Bi-directional adaptive enhanced A* algorithm for mobile robot navigation

Atef Gharbi
Department of Information Systems, Faculty of Computing and Information Technology, Northern Border University, Rafha , Saudi Arabia

Received 26 December 2023
Revised 16 March 2024
1 April 2024
Accepted 6 April 2024

Abstract

Purpose – The present paper aims to address challenges associated with path planning and obstacle avoidance in mobile robotics. It introduces a pioneering solution called the Bi-directional Adaptive Enhanced A* (BAEA*) algorithm, which uses a new bidirectional search strategy. This approach facilitates simultaneous exploration from both the starting and target nodes and improves the efficiency and effectiveness of the algorithm in navigation environments. By using the heuristic knowledge A*, the algorithm avoids unproductive blind exploration, helps to obtain more efficient data for identifying optimal solutions. The simulation results demonstrate the superior performance of the BAEA* algorithm in achieving rapid convergence towards an optimal action strategy compared to existing methods.

Design/methodology/approach – The paper adopts a careful design focusing on the development and evaluation of the BAEA* for mobile robot path planning, based on the reference [18]. The algorithm has remarkable adaptability to dynamically changing environments and ensures robust navigation in the context of environmental changes. Its scale further enhances its applicability in large and complex environments, which means it has flexibility for various practical applications. The rigorous evaluation of our proposed BAEA* algorithm with the Bidirectional adaptive A* (BAA*) algorithm [18] in five different environments demonstrates the superiority of the BAEA* algorithm. The BAEA* algorithm consistently outperforms BAA*, demonstrating its ability to plan shorter and more stable paths and achieve higher success rates in all environments.

Findings – The paper adopts a careful design focusing on the development and evaluation of the BAEA* for mobile robot path planning, based on the reference [18]. The algorithm has remarkable adaptability to dynamically changing environments and ensures robust navigation in the context of environmental changes. Its scale further enhances its applicability in large and complex environments, which means it has flexibility for various practical applications. The rigorous evaluation of our proposed BAEA* algorithm with the Bi-directional adaptive A* (BAA*) algorithm [18] in five different environments demonstrates the superiority of the BAEA* algorithm.

Research limitations/implications – The rigorous evaluation of our proposed BAEA* algorithm with the BAA* algorithm [18] in five different environments demonstrates the superiority of the BAEA* algorithm. The BAEA* algorithm consistently outperforms BAA*, demonstrating its ability to plan shorter and more stable paths and achieve higher success rates in all environments.

Originality/value – The originality of this paper lies in the introduction of the bidirectional adaptive enhancing A* algorithm (BAEA*) as a novel solution for path planning for mobile robots. This algorithm is characterized by its unique characteristics that distinguish it from others in this field. First, BAEA* uses a unique bidirectional search strategy, allowing to explore the same path from both the initial node and the target node. This approach significantly improves efficiency by quickly converging to the best paths and using A* heuristic knowledge. In particular, the algorithm shows remarkable capabilities to quickly recognize shorter and more stable paths while ensuring higher success rates, which is an important feature for time-sensitive applications. In addition, BAEA* shows adaptability and robustness in dynamically changing environments, not only avoiding obstacles but also respecting various constraints, ensuring safe path selection. Its scale

Funding: This paper has been completed without the support of external funding sources.
Conflict of interest: The authors declare that they have no conflict of interest.

Applied Computing and Informatics
Emerald Publishing Limited
e-ISSN: 2210-8327
p-ISSN: 2634-1964
DOI 10.1108/ACI-12-2023-0195

further increases its versatility by seamlessly applying it to extensive and complex environments, making it a versatile solution for a wide range of practical applications. The rigorous assessment against established algorithms such as BAA* consistently shows the superior performance of BAEA* in planning shorter paths, achieving higher success rates in different environments and cementing its importance in complex and challenging environments. This originality marks BAEA* as a pioneering contribution, increasing the efficiency, adaptability and applicability of mobile robot path planning methods.

Keywords Mobile robot, Path planning, A* algorithm, Bi-directional adaptive enhanced A* algorithm (BAEA*)

Paper type Research paper

1. Introduction

To enhance production efficiency, mobile robots are increasingly deployed within industrial settings. As the utilization of mobile robots continues to expand, the imperative to enhance their navigational capabilities becomes increasingly apparent. A multitude of research studies have been conducted in pursuit of refining the autonomous navigation systems of these robots. Consequently, path planning has emerged as a focal point of research and development. Path planning methodologies facilitate the movement of mobile robots from their initial positions to the designated target locations, ensuring collision-free navigation through complex environments. If a mobile robot cannot traverse an unfamiliar environment safely, it may result in detrimental consequences, encompassing environmental harm along with various associated losses such as time and production inefficiencies. Hence, the incorporation of effective path-planning strategies for mobile robots operating in uncharted environments holds substantial potential to deliver significant value and influence.

Path planning strategies are contingent upon environmental characteristics, which can be broadly categorized into three types: known environments [1], partially known environments [2], and unknown environments [3]. Furthermore, path planning can be categorized into two primary modes: static and dynamic, contingent on the nature of the obstacles encountered. Static path planning denotes a scenario in which the spatial configuration of obstacles remains invariant over time [4], while dynamic path planning pertains to situations where obstacles can move within the environment over the time [5].

