Tao *et al.*
Chinese Journal of Mechanical Engineering (2024) 37:32
https://doi.org/10.1186/s10033-024-01014-8

ORIGINAL ARTICLE

Open Access

Heuristic Expanding Disconnected Graph: A Rapid Path Planning Method for Mobile Robots

Yong Tao[1,2*], Lian Duan[3], He Gao[2], Yufan Zhang[3], Yian Song[1] and Tianmiao Wang[1]

Abstract
Existing mobile robots mostly use graph search algorithms for path planning, which suffer from relatively low planning efficiency owing to high redundancy and large computational complexity. Due to the limitations of the neighborhood search strategy, the robots could hardly obtain the most optimal global path. A global path planning algorithm, denoted as EDG*, is proposed by expanding nodes using a well-designed expanding disconnected graph operator (EDG) in this paper. Firstly, all obstacles are marked and their corners are located through the map pre-processing. Then, the EDG operator is designed to find points in non-obstruction areas to complete the rapid expansion of disconnected nodes. Finally, the EDG* heuristic iterative algorithm is proposed. It selects the candidate node through a specific valuation function and realizes the node expansion while avoiding collision with a minimum offset. Path planning experiments were conducted in a typical indoor environment and on the public dataset CSM. The result shows that the proposed EDG* reduced the planning time by more than 90% and total length of paths reduced by more than 4.6%. Compared to A*, Dijkstra and JPS, EDG* does not show an exponential explosion effect in map size. The EDG* showed better performance in terms of path smoothness, and collision avoidance. This shows that the EDG* algorithm proposed in this paper can improve the efficiency of path planning and enhance path quality.

Keywords Global path planning, Mobile robot, Expanding disconnected graph, Edge node, Offset

1 Introduction

Autonomous navigation of mobile robots is widely used in the fields of entertainment, medicine, rescue, education, and agriculture [1]. As an important part of navigation technology, path planning involves the creation of a collision-free path from the initial position to the target position and can be divided into two types: local and global [2]. This study aims to improve global path planning approaches, which mainly include spatial sampling, swarm intelligence, and graph planning [3–5]. Spatial sampling solutions mainly comprise the probabilistic roadmap method (PRM) [6, 7] and rapidly-exploring random trees (RRT) [8, 9]. They can effectively solve planning problems in high-dimensional search spaces and complex constraints [2]. However, the solution obtained by PRM or RRT is not globally optimal and has some randomness [6, 9].

Genetic algorithms (GA) [10, 11] have a clear principle and inherent parallel nature but have disadvantages such as low searching efficiency and the tendency to fall into a local optimum [10]. The particle swarm optimization (PSO) algorithm [12, 13] has the advantages of a few adjustable parameters and simple algorithm

*Correspondence:
Yong Tao
taoy@buaa.edu.cn
[1] School of Mechanical Engineering & Automation, Beihang University, Beijing 100191, China
[2] Aero-Engine Research Institute, Beihang University, Beijing 102206, China
[3] School of Large Aircraft Engineering, Beihang University, Beijing 100191, China

implementation but has the shortcomings of a slow converging rate [13]. The ant colony optimization (ACO) algorithm [14, 15] has the advantages of positive feedback and strong heuristics. It also has the disadvantages of a large computation amount, a slow converging rate, and the tendency to fall into a local optimum [15].

Most graph planning algorithms discretize environment models and describe the environment by graphs [16]. The vertices of a graph generally represent the positions that a robot can reach. The edges represent the local paths that the robot can choose. The main algorithms include graph search, visibility graphs [17, 18], and Voronoi [19–21].

Graph search algorithms rasterize a map and then perform path searching to find the shortest path and achieve optimal efficiency. Dijkstra (DA) is the most classical graph search algorithm for finding the shortest path in weighted graphs [4]. DA is simple to implement and can obtain the optimal solution. Due to its high time complexity, its running time will increase significantly with increasing map size. Based on DA, A* (or A-Star) innovatively designs a heuristic function by using a priori information of the starting and target points. It reduces the number of search nodes and improves the efficiency of path search [4]. Restricted by the neighborhood searching strategy, A* has a certain probability of not obtaining the optimal solution that any-angle or angle-restricted search algorithms can obtain. As the map size increase, A* shows an exponential increase in computational complexity [22].

Weighted-A* [23, 24] optimizes the weight of the heuristic function and reduces the number of inflection points but has increased the path length and planning time. Smooth-A * [25] improves path smoothness by adding a path smoothing module based on A*. Its planning time increases by a factor of 4 to 7 compared to A*. Singh et al. [26, 27] introduced a safety distance constraint for the optimal waypoint in the heuristic function. They made the planned path for a robot far away from obstacles and increased the operational safety of the robot. To plan a collision-free and smooth path with minimum cost, Tang et al. [28] proposed Geometric-A* by combining the advantages of A* and interpolation algorithms and optimizing them by geometric rules. Multi-Heuristic-A* [29, 30] uses multiple heuristic functions to avoid becoming trapped in a local optimum. These improvements to the heuristic function optimize A* in some aspects but reduce its planning efficiency or path quality.

In conventional graph search algorithms, nodes are expanded within the eight neighborhoods of the current node. Zhang et al. [31] proposed 24-neighborhood-A*, which improved path smoothness to some extent but reduced path planning efficiency due to increased computation amount. Islam et al. [32, 33] proposed A*-Connect, which increased the path planning efficiency but could not achieve a global optimum. Harabor et al. [34, 35] proposed the jump point search (JPS) method, which greatly improved the solution-solving speed. Li et al. [36] proposed the A* method based on region search. It significantly reduced the search space but did not provide a generalized region partitioning method and had high restrictions for the map. Li et al. [37] further optimized the region-based searching strategy in 2021 and proposed sparse A*, which could effectively reduce the search space. Gong [38] introduced the concepts of convex corner points and neighbor relations and proposed a successor node expansion strategy based on neighbor relations. It showed an exponential increase in the number of convex corner points as the map size increased and had lower planning efficiency than A*. Yonetani et al. [39] proposed Neural A*, a novel data-driven search method for path planning problems. The canonical A* search algorithm is reformulated as differentiable and combined with a convolutional encoder to form an end-to-end trainable neural network planner. Neural A* solves path planning problems by encoding problem instances into a bootstrap graph and then using the bootstrap graph to perform a differentiable A* search. By learning to match the search results with the real paths provided by the expert, Neural A* can accurately and efficiently generate paths that are consistent with the real situation. Marcucci et al. [40] focus on collision-free motion planning, which is cost and trajectory constrained in terms of shape, duration, and speed, and propose a path planning framework that enables convex optimization to efficiently and reliably plan paths around trajectories around obstacles. A practical convex relaxation of the planning problem is devised. We show that this relaxation is typically so tight that inexpensive post-processing of its solution is almost always sufficient to identify globally optimal collision-free trajectories.

The above improvements to path planning algorithms only focused on the local optimum instead of the global optimum. In large grid maps, the high redundancy and large computational complexity of conventional graph search algorithms will lead to high time consumption and reduce planning efficiency. Meanwhile, limited by the neighborhood searching strategy, these algorithms failed to obtain a global optimum path. This paper attempts to mitigate this deficiency by introducing an expanding disconnected graph (EDG) and respective heuristic iterative algorithm (EDG*) to solve the global path planning problems of mobile robots.

The main contributions of this paper are as follows:

(1) The "Obstacle partitioning" and "corner points searching" are proposed for grid maps. The adjacent occupancy grids are marked as the same obstacle. Stronger obstacle semantic information that helps global and local path planning is provided by searching all obstacle corner points.
(2) The expanding disconnected graph (EDG) operator is proposed. It takes line segments as nodes and randomly selects points outside the collision polygon as new vertices of the graph to achieve expansion of disconnected edge nodes.
(3) The EDG heuristic iterative algorithm (EDG*) is proposed. It selects the next edge node to be expanded based on a well-designed evaluation function. It avoids obstacles and expands the current edge node to generate the new edge nodes utilizing the minimum offset, thus rapid iteration completing optimal path planning tasks.

The rest of this paper is structured as follows: Section 2 defines some necessary basic concepts, mainly including graphs, paths, offset points, and collision polygons. Section 3 introduces EDG and constructs the EDG* heuristic iterative algorithm, mainly including heuristic information, and map pre-processing. Section 4 builds the path planning experimental environment and presents the experimental results analysis. Finally, Section 5 concludes the paper.

