AIMS Mathematics, 9(5): 12011–12042.
DOI: 10.3934/math.2024587
Received: 28 January 2024
Revised: 04 March 2024
Accepted: 12 March 2024
Published: 27 March 2024

http://www.aimspress.com/journal/Math

Research article

GAO-RRT*: A path planning algorithm for mobile robot with low path cost and fast convergence

Lijuan Zhu[1], **Peng Duan**[1,2,*], **Leilei Meng**[1,*] **and Xiaohui Yang**[2]

[1] School of Computer Science, Liaocheng University, Shandong Liaocheng, 252059, China
[2] Shandong Provincial Key Laboratory of Network Based Intelligent Computing, Shandong Jinan, 250022, China

* **Correspondence:** Email: duanpeng@lcu.edu.cn, mengleilei@lcu.edu.cn.

Abstract: Path planning is an essential research topic in the navigation of mobile robots. Currently, rapidly-exploring random tree star (RRT*) and its variants are known for their probabilistic completeness and asymptotic optimality, making them effective in finding solutions for many path planning problems. However, slow convergence rate of the RRT* limits its practical efficiency. To address this problem, this paper proposed an enhanced RRT* algorithm by refining the extension process of the exploring tree. This enhancement aims to guide the tree approaching to obstacles (GAO) while exploring toward the target point. First, GAO-RRT* employed a dual-weighted sample strategy instead of random sample to guide search direction of the exploring tree. Second, a variable step size extension strategy was adopted to increase the efficiency of node generation, balancing searching time and path safety in regions with different obstacles densities. Third, growth status of new nodes was monitored in real-time, and a reverse growth strategy was proposed to guide the exploring tree to escape local optima. In addition, parent node creation procedure for new nodes was used to produce a better initial path. Finally, the proposed GAO-RRT* was compared with three state of the art algorithms on 16 different instances of four representative environments. Compared to RRT*, Quick-RRT* (Q-RRT*), and Fast-RRT* (F-RRT*), the results showed that (1) the average path cost of initial solutions obtained by GAO-RRT* decreased by 38.32%, 29.69%, and 20.44%, respectively; and (2) the average convergence time of solution obtained by GAO-RRT* to suboptimal ($1.05*C_{best}$) was reduced by 71.22%, 69.69%, and 58.37%, respectively. Simulation results indicated that GAO-RRT* outperforms the compared algorithms in terms of path cost and convergence speed.

Keywords: mobile robots; path planning; GAO-RRT*; dual-weighted sample strategy; variable step size extension strategy; reverse growth strategy
Mathematics Subject Classification: 68T40, 68W20

1. Introduction

Mobile robots have attracted considerable attention and have broad applications in many fields [1]. These applications span industrial automation [2–4], service robots [5, 6], autonomous driving [7, 8], and medical robots [9]. In the research of related technologies for mobile robots, path planning stands out as a critical core one. Indeed, the primary goal of path planning is to find a collision-free path from initial state to target state within the configuration space. Simultaneously, it needs to satisfy multiple optimization constraints, ensuring that the path is both short and smooth [10].

Research on path planning is gaining increasing popularity. Currently, path planning algorithms mainly encompass artificial potential field (APF) algorithms [11], intelligent bionic algorithms [12–14], grid-based search algorithms [15, 16], and sampling-based algorithms [17]. The fundamental concept of the APF method is to construct a repulsive potential field around obstacles and an attractive potential field around target point. The resulting force, comprising both repulsion and attraction, guides robot's movement within the composite potential field. The paths obtained by this method are smooth and reliable. However, a notable challenge with APF is its susceptibility to falling into local minima. Currently, APF utilizes potential environmental fields to enhance search efficiency and is frequently integrated with other path planning algorithms [18]. Intelligent bionic algorithms, drawing inspiration from bionic principles in nature, have been employed to solve path planning problems. Commonly used algorithms include the genetic algorithm [12], ant colony algorithm [13], and particle swarm optimization algorithm [14], each offering distinct advantages and suitability for optimizing complex path planning problems. Nevertheless, to solve path planning problems in complex environments, the reseachers usually design many optimized operators, inevitablely increasing the complexity of algorithms. In grid-based search methods, A* [15] algorithm is the most famous and widely applied one. It uses graph theory to discrete state spaces to solve path planning problems. In A* algorithm, it is assumed that each state corresponds to a grid point, ensuring integrity and optimality of the resolution. However, the computation time and used memory space grow exponentially as the complexity or dimension of environment increases. Sampling-based search algorithms are well-suited for scenes with unknown environmental information, thanks to their minimal requirements on the overall map environment representation [19]. One well-known example is the rapidly-exploring random tree (RRT) algorithm [17], which is widely acclaimed for its efficient exploration of state space. It has been proved probabilistically complete, meaning that if a feasible path exists, the algorithm is guaranteed to find a solution along with the number of samples. However, the path searched by the RRT algorithm is usually tortuous or not the optimal one [20].

