IJRCS

International Journal of Robotics and Control Systems
Vol. 6, No. 2, 2026, pp. 1476-1501
ISSN 2775-2658
http://pubs2.ascee.org/index.php/ijrcs

ASCEE

Structural and Computational Analysis of A* and Backtracking for Deterministic Offline Grid-Based Path Planning

Phichitphon Chotikunnan [a,1], Sunisa Wichaiwong [a,2], Kotchapon Bunnak [a,3], Rawiphon Chotikunnan [a,4], Wanida Khotakham [a,5], Yutthana Pititheeraphab [a,6], Tasawan Puttasakul [a,7], Nuntachai Thongpance [a,8,*]

[a] College of Biomedical Engineering, Rangsit University, Pathum Thani 12000, Thailand
[1] phichitphon.c@rsu.ac.th; [2] sunisa.w65@rsu.ac.th; [3] kotchapon.b65@rsu.ac.th; [4] rawiphon.c@rsu.ac.th; [5] wanida.k@rsu.ac.th; [6] yutthana.p@rsu.ac.th; [7] tasawan.p@rsu.ac.th; [8] nuntachai.t@rsu.ac.th
* Corresponding Author

ARTICLE INFO

Article history
Received January 12, 2026
Revised March 04, 2026
Accepted May 03, 2026

Keywords
A*;
Path Planning;
Backtracking;
Grid Map;
Simulation;
Multi-Criteria

ABSTRACT

Offline path planning in deterministic grid environments requires balancing shortest-path optimality with computational scalability as structural branching complexity increases. This study presents a controlled simulation-based analysis of two search paradigms: the heuristic-guided A* algorithm and the exhaustive Backtracking method. The main contribution is a unified multi-criteria evaluation framework that enables systematic structural comparison under identical movement constraints, allowing transparent assessment of path optimality, turning behavior, and computational search effort. Twelve deterministic grid maps were constructed, including six maps of size 6×6 and six maps of size 12×12, with fixed start and goal positions to investigate the impact of structural complexity. Both algorithms were implemented in C using four-directional unit-cost movement and evaluated using multiple metrics, including path length, number of turns, movement score, expanded nodes, neighbor checks, and backtracking operations. Simulation results show that both algorithms consistently achieve shortest-path optimality, producing paths of 10 steps in 6×6 maps and 22 steps in 12×12 maps. However, substantial differences appear in computational effort. In the 12×12 maps, A* expanded between 27 and 70 nodes, whereas Backtracking expanded up to 943 nodes and required up to 3740 neighbor checks and more than 900 backtracking operations in highly branched configurations. These results demonstrate exponential growth in exhaustive exploration compared with polynomial growth under heuristic guidance, highlighting the structural trade-off between exhaustive completeness and heuristic-guided computational scalability in deterministic offline grid planning.

1. Introduction

Path planning is a fundamental capability in autonomous robotics because it determines whether a mobile robot can reach a target safely, efficiently, and reliably under environmental constraints. In practical applications such as industrial logistics, warehouse automation, and service robotics, navigation performance depends not only on geometric feasibility but also on the interaction between planning algorithms, environmental structures, obstacle configurations, and system limitations.

Consequently, path-planning algorithms remain an important research topic encompassing classical graph-search techniques, heuristic approaches, and modern learning-based navigation frameworks [1]-[4]. Survey studies consistently report that algorithm performance varies depending on environmental characteristics, objective functions, and computational limitations, indicating that no single algorithm consistently outperforms others across all scenarios. Therefore, carefully designed experimental frameworks are necessary to analyze algorithmic behavior under controlled conditions and to evaluate performance trade-offs without introducing unnecessary environmental complexity [5].

Recent developments in robotic navigation have expanded evaluation criteria beyond simple shortest-path computation. Navigation performance is influenced by environmental representation, system constraints, and the interaction between planning and sensing components [6]. Studies also highlight the diversity of classical, heuristic, and metaheuristic planning approaches used in robotic navigation systems [7]. In parallel, learning-based navigation frameworks, particularly those based on deep reinforcement learning, have attracted increasing attention [8], while multi-robot navigation research emphasizes efficiency and coordination in practical deployments [9]. Research on task and motion planning further demonstrates that practical autonomy often requires integrating symbolic task objectives with geometric feasibility [10]. In addition, energy-aware navigation studies indicate that route selection and motion characteristics can significantly affect operational efficiency, encouraging evaluation metrics that consider multiple performance aspects beyond path length alone [11], [12].

Despite the growing interest in learning-based navigation, interpretable environmental representations remain essential when the objective is to isolate algorithmic search behavior and enable fair comparisons among planning strategies. Grid-based modeling is widely adopted due to its transparency, reproducibility, and compatibility with graph-search and discrete optimization techniques. Previous studies demonstrate that discretization choices, grid resolution, neighborhood connectivity, and movement constraints can significantly influence both path optimality and computational complexity [13], [14]. Furthermore, learning-based navigation systems operating in structured environments often rely on simplified map representations during development and evaluation to ensure repeatability and controlled experimental conditions [15]. Consequently, deterministic grid environments remain appropriate when the objective is to analyze structural algorithmic behavior rather than reproduce full perceptual complexity.

In this study, the offline path-planning problem is defined as determining a collision-free path between a predefined start position and goal position within a known two-dimensional grid environment containing static obstacles. The environment is discretized into grid cells where each cell represents a traversable or blocked location. The navigation model assumes four-directional connectivity (up, down, left, and right) with uniform movement cost between adjacent cells. Obstacles are represented as non-traversable grid cells that constrain feasible path construction. These assumptions enable controlled evaluation of search behavior while maintaining consistency across all experimental scenarios.

Heuristic-guided graph-search algorithms remain widely used baseline methods in classical path-planning research due to their transparency and computational efficiency. Lifelong Planning A* demonstrates how heuristic search methods can reuse previous computations when environmental costs change, highlighting the continuing relevance of heuristic-guided planning in modern robotic systems [16]. Analyses of A*-based pathfinding emphasize common evaluation metrics such as explored states, node expansions, and runtime performance [17]. Furthermore, studies on environmental representation and occupancy modeling show that map configurations strongly influence path feasibility and search complexity, emphasizing the importance of controlling obstacle distributions and connectivity during benchmarking experiments [18]. Alternative map representations such as visibility graphs combined with quadtree decomposition further demonstrate how representation choices influence search difficulty and path structure [19].

Various extensions of the A* algorithm have been proposed to improve efficiency and adaptability across different navigation scenarios. Bidirectional search and Jump Point Search

techniques reduce node expansions while maintaining acceptable path quality in grid environments [20]. Other studies investigate algorithm robustness under varying obstacle densities, showing that heuristic design and cost modeling significantly influence computational efficiency and trajectory characteristics [21], [22]. Hybrid approaches combining graph-search algorithms with optimization techniques such as simulated annealing or evolutionary heuristics have also been explored [23], [24]. Application-specific adaptations of A* have been developed for autonomous ground vehicles and unmanned aerial vehicles, where maneuver constraints and domain-specific cost functions must be incorporated into the planning framework [25], [26]. These developments indicate that A* continues to serve as a flexible baseline algorithm capable of supporting diverse navigation objectives [27], [28].

Beyond deterministic graph-search approaches, alternative planning paradigms have been developed to address continuous and high-dimensional search spaces. Sampling-based methods such as probabilistic roadmaps (PRM) and rapidly exploring random trees (RRT*) efficiently explore large configuration spaces while maintaining probabilistic completeness [29], [30]. Other classical techniques such as radial cell decomposition divide the environment into manageable regions to facilitate systematic path construction [31]. Informative path-planning frameworks further extend navigation objectives by incorporating criteria such as information gain and exploration efficiency [32]. In practical navigation systems, perception and environmental representation also play critical roles. LiDAR-based mapping systems designed for dynamic environments aim to maintain consistent maps despite moving objects [33], while visual SLAM-based navigation systems demonstrate how sensing conditions influence navigation performance in structured environments such as healthcare facilities [34]. Vision-based navigation approaches using neural radiance field representations further illustrate how detailed environmental models can support planning and control in complex environments [35].

Learning-based navigation has also attracted increasing research attention. UAV navigation systems developed for warehouse inventory operations illustrate how navigation strategies can be integrated with mission objectives in structured environments [36]. Representation learning techniques such as contrastive world models enable robots to construct compact visual representations that support planning and decision-making [37]. Incremental learning frameworks for deep reinforcement learning further allow navigation systems to adapt continuously to changing environments [38]. However, these approaches also introduce additional uncertainty and learning dynamics, reinforcing the importance of controlled evaluation environments when analyzing fundamental planning behavior [39].

Alongside heuristic-guided methods, exhaustive search paradigms remain valuable for analyzing navigation problems because they systematically explore search spaces without heuristic bias. Backtracking-based exploration has been applied to offline robot transport and coverage planning tasks in grid environments, allowing researchers to analyze how branching patterns and map structures influence exploration behavior [40]-[42]. Similar enumeration-based approaches have also been used in directed energy path planning and graph traversal problems, demonstrating that traversal order and search completeness remain important considerations in optimization tasks [43], [44]. In addition, metaheuristic optimization techniques are frequently applied when planning problems involve multiple objectives or complex search landscapes. Hybrid methods combining deterministic shortest-path algorithms with population-based optimization have been explored in applications such as PSO-enhanced Dijkstra route planning and hybrid metaheuristic scheduling systems [45], [46]. Surveys of nature-inspired optimization algorithms highlight the wide range of metaheuristic techniques available for solving complex optimization problems [47]. Consequently, multi-objective path planning has become increasingly important because real robotic systems often need to balance competing objectives such as safety, travel time, and energy consumption [48]-[54].

Simulation plays a central role in evaluating path-planning algorithms under controlled experimental conditions. Simulation environments enable repeatable experiments and systematic variation of scenario complexity, making them suitable for analyzing algorithmic behavior independently from hardware constraints. Previous studies emphasize that transparent and

reproducible simulation frameworks are essential for reliable comparative evaluations [55]-[57]. In robotics research, simulation-based experiments allow navigation algorithms to be tested without interference from perception noise or hardware limitations [58], [59]. Maze-navigation simulations and physics-based robotic experiments further illustrate how simulation supports systematic evaluation under clearly defined environmental assumptions [60], [61].

In mobile robotic systems, path characteristics influence not only navigation decisions but also motion execution and control stability. The geometric properties of a path, including turning frequency and maneuvering patterns, can significantly affect control performance during movement. Studies on self-balancing robots show that trajectory geometry directly influences motion stability [62]. Research on two-wheeled robots integrating path planning with trajectory tracking control further indicates that turning behavior and path smoothness affect tracking accuracy and system stability [63]. Robust navigation approaches based on fuzzy logic and uncertainty-aware planning highlight the impact of environmental variability on navigation performance [64]. Additionally, studies on human-robot cooperation and multi-robot coordination demonstrate that navigation decisions often interact with higher-level task allocation mechanisms [65], [66].