2 Basic Definitions

The generalized graph $G = (V, E)$ consists of a set of vertices $V = \{v_i\}$ and a binary set of edges $E = \{e_{ij}\}$ defined on V. In many studies [16], the vertices and edges of a graph are not represented explicitly but are specified implicitly by the staring vertex v_s and Γ defined in V. By applying Γ to v_{i-1}, v_i and e_{ij} are obtained. Γ is applied from v_s to the target vertex v_t in order. This paper explicitly provides all vertices and edges of the current state graph G.

In the generalized graph, any two vertices can be connected to form an edge, regardless of whether they are connected or not. Assume e_{ij} is an element in E; then, e_{ij} is the edge node connecting v_i and v_j. If v_i can reach v_j by passing through e_{ij}, e_{ij} is called a connected edge node. Otherwise, it is a disconnected edge node. The object that causes the disconnection of e_{ij} is called the obstacle. As shown in Figure 1, e_{s1} and e_{st} are disconnected edge nodes. e_{1t}, e_{s4}, and e_{4t} are connected edge nodes. The occupancy grid of Obstacle 1 is the obstacle.

This paper focuses on the cost graph of edge nodes. c_{ij} represents the optimal estimation cost of e_{ij}. If e_{ij} is connected, the practical cost c^*_{ij} of e_{ij} is its optimal estimation cost, that is, $c^*_{ij} = c_{ij}$; otherwise, $c^*_{ij} > c_{ij}$.

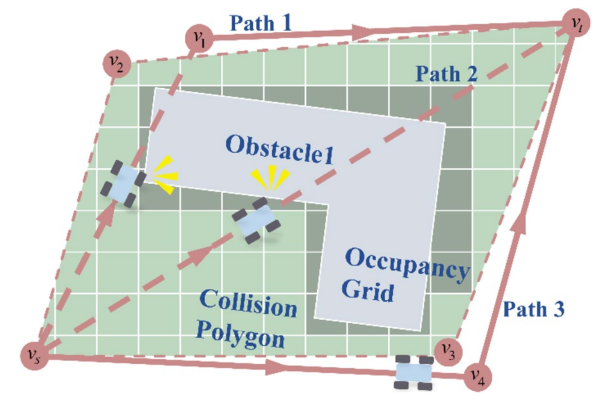

Figure 1 Schematic of basic definitions in the generalized graph

The generalized path from v_s to v_t consists of the ordered edge node set $P = \{e_{si}, e_{ij}, e_{jk}, ..., e_{nt}\}$, where e_{jk} is the successor edge node of e_{ij}. P is a connected path if and only if P does not contain disconnected edge nodes. Otherwise, P is a disconnected path. As shown in Figure 1, $P_1 = \{e_{s1}, e_{1t}\}$ and $P_2 = \{e_{st}\}$ are disconnected paths, and $P_2 = \{e_{s4}, e_{4t}\}$ is a connected path.

The cost of P is the sum of the costs of all edge nodes on the path. In a certain state of G, if the path with minimum cost from v_s to v_t is a disconnected path, it is the generalized optimum path from v_s to v_t. Its cost is the optimal estimation cost from v_s to v_t, denoted by $f(v_s, v_t)$. Otherwise, it is the practical optimum path from v_s to v_t. Its cost is the practical optimal cost from v_s to v_t, denoted by $f^*(v_s, v_t)$.

The shortest distance from v_m to e_{ij} is called the offset of v_m with respect to e_{ij}. The maximum offset of all points on P with respect to e_{st} is called the maximum offset of P. The points with the maximum offset on P are called the maximum offset points. In Figure 1, v_1 is the maximum offset point of $P_1 = \{e_{s1}, e_{1t}\}$.

A robot can move clockwise or counterclockwise to avoid a single obstacle O_i. In all paths that avoid O_i, the minimum value among the maximum offsets of the paths is called the minimum offset of O_i with respect to e_{ij}. The points with the minimum offset are called the minimum offset points. The minimum value among the maximum offsets of the paths in another obstacle avoidance direction is called the subminimum offset of O_i with respect to e_{ij}. The points with the subminimum offset are called the subminimum offset points. The closed polygon formed by the minimum offset points and the subminimum offset points connected with v_i and v_j, respectively, is called the collision polygon of O_i with respect to e_{ij}. As shown in Figure 1, the minimum offset point of Obstacle 1 with respect to e_{st} is v_2. The subminimum offset point is v_3, and the collision polygon is CP_{2s3t}.

The maximum value of the minimum offsets of all obstacles with respect to e_{ij} is called the minimum offset of e_{ij}. The subminimum offset of the corresponding obstacle with respect to e_{ij} is called the subminimum offset of e_{ij}. The points with the minimum offset and the subminimum offset are called the minimum offset points and subminimum offset points, respectively. As shown in Figure 1, the minimum offset point of e_{st} is v_2, and the subminimum offset point is v_3.

The operator Γ on the generalized graph is defined as follows. When e_{ij} is disconnected, the operation of adding a random vertex v_m outside the collision polygon of O_i with respect to e_{ij} and performing expansion to generate new edge nodes e_{im} and e_{jm} is called the Γ operation of e_{ij}. If both e_{im} and e_{jm} are connected, v_i and v_j can be indirectly connected through v_m. Obviously, applying Γ in connected edge nodes is meaningless.

As shown in Figure 1, e_{st} is a disconnected edge node. After vertex v_4 is added, it connects with v_s and v_t to form e_{s4} and e_{4t}, both of which are connected. This means that v_s and v_t are indirectly connected through v_4. To indirectly connect v_s and v_t through v_1, Γ needs to be applied to e_{s1}.

3 EDG* Heuristic Iterative Algorithm

3.1 The A* Algorithm

As the most widely used path planning algorithm for grid maps, A* adds heuristic information based on DA, making the algorithm search in a directional way. With vertices as nodes, A* selects the node with the minimum $f(n)$ in the open list as the current node. It will be moved to the closed list after being expanded. Then, the successor nodes that meet the requirements are stored in the open list. This process is repeated until the target node is found. The node cost $f(n)$ can be calculated as follows [25]:

$$f(n) = \alpha g(n) + \beta h(n), \tag{1}$$

where $f(n)$ denotes the optimal estimation cost from the starting node to the target node through the current node. $g(n)$ denotes the practical cost from the starting node to the current node. $h(n)$ denotes the optimal estimation cost from the current node to the target node. α and β are the weight coefficients. To balance the running time, path cost, and other factors, Euclidean distance is generally used to calculate $g(n)$, and Manhattan distance is used to calculate $h(n)$.

Considering the planning problems in Figure 2 [16], v_s is the initial position of the robot, and v_t is the target position. As mentioned above, A* implicitly specifies the graph G. The robot has eight moving directions at each grid, as shown by the arrows at v_i in Figure 2. v_s and v_2 can be directly connected, but limited by the searching strategy, e_{s2} will not be recognized. As A* expands in the diagonal direction, a path segment that crosses the wall corner is likely to be formed, as shown in Figure 2. It may cause the collision of the robot with the obstacle. Additionally, A* performs a large number of searches in the collision polygon. Most of the gray grids in Figure 2 are not related to the ultimate path, but multiple operations will be performed on them, which increases the computation amount.

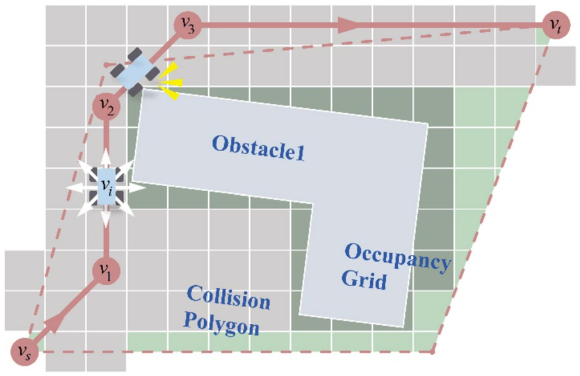

Figure 2 Schematic of the A* algorithm on the node search strategy, collision with the corner, etc.

3.2 Expanding Disconnected Graph

In connected graphs, path planning problems are generally defined as graph search problems. That is, the connected graphs are implicitly specified through $v_s \in \{v_s, v_t\}$ and Γ. Path planning is completed in the process of explicating graphs. In this paper, path planning is described as the process of expanding a disconnected graph into a connected graph. Specifically, with a focus on the graph G formed by the starting vertex v_s, target vertex v_t, and initial edge node e_{st}, this paper performs Γ on the disconnected edge nodes in G and updates G until it becomes a connected graph.