A multitude of algorithms dedicated to path planning have been advanced over the time. The authors introduced a vast amount of research on fuzzy-logic approaches for collision-free path planning of manipulator robots, highlighting their effectiveness and advantages over traditional methods [6]. A major drawback of using fuzzy logic for collision avoidance in path planning is its reliance on human-defined membership functions and fuzzy rules, which can be subjective and prone to errors in complex environments.

The Particle Swarm Optimization (PSO) algorithm is employed to prevent Robot collisions with obstacles in path planning, however, while doing so, PSO can get stuck in local optima and prioritize exploitation over exploration, leading to suboptimal paths and poor performance in dynamic environments, particularly when avoiding obstacles [7]. An adaptive ant colony algorithm enhances the optimization of the path planning of indoor mobile robots [8]. However, the Ant Colony algorithm can struggle with finding the optimal path around obstacles due to its reliance on pheromones, which can lead to inefficient exploration and stagnation in complex environments. Bio-inspired neural network are reported to optimize the robot's path to efficiently cover the entire hull surface, reducing energy consumption during maintenance operations [9]. However, to ensure collision-free paths in complex environments due to the nature of black boxes and the possibility of over-fitting, it leads to unpredictable behavior and safety concerns. The multi-robot path planning method leverages reinforcement learning techniques to coordinate the movement of multiple robots in complex environments, facilitating efficient path planning [10]. However, employing reinforcement learning for collision avoidance necessitates substantial training

data and exploration efforts. This can result in potential instability, slow convergence, and difficulties in adapting to dynamic environments where obstacles may move unexpectedly. The Reinforcement Learning-Based Grey Wolf Optimizer (RLGWO), proposed in Ref. [11], is a novel algorithm tailored for addressing path-planning challenges encountered by Unmanned Aerial Vehicles (UAVs). This method aims to enhance the efficiency and effectiveness of UAV path planning across diverse environments. However, RLGWO's dependency on reward functions for collision avoidance may result in suboptimal paths in complex environments. This challenge arises from the difficulty in designing comprehensive reward functions that accurately capture all pertinent factors, potentially causing the optimizer to prioritize exploration over achieving collision-free paths. The Efficient Q-Learning (EQL) algorithm introduced in Ref. [12] aims to address the challenges of slow convergence and suboptimal performance, ensuring the rapid generation of an optimal collision-free path. However, EQL may prioritize fast convergence over comprehensive exploration, which could result in neglecting safer but less immediately rewarding paths that effectively avoid collisions with obstacles. In the field of multi-agent systems, the proposed model [13] uses flexible Q-learning techniques to navigate decentralized multi-agent robots within continuous state spaces. Despite its potential, the approach faces challenges related to the overhead of communication and scalability as the number of agents increases. The continuous exchange of information required to update the Q-table of agents may constrain system resources, resulting in performance gaps and constraints on the supervision of large-scale multi-agent systems. Unmanned Aerial Vehicles (UAVs) were proposed for the detection of aquaculture cages, with Q-learning and SARSA used to enhance energy efficiency and flight safety [14]. However, the study highlighted deficiencies, notably the lack of energy optimization strategies and quantitative indicators. The Improved Velocity Potential Field (IVPF) algorithm is introduced for tracking robot arms in medical scenarios, enhancing obstacle avoidance and ensuring safety for both human operators and robot arms [15]. However, the study lacks extensive validation in real-world medical environments and may benefit from further research into alternative trajectory planning approaches for a comprehensive evaluation. Neural network models and sophisticated algorithms were used to improve the planning of paths and obstacle avoidance for clean robots operating in dynamic environments [16]. Nevertheless, the research acknowledges the need for additional validation in different realistic scenarios and highlights potential challenges associated with the integration of neural network models into robotic systems. Reinforcement learning (RL)-based techniques for dynamic obstacle avoidance and path planning are widely used [17]. Nonetheless, limited validation under diverse scenarios may limit the generality of these findings.

The Bi-directional adaptive A* algorithm (BAA*) introduced in Ref. [18] for path planning of unmanned aerial vehicles (UAVs) in complex environments is based on the A* algorithm and integrates both an adaptive step strategy and an adaptive weight strategy to efficiently compute optimal paths for large-scale Unmanned Aerial Vehicles (UAVs) while considering multiple constraints. The adaptive step strategy dynamically modifies the step size of the A* algorithm by assessing the local environment around the current node, enhancing convergence speed. Meanwhile, the adaptive weight strategy aims to strike a balance between the convergence rate and the quality of the generated path during the planning process.

Our approach draws inspiration from the BAA* algorithm outlined in Ref. [18], yet it is essential to underscore the substantial divergences and novelties inherent in our proposed Bi-directional adaptive Enhanced A* algorithm (BAEA*) in comparison to Ref. [18]. The paramount distinction between our BAEA* and BAA* [18], which bears the greatest significance, lies in our novel proposition about the computation of gobs(N). This parameter is pivotal as it quantifies the cost associated with obstacle avoidance in our model, a unique

feature absent in the work presented in Ref. [18]. Another notable differentiation between our study and [18] is manifested in the Pseudo code of the bi-directional adaptive Enhanced A* algorithm (BAEA*). Our approach emphasizes the simplification and enhanced practicality of the algorithm's implementation, rendering it more straightforward and efficient compared to the version presented in Ref. [18]. Another distinction between our study and the prior work discussed in Ref. [18] lies in the context of application. The authors in Ref. [18] focused on Unmanned Aerial Vehicles (UAVs), whereas our research considers mobile robotic platforms. The secondary point of distinction between our study and [18] resides in the dimensionality under consideration. In Ref. [18], the authors conducted their research in the three-dimensional space, explicitly encompassing the (x, y and z) coordinates. In contrast, our investigation is grounded in a two-dimensional context, exclusively focusing on the (x, y) plane.