Despite the extensive body of literature on path-planning algorithms, relatively few studies systematically examine the structural differences between heuristic-guided search and exhaustive search strategies under controlled experimental conditions. Although modern path-planning research often focuses on advanced heuristic variants or sampling-based methods, comparing a heuristic-guided algorithm with an exhaustive search strategy remains valuable for understanding fundamental search behavior. The Backtracking algorithm systematically explores all feasible paths without relying on heuristic guidance, guaranteeing search completeness but potentially incurring higher computational cost. In contrast, A* uses heuristic evaluation to guide exploration toward promising regions of the search space. Therefore, comparing these two fundamentally different search paradigms provides insight into the trade-off between heuristic-guided efficiency and exhaustive-search completeness under identical experimental conditions. In previous work, the Backtracking algorithm was applied to offline robot transport in grid environments and demonstrated feasibility through simulation-based evaluation [40]. Building upon that foundation, the present study extends the analysis toward a systematic structural comparison of search behavior by progressively increasing grid-map complexity.

The main contribution of this study is the development of an interpretable experimental framework that enables systematic structural comparison between heuristic-guided search scalability and exhaustive-search completeness in controlled grid-based environments. The proposed framework evaluates algorithm performance using multiple metrics including computational search effort, path optimality, turning characteristics, and movement smoothness. By analyzing how search strategies respond to increasing structural complexity, the study provides deeper insight into the trade-offs between heuristic-guided efficiency and exhaustive-search completeness. The findings provide a transparent and reproducible baseline for understanding fundamental search behavior and may support future extensions toward dynamic environments, multi-objective path planning formulations, and perception-integrated robotic navigation systems.

2. Method

This study presents a controlled comparative analysis of grid-based path-planning algorithms under deterministic offline simulation conditions. The primary objective is not to generalize the results to high-dimensional robotic configuration spaces, but rather to examine behavioral differences between distinct search paradigms within structured two-dimensional grid environments. The analysis focuses on computational behavior, search efficiency, and solution quality under controlled and reproducible experimental settings.

Two distinct search strategies were selected for evaluation: the A* algorithm, which represents heuristic-guided informed search, and the Backtracking algorithm, which represents exhaustive

depth-first exploration. The inclusion of Backtracking aims to provide a computational baseline illustrating the effects of exhaustive search in contrast to heuristic-guided exploration. Although Backtracking is not considered a modern standard for shortest-path algorithms in contemporary robotics research, it provides a clear reference for examining trade-offs among search completeness, exploration behavior, and computational effort when compared with heuristic-driven approaches such as A*.

Both algorithms were implemented in the C programming language and executed under identical experimental conditions to ensure fairness and reproducibility. All evaluations were conducted within a simulated grid environment, and no physical robot or real-time system was involved during the experiments. The environment is modeled as a discrete two-dimensional grid with four-directional connectivity (up, down, left, and right), uniform movement cost between adjacent cells, and static obstacle representation. To maintain consistency across experiments, the start and goal positions were fixed for all map configurations.

Fig. 1 illustrates the overall experimental workflow employed in this study to evaluate the A* and Backtracking algorithms in grid-based path planning. The procedure begins with grid map initialization, where the simulated environment is configured as a discrete two-dimensional grid with predefined obstacle configurations. After the environment is established, the start and goal positions are defined to specify the navigation task. Both algorithms are then executed under identical environmental conditions to ensure a fair and unbiased comparison. The A* algorithm performs heuristic-guided search using the Manhattan-distance heuristic, while the Backtracking algorithm explores the search space exhaustively through depth-first search. Once a feasible path is identified, the resulting trajectory is extracted for further analysis. The generated paths are subsequently evaluated using several path-quality metrics, including path distance, number of turns, and movement score. In addition to path characteristics, computational effort metrics such as the number of expanded nodes, neighbor checks, and execution time are recorded to assess algorithmic efficiency. Finally, the performance data obtained from both algorithms are systematically compared to analyze differences in search behavior, computational cost, and overall path-planning performance across the evaluated grid environments.

Fig. 2 illustrates the creation of twelve deterministic, manually constructed grid maps for experimental assessment. The collection consists of six maps with dimensions of 6 × 6 and six maps with dimensions of 12 × 12. The maps vary in obstacle density, corridor width, branching configuration, and maze-like structures. The deterministic design enables a systematic assessment of algorithmic performance under controlled variations in environmental complexity while maintaining consistent boundary conditions. The selected maps, although lacking extensive randomized environments, demonstrate increasing structural complexity, enabling the analysis of scaling tendencies in both smaller and moderately larger grid environments. The performance evaluation encompassed multiple metrics, including path distance, number of turns, movement score, node expansion count, neighbor checks, execution time, and memory utilization. Memory usage was evaluated based on the data structures required by each algorithm. In the Backtracking implementation, recursion depth was considered an indicator of stack memory utilization during the search process. This multi-criteria evaluation enables a comprehensive analysis of path quality and computational search effort. The methodological framework seeks to isolate algorithmic behavior in deterministic offline environments. This approach enables a clear comparison between heuristic-guided search and exhaustive exploration while ensuring experimental consistency and analytical rigor.

2.1. Simulation Environment Design and Grid Map Representation

The simulation environment was defined as a two-dimensional square grid map of size $N \times N$, where each grid cell represents a discrete navigable spatial unit for the robot. Two map configurations were evaluated in this study, namely $N = 6$ and $N = 12$, to investigate algorithmic performance under different spatial complexities.

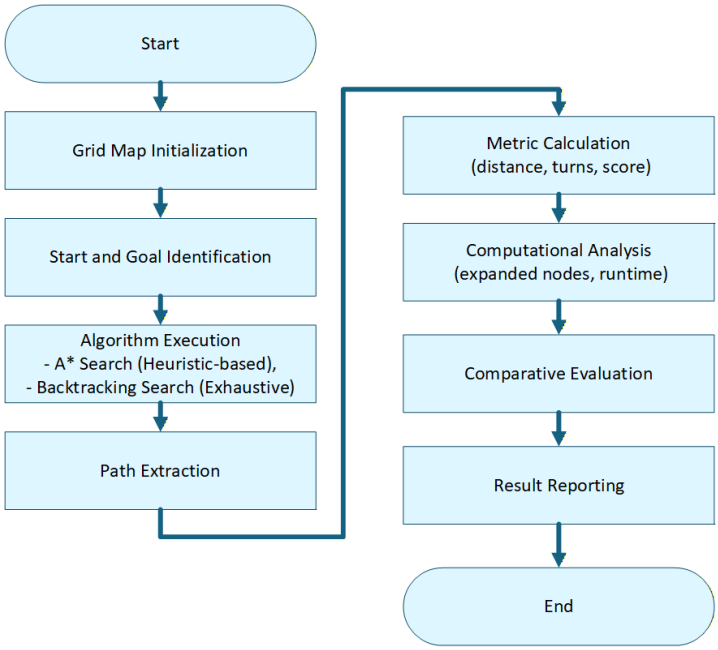


Fig. 1. Flowchart of the experimental framework for comparing A* and backtracking algorithms in grid-based path planning

The environment was represented using a binary occupancy matrix as defined in (1).

$$m \in \{0,1\}^{N \times N} \tag{1}$$

where $m(x, y) = 1$, denotes a traversable cell and $m(x, y) = 0$, denotes an obstacle or restricted region. The map was assumed to be static, meaning that obstacle positions remained fixed throughout each experiment. Path planning was performed under an offline computation framework in which the complete map information was available prior to execution. No dynamic obstacle updates or stochastic elements were introduced, thereby ensuring deterministic and reproducible experimental conditions.

To maintain structural independence between obstacle configuration and terminal states, the initial and goal positions were defined separately from the occupancy matrix using a dedicated coordinate definition. Across all evaluated scenarios, the robot started from the upper-left corner (0, 0) and aimed to reach the lower-right corner $(N - 1, N - 1)$.

The robot operated under a four-neighbor connectivity model, permitting movement only in the cardinal directions: right, left, down, and up. Accordingly, the action set was defined as (2).

$$A = \{(0,1), (0,-1), (1,0), (-1,0)\} \tag{2}$$

Given the current state $n = (x, y)$, a transition to the subsequent state $n' = (x', y')$is governed by (3).

$$(x', y') = (x, y) + a,\ a \in A \tag{3}$$

Each valid movement between adjacent cells incurs a uniform cost of one unit. This uniform edge-cost model ensures consistency across all simulation experiments and supports compatibility with the heuristic formulation employed in the A* algorithm.

To validate state transitions, a safety function was defined as follows (4).

$$\text{Safe}(x, y) = \begin{cases} 1, & 0 \le x < N,\ 0 \le y < N,\ m(x, y) = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

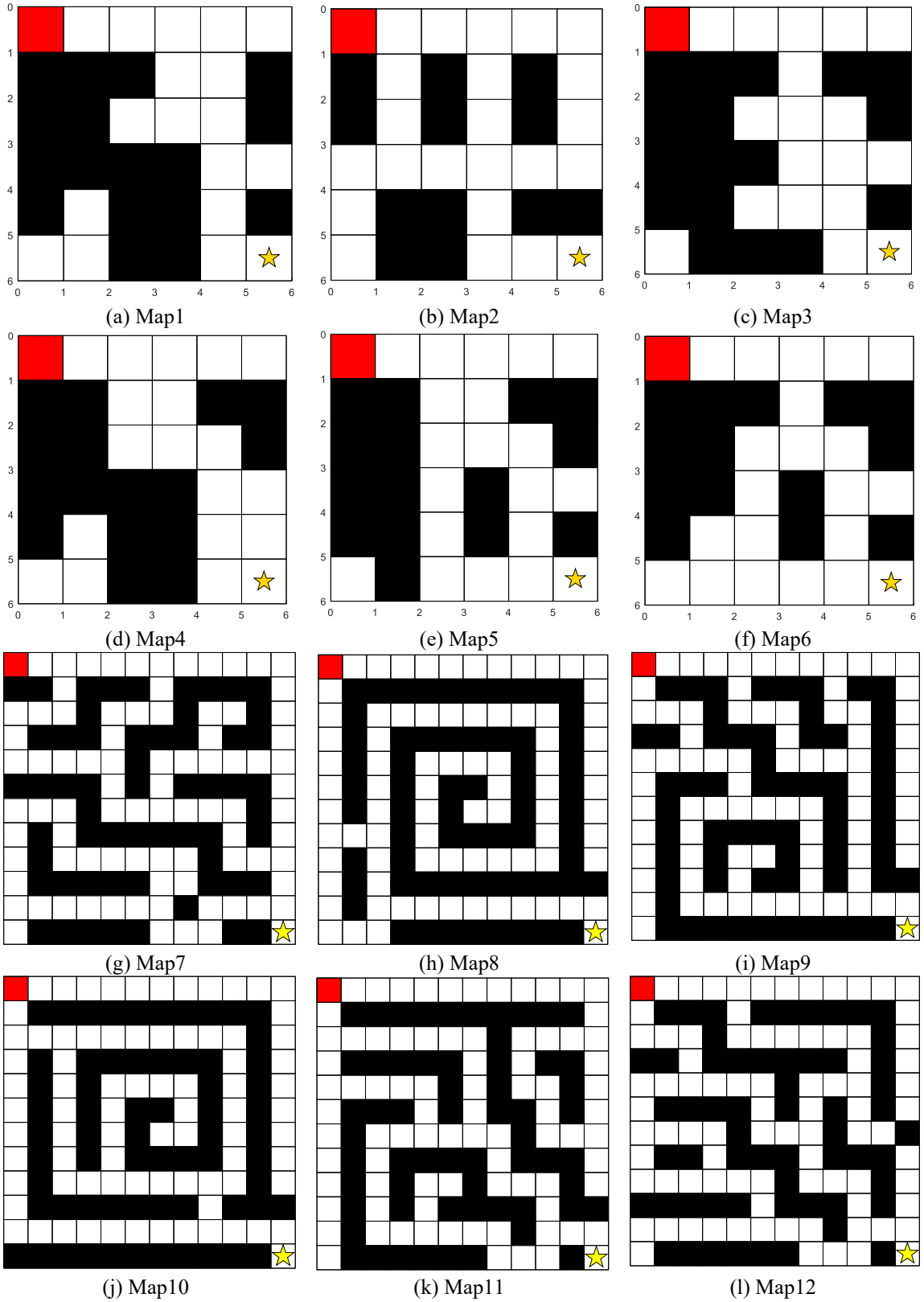


Fig. 2. Deterministic grid environments used for the comparative evaluation of A* and backtracking algorithms, consisting of six 6×6 maps (a–f) and six 12×12 maps (g–l)

A transition to state $n' = (x', y')$ is permitted only if $Safe(x', y') = 1$. This formulation ensures that both the A* and Backtracking algorithms operate under identical environmental

constraints, boundary conditions, and movement rules across all simulation experiments. Fig. 3 conceptually illustrates the four-directional movement model and the verification of the safety criterion $Safe(x, y)$.

