $G$, which serves as the navigation environment for the AMR. This grid is structured as a two-dimensional occupancy map, where each cell is categorized. The grid-based representation ensures that the ACO algorithm has a clear, structured workspace for computing and optimizing navigation paths. Additionally, the starting position $S_p$ the goal position $G_p$ are pre-defined, serving as reference points for the pathfinding process. Once the environment is structured, the algorithm initializes the pheromone distribution across all traversable edges. The pheromone levels, denoted as $\tau_0$, are uniformly assigned at the beginning of the simulation. This uniform pheromone distribution is essential to ensure unbiased exploration in early iterations, particularly in dynamic environments where obstacles may shift over time. Following pheromone initialization, ants are placed at the starting point $S_p$. These ants function as search agents within the ACO framework, navigating through the grid-based environment to identify the shortest and most efficient path to the goal position $G_p$. If any discrepancies are detected, the system reinitializes the affected parameters to maintain accuracy and stability. This step prevents misconfigurations and ensures smooth execution of the IECACO path planning system. Once the initialization phase is completed, the system transitions into the Ant Iteration Phase which is labelled as A in the Fig. 4, where the core path optimization process begins.

### 3.3. The Ant Iteration Phase

Fig. 5 shows the flowchart of the Ant Iteration Phase in IECACO. The execution begins immediately after the Initialization Phase (label A) and may interface with the Path Retrieval Phase (Label B and C). The Ant Iteration Phase is a fundamental stage in the IECACO algorithm, where virtual ants actively explore the grid-based environment to identify an optimal path from the point $S_p$ to the $G_p$. This phase builds upon the initialized pheromone distribution and heuristic values, guiding ants' movements through a probabilistic decision-making process. Unlike traditional ACO, which relies solely on pheromone reinforcement, the proposed IECACO improves path planning efficiency by integrating global path storage and dynamic obstacle avoidance mechanisms.

Each ant begins at the $S_p$ and moves iteratively through neighboring nodes by evaluating pheromone levels, $\tau_{ij}$ and heuristic desirability, $\eta_{ij}$. The probability of selecting the next node is determined by the state transition probability function can be expressed as

$$P_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}]^{\beta}}{\sum_{l\in N_i}[\tau_{il}(t)]^{\alpha}[\eta_{il}]^{\beta}} \tag{8}$$

Let $\eta_{ij}(t)$ is heuristic information which represents the desirability if moving from node $i$ to node $j$, often calculates as.

$$\eta_{ij}(t) = \frac{1}{d_{ij}} \tag{9}$$

where, $d_{ij}$ is the Euclidean distance between nodes, promoting shorter paths. Meanwhile, the path selection of this phase, $\eta_{il}$ is denoted as the heuristic desirability of the edge between node $i$ and node $l$, often inversely proportional to distance. If a neighboring node $l$ has a high pheromone level and heuristic desirability, it will increase the denominator, reducing the probability of choosing the node $j$. Conversely, if the neighboring nodes have low desirability, the probability of selecting $j$ increases. As ants navigate, pheromone levels are dynamically updated to reflect changes in the environment. IECACO modifies the classic pheromone update rule to account for dynamic obstacles, ensuring that blocked paths lose influence while newly opened paths gain reinforcement. As ants traverse the grid, the pheromone update rule in a dynamic environment can be expressed as.

$$\tau_{ij}(t+1) = (1-\rho)\cdot\tau_{ij}(t) + \Delta\tau_{ij}(t) + \delta_{ij}(t) \tag{10}$$

where, $\rho$ is the pheromone evaporation rate and $\Delta\tau_{ij}(t)$ the pheromone reinforcement for paths successfully used by ants. Meanwhile, the adjustment factor $\delta_{ij}(t)$ is a binary indicator function that denotes whether the edge between node $i$ and $j$ is part of the best path found by an ant during iteration It is used in the pheromone update rule to selectively reinforce only those edges that contribute to a successful or optimal solution. Mathematically, it is defined as

$$\delta_{ij}(t) = \begin{cases} -\tau_{ij}(t), & \text{if path is blocked} \\ 0, & \text{if path is remain unchanged} \\ \Delta\tau_{ij}(t), & \text{if path is newly opened} \end{cases} \quad (11)$$

If an obstacle is newly encountered, the ant avoids it and recalculates a new path. If an obstacle was previously known, the system proceeds to the Path Retrieval Phase (Label B) to check for stored solutions. And if no stored solution exists, the system computes a new path using standard ACO principles.

### 3.4. The Path Retrieval Phase

The Path Retrieval Phase in the proposed IECACO system enhances computational efficiency by utilizing previously computed paths stored in the global path storage $R$, hosted within the edge cloud infrastructure. The flowchart representing the operations within this phase is shown in Fig. 6. At the beginning of the phase, triggered from the Ant Iteration Phase (label B), the system checks the $R$ for any previously stored paths that match the current configuration. If a matching path is found, it is retrieved and immediately applied to the AMR, bypassing the need for a full re-execution of the Ant Iteration Phase. This significantly reduces computational cost and time, particularly when dynamic obstacles cause only minor environmental changes. If no suitable path is found, the system reverts to standard ACO path planning logic (label C). In both cases, retrieved paths are temporarily stored in local memory, allowing faster access for repeated queries within the same simulation run. This temporary storage avoids redundancy and accelerates execution without compromising accuracy.

This phase minimizes redundant calculations and ensures that the ACO algorithm operates efficiently in dynamic environments. At the beginning of this phase, the system queries the $R$ for any previously computed paths matching the current start and goal point. The retrieved paths are directly incorporated into the ant's movement strategy:

- If a matching path is found, it is retrieved and applied immediately, by passing the need for a full iterative process.
- If no stored path exists, the system defaults to standard ACO path computation, ensuring that the AMR can still operate effectively in unknown scenarios.
- All retrieved paths are temporarily stored in local memory, ensuring faster access for repeated queries within the same iteration.