To overcome the limitations of the RRT algorithm, researchers have put forward various variants to solve path planning problems in many fields. Among the variants, a notable milestone is the RRT* algorithm proposed by Karaman and Frazzoli [21]. RRT* aims to improve efficiency and optimality by minimizing the total path cost through the ChooseParent and Rewire procedures. RRT* revises the parent vertex of nodes based on the cost function in each iteration, resulting in superior paths compared to the basic RRT algorithm [22]. Despite its advantages, RRT* still faces challenges in complex environments such as ineffective node extension and redundant sampling, leading to slow convergence speed. To accelerate convergence speed, Nasir investigated the RRT*-Smart [23], which incorporates an exploration-exploitation strategy to intelligently sample and optimize the path, ultimately improving the convergence speed and reducing path cost. However, the quality of the final path achieved by RRT*-

Smart is influenced by the quality of the initial solution [24]. In addition to the method, Informed-RRT* [25] introduces an informed sampling strategy by constructing an elliptical sampling region to provide a more focused and informed sampling space based on the starting point, the target point, and the current path length. By concentrating sampling efforts in regions likely to yield lower-cost paths, Informed-RRT* accelerates the search for an optimal solution. Nevertheless, it's worth noting that Informed-RRT* still relies on the RRT* to generate the initial solution. While this can enhance the search efficiency, it may also introduce some level of computational overhead due to the additional computations involved in constructing the informed sampling region. To address this problem, Qureshi and Howie [26] proposed the Potential RRT* (P-RRT*) by incorporating APF into the RRT*. P-RRT* addressed the challenge of reducing expansion time toward the target, and it is particularly useful in scenarios where the configuration space is complex. While APF is effective in guiding exploration, it's important to note that P-RRT* may encounter the challenge of local minima. Furthermore, Fan [27] proposed the improved Bi-P-RRT* algorithm, which introduces the goal-biased strategy and APF method to further improve the search efficiency. In addition to adopting a goal-directed approach, researchers also improve the planning efficiency of the RRT* algorithm by modifying the random expansion of the tree structure [28]. For example, in order to enhance exploration efficiency, Jeong proposed a novel improvement, the Q-RRT* [29], which adjusts the optimization module of RRT* based on triangle inequality. Unlike traditional RRT*, Q-RRT* expands the range for selecting parent nodes. This modification allows the algorithm to explore a broader region of the configuration space during the tree expansion process. However, it may face limitations in narrow or complex environments, potentially leading to suboptimal paths [30]. Similar to Q-RRT*, designed to improve the efficiency of exploration and reduce path cost, Liao proposed the F-RRT* [31]. F-RRT* introduces two specific procedures, namely FindReachest and CreateNode. The FindReachest procedure aims to identify a parent node that is close to obstacles for a new node, while the CreateNode procedure aims to reduce the path cost by creating parent node close to obstacles. Although F-RRT* presents advantages in terms of reducing path length, it may be associated with a trade-off in convergence speed, especially in the environments with local traps.

The algorithm proposed in this study is driven by the primary challenges observed in existing path planning methods. First, it recognizes that optimal paths typically traverse regions close to obstacles, which are often underutilized by current methods, leading to higher path costs. Second, many RRT*-based path planning algorithms lack of node growth state detection, heavily relying on the randomness of sampling for new node generation. Third, existing methods lack an occasional reverse exploring mechanism, which proves beneficial in navigating challenging scenarios like local traps. To address these problems, this paper proposed a novel algorithm, GAO-RRT*, which generates better initial solutions and achieves fast convergence. The main contributions of this study are summarized as follows.

(1) A dual-weighted sampling strategy is developed to guide the expansion of the exploring tree, allowing it to not only go toward the target point, but also approach obstacles to shorten path length. This strategy makes the RRT*-based planner more flexible and adaptable to complex environments.

(2) The current growth status of the exploring tree is monitored and then used to improve the strategy for generating new nodes, in which a reverse growth strategy is proposed.

(3) The extension of new nodes no longer depends on a predefined step size. The dynamic variable step size, combined with the reverse growth strategy, is closer to simulating the growth mode of trees

in the natural environment, jumping out of local traps, shortening the path length, and accelerating the convergence rate of the proposed algorithm.

The rest of this paper is organized as follows. Section 2 formalizes two path planning problems and introduces the state of the art RRT-based algorithms. Section 3 describes the proposed GAO-RRT* and its detailed components. Section 4 presents the simulation environments, parameter selection, results, and comparative analysis with the outstanding algorithms. Finally, conclusion is summarized in Section 5.

2. Background

In this section, two path planning problems are formalized and the state of the art RRT-based algorithms, such as RRT*, Q-RRT*, and F-RRT*, are introduced.

2.1. Path planning problems

Let X denote the configuration space, X_{obs} is the obstacle region, and $X_{free} = X/X_{obs}$ is the free space. (X, x_{start}, X_{goal}) describes a path planning problem, where $x_{start} \in X_{free}$ is the initial state and $X_{goal} \subset X_{free}$ is the target region. Indeed, X_{goal} represents a circular region centered on the target point x_{goal}. Let a continuous function with bounded variation $\sigma(\tau) : [0, 1] \to X$ denote a path. If $\forall\tau \in [0, 1]$ is satisfied for $\sigma(\tau) \in X_{free}$, the path is a collision-free one.

Problem 1. (Feasible path planning) For the path planning problem (X, x_{start}, X_{goal}), the objective is to find a collision-free path from an initial state x_{start} to the target region X_{goal}, represented as

$$\sigma(\tau) : [0, 1] \to X_{free}, \sigma(0) = x_{start}, \sigma(1) \in X_{goal}. \tag{2.1}$$

If no solution exists, report failure.

Problem 2. (Optimal path planning) Let $\sum$ be the set of all the feasible paths and $c(\cdot)$ be the cost function of a feasible path. The optimal path planning problem can be defined as σ^*, which aims to find a feasible path with the lowest value of the cost function, such that

$$\sigma^* = \arg\min_{\sigma\in\sum}\{c(\sigma) \mid \sigma(0) = x_{start}, \sigma(1) \in X_{goal}, \forall\tau \in [0, 1], \sigma(\tau) \in X_{free}\}. \tag{2.2}$$

*2.2. RRT**

RRT*, as an improved variant of the RRT, is widely used in the field of robotics and path planning. It provides a powerful algorithmic framework for finding optimal paths in complex and high-dimensional spaces. The pseudocode of RRT* is presented in Algorithm 1. The most distinction between RRT and RRT* lies in the additional steps taken by RRT*, ChooseParent, and Rewire procedures. Following the generation of a new node, RRT* incorporates the ChooseParent procedure to identify a superior parent node from the existing nodes for the new node. In the Rewire procedure, RRT* optimizes the tree structure by considering alternative connections, improving the cost of existing paths. Specifically, the primary goal of ChooseParent is to select an optimal parent for a newly generated node x_{new} within X_{near} and evaluate whether the chosen nodes have the potential to reduce path cost. If such an improvement is possible, the parent node of x_{new} is updated. On the contrary, the aim of Rewire is to assess and potentially update the parent-child relationships on the tree by checking whether the path cost from

x_{init} to the node of X_{near} via its original parent is greater than that via x_{new} as parents. A concise overview of some procedures employed by the RRT* algorithm is provided.