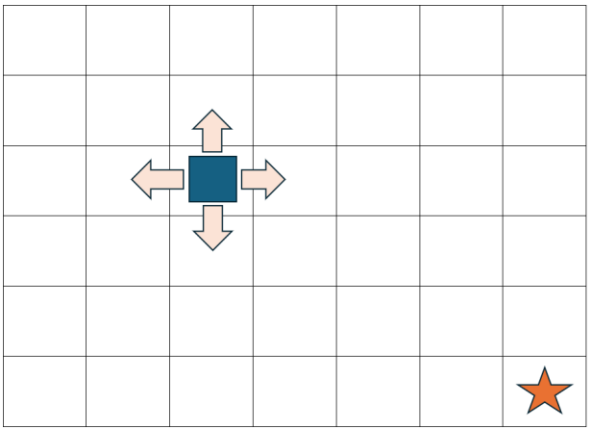


Fig. 3. conceptually illustrates the four-directional movement model and the verification of the safety criterion *Safe*(x,y)

2.2. A* Algorithm for Evaluation-Function-Based Path Planning

The A* algorithm is an informed heuristic search method that expands the most promising state at each iteration based on an evaluation function. The evaluation function combines the accumulated path cost from the start state with an estimated cost to the goal state and is defined as (5).

$$f(n) = g(n) + h(n) \tag{5}$$

where $n = (x_n, y_n)$ denotes the current grid state, $g(n)$represents the accumulated cost from the start state to state n, and $h(n)$ denotes the heuristic estimate of the remaining cost from state n to the goal state.

Since the grid model defined in Section 2.1 adopts four-directional connectivity with uniform unit cost, each transition between adjacent cells incurs a cost of one. Accordingly, the accumulated cost function is defined recursively as (6).

$$g(n) = g\big(\text{parent}(n)\big) + 1, g(n_{\text{Start}}) = 0 \tag{6}$$

where $parent(n)$ denotes the predecessor of state $n$in the search tree, and n_{start} represents the initial state.

The heuristic function used in this study is the Manhattan distance, which is suitable for grid-based environments with four-directional movement. The heuristic function is defined as (7).

$$h(n) = | x_n - x_g | + | y_n - y_g | \tag{7}$$

where (x_g, y_g) denotes the coordinates of the goal state. Under uniform unit-cost movement and four-directional connectivity, the Manhattan heuristic is both admissible and consistent. Consequently, the A* algorithm guarantees optimality within the defined grid environment.

The implementation of the A* algorithm employs two primary data structures: the OPEN set and the CLOSED set. The OPEN set contains discovered states that are candidates for expansion, whereas the CLOSED set stores states that have already been expanded and will not be revisited.

At each iteration, the algorithm selects the state $n \in OPEN$ that has the minimum evaluation value $f(n)$. The selected state is transferred to the CLOSED set. If the selected state corresponds to the goal state, the search terminates and the optimal path is reconstructed by tracing parent links from the goal state back to the start state.

For each expanded state n, neighboring states n' are generated according to the four-directional movement model defined in Section 2.1. A neighboring state is considered only if it satisfies the safety condition $Safe(x', y') = 1$ and is not already contained in the CLOSED set. A* temporary accumulated cost is then computed as defined in (8).

$$g_{\text{temp}}(n') = g(n) + 1 \tag{8}$$

The neighboring state is updated if the condition defined in (9) is satisfied.

$$n' \notin OPEN \text{ or } g_{temp}(n') < g(n') \tag{9}$$

When this update condition holds, the parent of n' is reassigned according to (10).

$$parent(n') = n \tag{10}$$

And its cost values are updated as defined in (11).

$$g(n') = g_{temp}(n'), \qquad f(n') = g(n') + h(n') \tag{11}$$

If n' is not already present in the OPEN set, it is inserted accordingly.

The procedural structure of the implemented A* algorithm is summarized in Fig. 4. The algorithm begins by inserting the start state into the OPEN set with an initial cost of zero, while the CLOSED set is initialized as empty. During each iteration, the state with the minimum evaluation value is expanded, and neighboring states are evaluated under the defined movement and safety constraints.

This formulation directly corresponds to the implemented program structure, including node expansion, neighbor evaluation, OPEN insertion, cost relaxation, and parent tracking. The explicit mathematical definitions facilitate accurate measurement of computational metrics such as expanded nodes, neighbor checks, and relaxation updates, thereby ensuring a rigorous and fair comparison with the Backtracking algorithm.

2.3. Backtracking Algorithm for Exhaustive Path Exploration and Optimal Path Determination

In this study, the Backtracking algorithm is employed to exhaustively explore all feasible paths from the initial state to the goal state within the grid environment. The algorithm strictly adheres to the safety constraint defined in Section 2.1 and enforces the restriction that no grid cell may be revisited within the same path.

To prevent revisiting states during depth-first exploration, a visitation matrix is defined as (12).

$$Visited(x, y) \in \{0,1\} \tag{12}$$

where $Visited(x, y) = 1$ indicates that the cell $(x, y)$is currently included in the active path, and $Visited(x, y) = 0$ indicates that the cell is not part of the current path and is therefore eligible for exploration.

Let the current path be represented as an ordered sequence of states defined in (13).

$$P = \{n_0, n_1, \dots, n_{L-1}\} \tag{13}$$

where $n_0 = n_{start}$, n_{L-1} denotes the most recently added state, and L represents the number of states in the path.

The Backtracking algorithm operates recursively using depth-first search principles. When the algorithm is at state $n_k = (x_k, y_k)$, it attempts to expand to adjacent states n_{k+1} generated under the

four-directional movement model defined in Section 2.1. A candidate state $n_{k+1} = (x_{k+1}, y_{k+1})$ is considered valid only if it satisfies the safety and visitation constraints defined in (14) and (15).

```
Input:
  Start node s, Goal node g
  Grid map m, Action set A, Safety function Safe(·)

Output:
  Shortest path from s to g, or failure

1:  Initialize OPEN with s
2:  Initialize CLOSED as empty
3:  Set g(s) = 0
4:  Compute h(s) using Manhattan distance
5:  Set f(s) = g(s) + h(s)
6:  Set parent(s) = NULL

7:  while OPEN is not empty do
8:      Select node n in OPEN with the smallest f(n)
9:      Move n from OPEN to CLOSED

10:     if n is the goal node then
11:         Reconstruct and return the path
12:     end if

13:     for each possible movement direction do
14:         Generate neighboring node n'
15:         if n' is unsafe or already in CLOSED then
16:             Skip this neighbor
17:         end if

18:         Compute temporary cost g_temp = g(n) + 1
19:         if n' is not in OPEN or g_temp < g(n') then
20:             Update parent of n' to n
21:             Update g(n'), h(n'), and f(n')
22:             Add n' to OPEN if not already included
23:         end if
24:     end for
25: end while

26: return failure
```

Fig. 4. Pseudocode of the A* algorithm for grid-based path planning

$$Safe(x_{k+1}, y_{k+1}) = 1 \tag{14}$$

$$Visited(x_{k+1}, y_{k+1}) = 0 \tag{15}$$

If both conditions hold, the state is appended to the current path as defined in (16).

$$P_{k+1} = P_k \cup \{n_{k+1}\} \tag{16}$$

Simultaneously, the visitation matrix is updated as (17).

$$Visited(x_{k+1}, y_{k+1}) = 1 \tag{17}$$

The algorithm is then recursively invoked from state n_{k+1}.

When the current state corresponds to the goal state n_{goal}, the constructed path is treated as a complete solution. The solution is evaluated according to the predefined path-quality criteria described in Section 2.4, and it is compared against the best solution identified thus far.

After exploring all feasible extensions from a state, the algorithm performs a backtracking operation. The backtracking step removes the most recently appended state from the path and resets its visitation status as defined in (18) and (19).

$$P_k = truncate(P_{k+1}) \tag{18}$$

$$Visited(x_{k+1}, y_{k+1}) = 0 \tag{19}$$

where $truncate(\cdot)$ removes the last element of the ordered path sequence.

If

$$P_{k+1} = (n_0, n_1, \dots, n_k, n_{k+1}),$$

then

$$truncate(P_{k+1}) = (n_0, n_1, \dots, n_k).$$

This process continues until all feasible paths from the start state to the goal state have been exhaustively explored.

Let $\mathcal{P}_{sol}$ denote the set of all distinct solution paths that reach the goal state during exhaustive exploration. The total number of such solution paths is defined in (20).

$$N_{sol} = |\mathcal{P}_{sol}| = |\{P_i \mid P_i \text{ is a valid path and } n_{L-1} = n_{goal}\}| \tag{20}$$

In this study, the Backtracking algorithm systematically enumerates all feasible solution paths for each grid map. The value of N_{sol} reflects the structural complexity of the environment and the number of alternative routing possibilities. In addition, the number of dead ends encountered and the total number of backtracking operations are recorded to characterize the computational burden associated with exhaustive search strategies. The procedural structure of the implemented Backtracking algorithm is summarized in Fig. 5.

2.4. Path Quality Evaluation: Distance, Number of Turns, and Movement Scoring Function

To ensure a fair and consistent comparison between the A* and Backtracking algorithms, this study employs a unified set of path evaluation metrics. Three principal evaluation dimensions are considered: path length (number of steps), number of turns, and a movement score that reflects path smoothness and directional change penalties.

Let a solution path be defined as an ordered sequence of grid states, as defined in (21).

$$P = \{n_0, n_1, \dots, n_{L-1}\} \tag{21}$$

where L denotes the total number of states in the path, $n_0 = n_{start}$, and $n_{L-1} = n_{goal}$.