Importantly, the path solution updates to the global path storage are handled asynchronously. Once a new path is computed, it is offloaded to the edge cloud without interrupting the ongoing iteration cycle. To incorporate past solutions effectively, IECACO integrates a retrieval influence factor $R_{ij}(t)$ into the decision-making and pheromone update process. $R_{ij}(t)$ represents the retrieval influence factor, indicating whether the path should be retrieved. This factor reflects the reliability or frequency of a previously stored path between nodes $i$ and $j$, as recorded in the global path storage. $R_{ij}(t)$ can be defined as.

$$R_{ij}(t) = \begin{cases} 1, & \text{path retrieval is needed} \\ 0, & \text{otherwise} \end{cases} \quad (12)$$

As $R_{ij}(t)$ increases, the probability $P_{ij}(t)$also increases, ensuring that paths stored in global storage are prioritized when they demonstrate high reliability and effectiveness. By integrating $R_{ij}(t)$

into the pheromone update rules, the algorithm reinforces the retrieved path's pheromone levels based on its utility, promoting these paths in future iterations. Let $\alpha, \beta$ and $\gamma$ are weight parameters controlling pheromone, heuristic, and retrieval influence, respectively. The state transition probability for an ant moving from node $i$ to node $j$ is adjusted as.

$$P_{ij}(t) = \frac{[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[R_{ij}(t)]^{\gamma}}{\sum_{k\in N_i}[\tau_{ij}(t)]^{\alpha}[\eta_{ij}(t)]^{\beta}[R_{ij}(t)]^{\gamma}} \quad (13)$$

where, $\tau_{ij}(t)$ denotes as pheromone level on edge $(i, j)$, in the equation represents the reinforcement of previously traveled paths which explains the higher pheromones levels indicates frequently used or optimal paths. In IECACO, when a path is retrieved, its pheromone level is updated to reinforce its usefulness in future iterations. The pheromone update rule is modified to include the retrieval influence factor as in (14). Let $\lambda$ is a scaling factor controlling retrieval influence. By integrating past solutions into pheromone updates, IECACO ensures that effective paths retain their influence while still allowing new exploration when needed.

$$\tau_{ij}(t+1) = (1-\rho)\tau_{ij}(t) + \Delta\tau_{ij}(t) + \lambda R_{ij}(t) \quad (14)$$

These two equations fully capture the retrieval mechanism's role in path evaluation and reinforcement $\gamma$ controls the strength of retrieval influence during decision-making, while $\lambda$ governs its weight in pheromone reinforcement. This integration reduces unnecessary computation and encourages the reuse of successful prior paths, thus improving convergence speed in dynamic environments.

This $\tau_{ij}(t+1)$ in this phase is different compared to an Ant Iteration Phase, where it only reflects on retrieval confidence in (12). The update rule is implemented during or after path retrieval when a previously computed path is reused from $R$. The update reinforces successful stored paths to encourage reuse in similar future conditions. Meanwhile in Ant Iteration Phase, the update rule is implemented during ant exploration especially in dynamic environments where new obstacles appear or paths open up.

The proposed IECACO algorithm which integrates ACO approach with a path retrieval mechanism is outlined in Algorithm 1. This algorithm outlines the step-by-step execution of the method, including the pheromone update procedures and the interconnection between each phase. Additionally, Algorithm 1 illustrates how the associated formulations are integrated throughout the process. The path retrieval mechanism is embedded within the edge cloud framework, enabling access to a Global Path Storage module.

## 4. Results and Discussion

The proposed algorithm is developed and simulated in the environment of MATLAB 2023b and a PC with a 1.19GHz Intel ® Core™ i5-1035 processor with 12GB memory. The operating system device is Windows 11, and an NVIDIA GeForce MX330 GPU was employed to support computational tasks and visualization. This simulation setup provided a stable and consistent environment for assessing the computational efficiency of IECACO against the baseline ACO algorithm.

### 4.1. Initial Experimental Setup

For the initial comparative analysis, both the traditional ACO and proposed IECACO algorithms were evaluated on a simplified 4 × 4 grid map. The experimental parameters are presented in Table 2. The IECACO algorithm was tested with four different parameter configurations, varying heuristic information and pheromone strength. The selection of these parameter values is based on ranges and tuning strategies adopted in prior studies on ACO and its variants, such as those presented in [10], [21] and [22]. This allowed for a detailed analysis of their impact on convergence speed and

computational efficiency. The study aimed to determine an optimal balance between heuristic guidance and pheromone-based reinforcement.