The core of EDG is to find a collision-free path from v_s to v_t by expanding the disconnected edge nodes in G. Through a rough description of how the algorithm works, it is clear what is meant by expanding a disconnected graph. EDG randomly selects a disconnected edge node in G as the current edge node, performs Γ on the edge node, updates graph G, and randomly selects the next disconnected edge node and performs Γ on it. If there is a connected path in G, that is, a collision-free path from v_s to v_t. As shown in Figure 3, if the robot needs to avoid Obstacle 1, it is unreasonable to perform path searching in the collision polygon as in A*. Hence, for the Γ operation in EDG, random points outside the collision polygon need to be selected as new vertices of graph to complete the planning tasks. In the generalized graph considering the planning problems in Figure 2, the initial edge node interacts with the obstacle and is the unique disconnected edge node in G. The Γ operation is performed on

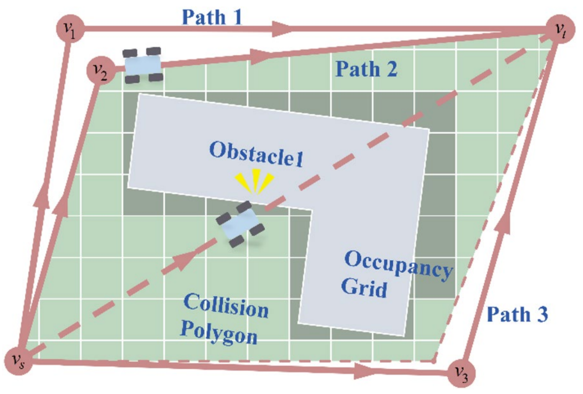

Figure 3 Schematic of the expanding disconnected graph

e_{st}, and points v_1 and v_3 are selected randomly outside the collision polygon as the new vertices of G. In this case, there are two connected paths that is Path 1 and Path 3. The cost of Path 3 is lower than that of Path 1. Therefore, Path 3 is the solution obtained by EDG.

In the single-obstacle map, Path 2 also exists, and its cost is lower than that of Path 1 and Path 3. It's the optimal path that the EDG could have obtained in the grid. In cases of multi-obstacle maps, selection of disconnected edge node for the Γ operation and a point outside the collision polygon as a new vertex of the graph for planning of an optimal collision-free path similar to Path2 would be left to EDG*.

3.3 Heuristic Information

To obtain the optimum path with the fewest edge nodes expansions, the algorithm needs to continuously make as many informed decisions as possible about the next edge node to be expanded and how to expand it. The expanded edge nodes or the successor edge nodes obtained through expansion should be on the optimum path as much as possible. Additionally, if the edge nodes that may be on the optimum path are ignored, the optimal solution may not be found. Efficient algorithms need some heuristic information to determine which edge node should be expanded next and how to expand it. Let $f^*(e_{ij})$ be the practical optimal cost of the practical optimum path from v_s to v_t through e_{ij}. Let $f^*(v_s, v_t)$ be the practical cost of the practical optimum path from v_s to v_t. If e_{ij} is located on the optimum path, then $f^*(e_{ij}) = f^*(v_s, v_t)$, and vice versa. If e_{ij} is not located on the optimum path, then $f^*(v_s, v_t) < f^*(e_{ij})$, and vice versa. Let $f(e_{ij})$ be the optimal estimation cost of the generalized optimum path from v_s to v_t through e_{ij}. Then, $f(e_{ij})$ is the optimal estimation of $f^*(e_{ij})$ and $f(e_{ij}) < f^*(e_{ij})$, and vice versa. If e_{ij} is located on the optimum path, then $f(e_{ij}) < f^*(e_{ij}) = f^*(v_s, v_t)$, but the converse is not necessarily true. If e_{ij} is not located on the optimum path, then $f^*(v_s, v_t) < f(e_{ij})$ is not necessarily true. If $f^*(v_s, v_t) < f(e_{ij})$ holds, it is certain that e_{ij} is not on the optimum path. Overall, the smaller $f(e_{ij})$ is, the more likely the edge node is to locate on the practical optimum path. Although $f^*(v_s, v_t)$ is not a priori, using the optimal estimation $f(e_{ij})$ of $f^*(e_{ij})$ as the edge node evaluation function is reasonable. It can be written as a sum of three terms:

$$f(e_{ij}) = \alpha g(e_{ij}) + \beta h_1(e_{ij}) + \gamma h_2(e_{ij}). \tag{2}$$

For a certain edge node e_{ij} in P, $h_1(e_{ij}) = c_{ij}$, c_{ij} represents the optimal estimation cost of e_{ij}. $g(e_{ij})$ is the sum of the practical costs of the connected edge nodes among all edge nodes in P except for e_{ij}. $h_2(e_{ij})$ is the sum of the optimal estimation costs of the disconnected edge nodes among all edge nodes in P except for e_{ij}. α, β and γ are weight coefficients.

If $f^*(e_{ij})$ is the practical cost of $f(e_{ij})$ and is the sum of the practical costs $g^*(e_{ij})$, $h_1^*(e_{ij})$ and $h_2^*(e_{ij})$ of $g(e_{ij})$, $h_1(e_{ij})$ and $h_2(e_{ij})$, then $f(e_{ij}) \leq f^*(e_{ij})$ always holds. There exist three relations between e_{ij} and obstacles.

Case 1: if e_{ij} is collision-free with any obstacle, then e_{ij} is connected. The Γ operation is not required for e_{ij}. Robot can move along e_{ij} without the risk of collision with any obstacle.

Case 2: if e_{ij} collides with a unique obstacle, then e_{ij} is disconnected. The Γ operation is required for e_{ij}. Although it is possible for the robot to collide with other obstacles while avoiding this unique obstacle, it is assumed that the robot only needs to avoid this obstacle and the other obstacles will be ignored.

Case 3: if e_{ij} collides with multiple obstacles, then e_{ij} is disconnected. The Γ operation is required for e_{ij}. It is obvious that the robot needs to bypass multiple obstacles to reach the target vertex. It is assumed that the robot will preferentially bypass the obstacle corresponding to the minimum offset with respect to e_{ij}, and other obstacles will be ignored.

EDG* uses the optimal successor edge node generation method to perform the Γ operation. The process is as follows:

(1) Some points are selected from the minimum and subminimum offsets of the current disconnected edge node according to the preset rule.
(2) The selected points are added as new vertices of G.
(3) The new vertices are connected with two vertices of the current edge node to form new edge nodes.
(4) The current disconnected edge node is removed from G.

It is a special case that the offset from the new vertex to the predecessor of the current edge node is greater than the minimum offset of the latter. In this case, the new vertex also needs to form a new edge node with another vertex of the latter. If this is neglected, in some special planning problems, the generalized optimum path of the current state may not be generated correctly, resulting in a local optimum.

3.4 Map Pre-processing

In the grid map, each grid has only one attribute value, occupancy or blank. This lacks the most basic semantic property of which grids represent the same obstacles in a real-world context. An unpartitioned grid map cannot reflect the actual spatial distribution of obstacles faced by a robot, which is not conducive to global path planning. Whereas EDG* performs path planning on the partitioned map, a simple and efficient obstacle partitioning method was proposed in this paper. If there are other occupancy grids in the eight neighboring grids of an occupancy grid, these occupancy grids belong to the same obstacle. The corresponding map area is traversed to complete the partitioning.

EDG* also obtains heuristic information from obstacle offsets. In the grid map, it is easily proven that the minimum offset of an obstacle to an edge node can only come from the convex corner points of the obstacle. In the map pre-processing stage, finding the concave and convex corner points of each obstacle in advance is necessary.

In a nine-pane grid, if the blank grid v_i is geometrically opposite to the occupancy grid of an obstacle and the other two grids are blank grids, v_i is the convex corner point of the obstacle. If the other two grids are occupancy grids, v_i is the concave corner point of the obstacle. As shown in Figure 4, v_1 is the convex corner point of Obstacle 1. v_3 is the concave corner point of Obstacle 1.

A corner point search method with the time property was proposed. For an obstacle, the upper right corner of the occupancy grid that is the first time to join the obstacle must be the convex corner point of the obstacle. Starting from the convex corner point, the center of the nine-pane grid is moved counterclockwise along the boundary of the obstacle to search for corner points to obtain the corner points of the obstacle. It is shown in Figure 4. The corner points are connected in order of time. The closed polygon obtained is the approximate outer contour of Obstacle 1.