• *Sample*: Returns a random sample state x_{rand} from *map*. The method of calculating x_{rand} can be expressed as:

$$x_{rand} = \{x_i \,|x_i \in map\}. \tag{2.3}$$

• *Nearest*: Given V and x_{rand}, it returns the node $x_{nearest}$ that is closest to x_{rand} in V. The method of calculating $x_{nearest}$ can be expressed as:

$$x_{nearest} = \arg\min\{dist(x_i, x_{rand}) \,|x_i \in V\}. \tag{2.4}$$

• *Steer*: This function extends an incremental distance from $x_{nearest}$ to x_{rand} to generate a new state x_{new}.

• *CollisionFree*: This function checks whether the local path from x_{new} to $x_{nearest}$ is collision-free.

• *Near*: It returns a set of nodes in a hypersphere with a specific radius centered on x_{new}, where r_{near} is the radius. The method of calculating X_{near} can be expressed as:

$$X_{near} = \{x_i \,|dist(x_i, x_{new}) < r_{near}, x_i \in V\}. \tag{2.5}$$

Algorithm 1: RRT*

Input: $x_{start}, X_{goal}, n_{repeat}, map, r_{near}$
Output: $T = (V, E)$

$V \leftarrow \{x_{start}\}, E \leftarrow \emptyset$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow Sample(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollisionFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Near(V, x_{new}, r_{near})$;
    $x_{parent} \leftarrow ChooseParent(X_{near}, x_{nearest}, x_{new})$;
    $V \leftarrow V \cup \{x_{new}\}$;
    $E \leftarrow E \cup \{(x_{parent}, x_{new})\}$;
    $T \leftarrow Rewire(T, x_{new}, X_{near})$;
  end
end
return $T = (V, E)$;

*2.3. Q-RRT**

Compared with the RRT*, the Q-RRT* [29] optimizes and adjusts the ChooseParent and Rewire procedures. The pseudocode of Q-RRT* is displayed in Algorithm 2. In the ChooseParent procedures, the search range of the possible parent node of x_{new} includes not only X_{near}, but also the ancestor of X_{near}, where $d_{ancestor}$ denotes the depth of finding the parent node. According to the triangle inequality, a parent node that satisfies the requirement and reduces the path cost could be found. Q-RRT* also considers the depth problem in the Rewire-Q-RRT* procedure, including the ancestors of x_{new} in the candidate search range, which significantly reduces the path cost. Algorithm 3 gives the pseudocode of

the Rewire-Q-RRT* procedure. A concise overview of new procedures used in Q-RRT* is described as follows.

• *ancestor*: Given $T = (V, E)$, a node x_p, and a natural number $n \in N$, it returns the n-th parent of x_p.

• *Ancestry*: Given $T = (V, E)$, a node x_p, and a natural number $d_{ancestor}$, if the depth $d_{ancestor}$ is 0, it returns $\varnothing$; otherwise it returns $\bigcup_{i=1}^{d_{ancestor}} ancestor(T, x_p, i)$.

• *Cost*: Given a node x from V, it returns the full length of the path from x_{start} to x.

• *dist*: It returns the Euclidean distance between two nodes x_i and x_j.

• *Parent*: Given a node x from V, it returns the parent node of x.

Algorithm 2: Q-RRT*

Input: $x_{start}, X_{goal}, n_{repeat}, map, r_{near}, d_{ancestor}$
Output: $T = (V, E)$

$V \leftarrow \{x_{start}\}, E \leftarrow \varnothing$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow Sample(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollisionFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Near(V, x_{new}, r_{near})$;
    $X_{parent} \leftarrow Ancestry(T, X_{near}, d_{ancestor})$;
    $x_{parent} \leftarrow ChooseParent(X_{near} \cup X_{parent}, x_{nearest}, x_{new})$;
    $V \leftarrow V \cup \{x_{new}\}$;
    $E \leftarrow E \cup \{(x_{parent}, x_{new})\}$;
    $T \leftarrow Rewire\text{-}Q\text{-}RRT^*(T, x_{new}, X_{near}, d_{ancestor})$;
  end
end
return $T = (V, E)$;

Algorithm 3: Rewire-Q-RRT*

Input: $T, x_{new}, X_{near}, d_{ancestor}$
Output: $T = (V, E)$

for *each* $x_{near} \in X_{near}$ **do**
  for *each* $x_{from} \in \{x_{new}\} \cup Ancestry(T, x_{new}, d_{ancestor})$ **do**
    if $Cost(x_{from}) + dist(x_{from}, x_{near}) < Cost(x_{near})$ **then**
      if $CollisionFree(x_{from}, x_{near})$ **then**
        $x_{parent} \leftarrow Parent(x_{near})$;
        $E \leftarrow (E \backslash \{(x_{parent}, x_{near})\}) \cup \{(x_{from}, x_{near})\}$;
      end
    end
  end
end
return $T = (V, E)$;