In Section 2, we will elucidate the Path Planning Utilizing the Enhanced A* Algorithm. Section 3 will expound on the Bi-directional Adaptive Enhanced A* Algorithm, providing an in-depth analysis of its algorithmic procedures and key steps. Section 4 will develop Q-Learning extensively by providing a detailed clarification of its principle and method. Section 5 will present the experimental findings and results. The concluding section, Section 6, will encapsulate the study's conclusions and delineate avenues for future research.

2. Path planning using enhanced A* algorithm

The effectiveness of the A* algorithm depends heavily on the design of the heuristic function h (n). It plays a crucial role in guiding the search process by providing an estimate of the cost of reaching the goal from the current node. A good heuristic function helps the algorithm efficiently explore the most promising paths first, leading to faster and more optimal pathfinding. Equation (1) defines the heuristic function f(N) in the A* algorithm.

$$f(N) = g(N) + h(N) \tag{1}$$

Equation (1) consists of two main components g(N) and h(N):

(1) f(N): Cumulative cost estimate from the initial point to the destination via the current node n.

(2) g(N): The real cost from the starting point to the current node n, reflecting the accumulated cost up to this point.

(3) h(N): Estimated cost from the current node n to the goal. This is a heuristic estimate and is typically problem-specific.

2.1 Adaptive weight strategy

The A* algorithm can be dynamically adjusted to improve its performance in different problems and environments. This is achieved by adjusting the weights of the cost function g(N) and heuristic function h(N). When the weight of the heuristic function h(N) is minimized, the A* algorithm behaves more like Dijkstra's algorithm. This means that it will explore the entire state space to find the optimal path, but will take longer to converge. When the weight of the cost function g(N) is minimized, the A* algorithm behaves more like a best-first-search algorithm (BFS). This means that it will quickly find a path to the goal but it may not be the optimal path. The A* algorithm can strike a balance between the convergence speed and path quality by dynamically adjusting the weights of g(N) and h(N). The proposed algorithm for mobile robot (MR) path planning uses an adaptive weight strategy to dynamically adjust the weights g(N) and h(N). The goal is to determine a high-quality path promptly.

The evaluation function f(N) is defined in equation (2):

$$f(N) = \omega_g * g(N) + \omega_h * h(N) \quad (2)$$

where ω_g and ω_h are the weight factors of g(N) and h(N), respectively.

Weight factors ωg and ωh were determined using the following equations (3) and (4):

$$\omega_h = \begin{cases} \omega_{\max}, & D \geq D_{eff} \\ \omega_{\max} - (\omega_{\max} - \omega_{\min}) * \left(1 - \dfrac{D}{D_{eff}}\right), & D < D_{eff} \end{cases} \quad (3)$$

$$\omega_g = 1 \quad (4)$$

where ω_{max} and ω_{min} represent the maximum and minimum preset values for ω_h, respectively. D_{eff} is a threshold value that defines the boundary between obstacle-free and obstacle-rich areas.

In obstacle-free areas ($D \geq D_{eff}$), ω_h is set to its maximum value to accelerate convergence. In obstacle-rich areas ($D < D_{eff}$), ω_g is adaptively increased to find an optimal path within the obstacles. This adaptive weight strategy allows the algorithm to find high-quality paths promptly, even in complex environments with many obstacles.

2.2 Real cost g(N)

To account for the specific constraints of MRs, such as distance cost and the need for a collision-free path, the expression for the cost function g(N) was modified as shown in equation (5):

$$g(N) = \omega_1 * g_{obs}(N) + \omega_2 * g_{cost}(N) \quad (5)$$

where:

(1) $g_{obs}(N)$ is the cost from the current position N to the nearest obstacle. It relies on the distance between the robot and the obstacle.

(2) $g_{cost}(N)$ is the Manhattan distance cost from the current position N to the initial node Pinit.

(3) ω_1 and ω_2 are the weights for the obstacle cost and distance cost factors, respectively.

The objective function g(N) in Equation (5) is minimized by the optimization process.

The cost function is defined in equation (6):

(1) If D is greater than D_{eff}, the cost is zero.

(2) If x is less than or equal to D_{eff}, the cost is D_{eff}/D.

Threat area cost ($g_{obs}(N)$)

$$g_{obs}(N) = \begin{cases} 0, & D > D_{eff} \\ \dfrac{D_{eff}}{D}, & D \leq D_{eff} \end{cases} \quad (6)$$

(1) $g_{obs}(N)$ represents the threat area cost of the robot.

(2) D signifies the distance between the robot and the obstacle.

(3) Obstacles are characterized by one radius: D_{eff} for the detection radius.

These equations were used to calculate the energy consumption and threat area costs associated with the path planning of the robot.

2.3 Heuristic function h(N)

In contrast to the traditional A* algorithm, the heuristic function h(N) in this approach was determined using the Manhattan distance metric rather than relying on the Euclidean metric. This choice is based on the observation that the optimization results vary depending on the geometric measurement used for the heuristic function. A heuristic function based on the Euclidean distance tends to explore a larger space, which can lead to increased memory usage and computational load owing to the expensive square-root calculations involved. Conversely, a heuristic function grounded in the Manhattan distance metric provides a more accurate exploration of space with a reduced number of superfluous nodes. This makes the Manhattan distance a better choice for ensuring the efficiency and path quality of the proposed algorithm as shown in equation (7).

$$h(N) = |n_x - goal_x| + |n_y - goal_y| \tag{7}$$

where:

(1) (n_x, n_y) are the coordinates of the node N.