3.5 Algorithm Description of EDG*

The EDG that introduces the evaluation function shown in Eq. (2) and the optimal child node generation method is denoted as EDG*. By means of $f(e_{ij})$, the costs of all disconnected edge nodes are evaluated. The evaluation

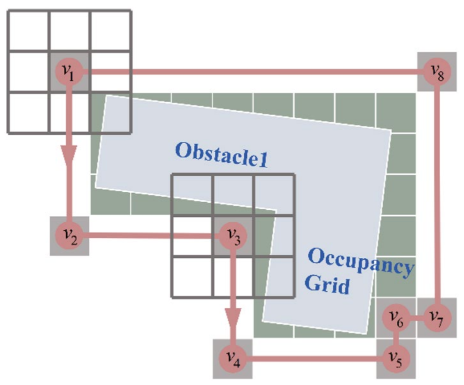

Figure 4 Schematic of map pre-processing

results are used to guide EDG* to select the next edge node for Γ from the disconnected edge nodes. The disconnected edge node with the minimum $f(e_{ij})$ will be the edge node to be expanded next. If there are multiple edge nodes with the minimum $f(e_{ij})$, the edge node with the maximum h_1 is selected. If there are still multiple edge nodes, the edge node is selected based on the breadth priority principle. The priority queue Q_{open} is used to store all leaf edge nodes in the current state. If the current edge node is disconnected, move it to the set S_{closed}; otherwise, move it to the set S_{candi}.

Algorithm 1 EDGstar_Pathfingding

```
Input : map_obstacle, start vertex $v_s$, target vertex $v_t$
Output : $result$ of path planning
1: $e_{st} \Leftarrow (v_s, v_t)$, $Q_{open} \Leftarrow \{e_{st}\}$, $S_{candi} \Leftarrow \phi$, $S_{closed} \Leftarrow \phi$;
2: while $Q_{open} \neq \phi$ do
3:     $n \Leftarrow$ GetHead$(Q_{open})$;
4:     if $n$ is disconnected then
5:         move $n$ to $S_{closed}$;
6:         $\Gamma(n)$;
7:         calculate $f$ for each successor of $n$;
8:         move each successor not in $S_{closed}$ or $Q_{open}$ to $Q_{open}$
9:         update $f$ of all leaf edge nodes in graph $G$;
10:        continue form line 2;
11:    else
12:        move $n$ to $S_{candi}$;
13:        update $f$ of all leaf edge nodes in graph $G$;
14:        if $h_2(n) = 0$ then
15:            break form line 2;
16:        end if
17:    end if
18: end while
19: return all edge nodes with $h_2 = 0$ in $S_{candi}$ as $result$.
```

EDG* must obtain the estimates $g(e_{ij})$ and $h(e_{ij})$ of $g^*(e_{ij})$ and $h^*(e_{ij})$, respectively. For path planning in a two-dimensional grid map, the cost mainly comes from the distance information in the map. The common forms of cost are Euclidean and Manhattan distances. In this paper, Euclidean distance is used to calculate all the costs. It can ensure the algorithm's time efficiency and convergence with the minimum number of expanded edge nodes.

4 Path Planning Experiments

To verify the effectiveness of EDG* and to objectively evaluate its planning efficiency, path quality, and environmental adaptability, the following three experiments were conducted by considering the map size and environmental complexity.

Experiment 1: Evaluate the map size exponential explosion effect of the evaluation metrics of EDG*. Experiment 1 was conducted in the same typical indoor environment. By setting different resolutions, six maps of different sizes from 32×32 to 1024×1024 were created. The starting vertex of planning problems was located at the lower-left corner of the map. The target vertex was in the upper-right corner. Due to the random error in the running time, each planning problem was run 20 times independently. The running time of each problem presented in Experiment 1 was the average time of the 20 runs.

Experiment 2: Evaluate the path planning performance of EDG* on maps of different environments. Experiment 2 used the map dataset of 30 cities openly available on the Moving AI Lab [41, 42]: City Street Maps dataset (CSM). The map size ranged from 256×256 to 1024×1024. A maximum size of 1024×1024 was used in this paper. The starting and target vertices of the planning problems and the number of runs were the same as in Experiment 1.

Experiment 3: Evaluate the overall performance and robustness of EDG* under ultra-large planning problems. Experiment 3 was conducted in CSM, with the starting and target vertices specified by the scan file of CSM. CSM presented a total of 113200 planning problems with different starting and target vertices in 30 city street maps and the optimal length of benchmarks. The total running time presented in Experiment 3 was the average time of three independent runs.

At the start of each experiment, evaluation metrics that matched the purpose of the experiment were set. Additionally, each experiment included the same experimental setup as follows.

(1) Comparison algorithms: A*, DA, and JPS were selected as the comparison algorithms. A* and JPS versions were selected from the top starred versions on GitHub over the past 5 years. The selected A* and JPS were open-sourced and maintained by Yu Hu from Shanghai Jiao Tong University, China. The heuristic information of the selected A* was set to 0 to obtain the DA used in this paper.

(2) Heuristic information: A* heuristic information was the Manhattan distance from the current vertex to the target vertex. DA had no heuristic information. JPS heuristic information was the Euclidean distance from the current vertex to the target vertex. All the distances in EDG* were Euclidean distances. The heuristic information was given by Eq. (2). Each weight coefficient was 1.

(3) Programming language: All algorithms were written in C++.

(4) Hardware environment: All algorithms were run on a computer with a 4.00 GHz Intel Core i7-6700K CPU and 3200 MHz 16G RAM.

4.1 Experiment 1

In Experiment 1, the exponential explosion effects of EDG* planning efficiency and path quality on map size were investigated to evaluate the optimization effects of EDG* compared to the competitive algorithms. The planning efficiency was evaluated based on the number of algorithm iterations, number of successor nodes, and running time, with the running time as the main evaluation metric. The path quality was evaluated based on path smoothness, collision avoidance performance, and length. In this paper, the path smoothness was characterized by the number of path turns. The path collision avoidance performance was characterized by the number of path collisions with wall corners.

Figure 5 shows the path planning results of the four algorithms in the 1024×1024 indoor environment map. The black grid is the occupancy grid. The white grid is the blank grid. The folded line is the path generated by the corresponding algorithm. The detailed experimental results are shown in Tables 1 and 2.

As seen in Table 1, under the same environment but with different map resolutions, A*, DA, JPS, and EDG* could accomplish the specified path planning tasks. EDG* outperformed A*, DA, and JPS in terms of the number of algorithm iterations, number of successor nodes, and running time. Both the number of algorithm iterations and the number of successor nodes of A* and DA increased exponentially with map size. The JPS and EDG* were basically not affected by the map size, as shown in Figures 6 and 7. The running time of all four algorithms increased with map size. The running times of A*, DA, and JPS were greatly affected by the map size, exponentially increasing with the map size. The running time of EDG* increased at a slower speed, as shown in Figure 8.

As seen in Table 1, under the same environment but with different map resolutions, A*, DA, JPS, and EDG* could accomplish the specified path planning tasks. EDG* outperformed A*, DA, and JPS in terms of the number of algorithm iterations, number of successor nodes, and running time. Both the number of algorithm iterations and the number of successor nodes of A* and DA increased exponentially with map size. The JPS and EDG* were basically not affected by the map size, as shown in Figures 6 and 7. The running time of all four algorithms increased with map size. The running times of A*, DA, and JPS were greatly affected by the map size, exponentially increasing with the map size. The running time of EDG* increased at a slower speed, as shown in Figure 8.

According to Table 2, EDG* outperformed A*, DA, and JPS in terms of the number of path turns, number of corners passed by the path, and path length. The four algorithms exhibited no significant exponential expansion effect on the map size in the number of path turns and the number of corner paths passed through. The path lengths of the four algorithms did not differ significantly, increasing with the map size.

4.2 Experiment 2

Experiment 2 investigated the optimization degrees of EDG* in terms of planning efficiency and path quality on maps of different environments compared to the competitive algorithms. The optimization rate of the six metrics of EDG* compared to the competitive algorithms was evaluated in the range of 0–100% [43] and calculated as follows:

$$RE = \max\left\{\frac{E_i - E_{EDG*}}{E_i} \times 100\%, 0\right\}, \quad (3)$$

where E_i is the corresponding metric value of the competitive algorithms. E_{EDG*} is the corresponding metric value of EDG*. RE characterizes the degree of optimization of this metric of EDG* compared to the competitive algorithms. Figure 9 shows some of the planning results of the four algorithms in CSM.