The path length, expressed as the total number of transitions between consecutive states, is defined in (22).

$$D = L - 1 \tag{22}$$

To quantify directional changes, the movement vector at step $i$is defined in (23).

$$\begin{gathered} \Delta_i = n_i - n_{i-1} \\ \Delta_i = (x_i - x_{i-1},\, y_i - y_{i-1}), i = 1, 2, \dots, L-1 \end{gathered} \tag{23}$$

Using the movement vectors defined in (23), the total number of turns is defined in (24) as the number of instances in which the movement direction differs from that of the preceding step.

$$T = \sum_{i=2}^{L-1} \mathbf{1}(\Delta_i \neq \Delta_{i-1}) \tag{24}$$

where $1(\cdot)$ denotes the indicator function, which returns 1 when the specified condition is satisfied and 0 otherwise.

```
Input:
    Start state n0
    Goal state Goal
    Grid map with safety function Safe(x, y)

Output:
    Set of solution paths P_sol
    Total number of solution paths N_sol

Initialize:
    P = { n0 }
    Visited(x, y) = 0 for all grid cells
    Visited(n0) = 1
    P_sol = □
    N_sol = 0

Procedure Backtrack(n_k):

    if n_k = Goal then
        P_sol = P_sol ∪ { P }
        N_sol = N_sol + 1
        return
    end if

    for each neighboring state n_{k+1} of n_k do
        if Safe(x_{k+1}, y_{k+1}) = 1 and Visited(x_{k+1}, y_{k+1}) = 0 then
            P = P ∪ { n_{k+1} }
            Visited(x_{k+1}, y_{k+1}) = 1

            Backtrack(n_{k+1})

            P = { n_0, n_1, …, n_k }
            Visited(x_{k+1}, y_{k+1}) = 0
        end if
    end for

End Procedure

Call Backtrack(n0)
```

Fig. 5. Pseudocode of the backtracking-based exhaustive path exploration algorithm

In addition to distance and number of turns, a movement scoring function is introduced to penalize directional changes. The step-wise cost is defined in (25).

$$c_i = \begin{cases} 1, & i = 1 \text{ or } \Delta_i = \Delta_{i-1} \\ 2, & i \geq 2 \text{ and } \Delta_i \neq \Delta_{i-1} \end{cases} \tag{25}$$

The cumulative movement score of a path is then defined in (26).

$$\text{Score} = \sum_{i=1}^{L-1} c_i \tag{26}$$

This formulation embodies the principle that directional changes require greater control effort than linear movements; consequently, turns are assigned a higher penalty to enable the score to accurately represent the overall uniformity of the path.

The determination of the optimal path in this study employs a lexicographic ranking approach to distinctly prioritize primary and secondary objectives. Initially, the path with the shortest distance D, as defined in (22), is selected. If multiple paths share the same distance, the path with the minimal number of turns T, as defined in (24), is selected. If ties persist, the path with the lowest movement score, as defined in (26), is chosen. This hierarchical decision rule ensures consistent and transparent best-path selection across both A* and Backtracking algorithms under identical evaluation criteria.

The definitions of the path quality metrics and corresponding scoring functions used for best-path selection are summarized in Table 1.

Table 1. Definitions of path quality metrics and scoring functions used for best-path selection

Metric	Symbol	Definition	Equation	Preferred Value
Path length (steps)	D	Total number of state transitions in the path	$D = L - 1$	Smaller
Number of turns	T	Number of direction changes between consecutive steps	$T = \sum_{i=2}^{L-1} \mathbf{1}(\Delta_i \neq \Delta_{i-1})$	Smaller
Movement score	Score	Sum of step-wise costs (straight = 1, turn = 2)	$\text{Score} = \sum_{i=1}^{L-1} c_i$	Smaller
Step cost	c_i	Cost assigned to step $i$based on direction change	$c_i \in \{1,2\}$	Smaller

2.5. Computational Performance Metrics and Experimental Data Collection

To analyze the computational cost of both algorithms, this study defines a set of quantitative performance metrics that can be directly counted from the algorithmic execution and enable one-to-one comparison. The common metrics applied to both algorithms include the number of expanded states N_{exp}, which counts each time a state is selected for expansion; the number of neighbor checks N_{nbr}, which counts each attempted movement to an adjacent cell; the number of rejections due to obstacles or boundary violations N_{blk}, which counts occurrences where $\text{Safe}(x, y) = 0$; and the number of skipped states N_{skp}, which records cases where a state is ignored because it has already been placed in the CLOSED set or marked as visited.

For the A* algorithm, additional algorithm-specific metrics are recorded, including the number of first-time insertions into the OPEN set N_{open} and the number of cost relaxations N_{relax}, which reflect instances where a better path to an already discovered state is identified through an update of the accumulated cost $g(n)$. These metrics provide insight into the dynamic behavior of the heuristic search process. For the Backtracking algorithm, additional metrics are recorded to characterize the burden of complete search, including the total number of solution paths discovered N_{sol}, the number of dead ends encountered N_{dead}, and the number of backtracking operations N_{back}. These indicators reflect the structural complexity of the search space and the exhaustive nature of the algorithm.

The experimental results for each map are stored as structured reports containing the best path obtained by A* and Backtracking, along with the corresponding path quality metrics, namely distance D, number of turns T, and movement score, and all computational performance metrics defined above. These data are subsequently used for comparative analysis in the Results and Discussion section. The comprehensive compilation of computational performance metrics for both algorithms is presented in Table 2.

Table 2. List of computational performance metrics recorded for the A* and Backtracking algorithms

Metric	Symbol	Computational Meaning	Applicable Algorithm
Expanded states	N_{exp}	Number of times a state is selected for expansion	A*, Backtracking
Neighbor checks	N_{nbr}	Number of attempts to move to adjacent cells	A*, Backtracking
Obstacle / boundary rejections	N_{blk}	Number of times $\text{Safe}(x, y) = 0$	A*, Backtracking
Skipped states	N_{skp}	Skipped due to CLOSED or visited status	A*, Backtracking
First OPEN insertions	N_{open}	Number of states first inserted into OPEN	A* only
Cost relaxations	N_{relax}	Number of times a better path is found	A* only
Total solutions found	N_{sol}	Number of paths from Start to Goal discovered	Backtracking only
Dead ends	N_{dead}	Number of times no further expansion is possible	Backtracking only
Backtracking operations	N_{back}	Number of backward steps after exploration	Backtracking only

2.6. Theoretical Time Complexity of the Investigated Algorithms

To support the interpretation of the experimental results, this study provides a conceptual discussion of the theoretical time complexity of the two investigated algorithms.

The Backtracking algorithm performs exhaustive exploration of all feasible paths from the start state to the goal state. Since the movement model permits at most four possible directions at each state, the branching factor is bounded by $b \leq 4$. Let the search depth be denoted by n, corresponding to the maximum length of a feasible path. The recurrence relation describing the worst-case expansion behavior is defined in (27).

$$T(n) = b \cdot T(n-1), b \leq 4 \tag{27}$$

with the base condition $T(0) = 1$.

Solving the recurrence relation defined in (27) yields the asymptotic time complexity expressed in (28).

$$T(n) = O(b^n) \tag{28}$$

This result indicates exponential growth in computational cost as the depth of the search increases. In grid environments containing multiple branching corridors or alternative routes, the number of feasible paths increases rapidly, which explains the substantial growth in node expansions, dead ends, and backtracking operations observed in the experimental results.

In contrast, the A* algorithm reduces unnecessary exploration by employing an evaluation function and prioritizing expansion of states with the smallest $f(n)$. Let the total number of grid cells be defined as (29).

$$V = N \times N \tag{29}$$

And the approximate number of edges be expressed as (30).

$$E \approx 4V \tag{30}$$

In the present implementation, the selection of the next state from the OPEN set is performed through a full scan over all candidate states. Consequently, each selection step incurs a cost proportional to $O(V)$ in the worst case. Since each state can be inserted into the OPEN set at most once under consistent heuristic behavior, the overall worst-case computational cost is bounded above by (31).

$$O(V^2) \tag{31}$$

Although this bound is higher than the theoretical complexity achievable with priority-queue-based implementations, it remains polynomial and grows significantly slower than the exponential complexity of Backtracking.

This theoretical analysis aligns with the experimental observations, where A* consistently exhibited substantially fewer node expansions and neighbor evaluations compared with Backtracking across all tested maps. The conceptual contrast between exhaustive branching in Backtracking and heuristic-guided focused exploration in A* is illustrated in Fig. 6.

3. Results of Simulation

The simulation tests were conducted on twelve grid-based environments with two map scales: six maps of size 6 × 6 and six maps of size 12 × 12. In all experiments, the start position was fixed at coordinate (0,0), while the goal position was located at the diagonally opposite corner of the grid.

Both A* and Backtracking were implemented under identical simulation conditions, including uniform step cost, full map observability, and four-directional movement constraints.

The maps were intentionally designed as deterministic structural test cases rather than randomly generated environments. This design allows controlled analysis of search behavior under different obstacle arrangements and branching patterns while avoiding the variability introduced by large random datasets. The objective is to examine how heuristic-guided search (A*) and exhaustive search (Backtracking) behave under comparable grid configurations.

For each map, the simulation outputs include the optimal paths produced by both algorithms together with several quantitative metrics. Path-quality metrics include path length, the number of turns, and a movement score representing directional smoothness. In addition, computational-effort metrics were recorded during the search process, including the number of expanded nodes, neighbor checks, blocked or out-of-bound rejections, skipped states due to CLOSED-set membership or previously visited conditions, and algorithm-specific operations such as OPEN insertions, cost relaxations, dead ends, and backtracking operations.

Fig. 7, Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12, Fig. 13, Fig. 14, Fig. 15, Fig. 16, Fig. 17, Fig. 18 illustrate the optimal paths obtained by A* and Backtracking across all twelve grid maps. Fig. 7, Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12 present the results for the six 6 × 6 maps, while Fig. 13, Fig. 14, Fig. 15, Fig. 16, Fig. 17, Fig. 18 show the results for the six 12 × 12 maps. Each figure displays the grid layout together with the computed path, including step indices and movement directions. This visualization enables direct observation of structural routing patterns and turning behavior within each environment.

Particular attention is given to the number of feasible solution paths and the number of backtracking operations observed during exhaustive search. These quantities serve as empirical indicators of the structural complexity of each map. Environments containing multiple feasible shortest paths or numerous dead ends tend to produce substantially higher backtracking counts, reflecting increased branching complexity within the search space.

By examining both path-quality metrics and computational-effort indicators across different map structures and grid sizes, the results provide insight into how heuristic-guided search behaves under increasing structural complexity. At the same time, the analysis highlights the practical scalability limitations associated with exhaustive exploration strategies.

3.1. Qualitative Path Analysis of A* and Backtracking

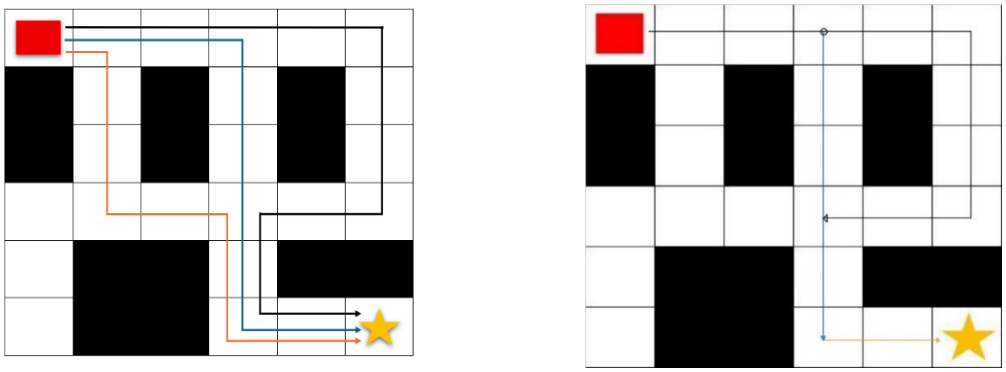
The qualitative path analysis emphasizes path optimality and directional characteristics across twelve grid maps, consisting of six 6 × 6 environments (Maps 1–6) and six 12 × 12 environments (Maps 7–12). In every evaluated case, both A* and Backtracking successfully identified the shortest path based on step count. In the 6 × 6 maps, the optimal path length was consistently 10 steps, whereas in the 12 × 12 maps, it was 22 steps. This confirms that the Manhattan-distance heuristic used in A* preserves optimality with respect to path length under the defined four-directional movement framework.