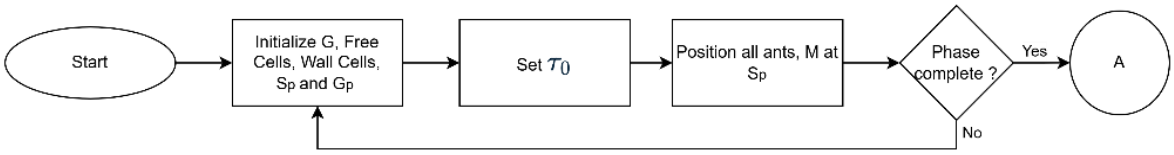


**Fig. 4.** The flowchart of initialization phase of IECACO

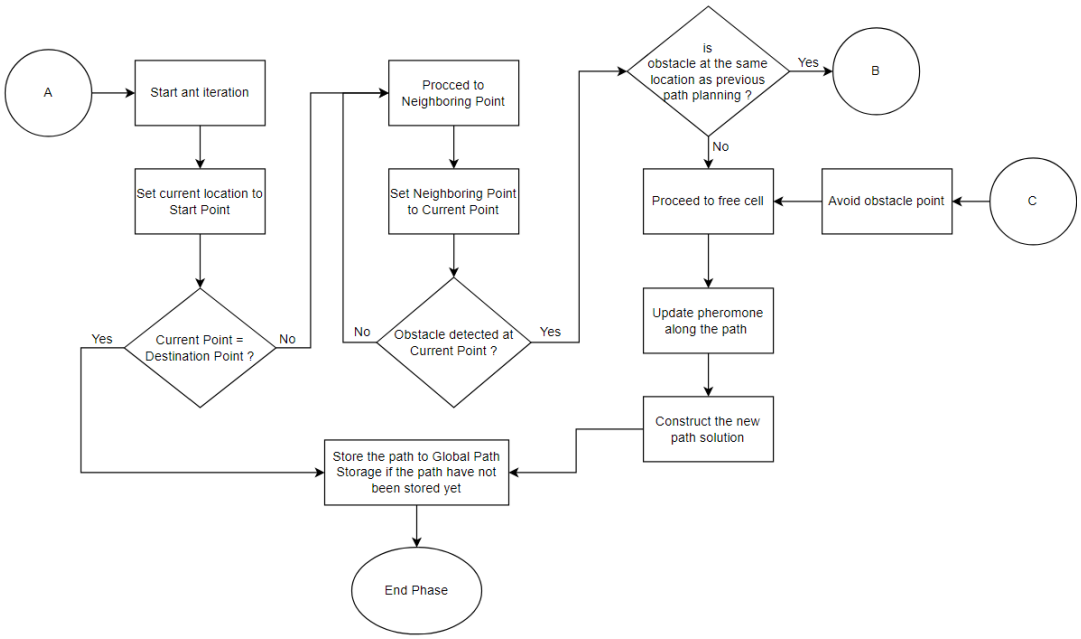


**Fig. 5.** The flowchart of the ant iteration phase in IECACO

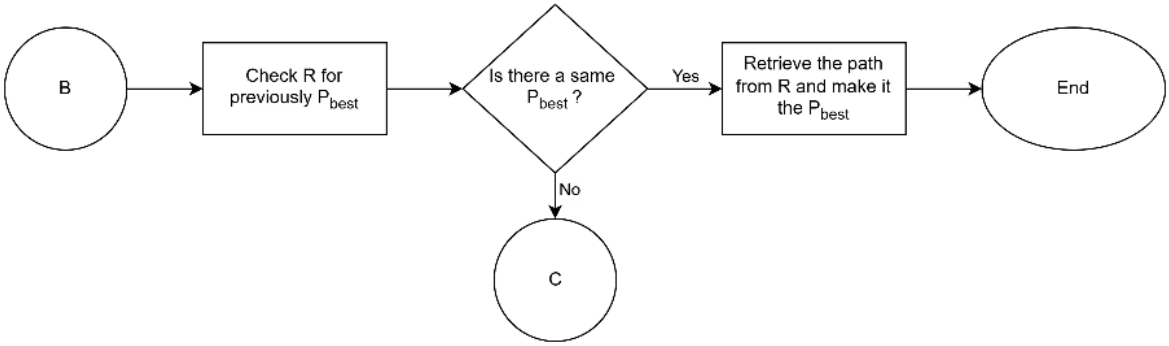


**Fig. 6.** The flowchart of the path retrieval phase of IECACO

**Table 2.** Experimental parameters values

| Variables Description | Variables | ACO [21] | IACO [21] | IECACO D1 [10] | IECACO D2 [10] | IECACO D3 [22] | IECACO D4 [21] |
|---|---|---|---|---|---|---|---|
| Influence of pheromone | α | 1 | 1 | 1 | 1 | 1 | 1 |
| Influence of heuristic information | β | 7 | 7 | 2 | 2 | 5 | 7 |
| Retrieval influence | γ | - | - | 1 | 1 | 1 | 1 |
| Scaling factor | λ | - | - | 0.5 | 0.5 | 0.5 | 0.5 |
| Pheromone evaporation rate | ρ | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 | 0.45 |
| Pheromone strength | *Q* | 400 | 400 | 100 | 200 | 100 | 400 |
| Number of ants | *M* | 50 | 50 | 50 | 50 | 50 | 50 |
| Maximum Number of Iterations | *I* | 100 | 100 | 100 | 100 | 100 | 100 |

The IECACO algorithm was evaluated against traditional ACO using quantitative metrics, including path quality, computational efficiency, and retrieval performance in static and dynamic environments, as presented in Table 3. The classical ACO lacks a path retrieval mechanism, resulting in no recorded data for path retrieval time, as classical ACO relies on recalculating paths in iterations.

**Algorithm 1: IECACO**

Input $G$:, $S_p$, $G_p$, $I$, $\eta$, $\tau$, $\alpha$, $\beta$, $\gamma$, $\rho$, $Q$, and $R$
Output: $P_{best}$, $L_k$
#Initialize Phase
Initialize $G$ with free cells and wall cells
Define $S_p$ and $G_p$
Initialize pheromone levels: set $\tau_{ij} = \tau_0$ for all edges $(i,j)$
Deploy ants $n$ at $S_p$
Initialize $R$
#Ant Iteration Phase
for iteration $t = 1{:}I$
for each ant $k = 1{:}n$
Set the current position $(x,y)$ to $S_p$
#Path Retrieval Phase
Check if a precomputed path exists in $R$ //Access $R$ via edge cloud
if a stored path exists, retrieve and follow $P_{best}$
Pheromone update (14) else
# Ant Iteration Phase
while $(x,y) \neq G_p$
Compute transition probability $P_{ij}(t)$
Move $k$ to node $j$ using (12)
if cell "1" is detected at $(i,j)$
if $(i,j)$ exist previously in $R$
Retrieve $P_{best}$ from $R$
else
Recompute the path to bypass the wall cell (8)
Update pheromone $\tau_{ij}(t+1)$ (10)
Store the $P_{best}$ in the iteration to $R$
End for each $n$
End $I$
Return $P_{best}$ $L_k$

**Table 3.** Evaluation metrics

| Metric | Details |
|---|---|
| Path Length (unit per cell) | Measures the total distance of the computed path ($L_k$) |
| Number of Iteration | The number of iterations required to achieve an optimal solution. |
| Computational Time (s) | The time taken to compute a feasible path |
| Path Retrieval Success Rate | Measures the success rate of the path retrieval of each simulation |
| Path Retrieval Time (s) | Measures the time taken to retrieve a previously computed path from Global Path Storage |

## 4.2. Statistics Analysis

To assess the statistical significance of the differences among the IECACO configurations, which are D1, D2, D3 and D4, a one-way ANOVA was performed on the experimental results obtained from 50 independent runs. The test was conducted to evaluate whether the mean values of key performance metrics path length, number of iterations, computational time, and retrieval time which is differ significantly across configurations. The null hypothesis $H_0$ was defined as

$$H_0: \mu_{D1} = \mu_{D2} = \mu_{D3} = \mu_{D4}$$

Table 4 shows the ANOVA summary table. The F statistic is 2.13, the F-critical is 3.01, and the p-value is 0.08. Therefore, since the value of F < F-critical and p > 0.05, we fail to reject the null hypothesis, indicating that the performance differences among configurations D1–D4 are not statistically significant. This confirms the consistency and stability of the IECACO algorithm.