Unlike A*, DA, and JPS, which use the neighborhood searching strategy, EDG* selects the next edge node by a well-designed evaluation function and avoids collisions,

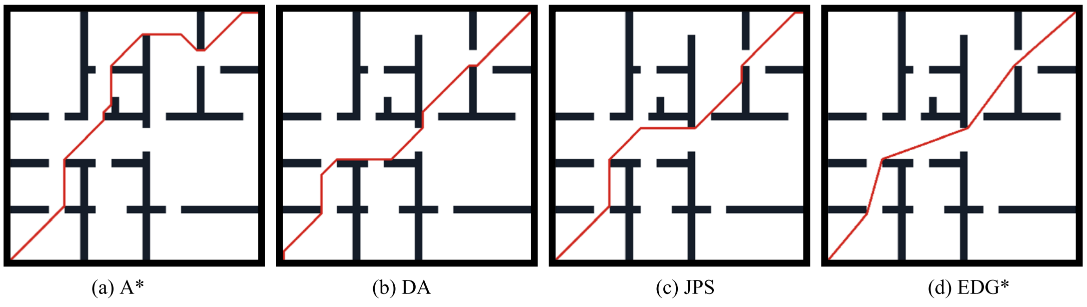

Figure 5 Comparison of paths planned by EDG* and three competitive algorithms on the 1024 × 1024 indoor environment map

Table 1 Comparison of evaluation metrics of path planning efficiency between EDG* and competitive algorithms on indoor environment maps of different sizes

Metrics	Map size	32 × 32	64 × 64	128 × 128	256 × 256	512 × 512	1024 × 1024
Number of iterations↓	A*	165	527	1739	6575	25009	97981
	DA	892	3568	14272	57088	228352	913408
	JPS	24	29	28	28	28	28
	EDG*	9	9	9	9	9	9
Number of nodes↓	A*	1104	3842	13196	51175	197482	778684
	DA	6039	26259	109515	447291	1807899	7269339
	JPS	33	38	37	37	37	37
	EDG*	12	12	12	12	12	12
Time↓(ms)	A*	2.2	7.1	24.2	87.7	334.5	1350.1
	DA	10.5	41.6	167.7	646.9	2547.9	10280.6
	JPS	0.3	0.7	2.4	8.8	34.7	139.0
	EDG*	0.2	0.3	0.3	0.4	0.7	1.2

Table 2 Comparison of evaluation metrics of path quality between EDG* and competitive algorithms on indoor environment maps of different sizes

Metrics	Map size	32×32	64×64	128×128	256×256	512×512	1024×1024
Number of turns↓	A*	7	11	11	11	11	11
	DA	9	9	9	9	9	9
	JPS	7	7	7	7	7	7
	EDG*	4	4	4	4	4	4
Number of corners↓	A*	4	6	6	6	6	6
	DA	4	4	4	4	4	4
	JPS	4	4	4	4	4	4
	EDG*	0	0	0	0	0	0
Length↓	A*	48.5	107.3	215.2	431.0	862.5	1725.6
	DA	48.5	98.5	198.4	398.1	797.7	1596.8
	JPS	48.5	98.5	198.4	398.1	797.7	1596.8
	EDG*	47.8	95.1	189.8	379.5	758.9	1517.7

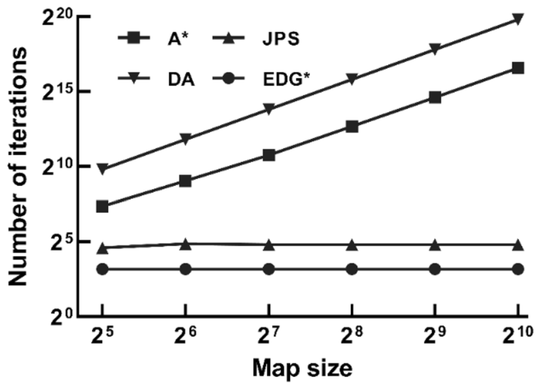

Figure 6 Comparison of the number of algorithm iterations between EDG* and three competitive algorithms on indoor environment maps of different sizes

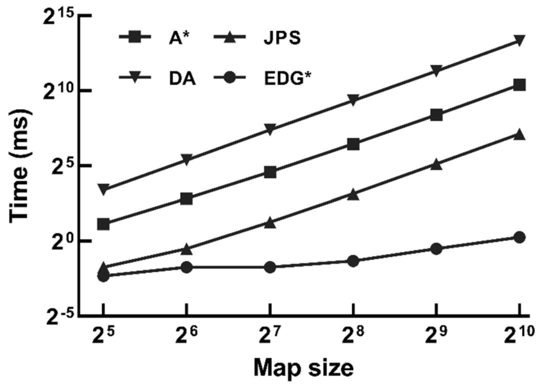

Figure 8 Comparison of the running time between EDG* and three competitive algorithms on indoor environment maps of different sizes

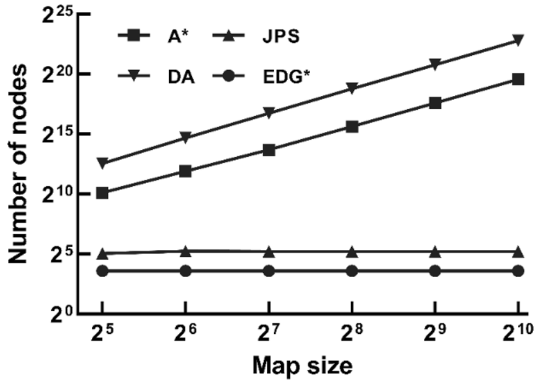

Figure 7 Comparison of the number of successor nodes between EDG* and three competitive algorithms on indoor environment maps of different sizes

expanding the edge node according to the minimum offset. It can reduce the number of algorithm iterations and the number of successor nodes by several orders of magnitude, reducing the time complexity and running time. Table 3 shows the experimental results of the number of algorithm iterations, number of successor nodes, and running time of the four algorithms on different environment maps. The results show that A*, DA, JPS, and EDG* can accomplish the specified path-planning tasks on maps of different environments. The number of algorithm iterations of EDG* was reduced by more than 95% on average compared with that of the three competitive algorithms. The number of successor nodes of EDG* was reduced by more than 90% on average compared with that of the three competitive algorithms. The running time of EDG* was decreased by 79.9%–99.2% compared with A*, with an average decrease of 94.3%. The running time of EDG* was reduced by 99.7%–99.9% compared with DA, with an average decrease of 99.9%. The running time of EDG* was reduced by 86.7%–99.7% compared with JPS, with an average decrease of 95.1%. The EDG* outperformed A*, DA, and JPS in terms of the iteration number, number of successor nodes, and running time.

In A* and JPS, the cost of expanding nodes along a diagonal direction was generally smaller. The dangerous paths involving a collision with corners in Figure 4 were likely to be generated.

The proposed EDG* expanded successor edge nodes according to the obstacle corner points obtained in advance. It greatly reduced the risk of collision between the robot and corners. The A*, DA, and JPS are angle-restricted search algorithms because they can only search in specific directions. The paths planned by these algorithms are not optimum global paths. EDG* allows the robot to go in any direction and thus belongs to any-angle search algorithms. It can reduce the number of path turns, improve path smoothness, and reduce the path length.

Table 4 shows the experimental results of the number of path turns, number of corners passed through, and path length of the four algorithms on different environment maps. The number of turns of EDG* was reduced by more than 80% on average compared with that of the three competitive algorithms. The number of corners passing through of EDG* was reduced by more than 95% on average compared with that of the three competitive algorithms. The path length of EDG* was reduced by 0.8%–12.0% compared with that of A*, with an average decrease of 4.9%, and by 0.8%–5.7% compared with that of DA and JPS, with an average decline of 3.7%. The results show that EDG* outperformed A*, DA, and JPS in terms of path smoothness, collision avoidance performance, and path length.

4.3 Experiment 3

The CMS dataset provided 113200 planning problems in 30 maps as well as the optimal length of benchmarks. Experiment 3 used four algorithms to solve the 113200 planning problems independently and defined the metrics in Table 5 to evaluate the algorithms.