Although both algorithms consistently achieved optimal path length, differences emerged in certain environments when additional path quality metrics, namely the number of turns and the movement score, were examined. These differences are structurally related to the existence of multiple feasible shortest paths exhibiting distinct turning patterns. The detailed path quality results for the 6 × 6 maps are summarized in Table 3, while the corresponding results for the 12 × 12 maps are presented in Table 4.

In the 6 × 6 environments, Maps 1, 2, 3, and 6 produced identical results for both algorithms across all evaluated metrics, as shown in Table 3. These maps exhibit either a unique shortest path or limited structural branching. However, in Maps 4 and 5, multiple shortest paths are present. In these cases, Backtracking, through exhaustive exploration, identified paths with fewer turns and lower movement scores compared to those selected by A*. This difference reflects the fact that A*

terminates once a shortest path is found according to its evaluation function, without exploring all alternative shortest paths.

Similar structural behavior was observed in the 12 × 12 environments. As indicated in Table 4, in some maps, such as Maps 7, 8, 9, 11, and 12, both algorithms produced identical directional characteristics. In Map 10, which exhibits higher structural branching complexity, Backtracking identified a shortest path with fewer turns than the path selected by A*. This result indicates that when multiple shortest paths exist, exhaustive enumeration allows further refinement according to secondary criteria.



(a) Backtracking for exhaustive exploration with multiple branches

(b) A* for heuristic-guided focused search

Fig. 6. Conceptual illustration of branching explosion in Backtracking compared with heuristic-guided search behavior in A*

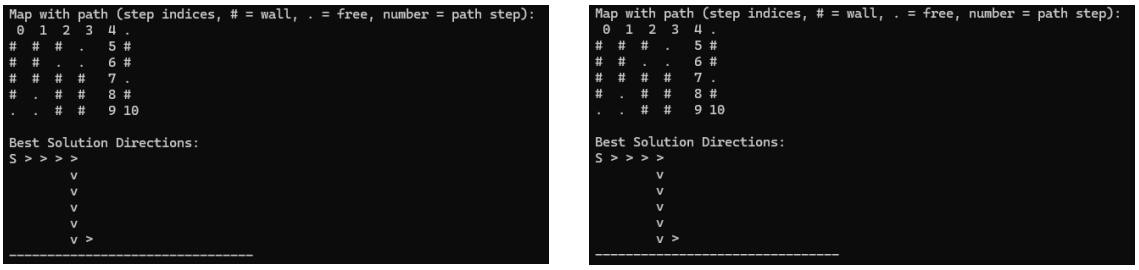


Fig. 7. Best paths obtained by A* and backtracking for map1

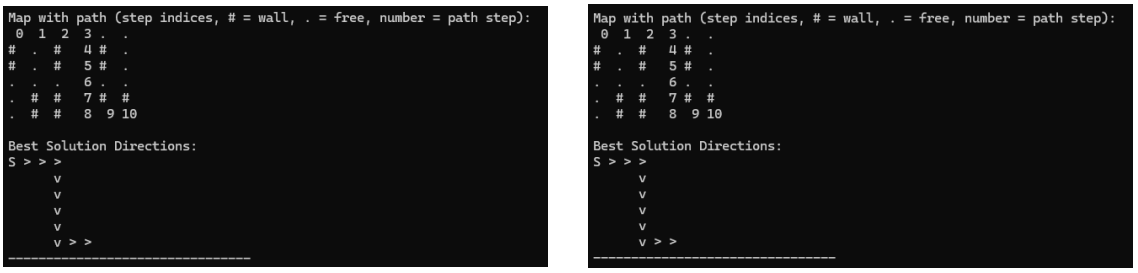


Fig. 8. Best paths obtained by A* and backtracking for map2

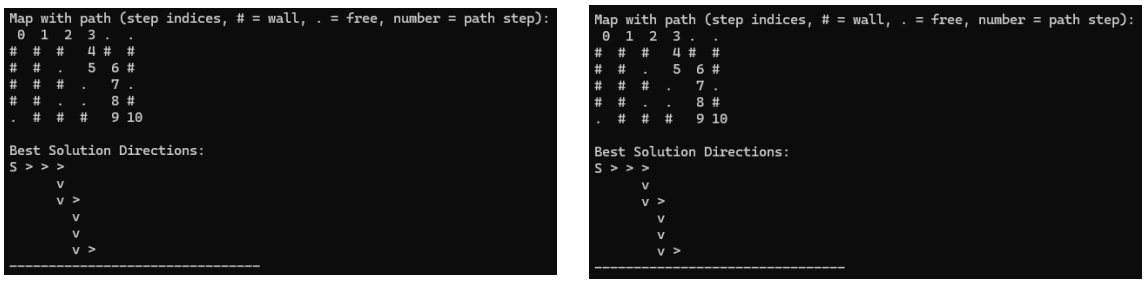


Fig. 9. Best paths obtained by A* and backtracking for map3

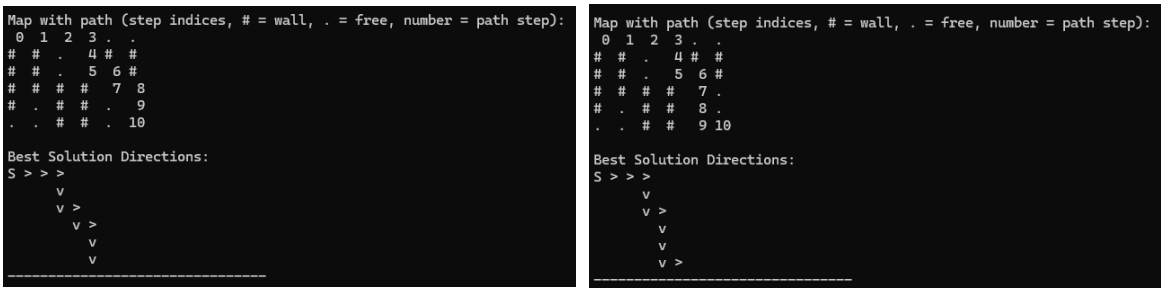


Fig. 10. Best paths obtained by A* and backtracking for map4

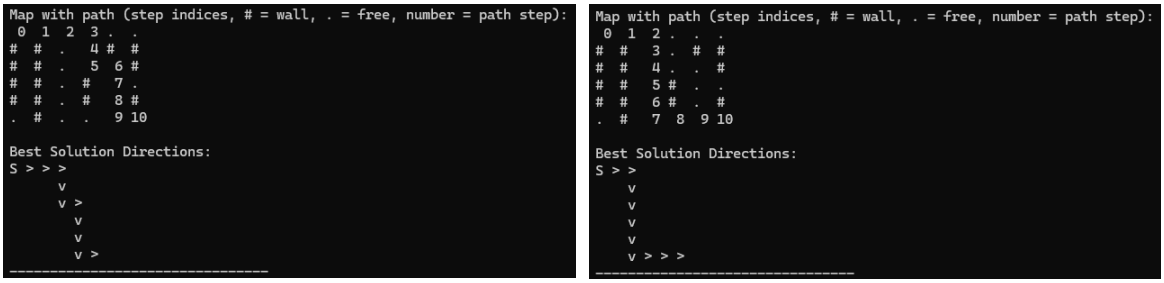


Fig. 11. Best paths obtained by A* and backtracking for map5

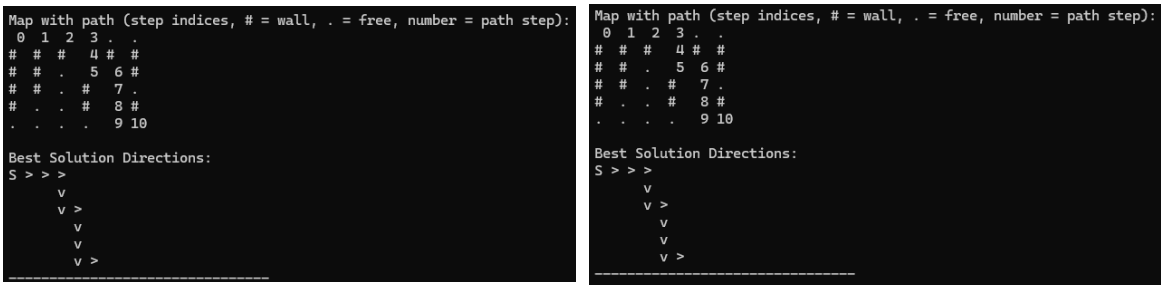


Fig. 12. Best paths obtained by A* and backtracking for map6

Fig. 13. Best paths obtained by A* and backtracking for map7

Importantly, these findings are not intended to establish a competitive comparison. Backtracking serves as a structural reference that reveals the full set of feasible shortest paths, whereas A* demonstrates the ability to obtain high-quality solutions without exhaustive search.

3.2. Analysis of Computational Search Effort and Exploration

The computational search effort of both methods, together with path quality, was evaluated using the quantitative performance measures outlined in the methodology section. The aim of this

analysis is not to competitively rank the methods, but to investigate how computational effort correlates with structural complexity and map size. Backtracking is treated as a comprehensive structural reference, whereas A* is assessed in terms of its computational feasibility under heuristic guidance. The computational effort results for the 6 × 6 maps are summarized in Table 5 and Table 6, while the results for the 12 × 12 maps are presented in Table 7 and Table 8.