**Table 4.** The ANOVA table

| Source of Variation | SS | df | MS | F | F-Critical |
|---|---|---|---|---|---|
| Between Groups | 0.51 | 3 | 0.17 | 2.13 | 3.01 |
| Within Groups | 15.92 | 46 | 0.35 | | |
| Total | 16.43 | 49 | | | |

## 4.3. Performance Evaluation

The evaluation of IECACO was conducted across 50 independent experimental runs to ensure reliability and consistency. Fig. 7 provides a visual comparison of the performance between the traditional ACO and the proposed IECACO across multiple performance metrics. In Fig. 7 (a), it is evident that IECACO consistently achieves convergence with fewer iterations than ACO. The

classical ACO displays a more erratic convergence behavior, requiring a significantly greater number of iterations to reach an optimal path solution. This difference highlights the efficiency of the path retrieval mechanism embedded in IECACO, which accelerates convergence, especially when reusable paths from global storage are available. The Fig. 7 (b) further supports this observation by showing that the paths generated by IECACO are consistently shorter than those produced by ACO. In Fig. 7 (c), the computational time for each method is compared, with ACO incurring a longer runtime due to repetitive computation and pheromone updates. Lastly, Fig. 7 (d) illustrates the path retrieval time for IECACO, highlighting the system's ability to reduce overall planning time through efficient reuse of previously computed paths.

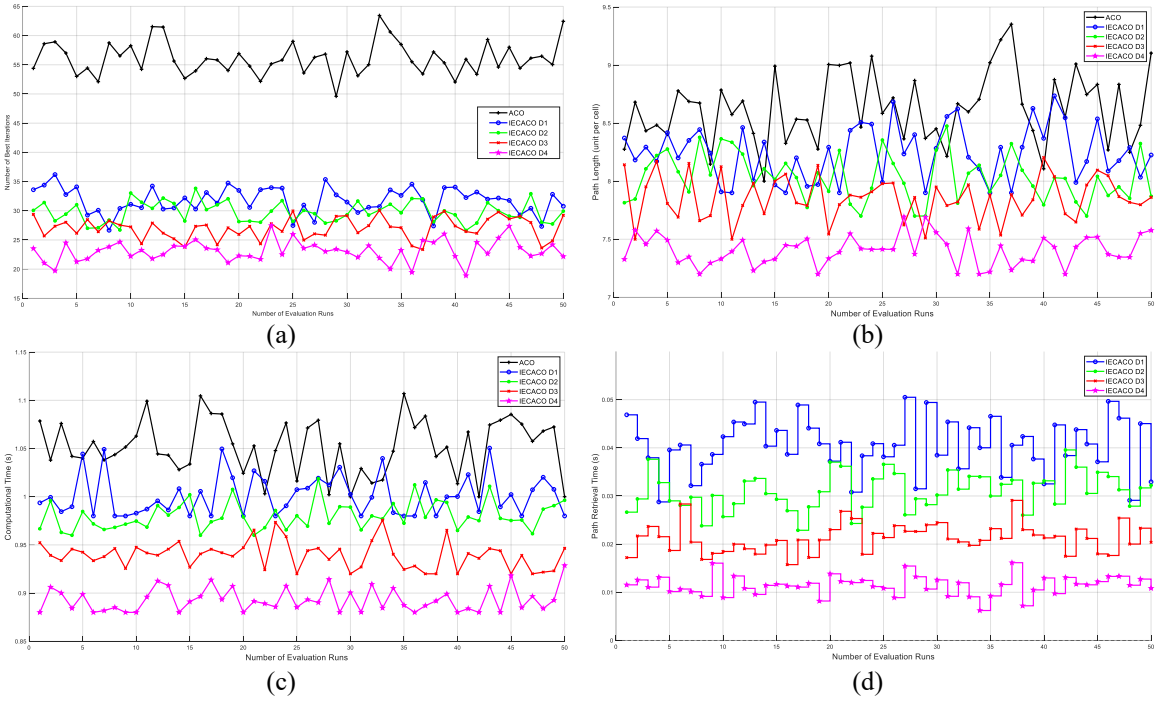


**Fig. 7.** The performance plot evaluations run against (a) number of iteration (b) path length (c) computational time (d) path retrieval time

Table 5 then compares the performance of the traditional ACO algorithm with different parameter values of the IECACO algorithm (D1-D4). Each metric is reported as Mean ± Standard Deviation, where the standard deviation (std) reflects how much the results varied across multiple simulation runs. These numerical insights reinforce the superiority of IECACO across all tested metrics, validating its suitability for efficient path planning in dynamic and resource-constrained environments.

**Table 5.** The average performance metrics across evaluation runs [1]

| Metric | ACO | IECACO | | | |
|---|---|---|---|---|---|
| | | D1 | D2 | D3 | D4 |
| Path Length (unit per cell) | 8.63 | 8.23+/-0.22 | 8.04+/-0.19 | 7.82+/-0.17 | 7.40+/-0.15 |
| Number of Best Iteration | 56 | 32+/-2.3 | 30+/-2.1 | 27+/-1.8 | 23+/-1.6 |
| Computational Time (s) | 1.05 | 1.00+/-0.06 | 0.98+/-0.05 | 0.94+/-0.04 | 0.89+/-0.03 |
| Reduction (%) | | 4.76 | 6.67 | 10.48 | 15.23 |
| Path Retrieval Time (s) | - | 0.041+/-0.004 | 0.031+/-0.22 | 0.022+/-0.002 | 0.011+/-0.002 |

## 4.4. IECACO Analysis

The performance improvements observed in IECACO compared to ACO can be attributed to the variations in key experimental parameters, as presented in Table 5. The tuning of these parameters

[1] The Mean ± Standard is computed based on the standard error $\sigma = \sqrt{\frac{1}{r-1}\sum_{n=1}^{r}(x_n - \mu)^2}$

directly influenced the efficiency of the path-planning process, particularly in terms of convergence speed, path length, and computational time.

The heuristic information parameter $\beta$, which govern the significance of heuristic values in decision-making, varied across different configurations. A higher $\beta$ value increases the algorithm's reliance on heuristic information, resulting in more informed and efficient path selection. This effect is evident in the shorter path lengths observed in IECACO. Furthermore, the pheromone strength $Q$ plays a crucial role in reinforcing optimal paths over multiple iterations where it employs progressively higher values across D1-D4, whereas ACO maintains at 100. Based on Table 5, D2 has lower heuristic information $\beta$ compared to D3. Higher $\beta$ increases the weight heuristic desirability, making the algorithm prioritize shorter and more direct paths. Meanwhile, in terms of $Q$, even though D3 has lower strength, it reinforces paths too strongly, potentially leading to premature convergence on suboptimal routes. This proves that D3 strikes a better balance between exploring new paths and exploiting previously successful ones. A higher $\beta$ integrate with a moderate value of $Q$, prevents excessive commitment to early solutions, allowing for more adaptive path refinement. This combination results in shorter paths, faster convergence, and reduced computational overhead, making D3 the superior configuration.