2.4. F-RRT*

F-RRT*, proposed by Liao et al. [31], is designed to improve the initial solution and convergence rate in path planning. The pseudocode of F-RRT* is shown in Algorithm 4. Indeed, F-RRT* optimizes path costs by creating a parent node for the new node x_{new}, instead of selecting it among the existing vertices. The creation process are divided into two steps, the FindReachest and CreatNode procedures. Inspired by Q-RRT*, the FindReachest procedure aims to find the reachable vertex $x_{reachest}$ from the ancestors of $x_{nearest}$, rather than searching a parent node near x_{new}. This particular $x_{reachest}$, once determined, is considered a candidate parent node for x_{new}. It should be noted that connecting $x_{reachest}$ and x_{new} makes the cost of the path from the x_{start} to x_{new} lower than connecting $x_{nearest}$ and x_{new}. In order to further minimize the path cost, the CreatNode procedure is implemented to create a new parent node x_{create} in proximity to the obstacle for x_{new}, leveraging triangle inequality among x_{new}, $x_{reachest}$, and the parent node of $x_{reachest}$.

Algorithm 4: F-RRT*

Input: $x_{start}, X_{goal}, n_{repeat}, map, r_{near}, d_{dichotomy}$
Output: $T = (V, E)$

$V \leftarrow \{x_{start}\}$,$E \leftarrow \varnothing$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow Sample(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  $x_{new} \leftarrow Steer(x_{nearest}, x_{rand})$;
  if $CollisionFree(x_{nearest}, x_{new})$ **then**
    $X_{near} \leftarrow Near(V, x_{new}, r_{near})$;
    $x_{reachest} \leftarrow FindReachest(T, x_{nearest}, x_{new})$;
    $x_{create} \leftarrow CreateNode(T, x_{reachest}, x_{new}, d_{dichotomy})$;
    if $x_{create} \neq \varnothing$ **then**
      $V \leftarrow V \cup \{x_{create}, x_{new}\}$;
      $E \leftarrow E \cup \{(Parent(x_{reachest}), x_{create}), (x_{create}, x_{new})\}$;
    else
      $V \leftarrow V \cup \{x_{new}\}$;
      $E \leftarrow E \cup \{(x_{reachest}, x_{new})\}$;
    end
    $T \leftarrow Rewire(T, x_{new}, X_{near})$;
  end
end
return $T = (V, E)$;

3. GAO-RRT*

This section provides a detailed description of the proposed GAO-RRT*. The core concept of GAO-RRT* is based on two key considerations: The nodes that make up an optimal path are usually close to obstacles [32] and occasional reverse growth may be beneficial for the exploring tree to overcome challenging situations. GAO-RRT* primarily enhances the process of generating new nodes, which

includes three procedures: GenerateNewNode, StatisticRecentCollisions, and ReverseNewNode. Figure 1 presents the flowchart of the proposed GAO-RRT*. In the GenerateNewNode procedure, the proposed GAO-RRT* employs a dual-weighted sample strategy instead of random sample. The strategy guides the exploring tree to grow toward the target point while approaching obstacles as much as possible to shorten the path length. In the StatisticRecentCollisions procedure, the current growth status of the exploring tree is monitored, and subsequent growth manner is determined based on the latest statistics. For the ReverseNewNode procedure, when the exploring tree encounters challenges in its growth, it alternates between reverse and forward growth attempts. It should be noted that the reverse growth attempt endows the proposed GAO-RRT* with the ability to escape local optima, thereby more flexibly responding to the challenges encountered.