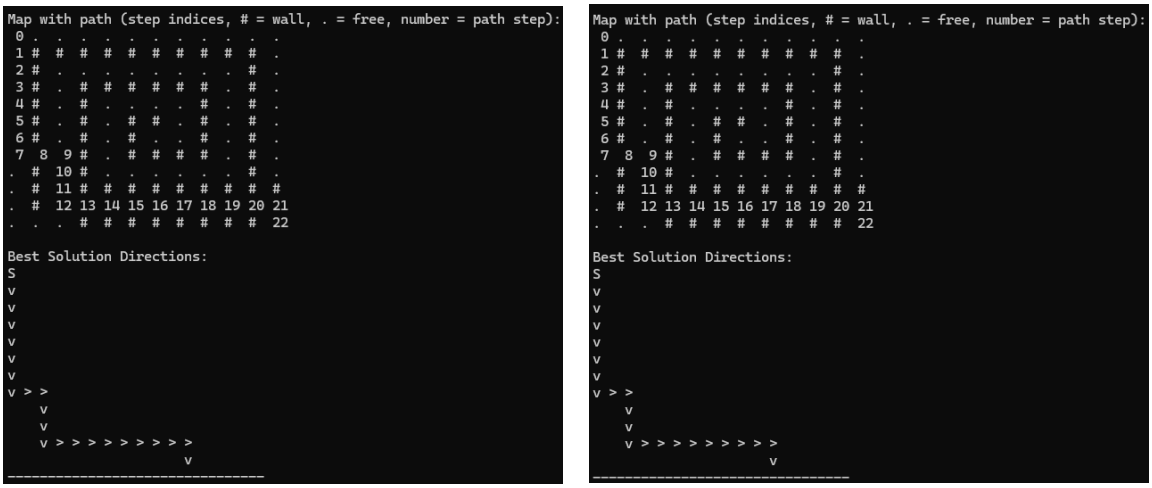


Fig. 14. Best paths obtained by A* and backtracking for map8

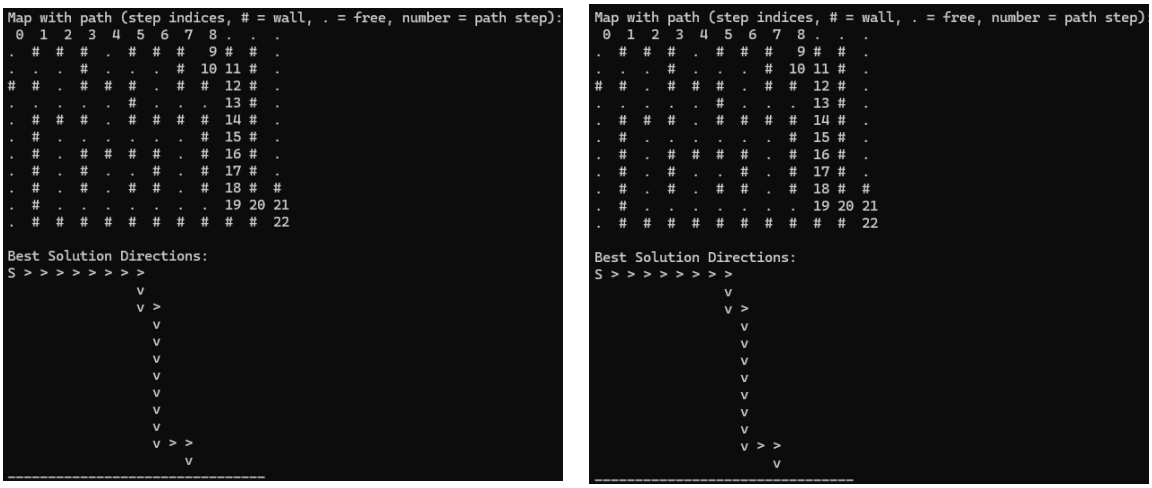


Fig. 15. Best paths obtained by A* and backtracking for map9

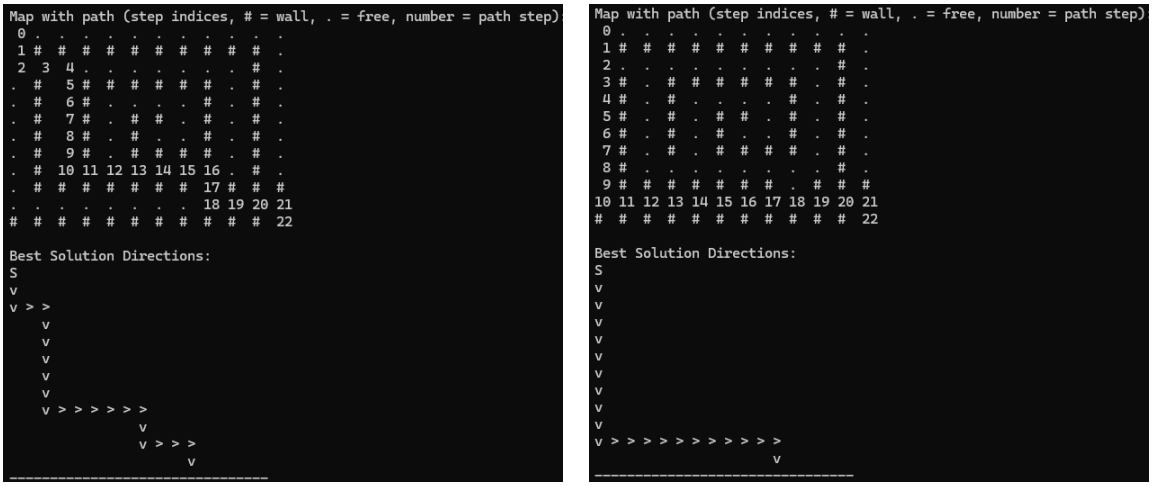


Fig. 16. Best paths obtained by A* and backtracking for map10

Fig. 17. Best paths obtained by A* and backtracking for map11

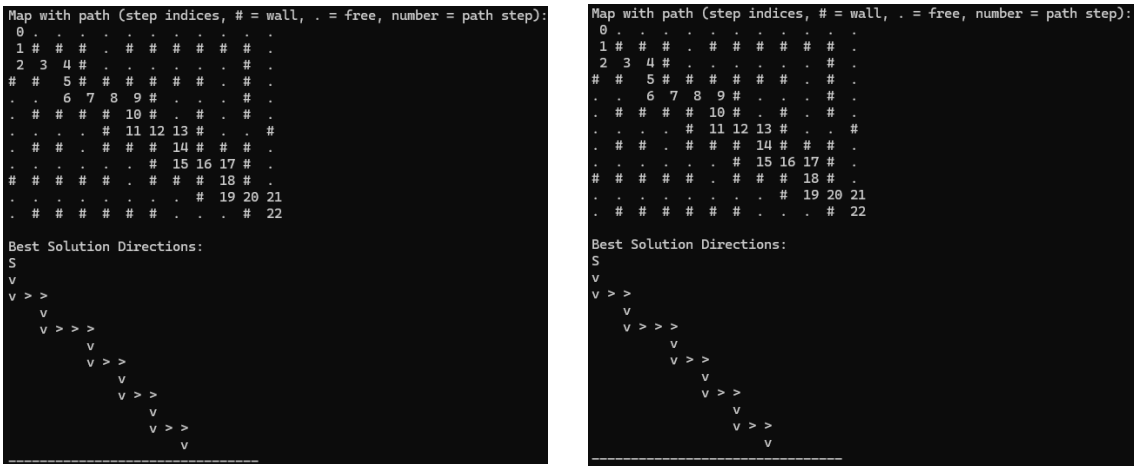


Fig. 18. Best paths obtained by A* and backtracking for map12

Table 3. Path quality results for 6×6 maps (maps 1-6)

Metric / Map	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
A* - Steps	10	10	10	10	10	10
Backtracking - Steps	10	10	10	10	10	10
A* - Turns	2	2	4	5	4	4
Backtracking - Turns	2	2	4	4	2	4
A* - Score	12	12	14	15	14	14
Backtracking - Score	12	12	14	14	12	14

Table 4. Path quality results for 12×12 maps (maps 7-12)

Metric / Map	Map 7	Map 8	Map 9	Map 10	Map 11	Map 12
A* - Steps	22	22	22	22	22	22
Backtracking - Steps	22	22	22	22	22	22
A* - Turns	1	4	5	6	8	10
Backtracking - Turns	1	4	5	2	8	10
A* - Score	23	26	27	28	30	32
Backtracking - Score	23	26	27	24	30	32

In the 6 × 6 settings, A* required a modest and consistent number of expanded nodes, ranging from 14 to 21, and neighbor checks between 52 and 80. These values indicate that the Manhattan

heuristic effectively restricted exploration to a confined region of the grid. Conversely, Backtracking demonstrated significantly higher computational demands even at this reduced scale. The number of expanded nodes ranged from 41 to 115, while neighbor checks reached up to 428 in structurally more complex maps. The number of backtracking operations varied from 40 to 114, reflecting differences in branching structure and the number of feasible solution paths.

The disparity becomes more evident in the 12 × 12 scenarios. For these larger maps, A* expanded between 27 and 70 nodes, depending on structural complexity, with neighbor checks ranging approximately from 104 to 276. Although computational effort increased relative to the 6 × 6 case, the growth remained manageable. In contrast, Backtracking exhibited markedly greater sensitivity to branching complexity. Expanded nodes ranged from 135 to 943, while neighbor checks escalated to 3740 in the most structurally intricate layout. Correspondingly, backtracking operations exceeded 900 in certain cases.

These results suggest that the number of backtracks serves as a practical indicator of structural search difficulty. Maps containing multiple shortest paths or numerous dead ends produced higher backtracking counts, thereby increasing the number of node expansions and neighbor evaluations. As map size and branching complexity increased, exhaustive exploration exhibited a rapid escalation in computational demand.

Importantly, this behavior reflects inherent differences in search strategy rather than competitive performance. Backtracking must explore all feasible paths to guarantee optimality under multi-criteria evaluation, whereas A* terminates once a shortest path is identified through heuristic guidance. Therefore, although Backtracking provides comprehensive structural insight into the search space, its computational cost may become impractical for larger or more complex environments, particularly under limited processing resources.

3.3. Summary and Considerations for Algorithm Choice

A summary of the key findings across all twelve grid maps is presented in Table 9. The results illustrate the fundamental trade-off between computational efficiency and multi-criteria path optimality. The A* algorithm consistently generated optimal shortest-path solutions while requiring substantially lower computational effort. In contrast, the Backtracking approach, although computationally more intensive, systematically identifies the optimal path according to the defined lexicographic multi-criteria evaluation framework.

The analysis highlights the structural distinction between heuristic-guided and exhaustive search strategies in deterministic grid-based environments. The A* algorithm effectively reduces the search space through heuristic guidance, allowing efficient identification of shortest paths without exploring all possible alternatives. Conversely, Backtracking performs exhaustive exploration of the search space, enabling the identification of alternative shortest paths that may exhibit improved directional characteristics, such as fewer turns or lower movement scores.

Table 5. Computational effort metrics of the A* algorithm for 6×6 maps (maps 1-6)

Metric / Map	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Expanded nodes	15	21	16	17	20	14
Neighbor checks	56	80	60	64	76	52
Blocked/out-of-bound rejections	24	36	25	26	33	25
Skipped due to closed state	15	21	16	18	21	12
OPEN insertions	16	22	18	17	20	16
Cost relaxations	15	21	17	16	19	15

The simulation framework employed in this study was designed to isolate algorithmic search behavior under controlled conditions. All experiments were conducted in deterministic two-dimensional grid environments using an offline planning framework with four-directional connectivity, uniform movement cost, and static obstacle configurations. These assumptions enable

consistent comparison between heuristic-guided and exhaustive search strategies while maintaining full reproducibility of the simulation results.

Although the simplified grid environment facilitates controlled structural analysis, the present study does not consider additional complexities commonly encountered in real-world robotic navigation, such as dynamic obstacles, three-dimensional environments, or non-holonomic motion constraints. These aspects represent important directions for future research and may further influence the relative performance and applicability of the investigated algorithms.