Experiment 3 aimed to verify the overall performance and robustness of the corresponding algorithms in ultra-large planning problems. The total running time

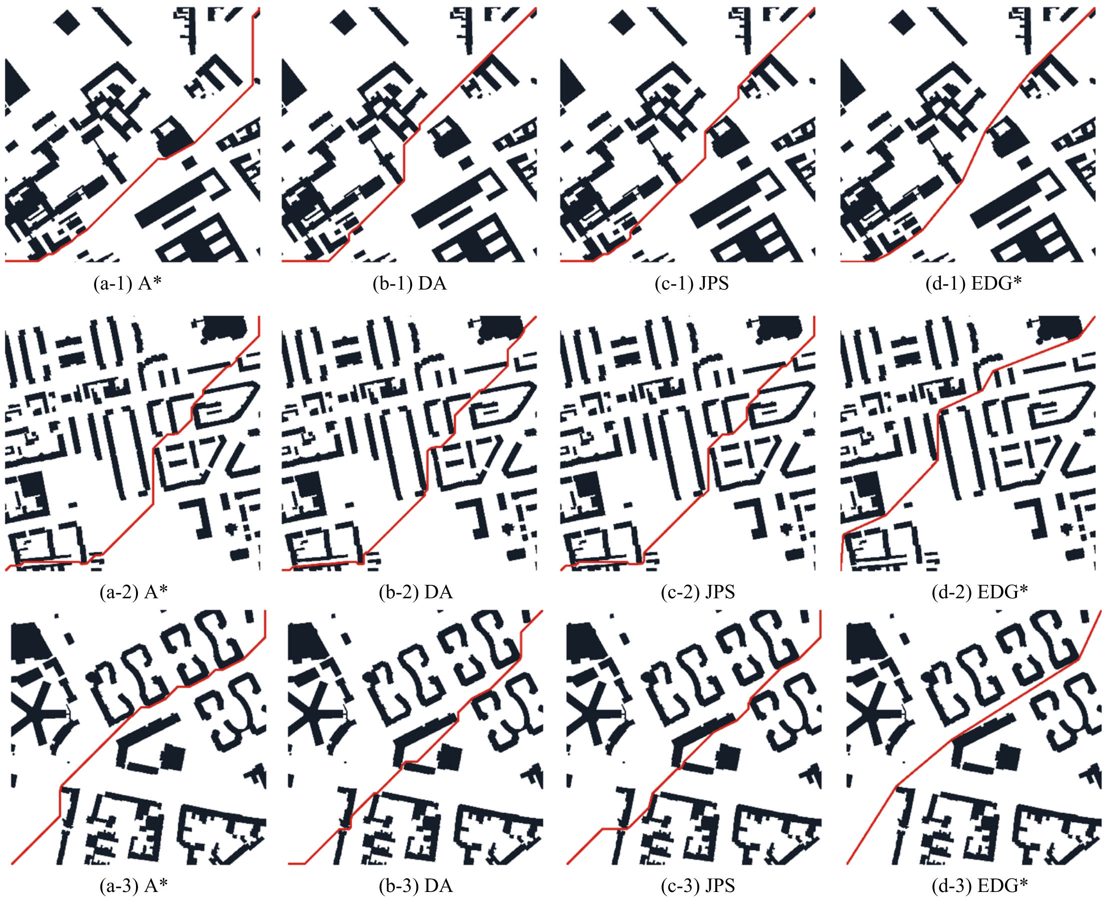

Figure 9 Comparison of paths planned by EDG* and the competing algorithm on the 1st to 3rd maps in the 1024 × 1024 CSM

Table 3 The ratio of evaluation metrics of path planning efficiency of EDG* to competitive algorithms on the maps in the 1024 × 1024 CSM

Metrics	**Number of iterations↓**			**Number of nodes↓**			**Time↓**		
Map number	**EDG* VS**			**EDG* VS**			**EDG* VS**		
	A* (%)	**DA (%)**	**JPS (%)**	**A* (%)**	**DA (%)**	**JPS (%)**	**A* (%)**	**DA (%)**	**JPS (%)**
1	99.9	99.9	97.0	99.9	99.9	95.1	95.8	99.8	88.3
2	99.9	99.9	98.0	99.9	99.9	96.1	98.9	99.9	96.2
3	99.7	99.9	98.5	99.9	99.9	97.1	94.7	99.9	95.0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
28	99.1	99.9	85.7	99.8	99.9	83.1	96.0	99.9	96.7
29	99.3	99.9	97.4	99.8	99.9	95.6	95.8	99.9	98.4
30	99.5	99.9	98.3	99.9	99.9	97.2	95.4	99.9	97.7
Min.	97.9	99.9	84.7	99.4	99.9	82.9	79.9	99.7	86.7
Avg.	99.5	99.9	96.1	99.9	99.9	94.3	94.3	99.9	95.1
Max.	99.9	99.9	99.4	99.9	99.9	99.2	99.2	99.9	99.7

in Table 6 is the average time of the corresponding algorithms running three times independently for 113200 planning problems. The running time included the time of loading pre-processed data and path planning time but excluded the map pre-processing time.

As shown in Table 6, the running time of EDG dropped by more than 99% compared with those of A* and DA and by 95.9% compared with that of JPS. Meanwhile, the path length of EDG* was reduced by 6.0% compared with that of A* and by 4.6% compared with those of DA and JPS.

As observed, EDG* found a valid solution for all 113200 planning problems, while A*, DA, and JPS failed to find valid solutions for seven problems. In terms of the optimal solution rate, EDG* outperformed the competitive algorithms. A* failed to find optimal solutions for 38612 problems, with an optimal solution rate of only 65.9%. DA and JPS failed to find optimal solutions for 99 problems. EDG* did not find optimal solutions for five problems. The non-optimal path cases are shown in Figure 10. For the No. 2261 path planning problem in the 13th map of the CSM dataset, the three comparison algorithms all give optimal solutions with path lengths less than the benchmark length, and the path lengths are all 1067.02, but the path solution length of the EDG* algorithm is 1068.23, while the benchmark optimal path length is 1067.57, as shown in the comparison diagram of group (a) in Figure 10. the No. 3099 path planning problem in the 14th map of the CSM dataset, the EDG* algorithm gives the optimal solution whose length is less than the benchmark's length, and the path length is 1148.33, but none of the three comparison algorithms gives the optimal solution, in which the A* algorithm gives the path solution length of 1246.55, and the other two comparison

Table 4 The ratio of evaluation metrics of path quality of EDG* to competitive algorithms on the maps in the 1024 × 1024 CSM

Metrics	**Number of turns↓**			**Number of corners↓**			**Length↓**		
Map number	**EDG* VS**			**EDG* VS**			**EDG* VS**		
	A* (%)	**DA (%)**	**JPS (%)**	**A* (%)**	**DA (%)**	**JPS (%)**	**A* (%)**	**DA (%)**	**JPS (%)**
1	95.8	52.6	91.3	99.9	99.9	99.9	6.5	3.2	3.2
2	84.7	91.1	84.7	99.9	99.9	99.9	4.8	4.1	4.1
3	98.7	96.9	98.6	99.9	99.9	99.9	5.8	4.8	4.8
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
28	60.0	53.8	60.0	99.9	99.9	99.9	2.3	2.3	2.3
29	76.2	90.6	76.2	96.6	96.9	96.6	2.9	2.9	2.9
30	93.8	93.8	93.8	95.1	94.0	95.1	4.0	4.0	4.0
Min.	46.2	33.3	46.2	87.5	83.0	93.0	0.8	0.8	0.8
Avg.	85.3	81.9	84.8	98.8	98.5	99.1	4.9	3.7	3.7
Max.	98.7	98.9	98.6	99.9	99.9	99.9	12.0	5.7	5.7

Table 5 Evaluation metrics defined in Experiment 3

Metrics	The definition of metrics
TT (ms)	Total time required for path planning 113200 times
AT (ms)	Average time spent on one path planning
TR	Time optimization rate of EDG* compared to competitive algorithms. It can be calculated by Eq. (3)
TBL	Sum of benchmarks' optimal length
ABL	Average of benchmarks' optimal length
TL	Total length of 113200 path planning problem
AL	Average length of 113200 path planning problem
LOR	Length optimization rate of EDG* compared to competitive algorithms. It can be calculated by Eq. (3)
NPS	Number of paths solved
NPI	Number of paths with no solution or invalid
NOPL	Number of optimum path lengths
OPLR	Optimum path length ratio
TMPT (ms)	Total map pre-processing time required for path planning 113200 times
AMPT (ms)	Average map pre-processing time required for path planning 113200 times

Table 6 Comparison of evaluation metrics defined in Experiment 3 between EDG* and competitive algorithms on 113200 path planning problems in the 1024 × 1024 CSM

Metrics	A*	DA	JPS	EDG*
TT (ms) ↓	56510903	592191778	10739162	435590
AT (ms) ↓	499.2	5231.4	94.9	3.8
TR (%)	99.2	99.9	95.9	/
TBL	85561250			
ABL	755.8			
TL↓	86586680	85319950	85319950	81426090
AL↓	764.9	753.7	753.7	719.3
LOR (%)	6.0	4.6	4.6	/
NPS↑	113193	113193	113193	113200
NPI↓	7	7	7	0
NOPL↑	74588	113101	113101	113195
OPLR↑ (%)	65.9	99.9	99.9	100.0
TMPT (ms)↑	0	0	0	3325
AMPT (ms)↑	0	0	0	0.03

algorithms give the path solution length of 1237.18, while the benchmark optimal path length is 1236.31, as shown in the comparison diagram of group (b) in Figure 10.