In this study, Path Retrieval Success is defined as the successful identification and reuse of a previously stored path from the global path storage $R$, that is remains valid under the current configuration of the environment. If no matching path is found or if the environmental changes invalidate the stored path, the system proceeds with a new computation using the IECACO procedure. The threshold for considering retrieval success varies based on the environment type. In static environments, a success rate exceeding 90% is considered high. In contrast, for dynamic environments, a success rate above 60% is deemed satisfactory due to the increased variability and complexity of the scenarios. Fig. 8 presents the Path Retrieval Success Rate across different runs. The results are evaluated based on (12), (14) where a value of 1 denotes a successful retrieval, and 0 indicates either a retrieval failure or a case where retrieval was unnecessary. The computed success rate percentage is plotted across multiple simulation runs. As shown in Fig. 8, the retrieval mechanism consistently achieved a 100% success rate in a static environment. Meanwhile, in dynamic settings, it maintained an average success rate of approximately 80%, confirming its robustness and adaptability in changing scenarios.

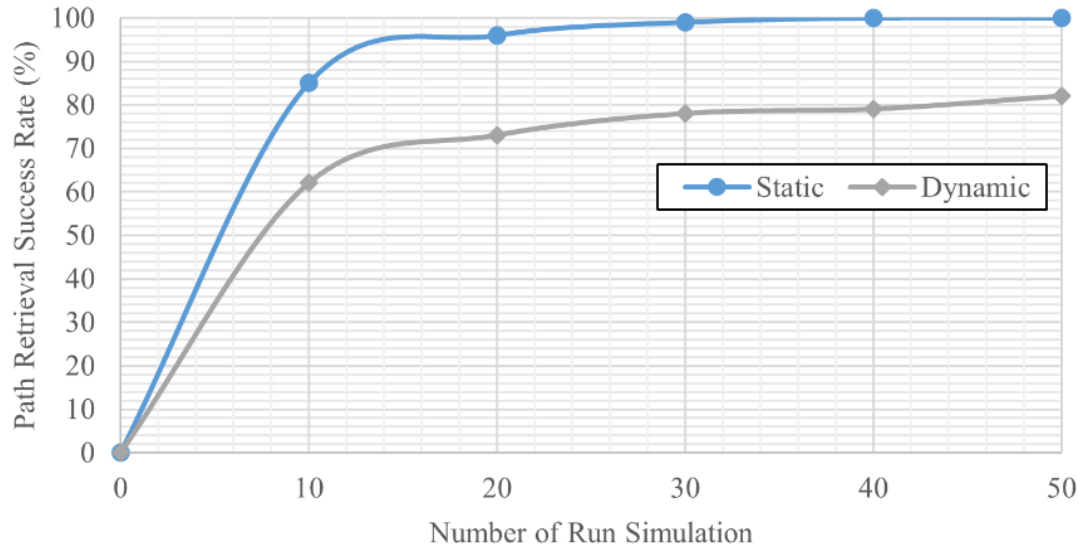


**Fig. 8.** The path retrieval success rate trends

### 4.4.1. Evaluation in Static Environment

To further assess the effectiveness of IECACO in a broader context, its performance is compared with the IACO [21], a multistep algorithm incorporating with edge cloud computing for path planning.

[21] aims for the shortest path with minimal execution time per iteration. This paper will validate the efficiency of IECACO, by conducting an additional simulation in a dynamic environment across 10 independent runs, demonstrating its adaptability and computational efficiency in dynamic scenarios in bigger workspace and different type of obstacles as described in Section 2.1.1. and Section 2.1.2. To ensure a fair and meaningful comparison with IACO method proposed in [21], this study adheres closely to the experimental setup described in that work and the key parameters such as α, β, ρ, and Q were kept identical across all algorithms as shown in Table 2.

For the result, Table 6 shows a depth comparison of the three algorithms in both environments. The table proves that IECACO outperforms both ACO and IACO across all three evaluation criteria. The convergence rate of IECACO is significantly faster IACO and ACO. In terms of path length, IECACO achieves the shortest final path. Furthermore, the computational time required by IECACO is the lowest among the three approaches, measuring 9.526 seconds. This represents a 41.4% reduction in computation time compared to ACO and a 15.4% improvement over IACO.

### 4.4.2. Evaluation in Dynamic Environment

To evaluate the performance of the proposed IECACO algorithm in a dynamic environment, the simulation was executed 10 times. The average values of the repeated experiments were listed in Table 6. These experimental results confirm that IECACO is highly effective in dynamic environments proving the implementation of the path retrieval mechanisms significantly reduces redundant computations, reinforcing IECACO's superiority over the other ACO-based approaches. Moreover, IECACO supports asynchronous retrieval updates to global storage, whereas IACO relies on synchronous multi-step searches. These architectural distinctions provide a stronger foundation for generalization to dynamic environments. However, it is acknowledged that both methods leverage edge computing differently, and this difference in design contributes to performance variation.

While this study evaluates IECACO using two grid map sizes, selected for initial benchmarking and comparative analysis with prior studies [21], it is acknowledged that larger and more complex grid scenarios were not explicitly simulated. The 4×4 grid was employed to allow detailed insights under a constrained and controlled setup, while the 20×20 grid reflects the setup used in the referenced baseline (IACO) in [21]. The decision to conduct 10 experimental runs was based on the observation that the performance trends stabilized early with minimal variance across repeated trials. While additional runs such as up to 20 runs, were also executed during internal validation, the results remained consistent with the initial trends. To maintain conciseness due to page limitations, these extended results are not included in the manuscript. Therefore, the decision to report 10 runs was considered sufficient to validate the performance of the algorithm, demonstrate trend stability in dynamic environments and ensure efficient use of edge cloud computing.