Table 6. Computational effort metrics of the backtracking algorithm for 6×6 maps (maps 1-6)

Metric / Map	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Expanded nodes	48	76	41	82	115	53
Neighbor checks	176	292	148	264	428	200
Blocked/out-of-bound rejections	78	137	64	102	186	91
Skipped due to visited state	51	80	44	81	128	57
Number of solution paths	4	3	4	16	8	3
Dead ends	12	11	12	9	20	13
Backtracks	47	75	40	81	114	52

Table 7. Computational effort metrics of the A* algorithm for 12×12 maps (maps 7-12)

Metric / Map	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Expanded nodes	27	48	60	70	59	52
Neighbor checks	104	188	236	276	232	204
Blocked/out-of-bound rejections	48	92	112	134	111	98
Skipped due to closed state	25	47	60	70	58	50
OPEN insertions	32	49	63	71	63	57
Cost relaxations	31	48	62	70	62	56

Table 8. Computational effort metrics of the backtracking algorithm for 12×12 maps (maps 7-12)

Metric / Map	Map 1	Map 2	Map 3	Map 4	Map 5	Map 6
Expanded nodes	943	135	440	230	233	445
Neighbor checks	3740	532	1744	908	916	1756
Blocked/out-of-bound rejections	1778	264	850	445	446	852
Skipped due to visited state	1020	134	455	234	238	460
Number of solution paths	8	2	4	3	4	6
Dead ends	94	5	37	13	19	42
Backtracks	942	134	439	229	232	444

Table 9. Summary of key findings across all twelve maps

Summary Aspect	Observations from the simulation
Path length optimality	Both A* and Backtracking produced identical step counts in all tested maps
Path quality in terms of turns and score	A* matched Backtracking in most maps; differences appeared in Maps 4, 5, and 10, where Backtracking identified paths with fewer turns and lower movement scores
Computational search effort	A* consistently required substantially fewer node expansions and neighbor checks than Backtracking
Influence of map structure	Maps with multiple shortest paths and complex branching patterns amplified qualitative and computational differences
Algorithm selection guideline	A* is suitable for computational efficiency and reliable shortest paths, while Backtracking is suitable for strict multi-criteria path optimality when computational cost is acceptable

4. Conclusion

This paper conducted a simulation-based structural analysis of the A* algorithm and the Backtracking method for offline robot path planning in two-dimensional grid environments under uniform evaluation conditions. The main aim was not to determine competitive superiority between the two approaches, but to analyze the behavior of different search paradigms as structural

complexity increases. The experimental results indicate that both algorithms consistently identify the shortest paths with identical step counts in all investigated scenarios, thereby validating the correctness of both methods with respect to minimum-distance navigation in grid-based environments. Discrepancies emerge only when additional path-quality criteria, such as turning behavior and movement smoothness, are evaluated in situations where multiple feasible shortest paths exist. In this study, the Backtracking method functions as a structural reference that reveals the full combinatorial characteristics of each map. The number of solution paths, dead ends, and backtracking operations provides empirical insight into branching diversity and search-space structure. In maps with intricate branching structures or numerous shortest-path alternatives, the number of backtracks increases substantially, reflecting the rapid growth of exhaustive search effort. In contrast, the A* algorithm demonstrates the ability to derive shortest-path solutions with significantly reduced computational cost through heuristic guidance. While A* does not enumerate all feasible shortest paths, the findings indicate that it consistently achieves optimal distance performance and, in most cases, maintains satisfactory directional quality while requiring considerably lower computational effort. The findings emphasize a structural distinction rather than a competitive trade-off between exhaustive completeness and heuristic-guided computational scalability. Backtracking guarantees full multi-criteria optimality through complete exploration; however, its computational cost increases markedly with branching complexity, which may limit its applicability in environments with constrained processing resources. A* provides a computationally efficient mechanism for obtaining reliable shortest-path solutions under practical resource constraints. Future research will extend this structural evaluation framework to larger and more complex environments and explore improved heuristic formulations that incorporate motion-quality considerations directly into the evaluation function. Such developments may enable heuristic-based search to achieve improved directional refinement while preserving computational scalability, thereby strengthening its applicability in practical robotic systems.

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

Funding: Research Institute, Academic Services Center, and College of Biomedical Engineering, Rangsit University.

Acknowledgment: The researcher acknowledges the financial assistance the research team received from the research institute, academic services center, and college of biomedical engineering at Rangsit University. The Ethics Review Board of Rangsit University has evaluated the study, reference number RSUERB2025-006, and confirmed that the research does not involve human subjects.

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

References

[1] L. Liu, X. Wang, X. Yang, H. Liu, J. Li, and P. Wang, "Path planning techniques for mobile robots: Review and prospect," *Expert Systems with Applications*, vol. 227, p. 120254, 2023, https://doi.org/10.1016/j.eswa.2023.120254.

[2] H. Qin, S. Shao, T. Wang, X. Yu, Y. Jiang, and Z. Cao, "Review of autonomous path planning algorithms for mobile robots," *Drones*, vol. 7, no. 3, p. 211, 2023, https://doi.org/10.3390/drones7030211.

[3] K. Katona, H. A. Neamah, and P. Korondi, "Obstacle avoidance and path planning methods for autonomous navigation of mobile robot," *Sensors*, vol. 24, no. 11, p. 3573, 2024, https://doi.org/10.3390/s24113573.

[4] N. Wang, X. Li, K. Zhang, J. Wang, and D. Xie, "A survey on path planning for autonomous ground vehicles in unstructured environments," *Machines*, vol. 12, no. 1, p. 31, 2024, https://doi.org/10.3390/machines12010031.

[5] R. Alqobali, M. Alshmrani, R. Alnasser, A. Rashidi, T. Alhmiedat, and O. M. D. Alia, "A survey on robot semantic navigation systems for indoor environments," *Applied Sciences*, vol. 14, no. 1, p. 89, 2024, https://doi.org/10.3390/app14010089.

[6] K. Karur, N. Sharma, C. Dharmatti, and J. E. Siegel, "A survey of path planning algorithms for mobile robots," *Vehicles*, vol. 3, no. 3, pp. 448–468, 2021, https://doi.org/10.3390/vehicles3030027.

[7] K. C. Ugwoke, N. A. Nnanna, and S. E. Y. Abdullahi, "Simulation-based review of classical, heuristic, and metaheuristic path planning algorithms," *Scientific Reports*, vol. 15, no. 1, p. 12643, 2025, https://doi.org/10.1038/s41598-025-96614-2.

[8] K. Zhu and T. Zhang, "Deep reinforcement learning based mobile robot navigation: A review," *Tsinghua Science and Technology*, vol. 26, no. 5, pp. 674-691, 2021, https://doi.org/10.26599/TST.2021.9010012.

[9] W. Chen *et al*., "A survey of autonomous robots and multi-robot navigation: Perception, planning and collaboration," *Biomimetic Intelligence and Robotics*, vol. 5, no. 2, p. 100203, 2025, https://doi.org/10.1016/j.birob.2024.100203.

[10] H. Guo, F. Wu, Y. Qin, R. Li, K. Li, and K. Li, "Recent trends in task and motion planning for robotics: A survey," *ACM Computing Surveys*, vol. 55, no. 13s, pp. 1–36, 2023, https://doi.org/10.1145/3583136.

[11] M. Wu, C. F. Yeong, E. L. M. Su, W. Holderbaum, and C. Yang, "A review on energy efficiency in autonomous mobile robots," *Robotic Intelligence and Automation*, vol. 43, no. 6, pp. 648–668, 2023, https://doi.org/10.1108/RIA-05-2023-0060.

[12] I. Lee, "Service robots: A systematic literature review," *Electronics*, vol. 10, no. 21, p. 2658, 2021, https://doi.org/10.3390/electronics10212658.

[13] J. P. Bailey, A. Nash, C. A. Tovey, and S. Koenig, "Path-length analysis for grid-based path planning," *Artificial Intelligence*, vol. 301, p. 103560, 2021, https://doi.org/10.1016/j.artint.2021.103560.

[14] R. Assari and Q. P. Gu, "Average sensitivity of breadth-first search algorithms on grids," *Combinatorial Algorithms*, pp. 173-187, 2025, https://doi.org/10.1007/978-3-031-98740-3_13.

[15] H. Y. Lin, K. L. Chang, and H. Y. Huang, "Development of unmanned aerial vehicle navigation and warehouse inventory system based on reinforcement learning," *Drones*, vol. 8, no. 6, p. 220, 2024, https://doi.org/10.3390/drones8060220.

[16] S. Koenig, M. Likhachev, and D. Furcy, "Lifelong planning A*," *Artificial Intelligence*, vol. 155, nos. 1-2, pp. 93-146, 2004, https://doi.org/10.1016/j.artint.2003.12.001.

[17] D. Foead, A. Ghifari, M. B. Kusuma, N. Hanafiah, and E. Gunawan, "A systematic literature review of A* pathfinding," *Procedia Computer Science*, vol. 179, pp. 507-514, 2021, https://doi.org/10.1016/j.procs.2021.01.034.

[18] J. Zhang, X. Wang, L. Xu, and X. Zhang, "An occupancy information grid model for path planning of intelligent robots," *ISPRS International Journal of Geo-Information*, vol. 11, no. 4, p. 231, 2022, https://doi.org/10.3390/ijgi11040231.

[19] W. Lee, G. H. Choi, and T. W. Kim, "Visibility graph-based path-planning algorithm with quadtree representation," *Applied Ocean Research*, vol. 117, p. 102887, 2021, https://doi.org/10.1016/j.apor.2021.102887.

[20] F. Wang, W. Sun, P. Yan, H. Wei, and H. Lu, "Research on path planning for robots with improved A* algorithm under bidirectional JPS strategy," *Applied Sciences*, vol. 14, no. 13, p. 5622, 2024, https://doi.org/10.3390/app14135622.

[21] H. Wang, X. Qi, S. Lou, J. Jing, H. He, and W. Liu, "An efficient and robust improved A* algorithm for path planning," *Symmetry*, vol. 13, no. 11, p. 2213, 2021, https://doi.org/10.3390/sym13112213.

[22] O. Amine and M. Mohammed, "Generating A-star algorithm admissible heuristics using a dynamic Dataloader on neural networks, enhanced with genetic algorithms, on a distributed architecture," *IEEE Access*, vol. 11, pp. 18356-18373, 2023, https://doi.org/10.1109/ACCESS.2023.3247773.

[23] Y. Chen, J. Pang, Z. Huang, Y. Gou, Z. Jiang, and D. Chen, "An improved A* algorithm based on simulated annealing and multidistance heuristic function," *International Journal of Intelligent Systems*, vol. 2025, no. 1, p. 5979509, 2025, https://doi.org/10.1155/int/5979509.

[24] J. Zhang, J. Wu, X. Shen, and Y. Li, "Autonomous land vehicle path planning algorithm based on improved heuristic function of A-Star," *International Journal of Advanced Robotic Systems*, vol. 18, no. 5, 2021, https://doi.org/10.1177/17298814211042730.

[25] Z. An, X. Rui, and C. Gao, "Improved A* navigation path-planning algorithm based on hexagonal grid," *ISPRS International Journal of Geo-Information*, vol. 13, no. 5, p. 166, 2024, https://doi.org/10.3390/ijgi13050166.

[26] Z. Zhang, J. Jiang, J. Wu, and X. Zhu, "Efficient and optimal penetration path planning for stealth unmanned aerial vehicle using minimal radar cross-section tactics and modified A-Star algorithm," *ISA Transactions*, vol. 134, pp. 42–57, 2023, https://doi.org/10.1016/j.isatra.2022.07.032.

[27] Y. Ou, Y. Fan, X. Zhang, Y. Lin, and W. Yang, "Improved A* path planning method based on the grid map," *Sensors*, vol. 22, no. 16, p. 6198, 2022, https://doi.org/10.3390/s22166198.