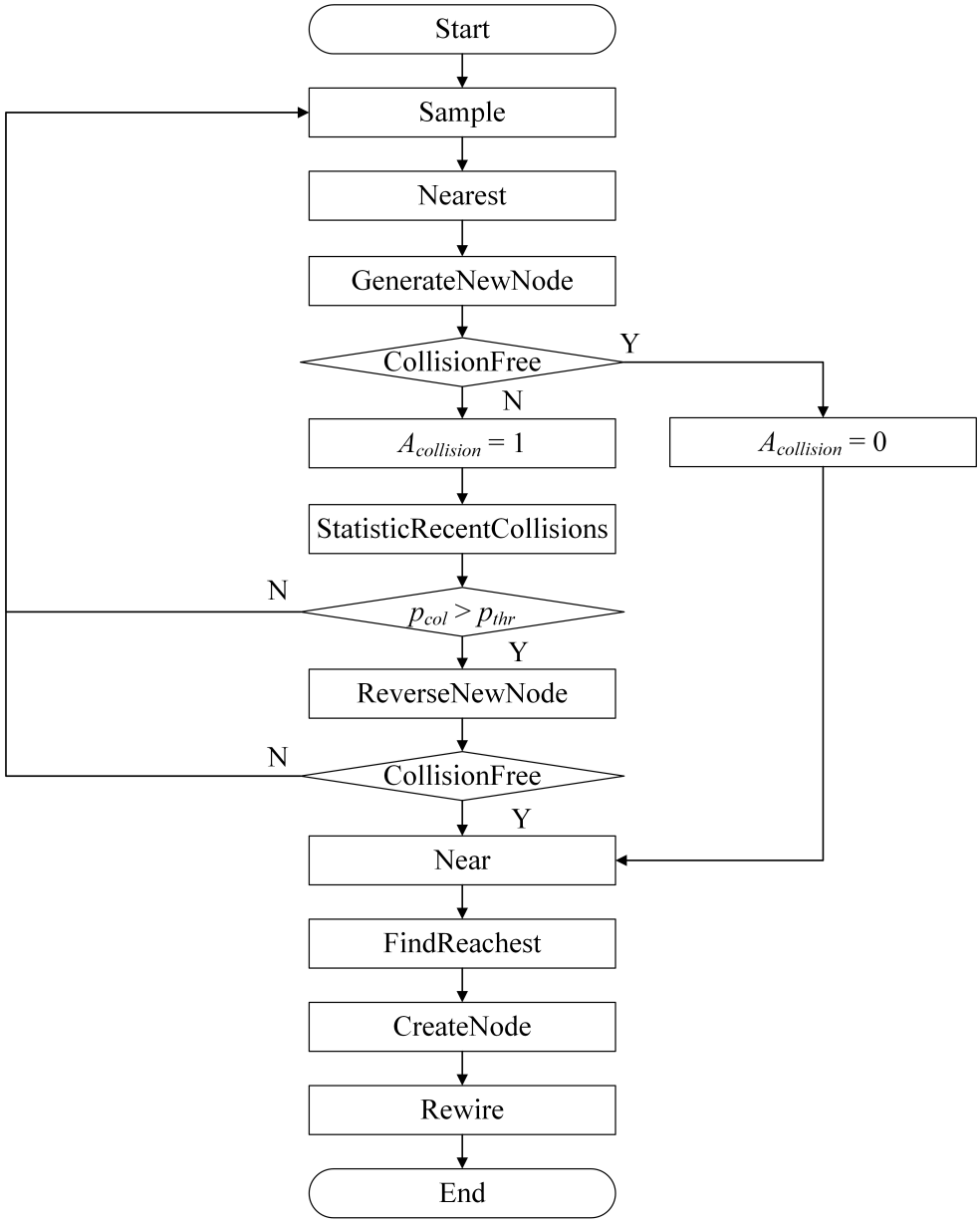


Figure 1. Flowchart of the proposed GAO-RRT*.

Algorithm 5 presents the pseudocode of the proposed GAO-RRT*. Specifically, the 5th line describes the GenerateNewNode procedure, the 10th line gives the StatisticRecentCollisions procedure, and lines 11 to 19 are employed to determine whether the ReverseNewNode procedure will be executed. It should be pointed out that in the proposed GAO-RRT*, we have incorporated the concept of creating nodes from the F-RRT*, as shown in lines 22 to 30 of Algorithm 5.

Algorithm 5: GAO-RRT*

Input: $x_{start}, x_{goal}, n_{repeat}, r_{near}, map, d_{dichotomy}, l_{step}, w_{obs}, p_{col}, p_{thr}, n_{iter}$
Output: $T = (V, E)$

1 $V \leftarrow \{x_{start}\}$,$E \leftarrow \varnothing$;
2 **for** $i = 1$ *to* n_{repeat} **do**
3 $x_{rand} \leftarrow Sample(i)$;
4 $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
5 $x_{new} \leftarrow GenerateNewNode(x_{rand}, x_{goal}, x_{nearest})$;
6 **if** $CollisionFree(x_{nearest}, x_{new})$ **then**
7 $A_{collision}(i) = 0$;
8 **else**
9 $A_{collision}(i) = 1$;
10 $p_{col} \leftarrow StatisticRecentCollisions(A_{collision}, i, n_{iter})$;
11 **if** $p_{col} > p_{thr}$ **then**
12 $x_{revnew} \leftarrow ReverseNewNode(T, x_{new}, x_{nearest}, d_{dichotomy})$;
13 **if** $\sim CollisionFree(x_{nearest}, x_{revnew})$ **then**
14 *continue*;
15 **end**
16 $x_{new} \leftarrow x_{revnew}$;
17 **else**
18 *continue*;
19 **end**
20 **end**
21 $X_{near} \leftarrow Near(V, x_{new}, r_{near})$;
22 $x_{reachest} \leftarrow FindReachest(T, x_{nearest}, x_{new})$;
23 $x_{create} \leftarrow CreateNode(T, x_{reachest}, x_{new}, d_{dichotomy})$;
24 **if** $x_{create} \neq \varnothing$ **then**
25 $V \leftarrow V \cup \{x_{create}, x_{new}\}$;
26 $E \leftarrow E \cup \{(Parent(x_{reachest}), x_{create}), (x_{create}, x_{new})\}$;
27 **else**
28 $V \leftarrow V \cup \{x_{new}\}$;
29 $E \leftarrow E \cup \{(x_{reachest}, x_{new})\}$;
30 **end**
31 $T \leftarrow Rewire(T, x_{new}, X_{near})$;
32 **end**
33 **return** $T = (V, E)$;

3.1. GenerateNewNode

The GenerateNewNode procedure aims to generate a new node that approaches obstacles while moving toward the target point. It employs a dual-weighted sample strategy, adjusting the weights of the guidance forces from x_{obs} and x_{goal} for $x_{nearest}$. This strategy enables the algorithm to strike a balance between the requirements of approaching obstacles and growing toward the target point, making it more flexible and adaptable to complex environments. To further illustrate the working process of this procedure, Figure 2 illustrated the new node generation using the dual-weighted sample strategy. As shown in Figure 2(a), the procedure initially identifies the obstacle point x_{obs} nearest to x_{rand} through the bwdist function, with x_{obs} highlighted in green. Subsequently, in Figure 2(b), utilizing the parallelogram rule, the combined attractive force of x_{obs} and x_{goal} guides the tree to extend a branch from $x_{nearest}$ to x_{new}. In detail, two weights, denoted by w_{obs} and (1-w_{obs}), respectively, are employed to balance the growth direction of the new node.