**Table 6.** Performance comparison in various environments

| Environment | Type | Number of Iteration | Path Length (unit/cell) | Computational Time (s) | Path Retrieval Time (s) |
|---|---|---|---|---|---|
| Static | ACO | 56 | 52.22 | 16.245 | - |
| | IACO [21] | 33 | 31.35 | 11.258 | - |
| | IECACO | 21 | 28.57 | 9.526 | - |
| Dynamic | | 23 | 29.57 | 11.97 | 8.36 |

## 5. Conclusion

In the context of Industry 4.0 and smart logistics, the proposed method has direct implications for optimizing autonomous material handling systems, warehouse navigation, and flexible manufacturing environments. IECACO aims to make the classical ACO faster, more flexible, and quicker to find solutions in changing environments. This mechanism is mathematically integrated into the ACO pheromone update and decision-making processes through the retrieval influence factor $R_{ij}(t)$ which is introduced as an adaptive weight in both pheromone reinforcement and transition probability. The modified pheromone update rule and probabilistic decision model represent a novel

contribution, allowing dynamic reuse and reinforcement of reliable paths. This dual-layered framework optimizes both path exploration and exploitation across dynamic planning scenarios. However, despite the promising result, it is acknowledged that the current implementation and evaluation of the IECACO algorithm are conducted entirely within a MATLAB simulation environment have several limitations. The dynamic environment experiments, as observed in Subsection 4.3., might not fully capture the breadth of real-world variability. Moreover, while the path retrieval mechanism improves computational efficiency in stable conditions, it may become less effective in highly dynamic environments where stored paths are frequently invalidated. This could lead to additional overhead due to repeated verification. While this allows for controlled experimentation, rapid iteration, and reproducibility, it does limit the direct applicability of the results to physical robotic systems. The absence of hardware-in-the-loop testing means that aspects such as sensor noise, actuator delay, communication latency, and hardware constraints are not fully captured in this study. In the simulation, obstacle positions are reallocated on a per-run basis, representing a form of environmental change that is structured yet non-deterministic. While real-world scenarios may involve continuous obstacle movement, sensor uncertainties, and more complex dynamics, the current simulation framework offers a controlled and reproducible platform for evaluating algorithmic adaptability. Nevertheless, the algorithm has been designed with practical deployment in mind, particularly through the use of edge cloud integration and modular path retrieval mechanisms that are compatible with real-time robotic architectures. Future research will focus on enhancing the scalability of IECACO by extending its evaluation to larger and more complex grid maps to examine its computational viability under increased state-space complexity. One direction involves optimizing memory management and retrieval indexing mechanisms to ensure efficient path selection in high-density obstacle environments. Additionally, validating IECACO on physical AMR platforms is a critical next step, which will involve real-time testing using onboard edge devices to evaluate performance under real-world latency, sensor noise, and network constraints. Another suggestion is to hybridize IECACO with deep reinforcement learning (DRL) to create an adaptive learning-based path planner. For instance, DRL can be employed to learn obstacle movement patterns in dynamic environments and inform the pheromone updating rules or retrieval selection heuristics, enhancing the system's responsiveness to non-deterministic changes. The proposed IECACO algorithm exhibits strong potential for broader application across various domains that involve combinatorial optimization and dynamic decision-making. This includes areas such as drone flight path optimization in urban air mobility, real-time traffic routing in intelligent transportation systems, and dynamic task scheduling in cloud and edge computing infrastructures.

**Author Contribution:** All authors contributed equally to the main contributor to this paper. All authors read and approved the final paper.

**Funding:** This research was funded by the Ministry of Higher Education Malaysia (MOHE) through the Fundamental Research Grant Scheme (FRGS) with grant number FRGS/1/2022/TK07/UTEM/02/33.

**Acknowledgment:** The authors wish to acknowledge the Faculty of Electrical Technology & Engineering (FTKE), Universiti Teknikal Malaysia Melaka (UTeM), Centre for Research & Innovation Management (CRIM) and the Kesidang Scholarship in providing administrative and technical support.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Reference

[1] J. R. Sánchez-Ibáñez, C. J. Pérez-del-Pulgar, and A. García-Cerezo, "Path Planning for Autonomous Mobile Robots: A Review," *Sensors*, vol. 21, no. 23, p. 7898, 2021, https://doi.org/10.3390/s21237898.

[2] X. Liu-Henke, T. Li, M. Göllner, S. Jacobitz, and J. Zhang, "Function development for self-localization through sensor data fusion for autonomous vehicles," *Asia Conference on Electronic Technology (ACET 2024)*, vol. 13211, pp. 87-94, 2024, https://doi.org/10.1117/12.3037706.

[3] I. Kubasáková, J. Kubáňová, D. Benčo, and N. Fábryová, "Application of Autonomous Mobile Robot as a Substitute for Human Factor in Order to Increase Efficiency and Safety in a Company," *Applied Sciences*, vol. 14, no. 13, p. 5859, 2024, https://doi.org/10.3390/app14135859.

[4] I. Ullah, D. Adhikari, H. Khan, M. S. Anwar, S. Ahmad, and X. Bai, "Mobile robot localization: Current challenges and future prospective," *Computer Science Review*, vol. 53, p. 100651, 2024, https://doi.org/10.1016/j.cosrev.2024.100651.

[5] B. Al-Tawil, A. Candemir, M. Jung, and A. Al-Hamadi, "Mobile Robot Navigation with Enhanced 2D Mapping and Multi-Sensor Fusion," *Sensors*, vol. 25, no. 8, p. 2408, 2025, https://doi.org/10.3390/s25082408.

[6] M. A. Taleb, G. Korsoveczki, and G. Husi, "Automotive navigation for mobile robots: Comprehensive Review," *Results in Engineering*, p. 105837, 2025, https://doi.org/10.1016/j.rineng.2025.105837.

[7] P. K. Panigrahi and S. K. Bisoy, "Localization strategies for autonomous mobile robots: A review," *Journal of King Saud University-Computer and Information Sciences*, vol. 34, no. 8, pp. 6019-6039, 2022, https://doi.org/10.1016/j.jksuci.2021.02.015.

[8] L. Yang, P. Li, S. Qian, H. Quan, J. Miao, M. Liu, Y. Hu, and E. Memetimin, "Path Planning Technique for Mobile Robots: A Review," *Machines*, vol. 11, no. 10, p. 980, 2023, https://doi.org/10.3390/machines11100980.

[9] S. N. L. K. N. Azmi, M. Rafique, N. I. A. Apandi, and N. A. Z. M. Noar, "Investigation of Autonomous Mobile Robot Path Planning with Edge Cloud Based on Ant Colony Optimization," *Smart and Sustainable Industrial Ecosystem Conference*, pp. 51-56, 2024, https://doi.org/10.1007/978-981-96-2806-3_9.

[10] T. Lv, J. Zhang, and Y. Chen, "A SLAM Algorithm Based on Edge-Cloud Collaborative Computing," *Journal of Sensors*, vol. 2022, no. 1, p. 7213044, 2022, https://doi.org/10.1155/2022/7213044.