EDG* needs to pre-process the map before planning. As indicated, for the 113200 planning problems in CSM, the time consumed by map pre-processing was 3325 ms, with the average time consumed by each map of 110.8 ms, and the average time consumed on each planning problem of 0.03 ms.

Since some of the planning problems from CSM have their starting and target vertices overlapping and the competitive algorithms failed to handle this, no solution was found for seven planning problems. To save the dataset integrity and the rigor of the experiment, the planning problems unfavorable to the competitive algorithms were not removed from Experiment 3.

The optimal solution rate of A* was much lower than that of the other three algorithms, probably because the heuristic distance of A* used the Manhattan distance by convention. The Manhattan distance significantly reduced the computation amount of the node cost update and running time. The probability of nodes expanding diagonally also increased significantly, resulting in the failure of A* to obtain the optimal solution that angle-restricted search algorithms could find. In an additional experiment, the A* heuristic information was replaced with the Eulerian distance. In this case, the optimal solution rate of A* was the same as that of DA and JPS, but its running time increased significantly. A* algorithm can hardly identify the global optimal solution that can be found by Any-angle Search algorithms due to the limitation of angle searching strategies. A* algorithm can neither identify the optimal solution that can be found by Angle-restricted Search algorithms using the Manhattan distance as the heuristic information was calculated by using Manhattan distance.

Experiment 3 verified the effectiveness and high robustness of EDG* in the 113200 planning problems from the CSM dataset. The results demonstrate that EDG* outperformed the competitive algorithms in terms of running time, path length, and optimum path rate.

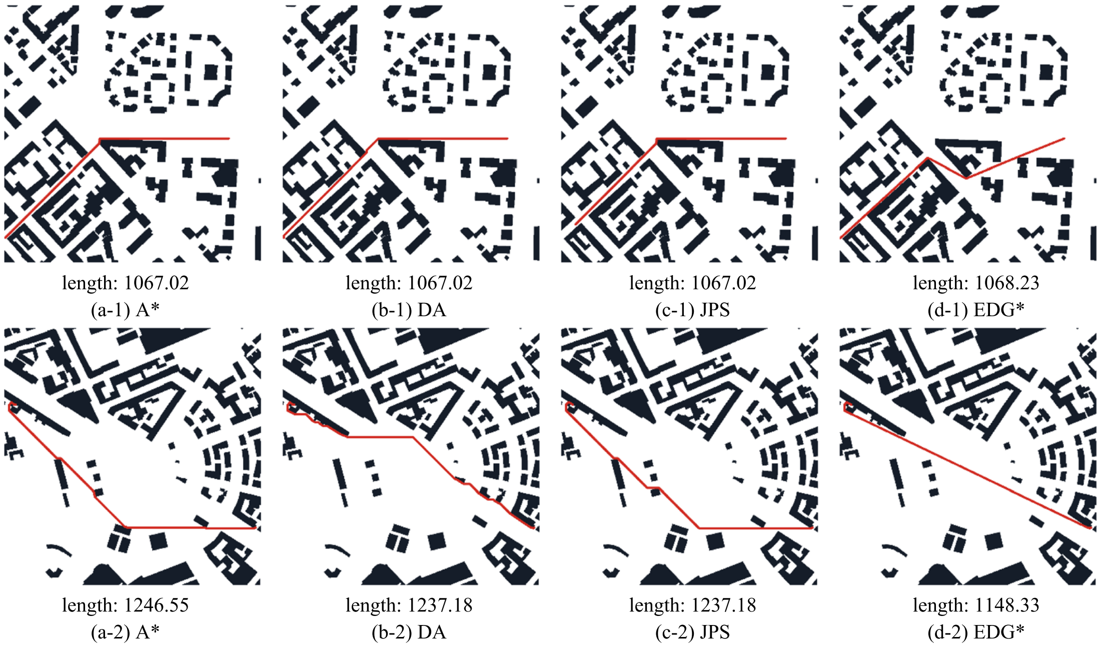

Figure 10 Comparison of EDG* and competitive algorithms for nonoptimal programming cases in the 1024 × 1024 CSM

5 Conclusions

Through improvements in the successor node expansion strategy of graph planning algorithms, an expanding disconnected graph (EDG) algorithm for global planning problems of mobile robots was proposed. A heuristic function called EDG* was presented to optimize EDG in path planning by finding edge nodes and expanding successor edge nodes in the pre-processed map. It replaced the operation on neighboring nodes in conventional graph planning algorithms, thus improving the path planning efficiency. In different complexity environment maps, EDG*'s running time dropped by more than 90% and total length of paths reduced by more than 4.6% compared with A*, DA, and JPS algorithms. Its planning efficiency has no exponential explosion dilemma on map size. The path quality evaluation metrics of EDG*, such as path length, smoothness, and collision avoidance, also outperformed those of the competitive algorithms. The extensive path planning experimental results indicate that EDG* application mitigated the low planning efficiency and poor path quality deficiencies of conventional algorithms.

This paper focused on proposing EDG* and verifying its effectiveness. In the future, more work will be put into further optimizing EDG* and using it for path planning in dynamic or 3D environments.

Acknowledgments
The authors would like to thank all the colleagues who contributed to this research.

Authors' Contributions
YT contributed by leading and supervising trials, providing constructive guidance, and reviewing manuscripts; LD contributed by providing methodology and design procedures to verify and write manuscripts; HG contributed by writing guidance; YZ contributed by data curation; YS contributed by visualization presentation; TW contributed by giving constructive guidance to the experiment, supervision, and review of manuscripts. All authors read and approved the final manuscript.

Funding
Supported by National Key Research and Development Program of China (Grant No. 2022YFB4700402).

Data Availability
The data that support the findings of this paper are openly available in GitHub at https://github.com/ldcookie/EDGstar_Pathfinding.

Declarations

Competing Interests
The authors declare no competing financial interests.

Received: 17 April 2023 Revised: 22 February 2024 Accepted: 12 March 2024
Published online: 28 March 2024

References
[1] B K Patle, L G Babu, A Pandey, et al. A review: on path planning strategies for navigation of mobile robot. *Defence Technology*, 2019, 15(4): 582–606.
[2] A Loganathan, N S Ahmad. A systematic review on recent advances in autonomous mobile robot Navigation. *Engineering Science and Technology, an International Journal*, 2023, 40: 101343.
[3] B Hichri, A Gallala, F Giovannini, et al. Mobile robots path planning and mobile multirobots control: A review. *Robotica*, 2022, 40(12): 4257–4270.
[4] M Abed, O Lutfy, Q Al-Doori. A review on path planning algorithms for mobile robots. *Engineering and Technology Journal*, 2021, 39(5A): 804–820.