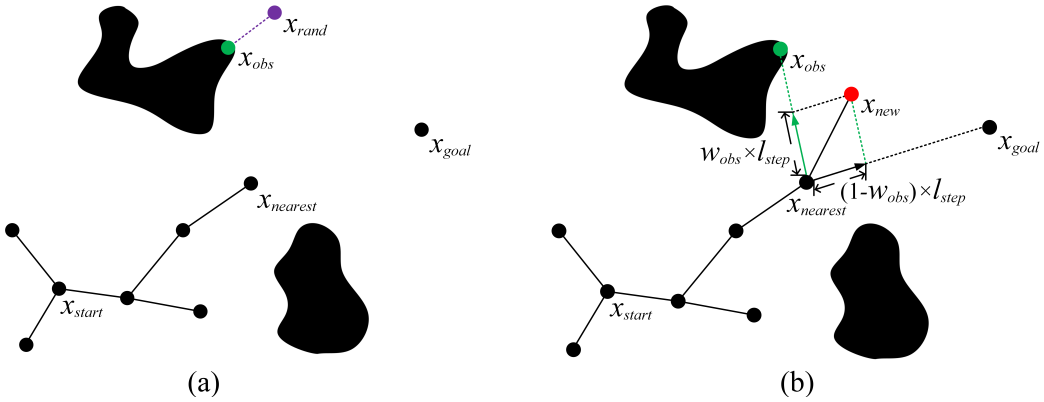


Figure 2. New node generation with the dual-weighted sample strategy. (a) Obtain the nearest obstacle point x_{obs} to x_{rand}; (b) new node extention process.

Algorithm 6 provides the pseudocode of the GenerateNewNode procedure. When the distance d between $x_{nearest}$ and x_{obs} is greater than a predetermined step size l_{step}, indicating a region with potentially fewer obstacles around the tree, the algorithm extends the new node x_{new} by l_{step}. Conversely, if the distance d is less than or equal to l_{step}, the algorithm extends new node x_{new} by d. It can be observed that, whether observed in line 5 or line 7 in Algorithm 6, the step size for each extension is a variable. This variable is influenced not only by the environment surrounding the exploring tree and the target point but also by the weights w_{obs} and (1-w_{obs}). In other words, a larger step size accelerates the search process in relatively open environments, while a smaller step size proves beneficial in avoiding collisions in more complex environments with obstacles. This adaptive step size enhances the search efficiency of the proposed algorithm.

Algorithm 6: GenerateNewNode

Input: $x_{rand}, x_{goal}, x_{nearest}$
Output: x_{new}

$L_{lable} \leftarrow bwdist(map)$;
$x_{obs} \leftarrow L_{lable}(x_{rand})$;
$d \leftarrow dist(x_{nearest}, x_{obs})$;
if $d > l_{step}$ **then**
    $x_{new} \leftarrow x_{nearest} + w_{obs} \times l_{step} \times (x_{obs} - x_{nearest})/dist(x_{obs}, x_{nearest}) + (1 - w_{obs}) \times l_{step} \times (x_{goal} - x_{nearest})/dist(x_{goal}, x_{nearest})$;
else
    $x_{new} \leftarrow x_{nearest} + w_{obs} \times d \times (x_{obs} - x_{nearest})/dist(x_{obs}, x_{nearest}) + (1 - w_{obs}) \times d \times (x_{goal} - x_{nearest})/dist(x_{goal}, x_{nearest})$;
end
return x_{new};

3.2. StatisticRecentCollisions

The StatisticRecentCollisions procedure is designed to monitor the current growth status of the exploring tree. Algorithm 7 outlines detailed steps for this procedure. As previously mentioned in the 7th and 9th lines of Algorithm 5, we used the binary values 0 and 1 to record the state of generating a new node. Indeed, $A_{collision}$ keeps track of all attempts to generate new nodes during the algorithm's execution. Consequently, it is easy to calculate the failure rate of generating new nodes in the most recent n_{iter} iterations, denoted as p_{col}. It should be noted that p_{col} refers to the frequency of collisions rather than probability. This indicator provides valuable insights into the extension status of the exploring tree. A higher p_{col} indicates suboptimal growth, prompting concerns about the potential occurrence of local minima. In such scenarios, it becomes imperative to promptly initiate a reverse growth process (outlined in lines 11 to 19 of Algorithm 5) to guide the tree out of its growth dilemma.

Algorithm 7: StatisticRecentCollisions

Input: $A_{collision}, i, n_{iter}$
Output: p_{col}

for $j = i : -1 : i - n_{iter}$ **do**
    if $j > 0$ **then**
        $sum = sum + A_{collision}(j)$;
    else
        $break$;
    end
end
$p_{col} \leftarrow sum/n_{iter}$;
return p_{col};

3.3. ReverseNewNode

The ReverseNewNode procedure aims to guide the exploring tree in escaping from the regions with challenging growth conditions. Algorithm 8 provides the pseudocode of this procedure. In detail, the exploring tree attempts to generate a new node x_{mid}, in the direction from $x_{nearest}$ to x_{new}. In the event of failure, the proposed procedure then utilizes mirror symmetry to determine the symmetrical point, x_{revnew}, of the midpoint x_{mid} with respect to $x_{nearest}$. Subsequently, the ReverseNewNode procedure repeats the above steps and attempts to generate a new node in a limited number of steps, which create numerous opportunities for the exploring tree to escape local optima. It should be noted that a parameter named $d_{dichotomy}$ and the distance between $x_{nearest}$ and x_{revnew} are used as the termination condition of the ReverseNewNode procedure. It is crucial to clarify that $d_{dichotomy}$ also serves as the termination condition of the dichotomy during the creation of x_{create} in Algorithm 4. This parameter sharing mechanism not only ensures that new available nodes can be obtained within a limited number of iterations, but also prevents situations where new generated nodes are too close to existing ones.