[11] T. Lv, J. Zhang, J. Zhang, and Y. Chen, "A path planning algorithm for mobile robot based on edge-cloud collaborative computing," *International Journal of System Assurance Engineering and Management*, vol. 13, pp. 594-604, 2022, https://doi.org/10.1007/s13198-021-01545-6.

[12] G. Tang, C. Tang, C. Claramunt, X. Hu and P. Zhou, "Geometric A-Star Algorithm: An Improved A-Star Algorithm for AGV Path Planning in a Port Environment," *IEEE Access*, vol. 9, pp. 59196-59210, 2021, https://doi.org/10.1109/ACCESS.2021.3070054.

[13] T. Liao, F. Chen, Y. Wu, H. Zeng, S. Ouyang, and J. Guan, "Research on Path Planning with the Integration of Adaptive A-Star Algorithm and Improved Dynamic Window Approach," *Electronics*, vol. 13, no. 2, p. 455, 2024, https://doi.org/10.3390/electronics13020455.

[14] Q. Meng, C. Qian, Z. Y. Sun, and S. Zhao, "Autonomous parking method based on improved A* algorithm and model predictive control," *Nonlinear Dynamics*, vol. 113, no. 7, pp. 6839-6862, 2025, https://doi.org/10.1007/s11071-024-10456-7.

[15] Y. Dai, W. Lv, S. Li, and M. Zong, “Improving the Lifelong Planning A-star algorithm to satisfy path planning for space truss cellular robots with dynamic obstacles,” *Robotica*, vol. 43, no. 4, pp. 1243-1257, 2025, https://doi.org/10.1017/S0263574725000256.

[16] L. Xu, M. Xi, R. Gao, Z. Ye, and Z. He, "Dynamic path planning of UAV with least inflection point based on adaptive neighborhood A* algorithm and multi-strategy fusion," *Scientific Reports*, vol. 15, no. 1, p. 8563, 2025, https://doi.org/10.1038/s41598-025-92406-w.

[17] M. Luo, X. Hou and J. Yang, "Surface Optimal Path Planning Using an Extended Dijkstra Algorithm," *IEEE Access*, vol. 8, pp. 147827-147838, 2020, https://doi.org/10.1109/ACCESS.2020.3015976.

[18] S. Alshammrei, S. Boubaker, and L. Kolsi, "Improved Dijkstra Algorithm for Mobile Robot Path Planning and Obstacle Avoidance," *Computers, Materials and Continua*, vol. 72, no. 3, pp. 5939-5954, 2022, https://doi.org/10.32604/cmc.2022.028165.

[19] S. W. Fan, W. L. Li, and Y. G. Sun, "An Integrated Trajectory Optimization and Simulation for a Construction Robot," *International Journal of Simulation Modelling*, vol. 24, no. 2, pp. 345-356, 2025, https://doi.org/10.2507/IJSIMM24-2-CO8.

[20] D. Liu and L. Xu, "Robot path planning and obstacle avoidance algorithm based on visual perception," *Neural Computing and Applications*, pp. 1-18, 2025, https://doi.org/10.1007/s00521-025-11358-4.

[21] Z. Bi, Y. Ma, X. Yang, and P. Zhou, "Research on wireless robot path planning under edge computing considering multistep searching and inflection points," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 6, p. e3825, 2021, https://doi.org/10.1002/ett.3825.

[22] P. Huang, L. Zeng, K. Luo, J. Guo, Z. Zhou and X. Chen, "ColaSLAM: Real-Time Multi-Robot Collaborative Laser SLAM via Edge Computing," *2021 IEEE/CIC International Conference on Communications in China (ICCC)*, pp. 242-247, 2021, https://doi.org/10.1109/ICCC52777.2021.9580413.

[23] E. Cui, D. Yang, H. Wang, and W. Zhang, "Learning-based deep neural network inference task offloading in multi-device and multi-server collaborative edge computing," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 7, p. e4485, 2022, https://doi.org/10.1002/ett.4485.

[24] X. Li, Y. Qin, H. Zhou, and Z. Zhang, "An intelligent collaborative inference approach of service partitioning and task offloading for deep learning based service in mobile edge computing networks," *Transactions on Emerging Telecommunications Technologies*, vol. 32, no. 9, p. e4263, 2021, https://doi.org/10.1002/ett.4263.

[25] S. Su, X. Ju, C. Xu and Y. Dai, "Collaborative Motion Planning Based on the Improved Ant Colony Algorithm for Multiple Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 3, pp. 2792-2802, 2024, https://doi.org/10.1109/TITS.2023.3250756.

[26] F. Martınez, H. Montiel, and L. Wanumen, "A deep reinforcement learning strategy for autonomous robot flocking," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 13, no. 5, pp. 5707-5716, 2023, http://doi.org/10.11591/ijece.v13i5.pp5707-5716.

[27] A. Momenikorbekandi and M. Abbod, "Intelligent Scheduling Based on Reinforcement Learning Approaches: Applying Advanced Q-Learning and State–Action–Reward–State–Action Reinforcement Learning Models for the Optimisation of Job Shop Scheduling Problems," *Electronics*, vol. 12, no. 23, p. 4752, 2023, https://doi.org/10.3390/electronics12234752.

[28] Y. Yang, L. Juntao, and P. Lingling, "Multi-robot path planning based on a deep reinforcement learning DQN algorithm," *CAAI Transactions on Intelligence Technology*, vol. 5, no. 3, pp. 177-183, 2020, https://doi.org/10.1049/trit.2020.0024.

[29] X. Chen, Y. Kong, X. Fang, and Q. Wu, "A fast two-stage ACO algorithm for robotic path planning," *Neural Computing and Applications*, vol. 22, no. 2, pp. 313-319, 2013, https://doi.org/10.1007/s00521-011-0682-7.

[30] Y. J. Shen and Q. T. Wang, "Mobile robot path planning with two stages based on hybrid intelligent optimisation algorithm," *International Journal of Robotics and Automation*, vol. 38, no. 6, pp. 416-429, 2023, https://dx.doi.org/10.2316/J.2023.206-0837.

[31] A. Zou, L. Wang, W. Li, J. Cai, H. Wang, and T. Tan, "Mobile robot path planning using improved mayfly optimization algorithm and dynamic window approach," *The Journal of Supercomputing*, vol. 79, no. 8, pp. 8340-8367, 2023, https://doi.org/10.1007/s11227-022-04998-z.