[5] M N A Wahab, S Nefti-Meziani, A Atyabi. A comparative review on mobile robot path planning: classical or meta-heuristic methods? *Annual Reviews in Control*, 2020, 50: 233–252.
[6] M Huppi, L Bartolomei, R Mascaro, et al. T-PRM: Temporal probabilistic roadmap for path planning in dynamic environments. *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Kyoto, Japan, October 23–27, 2022: 10320–10327.
[7] L E Kavraki, P Svestka, J C Latombe, et al. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 1996, 12(4): 566–580.
[8] Y Li, W Wei, Y Gao, et al. PQ-RRT*: An improved path planning algorithm for mobile robots. *Expert Systems with Applications*, 2020, 152: 113425.
[9] J Wang, M Meng, O Khatib. EB-RRT: Optimal motion planning for mobile robots. *IEEE Transactions on Automation Science and Engineering*, 2020, 17(4): 2063–2073.
[10] Y V Pehlivanoglu, P Pehlivanoglu. An enhanced genetic algorithm for path planning of autonomous UAV in target coverage problems. *Applied Soft Computing*, 2021, 112: 107796.
[11] M Nazarahari, E Khanmirza, S Doostie. Multi-objective multi-robot path planning in continuous environment using an enhanced genetic algorithm. *Expert Systems with Applications*, 2019, 115: 106–120.
[12] M D Phung, Q P Ha. Safety-enhanced UAV path planning with spherical vector-based particle swarm optimization. *Applied Soft Computing*, 2021, 107: 107376.
[13] C Huang, X Zhou, X Ran, et al. Adaptive cylinder vector particle swarm optimization with differential evolution for UAV path planning. *Engineering Applications of Artificial Intelligence*, 2023, 121: 105942.
[14] C Miao, G Chen, C Yan, et al. Path planning optimization of indoor mobile robot based on adaptive ant colony algorithm. *Computers & Industrial Engineering*, 2021, 156: 107230.
[15] C Ntakolia, D V Lyridis. A comparative study on ant colony optimization algorithm approaches for solving multi-objective path planning problems in case of unmanned surface vehicles. *Ocean Engineering*, 2022, 255: 111418.
[16] J R Sánchez-Ibáñez, C J Pérez-del-Pulgar, A García-Cerezo. Path planning for autonomous mobile robots: A review. *Sensors*, 2021, 21(23): 7898.
[17] W Lee, G H Choi, T Kim. Visibility graph-based path-planning algorithm with quadtree representation. *Applied Ocean Research*, 2021, 117: 102887.
[18] Q Li, F Xie, J Zhao, et al. FPS: Fast path planner algorithm based on sparse visibility graph and bidirectional breadth-first search. *Remote Sensing*, 2022, 14(15): 3720.
[19] W Chi, Z Ding, J Wang, et al. A generalized voronoi diagram-based efficient heuristic path planning method for RRTs in mobile robots. *IEEE Transactions on Industrial Electronics*, 2022, 69(5): 4926–4937.
[20] M Candeloro, A M Lekkas, A J Sørensen. A voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice*, 2017, 61: 41–54.
[21] B B K Ayawli, X Mei, M Shen, et al. Mobile robot path planning in dynamic environment using voronoi diagram and computation geometry technique. *IEEE Access*, 2019, 7: 86026–86040.
[22] T T Mac, C Copot, D T Tran, et al. A hierarchical global path planning approach for mobile robots based on multi-objective particle swarm optimization. *Applied Soft Computing*, 2017, 59: 68–76.
[23] X Lai, J Li, J Chambers. Enhanced center constraint weighted A* algorithm for path planning of petrochemical inspection robot. *Journal of Intelligent & Robotic Systems*, 2021, 102(4).
[24] K J C Fransen, J A W M Van Eekelen, A Pogromsky, et al. A dynamic path planning approach for dense, large, grid-based automated guided vehicle systems. *Computers & Operations Research*, 2020, 123: 105046.
[25] R Song, Y Liu, R Bucknall. Smoothed A* algorithm for practical unmanned surface vehicle path planning. *Applied Ocean Research*, 2019, 83: 9–20.
[26] Y Singh, S Sharma, R Sutton, et al. A constrained A* approach towards optimal path planning for an unmanned surface vehicle in a maritime environment containing dynamic obstacles and ocean currents. *Ocean Engineering*, 2018, 169: 187–201.
[27] H Zhang, M Li, L Yang. Safe path planning of mobile robot based on improved A* algorithm in complex terrains. *Algorithms*, 2018, 11(4): 44.
[28] G Tang, C Tang, C Claramunt, et al. Geometric A-Star algorithm: An improved A-Star algorithm for AGV path planning in a port environment. *IEEE Access*, 2021, 9: 59196–59210.
[29] F Islam, V Narayanan, M Likhachev. Dynamic multi-heuristic A*. *2015 IEEE International Conference on Robotics and Automation (ICRA)*, Seattle, Washington, USA, May 26-30, 2015: 2376–2382.
[30] K Mi, J Zheng, Y Wang, et al. A multi-heuristic A* algorithm based on stagnation detection for path planning of manipulators in cluttered environments. *IEEE Access*, 2019, 7: 135870–135881.
[31] J Zhang, Z Liu, Y Wang, et al. Research on effective path planning algorithm based on improved A* algorithm. *Journal of Physics: Conference Series*, Chengdu, China, November 7–9, 2022, 2188(1): 012014.
[32] F Islam, V Narayanan, M Likhachev. A*-Connect: Bounded suboptimal bidirectional heuristic search. *2016 IEEE International Conference on Robotics and Automation (ICRA)*, Stockholm, Sweden, May 16-21, 2016: 2752–2758.
[33] X Wu, L Xu, R Zhen, et al. Bi-Directional adaptive A* algorithm toward optimal path planning for large-scale UAV under multi-constraints. *IEEE Access*, 2020, 8: 85431–85440.
[34] D Harabor, A Grastien. Online graph pruning for pathfinding on grid maps. *Proceedings of the AAAI Conference on Artificial Intelligence*, San Francisco, California, USA, August 7-11, 2011, 25(1): 1114–1119.
[35] S Liu, M Watterson, K Mohta, et al. Planning dynamically feasible trajectories for quadrotors using safe flight corridors in 3-D complex environments. *IEEE Robotics and Automation Letters*, 2017, 2(3): 1688–1695.
[36] Z Li, Z Zhang, H Liu, et al. A new path planning method based on concave polygon convex decomposition and artificial bee colony algorithm. *International Journal of Advanced Robotic Systems*, 2020, 17(1): 172988141989478.
[37] Z Li, R Shi, Z Zhang. A new path planning method based on sparse A* algorithm with map segmentation. *Transactions of the Institute of Measurement and Control*, 2021, 44(4): 916–925.
[38] Y Gong, G Liu. Path planning method using convex corner to improve A* algorithm. *Computer Engineering and Applications*, 2022: 1–10. (in Chinese)
[39] R Yonetani, T Taniai, M Barekatain, et al. Path planning using neural A* search. *International Conference on Machine Learning(ICML 2021)*, Vienna, Austria, 2021: 12029–12039.
[40] T Marcucci, M Petersen, D von Wrangel, et al. Motion planning around obstacles with convex optimization. *Science Robotics*, 2023, 8(84): 1–11.
[41] N R Sturtevant. Benchmarks for grid-based pathfinding. *IEEE Transactions on Computational Intelligence and AI in Games*, 2012, 4(2): 144–148.
[42] L Zhang, Y Zhang, Y Li. Mobile robot path planning based on improved localized particle swarm optimization. *IEEE Sensors Journal*, 2021, 21(5): 6962–6972.
[43] R Yonetani, T Taniai, M Barekatain, et al. Path planning using neural A* search. *International Conference on Machine Learning*, Virtual, July 18–24, 2021: 12029–12039.

Yong Tao born in 1979, is an associate professor and a doctoral supervisor at *School of Mechanical Engineering & Automation, Beihang University*, *China*. He received his Ph.D. degree from *School of Mechanical Engineering and Automation, Beihang University, China*, in 2009. His research interests include intelligent robot control methods, robots for aviation, and robot integration applications.

Lian Duan born in 1996, is currently a master candidate at *School of Large Aircraft Engineering, Beihang University, China*. He received his Bachelor's degree from *China Agricultural University, China*, in 2021. His research interests include mobile robot autonomous positioning and path planning.

He Gao born in 1997, is currently a Ph.D. candidate at *Aero-Engine Research Institute, Beihang University, China*. He received his Bachelor's degree from *Northeast Forestry University, China*, in 2019. His research interests include mobile manipulation, motion planning, and trajectory optimization for autonomous mobile robots.

Yufan Zhang born in 2000, is a master candidate at *School of Large Aircraft Engineering, Beihang University, China*. He received his

Bachelor's degree from *Harbin Institute of Technology, China*, in 2022. His research interests include robot cluster control.

Yian Song born in 2001, is a master candidate at *School of Mechanical Engineering & Automation*, *Beihang University, China*. His research interests include mobile manipulators and robot cluster control.

Tianmiao Wang born in 1960, is a professor and a doctoral supervisor at *School of Mechanical Engineering & Automation, Beihang University, China*. He received his Ph.D. degree from *School of Computer Science, Northwestern Polytechnical University, China*, in 1990. His research interests include biomimetic robotics, medical robotics technology, and mobile robots.