(2) $(goal_x, goal_y)$ are the coordinates of the goal node P_{goal}.

The proposed algorithm uses a modified cost function and the Manhattan distance heuristic to improve the efficiency and path quality of MR path planning.

3. Bi-directional adaptive enhanced A* algorithm

The algorithm operates by conducting simultaneous forward and reverse searches of the initial and target positions. It maintains two OPEN tables and two CLOSED tables to track the explored nodes and their costs. Figure 1 shows the ongoing search process, which continues until the forward and reverse search converge to a mutual point, indicating a viable path is identified. The BAEA* algorithm uses problem-specific heuristics to guide forward and backward search processes. The paths are formulated with explicit collision verification, and the heuristic insights share convergence at the midpoint. The forward and backward search carefully explore the spatial domains guided by the heuristics and ensure the validity of their edges by validation of collisions.

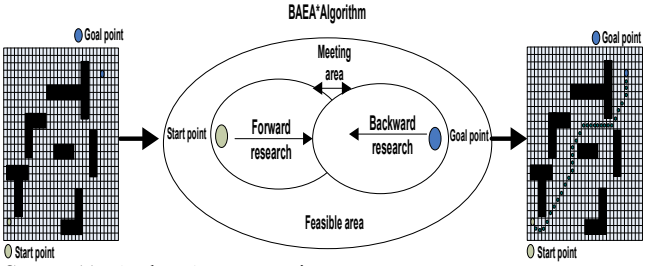


Figure 1.
Bidirectional BAEA* schematic (Backward and Forward research)

Source(s): Authors' own creation

Throughout the process, the algorithm considers various parameters denoted as ω_1, ω_2, D_{eff}, ω_{max}, ω_{min} and so on, which influence the search strategy and the quality of the path found. The algorithm is designed for efficient path planning in complex environments, and the use of Q-learning elements suggests that it may incorporate machine-learning techniques to enhance its performance. The detailed steps of the Bi-directional Adaptive Enhanced A* algorithm are in Algorithm 1.

Algorithm 1. Bi-directional Adaptive Enhanced A* algorithm

Input:

- Initial positions of the mobile robot (MR) and the goal
- parameters ω_1, ω_2, D_{eff}, ω_{max}, ω_{min}

Output:

Path node

Begin

While neither the forward nor reverse searches converged **do**

1. In the forward OPEN table, insert the forward search node.
2. In the reverse OPEN table, insert the reverse search node.
3. Build the parent-child relationship by moving the lowest-cost node from the forward OPEN table to the forward CLOSED table.
4. Build the parent-child relationship by moving the lowest-cost node from the reverse OPEN table to the reverse CLOSED table.
5. **If** neither the forward nor reverse searches have converged, then
   Continue;
   Else
   exit the loop.
   End if

End while

End

The path node can be obtained by following the parent-child relationship from the goal node to the initial node. The modified A* algorithm is a bidirectional search algorithm that includes the advantages of the A*. It is particularly well suited for mobile robot path planning, as it can efficiently find paths that avoid obstacles and satisfy other constraints, such as energy consumption and time limits. The algorithm operates by maintaining two search trees: one for the forward search and one for the reverse search. The forward search starts from the initial position of the MR and searches for a goal. The reverse search starts with the goal and

searches for the initial position. At each step, the algorithm selects the node with the minimum cost from the OPEN table of the forward search or the reverse search. The selected node is then moved to the CLOSED table of the corresponding search tree and its parent-child relationship is established. The algorithm terminates when the forward and reverse search trees meet, or when a certain number of iterations have been reached. The path node can then be obtained by following the parent-child relationship from the goal node to the initial node.

4. Path planning based on Q-learning algorithm

Our research on Q-learning methodology is based on previous research described in previous published work [19], in which we provide a comprehensive overview of the paradigms of machine learning. Consequently, we succinctly present the Q-learning procedure. To get a complete understanding, we directed readers to our previous publication [19].

The Q-learning algorithm contains four fundamental components: state, action, reward and Q-table. These elements are complexly defined in the application of the Q-learning algorithm. Reinforcement Learning (RL) is a paradigm of machine learning that aims to acquire optimal action through interactions within the environment to achieve a predetermined goal. Its structural framework consists of the following integral components:

(1) State (s): indicates the current configuration of the environment as the agent perceives it. It encapsulates all the relevant information needed for decision-making processes.

(2) Action (a): The selection of possible choices provided by the agent in each state. Action initiates a transition from one state to another and shapes the agent's trajectory.