[32] H. Heng, M. H. M. Ghazali, and W. Rahiman, "Comparative analysis of navigation algorithms for mobile robot," *Journal of Ambient Intelligence and Humanized Computing*, vol. 15, no. 12, pp. 3861-3871, 2024, https://doi.org/10.1007/s12652-024-04854-3.

[33] S. Zhang, J. Pu, Y. Si, and L. Sun, "Path planning for mobile robot using an enhanced ant colony optimization and path geometric optimization," *International Journal of Advanced Robotic Systems*, vol. 18, no. 3, 2021, https://doi.org/10.1177/17298814211019222.

[34] W. Zang, P. Yao, K. Lv, and D. Song, "A deep Q network assisted method for underwater gliders standoff tracking to the static target," *Neural Computing and Applications*, vol. 34, no. 23, pp. 20575-20587, 2022, https://doi.org/10.1007/s00521-022-07408-w.

[35] A. Figueroa, M.-C. Riff, and E. Montero, "Robots in Partially Known and Unknown Environments: A Simulated Annealing Approach for Re-Planning," *Applied Sciences*, vol. 14, no. 22, p. 10644, 2024, https://doi.org/10.3390/app142210644.

[36] Z. Zhang, P. Li, S. Chai, Y. Cui, and Y. Tian, "DGA-ACO: Enhanced Dynamic Genetic Algorithm—Ant Colony Optimization Path Planning for Agribots," *Agriculture*, vol. 15, no. 12, p. 1321, 2025, https://doi.org/10.3390/agriculture15121321.

[37] B. Zolghadr-Asli, "Ant Colony Optimization Algorithm," *Computational Intelligence-based Optimization Algorithms*, pp. 94-112, 2023, https://doi.org/10.1201/9781003424765-6.

[38] S. M. Almufti, R. P. Maribojoc, and A. V. Pahuriray, "Ant Based System: Overview, Modifications and Applications from 1992 to 2022," *Polaris Global Journal of Scholarly Research and Trends*, vol. 1, no. 1, pp. 1-8, 2022, https://doi.org/10.58429/pgjsrt.v1n1a85.

[39] F. Xu, Z. Qin, L. Ning, and Z. Zhang, "Research on computing offloading strategy based on Genetic Ant Colony fusion algorithm," *Simulation Modelling Practice and Theory*, vol. 118, p. 102523, 2022, https://doi.org/10.1016/j.simpat.2022.102523.

[40] W. Xia and L. Shen, "Joint resource allocation at edge cloud based on ant colony optimization and genetic algorithm," *Wireless Personal Communications*, vol. 117, no. 2, pp. 355-386, 2021, https://doi.org/10.1007/s11277-020-07873-3.

[41] T. Alfakih, M. M. Hassan and M. Al-Razgan, "Multi-Objective Accelerated Particle Swarm Optimization With Dynamic Programing Technique for Resource Allocation in Mobile Edge Computing," *IEEE Access*, vol. 9, pp. 167503-167520, 2021, https://doi.org/10.1109/ACCESS.2021.3134941.

[42] S. Ma, S. Song, J. Zhao, L. Zhai and F. Yang, "Joint Network Selection and Service Placement Based on Particle Swarm Optimization for Multi-Access Edge Computing," *IEEE Access*, vol. 8, pp. 160871-160881, 2020, https://doi.org/10.1109/ACCESS.2020.3020935.

[43] L. N. T. Huynh, Q.-V. Pham, X.-Q. Pham, T. D. T. Nguyen, M. D. Hossain, and E.-N. Huh, "Efficient Computation Offloading in Multi-Tier Multi-Access Edge Computing Systems: A Particle Swarm Optimization Approach," *Applied Sciences*, vol. 10, no. 1, p. 203, 2020, https://doi.org/10.3390/app10010203.

[44] T. Lv, J. Zhang, and Y. Chen, "A SLAM Algorithm Based on Edge-Cloud Collaborative Computing," *Journal of Sensors*, vol. 2022, no. 1, p. 7213044, 2022, https://doi.org/10.1155/2022/7213044.

[45] H. Wu, Y. Gao, W. Wang, and Z. Zhang, "A hybrid ant colony algorithm based on multiple strategies for the vehicle routing problem with time windows," *Complex & intelligent systems*, vol. 9, no. 3, pp. 2491-2508, 2023, https://doi.org/10.1007/s40747-021-00401-1.

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

[47] M. H. Mousa and M. K. Hussein, "Efficient UAV-based mobile edge computing using differential evolution and ant colony optimization," *PeerJ Computer Science*, vol. 8, p. e870, 2022, https://doi.org/10.7717/peerj-cs.870.

[48] Z. H. Ahmed, A. S. Hameed, M. L. Mutar, and H. Haron, "An Enhanced Ant Colony System Algorithm Based on Subpaths for Solving the Capacitated Vehicle Routing Problem," *Symmetry*, vol. 15, no. 11, p. 2020, 2023, https://doi.org/10.3390/sym15112020.

[49] I. A. Khalifa and S. A. Saleh, "A Review of Ant Colony Optimization for Solving 0-1 Knapsack and Traveling Salesman Problems," *European Journal of Applied Science, Engineering and Technology*, vol. 3, no. 2, pp. 87-99, 2025, https://doi.org/10.59324/ejaset.2025.3(2).08.

[50] M. Dorigo, T. Stützle, "Ant Colony Optimization: Overview and Recent Advances," *Handbook of Metaheuristics*, pp. 311-351, 2019, https://doi.org/10.1007/978-3-319-91086-4_10.

[51] C. -C. Hsu, R. -Y. Hou and W. -Y. Wang, "Path Planning for Mobile Robots Based on Improved Ant Colony Optimization," *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 2777-2782, 2013, https://doi.org/10.1109/SMC.2013.474.

[52] D. Farinati and L. Vanneschi, "A survey on dynamic populations in bio-inspired algorithms," *Genetic Programming and Evolvable Machines*, vol. 25, no. 2, p. 19, 2024, https://doi.org/10.1007/s10710-024-09492-4.

[53] L. B. Amar and W. M. Jasim, "Hybrid metaheuristic approach for robot path planning in dynamic environment," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 4, pp. 2152-2162, 2021, https://doi.org/10.11591/eei.v10i4.2836.

[54] M. Teymournezhad and O. K. Sahingoz, "Path planning of Mobile Robot in Dynamic Environment by Evolutionary Algorithms," *2023 2nd International Conference on Computational Systems and Communication (ICCSC)*, pp. 1-6, 2023, https://doi.org/10.1109/ICCSC56913.2023.10142971.