Algorithm 8: ReverseNewNode

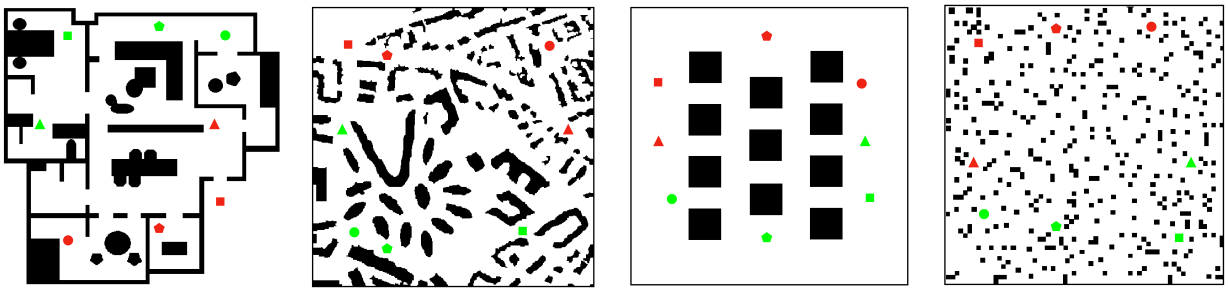
Input: $T, x_{new}, x_{nearest}, d_{dichotomy}$
Output: x_{revnew}

$x_{revnew} \leftarrow x_{start}$;
$d_{threshold} \leftarrow d_{dichotomy}$;
while $dist(x_{nearest}, x_{revnew}) > d_{threshold}$ && $d_{dichotomy} > 0$ **do**
  $d_{dichotomy} = d_{dichotomy} - 1$;
  $x_{mid} = (x_{nearest} + x_{new})/2$;
  if $CollisionFree(x_{nearest}, x_{mid})$ **then**
    $x_{revnew} \leftarrow x_{mid}$;
    break;
  else
    $x_{revnew} = 2x_{nearest} - x_{mid}$;
    if $CollisionFree(x_{nearest}, x_{revnew})$ **then**
      break;
    else
      $x_{new} \leftarrow x_{revnew}$;
    end
  end
end
return x_{revnew};

4. Simulation results and analysis

In this section, we conducted a comparative analysis to validate the performance of the proposed GAO-RRT* against the state of the art algorithms, such as RRT* [21], Q-RRT* [29], and F-RRT* [31]. The reasons for selecting the above three algorithms are as follows: (1) RRT* is a milestone of the sampling-based algorithm, which has been proven to effectively solve various path planning problems, and almost all new algorithms use it as a benchmark. (2) Unlike traditional RRT*, Q-RRT* expands the

range for selecting parent nodes to path length, which has demonstrated robust fast convergence ability. (3) F-RRT* optimizes the cost of paths by creating a parent node for the new node instead of selecting it among the existing vertices, which provides higher performance in the initial solution than almost all existing sampling-based algorithms. All compared algorithms are implemented using MATLAB R2019a, and the simulation experiments are executed on an Intel Core i7 processor with 32 GB RAM. Four representative environment maps, as shown in Figure 3, are employed to verify the effectiveness of GAO-RRT*. Indeed, Figure 3(a) represents an indoor residential environment based on real floorplan designs. Figure 3(b) illustrates an outdoor city park map sourced from Nathan Sturtevant's Mobile Artificial Intelligence Laboratory [33]. Figure 3(c) exhibits a simple regular environment, while Figure 3(d) displays a cluttered distribution of obstacles. The size of these environment maps is standardized at 500×500 pixels. To ensure the comprehensiveness of the tests, we strategically select the starting point and target point of the potential path in various directions of the map, including the main diagonal (MD), the secondary diagonal (SD), the horizontal axis (HA), and the vertical axis (VA). The start points are marked in red while the target points are marked in green, respectively. Each pair of coordinate points for directions MD, SD, HA, and VA is represented by rectangles, circles, triangles, and pentagons, respectively. Table 1 provides the coordinates of the starting and target points in four environment maps, corresponding to 16 different test instances.



(a) (b) (c) (d)

Figure 3. Environment maps for simulations. (a) Indoor environment; (b) Outdoor environment; (c) Regular environment; (d) Cluttered environment.

Table 1. Coordinates of starting and target points used for simulation experiments.

Environments	Path direction	Start point	Target point
Indoor environment	MD	(350,390)	(50,115)
	SD	(425,120)	(50,400)
	HA	(214,380)	(214,65)
	VA	(400,280)	(33,280)
Outdoor environment	MD	(60,60)	(365,420)
	SD	(68,423)	(405,70)
	HA	(215,455)	(215,45)
	VA	(75,120)	(430,120)
Regular environment	MD	(140,60)	(340,430)
	SD	(145,410)	(340,65)
	HA	(240,60)	(240,420)
	VA	(55,240)	(410,240)
Cluttered environment	MD	(60,60)	(420,420)
	SD	(30,375)	(375,65)
	HA	(280,45)	(280,440)
	VA	(35,200)	(395,200)

4.1. Choosing parameter

Appropriate parameter values have a significant impact on the algorithm's performance and practical applications. This impact extends beyond the convergence speed, robustness, and stability of the algorithm, encompassing the efficient utilization of resources. In this study, GAO-RRT* considers four key parameters, including the weight used to balance the growth direction of the new node (w_{obs}), the number of recent iterations when calculating the success rate of new node extension (n_{iter}), the predefined step size (l_{step}), and the search radius (r_{near}).