(3) Reward I: Represents the immediate feedback received by the environment when an action is performed in a particular state. The reward quantifies the desirable action of the agent. For example, if the robot is near an obstacle, it will get a negative reward, while arriving at the destination will get a positive reward. The distance between the agent and the nearest obstacle is symbolized by d_{obs}, where C_1 represents a fixed parameter that represents the reward to achieve the goal state (the negative value of $-C_1$ is associated with the proximity of the obstacle). The reward function includes additional scenarios, i.e. the proximity and distance to the target position. If the agent's position is closer from the target position without colliding to the target position, the agent receives a slight positive reward. On the contrary, if the agent's position moves further away from the target position without colliding, the agent receives a marginal negative reward. Dynamic rewards can be calculated as shown in equation (8).

$$r(s_t, a_t) = \begin{cases} C_1, S_t = S_g \\ -C_1, d_{obs} = 0 \\ C_2 * \dfrac{d_{t+1}}{D}, d_t < d_{t+1}, d_{obs} \neq 0 \\ -C_2 * \dfrac{d_{t+1}}{D}, d_t > d_{t+1}, d_{obs} \neq 0 \end{cases} \tag{8}$$

where d_t represents the Manhattan distance between the agent and the target, d_{obs} represents the minimum distance between the agent and the nearest obstacle, C_1 and C_2 represent a specific reward parameter obtained after the agent's actions.

(1) Q-function (Q): Estimate the value associated with performing specific actions within each State. It helps in decision-making processes by assessing the value of action and guiding the agent towards optimal choices.

The RL system operates through In iterative framework, allowing agents to interact with the environment, observe the state, perform actions, receive rewards, and adjust their policies and value functions based on these interactions. The main goal is for agents to obtain the best policies to maximize cumulative rewards over time. This learning process combines exploration (experience with new actions) and exploitation (use of the knowledge acquired to select the most beneficial actions), thus improving the agent's decision-making capabilities.

The Q-Learning algorithm, defined as Algorithm 2, orchestrates autonomous navigation in a predefined search area, using task nodes, starting points and target points as inputs to define objectives within the environment. The initialization stages include the setup of a Q-table, obstacle data, and an action space that defines a feasible agent action. The algorithm tries to iterate through an exploration-exploitation loop until a goal state is reached, by organizing the state space initialization for each iteration. In this cycle, the BAEA* algorithm determines the selection of actions based on current state and Q-values. The execution of the selected action leads the agent to a new state and generates feedback in the form of the rewards received and the resulting state. Afterward, Q-values are updated by combining the Q-learning formula to consider parameters such as learning rate α and discount factor γ. State transitions occur as the algorithm progresses and iterate until the final conditions are met. The integration of Q-learning and BAEA* promotes efficient path planning and learning-driven decision-making, enabling autonomous agents to navigate and learn in complex environments.

Algorithm 2. Q-Learning

Input: search area, tasks node, start point, goal point

Output: learning value, Q-table

Initialize: Q-table, obstacle information, action space $A = \{a1; a2; \ldots ; a_t\}$, action-value function $Q(s_t; a_t)$, where $s_t \in S$; $a_t \in A$

While S non-goal state **do**

Initialize the state space $S = \{s_1; s_2; \ldots ; s_t\}$

for each step in each round **do**

Use BAEA* algorithm to select an action a_{t+1}

Take the action a_t selected in the previous step to obtain the feedback value r_t and the new state s_{t+1}

$$Q(s_{t+1},a_{t+1}) = -(1 - \alpha) * Q(s_t,a_t) + \alpha * [r(s_t,a_t) + \gamma \max_{a_{t+1}} Q(s_{t+1}, a_{t+1})] \quad (9)$$

$s_t \leftarrow s_{t+1}$

End For

End While

This iterative algorithm explores and refined Q values dynamically by observing rewards and state transitions. Its objective is to improve the efficiency of the planning of robot paths within the designated search area by considering both obstacles and task nodes.

5. Experimental results

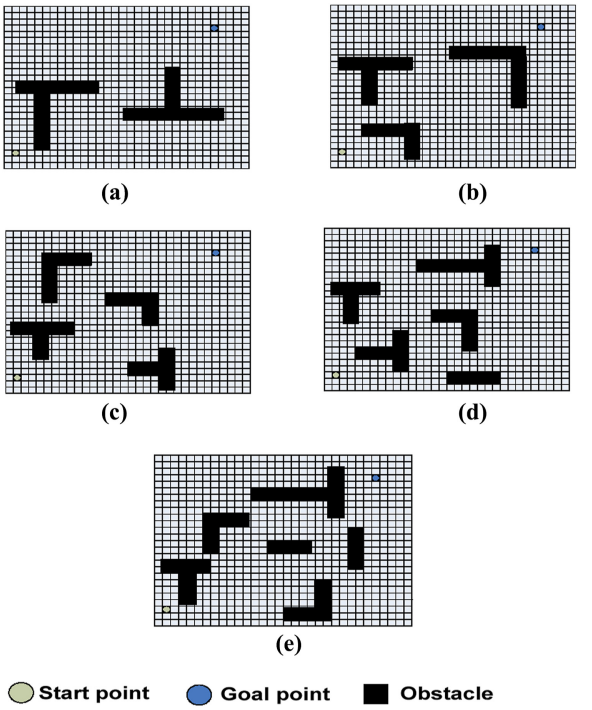
To evaluate the effectiveness of the proposed algorithm, some experiments were conducted. The mobile robot was initially placed at coordinates (20, 20), and the target destination and the threat zones located as it is indicated in Table 1. To evaluate the efficacy of our proposed approach, we carried out comparative assessments. The assessment of path planning was conducted based on mission and environmental details, as outlined in Table 1.