[28] C. Li, X. Huang, J. Ding, K. Song, and S. Lu, "Global path planning based on a bidirectional alternating search A* algorithm for mobile robots," *Computers and Industrial Engineering*, vol. 168, p. 108123, 2022, https://doi.org/10.1016/j.cie.2022.108123.

[29] Y. Zhou, Y. Lu, and L. Lv, "Grid-based non-uniform probabilistic roadmap-based AGV path planning in narrow passages and complex environments," *Electronics*, vol. 13, no. 1, p. 225, 2024, https://doi.org/10.3390/electronics13010225.

[30] J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, and L. Wang, "An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling," *Journal of Computational Science*, vol. 67, p. 101937, 2023, https://doi.org/10.1016/j.jocs.2022.101937.

[31] O. A. A. Salama, M. E. H. Eltaib, H. A. Mohamed and O. Salah, "RCD: Radial Cell Decomposition Algorithm for Mobile Robot Path Planning," *IEEE Access*, vol. 9, pp. 149982-149992, 2021, https://doi.org/10.1109/ACCESS.2021.3125105.

[32] P. Chintam, T. Lei, B. Osmanoglu, Y. Wang, and C. Luo, "Informed sampling space driven robot informative path planning," *Robotics and Autonomous Systems*, vol. 175, p. 104656, 2024, https://doi.org/10.1016/j.robot.2024.104656.

[33] Z. Zhou, X. Feng, S. Di and X. Zhou, "A LiDAR Mapping System for Robot Navigation in Dynamic Environments," *IEEE Transactions on Intelligent Vehicles*, pp. 1–20, 2023, https://doi.org/10.1109/TIV.2023.3328013.

[34] B. Fang, G. Mei, X. Yuan, L. Wang, Z. Wang, and J. Wang, "Visual SLAM for robot navigation in healthcare facility," *Pattern Recognition*, vol. 113, p. 107822, 2021, https://doi.org/10.1016/j.patcog.2021.107822.

[35] M. Adamkiewicz *et al*., "Vision-Only Robot Navigation in a Neural Radiance World," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4606-4613, 2022, https://doi.org/10.1109/LRA.2022.3150497.

[36] A. Pathak, S. Agrawal and R. Tripathi, "Autonomous Fleet Management System for Route Optimization Using Reinforcement Learning with Warehouse Cameras," *2025 IEEE Madhya Pradesh Section Conference (MPCON)*, pp. 897-902, 2025, https://doi.org/10.1109/MPCON66082.2025.11256627.

[37] J. Zhao, Y. Wang, Z. Cai, N. Liu, K. Wu and Y. Wang, "Learning Visual Representation for Autonomous Drone Navigation via a Contrastive World Model," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 3, pp. 1263-1276, 2024, https://doi.org/10.1109/TAI.2023.3283488.

[38] M. Luong and C. Pham, "Incremental learning for autonomous navigation of mobile robots based on deep reinforcement learning," *Journal of Intelligent & Robotic Systems*, vol. 101, no. 1, p. 1, 2021, https://doi.org/10.1007/s10846-020-01262-5.

[39] D. Lee, I. M. A. Nahrendra, M. Oh, B. Yu and H. Myung, "TRG-Planner: Traversal Risk Graph-Based Path Planning in Unstructured Environments for Safe and Efficient Navigation," *IEEE Robotics and Automation Letters*, vol. 10, no. 2, pp. 1736-1743, 2025, https://doi.org/10.1109/LRA.2024.3524912.

[40] P. Chotikunnan *et al*., "Application of backtracking algorithm for offline robot transport," *International Review of Automatic Control*, vol. 18, no. 1, pp. 1–11, 2025, https://doi.org/10.15866/ireaco.v18i1.25815.

[41] Y. Han, M. Shao, Y. Wu, and X. Zhang, "An improved complete coverage path planning method for intelligent agricultural machinery based on backtracking method," *Information*, vol. 13, no. 7, p. 313, 2022, https://doi.org/10.3390/info13070313.

[42] M. Govindaraju, D. Fontanelli, S. S. Kumar and A. S. Pillai, "Optimized Offline-Coverage Path Planning Algorithm for Multi-Robot for Weeding in Paddy Fields," *IEEE Access*, vol. 11, pp. 109868-109884, 2023, https://doi.org/10.1109/ACCESS.2023.3322230.

[43] B. Xiao, Z. Zhang, Q. Wang, B. Zhang, and S. Zheng, "TSP-based depth-first search algorithms for enhanced path planning in laser-based directed energy," *Precision Engineering*, vol. 93, pp. 224–236, 2025, https://doi.org/10.1016/j.precisioneng.2025.01.009.

[44] Z. Wang, W. Hu, G. Chen, C. Yuan, R. Gu and Y. Huang, "Towards Efficient Distributed Subgraph Enumeration Via Backtracking-Based Framework," *IEEE Transactions on Parallel and Distributed Systems*, vol. 32, no. 12, pp. 2953-2969, 2021, https://doi.org/10.1109/TPDS.2021.3076246.

[45] S. Sundarraj, R. V. K. Reddy, M. B. Basam, G. H. Lokesh, F. Flammini and R. Natarajan, "Route Planning for an Autonomous Robotic Vehicle Employing a Weight-Controlled Particle Swarm-Optimized Dijkstra Algorithm," *IEEE Access*, vol. 11, pp. 92433-92442, 2023, https://doi.org/10.1109/ACCESS.2023.3302698.

[46] M. Tanha, M. Hosseini Shirvani, and A. M. Rahmani, "A hybrid meta-heuristic task scheduling algorithm based on genetic and thermodynamic simulated annealing algorithms in cloud computing environments," *Neural Computing and Applications*, vol. 33, no. 24, pp. 16951–16984, 2021, https://doi.org/10.1007/s00521-021-06289-9.

[47] F. S. Gharehchopogh, M. Namazi, L. Ebrahimi, and B. Abdollahzadeh, "Advances in sparrow search algorithm: A comprehensive survey," *Archives of Computational Methods in Engineering*, vol. 30, no. 1, pp. 427–455, 2023, https://doi.org/10.1007/s11831-022-09804-w.

[48] Z. Wang, M. Zhang, S. Liang, S. Yu, C. Zhang, and S. Du, "A multi-objective path-planning approach for multi-scenario urban mobility needs," *Algorithms*, vol. 18, no. 1, p. 41, 2025, https://doi.org/10.3390/a18010041.

[49] K. S. Suresh, R. Venkatesan, and S. Venugopal, "Mobile robot path planning using multi-objective genetic algorithm in industrial automation," *Soft Computing*, vol. 26, no. 15, pp. 7387–7400, 2022, https://doi.org/10.1007/s00500-022-07300-8.

[50] J. Weise and S. Mostaghim, "A comparison of distance metrics for the multi-objective pathfinding problem," *Natural Computing*, vol. 22, no. 2, pp. 315-328, 2023, https://doi.org/10.1007/s11047-022-09908-z.

[51] D. Xiang, H. Lin, J. Ouyang, and D. Huang, "Combined improved A* and greedy algorithm for path planning of multi-objective mobile robot," *Scientific Reports*, vol. 12, no. 1, p. 13273, 2022, https://doi.org/10.1038/s41598-022-17684-0.

[52] J. Huang, Z. He, Y. Arakawa and B. Dawton, "Trajectory Planning in Frenet Frame via Multi-Objective Optimization," *IEEE Access*, vol. 11, pp. 70764-70777, 2023, https://doi.org/10.1109/ACCESS.2023.3294713.

[53] L. Zhu, K. Huang, Y. Wang and Y. Hu, "A Multi-Objective Path Planning Framework in Off-Road Environments Based on Deep Reinforcement Learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 26, no. 11, pp. 19163-19181, 2025, https://doi.org/10.1109/TITS.2025.3589586.

[54] Z. Ren, S. Rathinam, M. Likhachev, and H. Choset, "Multi-objective safe-interval path planning with dynamic obstacles," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 8154–8161, 2022, https://doi.org/10.1109/LRA.2022.3187270.

[55] J. Schoenfelder, S. Kohl, M. Glaser, S. McRae, J. O. Brunner, and T. Koperna, "Simulation-based evaluation of operating room management policies," *BMC Health Services Research*, vol. 21, no. 1, p. 271, 2021, https://doi.org/10.1186/s12913-021-06234-5.

[56] O. A. Nnene, J. W. Joubert, and M. H. Zuidgeest, "A simulation-based optimization approach for designing transit networks," *Public Transport*, vol. 15, no. 2, pp. 377–409, 2023, https://doi.org/10.1007/s12469-022-00312-5.

[57] S. Pawel, L. Kook, and K. Reeve, "Pitfalls and potentials in simulation studies: Questionable research practices in comparative simulation studies allow for spurious claims of superiority of any method," *Biometrical Journal*, vol. 66, no. 1, p. 2200091, 2024, https://doi.org/10.1002/bimj.202200091.

[58] C. Camargo, J. Gonçalves, M. Á. Conde, F. J. Rodríguez-Sedano, P. Costa, and F. J. García-Peñalvo, "Systematic literature review of realistic simulators applied in educational robotics context," *Sensors*, vol. 21, no. 12, p. 4031, 2021, https://doi.org/10.3390/s21124031.

[59] A. Meier, S. Carroccio, R. Dornberger, and T. Hanne, "Discussing the reality gap by comparing physics engines in kilobot simulations," *Journal of Robotics and Control*, vol. 2, no. 5, pp. 441–447, 2021, https://doi.org/10.18196/jrc.25120.

[60] I. R. Fahmi and D. J. Suroso, "A simulation-based study of maze-solving-robot navigation for educational purposes," *Journal of Robotics and Control*, vol. 3, no. 1, pp. 48–54, 2022, https://doi.org/10.18196/jrc.v3i1.12241.

[61] A. K. Vo, T. L. Mai, and H. K. Yoon, "Path planning for automatic berthing using ship-maneuvering simulation-based deep reinforcement learning," *Applied Sciences*, vol. 13, no. 23, p. 12731, 2023, https://doi.org/10.3390/app132312731.

[62] H. Maghfiroh and H. Probo Santoso, "Self-balancing robot navigation," *Journal of Robotics and Control*, vol. 2, no. 5, pp. 408–412, 2021, https://doi.org/10.18196/jrc.25115.

[63] I. A. Hassan, I. A. Abed, and W. A. Al-Hussaibi, "Path planning and trajectory tracking control for two-wheel mobile robot," *Journal of Robotics and Control*, vol. 5, no. 1, pp. 1–15, 2023, https://doi.org/10.18196/jrc.v5i1.20489.

[64] S. Hachani and E. Nechadi, "Type-2 fuzzy logic-based robot navigation in uncertain environments: Simulation and real-world implementation," *Journal of Robotics and Control*, vol. 6, no. 1, pp. 437–445, 2025, https://doi.org/10.18196/jrc.v6i1.25553.

[65] A.-N. Sharkawy, "Task location to improve human–robot cooperation: A condition number-based approach," *Automation*, vol. 4, no. 3, pp. 263–290, 2023, https://doi.org/10.3390/automation4030016.

[66] X. Bai, A. Fielbaum, M. Kronmüller, L. Knoedler and J. Alonso-Mora, "Group-Based Distributed Auction Algorithms for Multi-Robot Task Assignment," *IEEE Transactions on Automation Science and Engineering*, vol. 20, no. 2, pp. 1292-1303, 2023, https://doi.org/10.1109/TASE.2022.3175040.