Figure 2 represents each experimental scenario, the mission particulars, denoted by initial coordinates, were kept consistent with the mobile robot (MR) commencing at (20, 20). The

Table 1.
Target and obstacles in different test environments

Environment	Target	Obstacles
Env1	(390, 400)	(50, 79, 10); (155, 210, 15)
Env2	(450, 470)	(75, 90, 5); (145, 250, 15); (234, 327, 10)
Env3	(510, 430)	(110, 200, 5); (215, 324, 10); (320, 460, 10); (400, 510, 5)
Env4	(480, 530)	(135, 150, 5); (233, 267, 10); (345, 377, 5); (462, 525, 10); (513, 457,15)
Env5	(340, 500)	(125, 224, 10); (247, 355, 5); (378, 429, 10); (483, 562, 5) (529, 436,5); (511, 428,15)

Source(s): Authors' own creation



Source(s): Authors' own creation

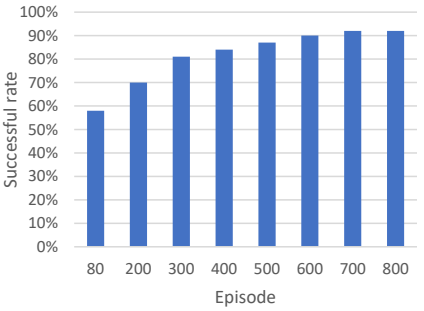
Figure 2.
Grid environment
(a) Environment 1,
(b) Environment 2,
(c) Environment 3,
(d) Environment 4, and
(e) Environment 5

environmental parameters encompassed diverse maps and zones featuring obstacles (O), which were characterized by the positions of obstacle sources and their corresponding radii, expressed in the format (x, y and r1). Figure 2 shows the grid environment layout, indicated as (a) environment 1, (b) environment 2, (c) environment 3, (d) environment 4 and (e) environment 5, each environment is characterized by the specific obstacles strategically placed in the grid.

The success rate is a crucial metric for determining the effectiveness of learning algorithms in training. If the mobile robot hits obstacles or fails to reach the destination within the highest number of iterations allowed during learning, the whole training session is regarded as a failure. In general, it is assumed that as more training episodes are added performance would improve, resulting in a larger success rate. To evaluate the impact of training on the success rate, measurements are taken every 100 episodes. This involves counting the number of successful episodes during training and subsequently calculating the success rate.

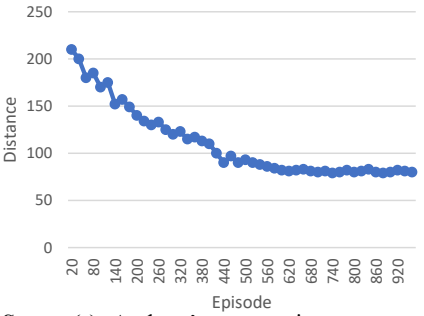
Figure 3 shows the training's influence on the success rate. Figure 3 indicates that the success rate consistently increases with additional episodes. At the outset, with just 80 training episodes, the success rate is 58%. As the algorithm undergoes hundreds of training iterations, reaching 600 training episodes, the success rate substantially improves to nearly 90%, a notably high success rate. The success rate experiences rapid growth in the early stages and remains at a high level beyond 600 episodes.

Algorithm performance is considered another important evaluation metric for training effectiveness which is path length. Based on Figure 4, the initially observed path length exhibited a high value due to the mobile robot's lack of knowledge of the environment, which



Source(s): Authors' own creation

Figure 3.
The success rate per episodes



Source(s): Authors' own creation

Figure 4.
The length of the planned path at various training episodes

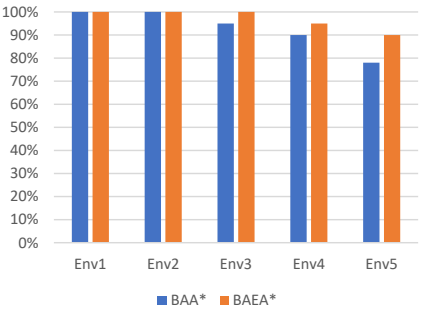
led it to prioritize exploration. As the number of training iterations increased significantly, the planned path length progressively decreased and ultimately converged to a stable state. In conclusion, the length of the planned path decreases as the algorithm is trained. This means that the algorithm is finding shorter and more efficient paths. Overall, the training results show that the proposed algorithm is effective in learning to plan paths in complex environments.

To rigorously assess the efficacy and accuracy of the proposed fusion methodology, distinct environmental configurations denoted as Env1, Env2, …, and Env5 were individually instantiated as separate map instances. To facilitate equitable comparisons among our proposed algorithm, denoted as the Bi-directional adaptive Enhanced A* algorithm (BAEA*), and the Bi-directional adaptive A* algorithm (BAA*) [18], we meticulously maintained uniformity in experimental conditions by subjecting each algorithm to precisely 40 iterations, and the algorithmic parameters were meticulously tuned through preliminary experimentation. To alleviate potential variability in the experiments, each algorithm was executed ten times within each distinct scenario. The evaluation criteria centered on the planned path's length and the mission's success rate, with success being defined by missions devoid of any collisions.

Figure 5 illustrates the operational outcomes of our proposed BAEA* algorithm across five discrete scenarios. These findings exemplify the proficient navigation capabilities of BAEA*, particularly in traversing complex terrains featuring diverse threat zones while avoiding collisions. It effectively maneuvers through areas with local minima to reach target positions surrounded by obstacles. Compared to the BAA* algorithms, our proposed algorithm showcases superior adaptability to diverse environments, ensuring secure task completion.

Figure 6 provides a comparative analysis of the planned path lengths between the two algorithms, underscoring BAEA*'s notable advantage in achieving shorter and more consistent path planning. The culmination of these experiments collectively validates that our proposed BAEA* algorithm surpasses traditional path-planning approaches, delivering superior performance, stability, and robustness.

Table 2 depicts the comparative performance of two algorithms, BAA* and BAEA*, across various environmental conditions. Each algorithm's effectiveness is assessed based on two key metrics: the cost of the generated path measured in nodes, and the time taken to compute the solution in seconds. Across all tested environments, BAEA* consistently demonstrates superior performance compared to BAA*. This superiority is notably reflected in the lower cost of the paths produced by BAEA*, indicating that it finds solutions that traverse fewer nodes in the search space. For example, in Environment 1, BAEA* achieves a

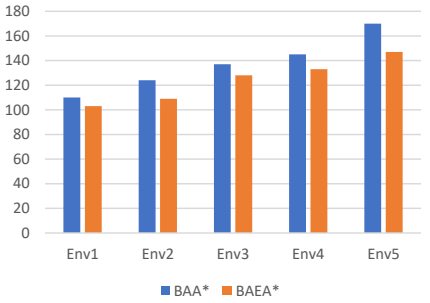


Source(s): Authors' own creation

Figure 5.
The average success rate of path planning in five distinct environmental settings

cost of 103 nodes, while BAA* incurs a higher cost of 110 nodes. This trend persists across all environments, with BAEA* consistently yielding lower path costs compared to BAA*. Additionally, BAEA* exhibits slightly faster computational times across most environments compared to BAA*. While the differences in execution times are relatively small, they still contribute to BAEA*'s overall efficiency advantage. For instance, in Environment 1, BAEA* completes the computation in 20.9 seconds, whereas BAA* takes slightly longer at 21.3 seconds. This pattern continues in subsequent environments, with BAEA* consistently showcasing marginally quicker execution times than BAA*. The improved performance of BAEA* can be attributed to its adaptive exploration strategy, which allows it to dynamically adjust its search process based on environmental conditions and task complexities. By intelligently exploring the search space, BAEA* can efficiently navigate through the environment and find high-quality solutions while minimizing computational overhead. In summary, the results clearly indicate that BAEA* offers superior performance over BAA* in terms of solution quality and computational efficiency across a range of environmental settings. This underscores the effectiveness of adaptive exploration techniques in enhancing the efficacy of path planning algorithms.

Figure 7 shows the path generated by two different path planning algorithms: (a) BAA* and (b) BAEA*. The two solutions obviously exhibit favorable characteristics; however, it can be observed that BAEA* produces a particularly smooth and short path in combination with BAA*.



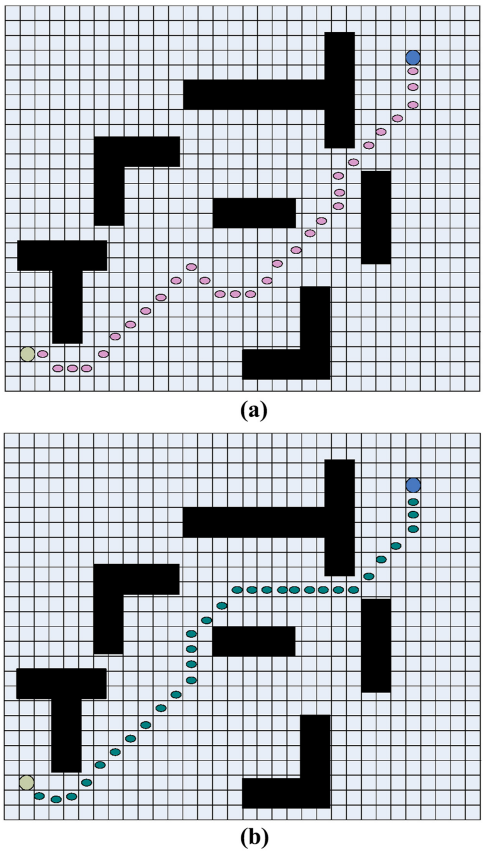
Source(s): Authors' own creation

Figure 6. Path length results in five distinct environmental settings

Methods	Environment	Cost (node)	Time(sec)
BAA*	Environment 1	110	21.3
BAEA*		103	20.9
BAA*	Environment 2	124	22.7
BAEA*		109	21.4
BAA*	Environment 3	137	23.8
BAEA*		128	23.5
BAA*	Environment 4	145	26.1
BAEA*		133	25.6
BAA*	Environment 5	170	29.3
BAEA*		147	26.2

Source(s): Authors' own creation

Table 2. Experimental comparison of path planning algorithms: cost and convergence across various environments



Source(s): Authors' own creation

Figure 7.
Smooth path generated by (a) BAA*, (b) BAEA*

6. Conclusion

A novel algorithm Bi-directional Adaptive Enhanced A* algorithm (BAEA*) was proposed based on [18]. Our proposed algorithm for mobile robot path planning offers several significant advantages. Firstly, it excels in efficiency, enabling real-time path identification. Secondly, it exhibits the capability to discern paths that not only avoid obstacles but also adhere to various constraints, enhancing safety and precision. Moreover, the algorithm demonstrates robustness in navigating through environmentally changeable surroundings, adapting effectively to alterations in the environment. Lastly, its scalability is a key asset, allowing it to be applied seamlessly to large and intricate environments, making it a versatile solution for a wide range of practical applications.

We tested our new algorithm BAEA* against the algorithm BAA*. We used five different environments, each with different obstacles and threat zones. We ran each algorithm 40 times in each environment to get an accurate comparison. We evaluated the algorithms on two criteria: the length of the path they planned and the success rate of the mission (i.e. whether the mobile robot reached its destination without colliding with an obstacle or threat zone). Our algorithm outperformed the other algorithm on both criteria. It was able to plan shorter and more stable paths, and it had a higher success rate in all five environments. This

shows that our algorithm is a better choice for path planning in complex and challenging environments.

Looking ahead, our research will focus on several key areas: (1) Dynamic obstacle avoidance, as the current work exclusively deals with static obstacles, introducing dynamic obstacles to the path planning process to enhance its complexity. (2) Transitioning our method from simulation to practical implementation on real-world robots, acknowledging that the ideal conditions assumed in this paper may not align with the complexities and uncertainties inherent in real-life environments.

References

1. Bottin M, Rosati G, Cipriani G. Iterative path planning of a serial manipulator in a cluttered known environment. In: Advances in Italian Mechanism Science: Proceedings of the 3rd International Conference of IFToMM Italy 3. Springer International Publishing; 2021. p. 237-44.
2. Ayawli BBK, Mei X, Shen M, Appiah AY, Kyeremeh F. Optimized RRT-A* path planning method for mobile robots in partially known environment. Inf Technol Control. 2019; 48(2): 179-94. doi: 10.5755/j01.itc.48.2.21390.
3. Chang L, Shan L, Jiang C, Dai Y. Reinforcement based mobile robot path planning with improved dynamic window approach in unknown environment. Autonomous Robots. 2021; 45(1): 51-76. doi: 10.1007/s10514-020-09947-4.
4. Choueiry S, Owayjan M, Diab H, Achkar R. Mobile robot path planning using genetic algorithm in a static environment. In: 2019 Fourth International Conference on Advances in Computational Tools for Engineering Applications (ACTEA). IEEE; 2019. p. 1-6.
5. Zhong X, Tian J, Hu H, Peng X. Hybrid path planning based on safe A* algorithm and adaptive window approach for mobile robot in large-scale dynamic environment. J Intell Robotic Syst. 2020; 99(1): 65-77. doi: 10.1007/s10846-019-01112-z.
6. Hentout A, Maoudj A, Aouache M. A review of the literature on fuzzy-logic approaches for collision-free path planning of manipulator robots. Artif Intelligence Rev. 2023; 56(4): 3369-444. doi: 10.1007/s10462-022-10257-7.
7. Das PK, Jena PK. Multi-robot path planning using improved particle swarm optimization algorithm through novel evolutionary operators. Appl Soft Comput. 2020; 92: 106312. doi: 10.1016/j.asoc.2020.106312.
8. Miao C, Chen G, Yan C, Wu Y. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. Comput Ind Eng. 2021; 156: 107230. doi: 10.1016/j.cie.2021.107230.
9. Muthugala MVJ, Samarakoon SBP, Elara MR. Toward energy-efficient online complete coverage path planning of a ship hull maintenance robot based on glasius bio-inspired neural network. Expert Syst Appl. 2022; 187: 115940. doi: 10.1016/j.eswa.2021.115940.
10. Bae H, Kim G, Kim J, Qian D, Lee S. Multi-robot path planning method using reinforcement learning. Appl Sci. 2019; 9(15): 3057. doi: 10.3390/app9153057.
11. Qu C, Gai W, Zhong M, Zhang J. A novel reinforcement learning based grey wolf optimizer algorithm for unmanned aerial vehicles (UAVs) path planning. Appl soft Comput. 2020; 89: 106099. doi: 10.1016/j.asoc.2020.106099.
12. Maoudj A, Hentout A. Optimal path planning approach based on Q-learning algorithm for mobile robots. Appl Soft Comput. 2020; 97: 106796. doi: 10.1016/j.asoc.2020.106796.
13. Ajabshir VB, Guzel MS, Bostanci E. A low-cost Q-learning-based approach to handle continuous space problems for decentralized multi-agent robot navigation in cluttered environments. IEEE Access. 2022; 10: 35287-301. doi: 10.1109/access.2022.3163393.
14. Tu GT, Juang JG. UAV path planning and obstacle avoidance based on reinforcement learning in 3d environments. Actuators. 2023; 12(2): 57. doi: 10.3390/act12020057.

15. Xia X, Li T, Sang S, Cheng Y, Ma H, Zhang Q, Yang K. Path planning for obstacle avoidance of robot arm based on improved potential field method. Sensors. 2023; 23(7): 3754. doi: 10.3390/s23073754.

16. Khanna S, Srivastava S. Path planning and obstacle avoidance in dynamic environments for cleaning robots. Q J Emerging Tech Innov. 2023; 8(2): 48-61.

17. Almazrouei K, Kamel I, Rabie T. Dynamic obstacle avoidance and path planning through reinforcement learning. Appl Sci. 2023; 13(14): 8174. doi: 10.3390/app13148174.

18. Wu X, Xu L, Zhen R, Wu X. Bi-directional adaptive A* algorithm toward optimal path planning for large-scale UAV under multi-constraints. IEEE Access. 2020; 8: 85431-40. doi: 10.1109/access.2020.2990153.

19. Gharbi A. A dynamic reward-enhanced Q-learning approach for efficient path planning and obstacle avoidance in mobile robotics. Appl Comput Inform. 2024. doi: 10.1108/aci-10-2023-0089.

Corresponding author
Atef Gharbi can be contacted at: atef.gharbi@nbu.edu.sa