Following preliminary testing, Table 2 displays the four factor levels of the four parameter values. In this study, the design of experiment (DOE) Taguchi method [34] is used to test the influence of these four parameters. As the four parameters are set with four factor levels, an orthogonal array $L_{16}(4^4)$ is used. Each parameter combination is independently run 30 times for statistical robustness, resulting in a total of $16 \times 30 = 480$ results. To determine the most suitable parameter values, the algorithm employs the relative percentage increase (RPI) [35] of the objective value as a performance measure, as illustrated in formula (4.1).

$$RPI = \frac{f_i - f_{best}}{f_{best}} \times 100 \tag{4.1}$$

where f_i refers to the average time value obtained by the algorithm when running a special combination of parameters, and f_{best} represents the best time value among 480 running results. The formula makes it clear that the smaller the value of RPI, the closer the time performance of the parameter combination is to the optimal value among all parameter combinations. In other words, a smaller RPI signifies superior time performance for that set of parameters. Table 3 presents the corresponding RPI values for 16 orthogonal parameter combinations. Additionally, Table 4 displays the average RPI values for each

parameter level along with its associated significance. The Delta represents the difference between the highest and lowest RPI values for each parameter, which indicates how much each parameter influences the algorithm. The Rank refers to the strength ranking of the parameters' influence for the algorithm. Notably, w_{obs} emerges as the most influential parameter, followed by l_{step}, n_{iter}, and r_{near}. Figure 4 illustrates the trend of factor levels for the four parameters. According to the analysis of the results, the optimal parameter configuration is determined to be $w_{obs} = 0.7, n_{iter} = 20, l_{step} = 20, r_{near} = 50$.

Table 2. Value levels for the four parameters.

Parameters	Parameter level			
	1	2	3	4
w_{obs}	0.3	0.5	0.7	0.9
n_{iter}	5	10	20	40
l_{step}	10	20	30	40
r_{near}	35	50	65	80

Table 3. Orthogonal combination of parameters and its corresponding RPI response value.

Experimental number	Parameters				RPI response value
	w_{obs}	n_{iter}	l_{step}	r_{near}	
1	1	1	1	1	1.1624
2	1	2	2	2	0.7897
3	1	3	3	3	1.0115
4	1	4	4	4	1.0489
5	2	1	2	3	0.2191
6	2	2	1	4	0.4590
7	2	3	4	1	0.0838
8	2	4	3	2	0.1751
9	3	1	3	4	0.1148
10	3	2	4	3	0.1596
11	3	3	1	2	0.2998
12	3	4	2	1	0.0828
13	4	1	4	2	0.2918
14	4	2	3	1	0.2594
15	4	3	2	4	0.0776
16	4	4	1	3	0.2851

Table 4. The average RPI response values and rank of the parameters.

Level	w_{obs}	n_{iter}	l_{step}	r_{near}
1	1.003	0.447	0.552	0.397
2	0.234	0.417	0.292	0.389
3	0.164	0.368	0.390	0.419
4	0.228	0.398	0.396	0.425
Delta	**0.839**	0.079	0.260	0.028
Rank	**1**	3	2	4

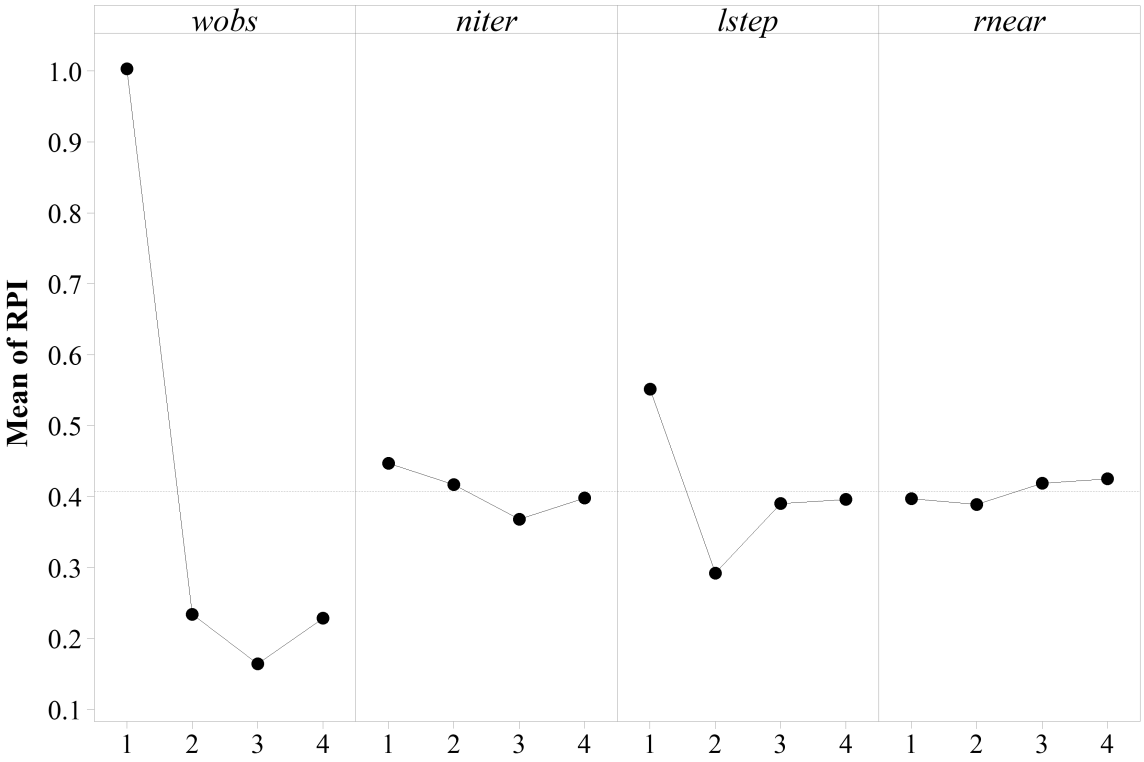


Figure 4. Factor level trend chart of four parameters.

4.2. Efficiency of the StatisticRecentCollisions procedure

To demonstrate the effectiveness of the StatisticRecentCollisions, we coded two types of GAO-RRT* algorithms. One is with all the components discussed in Section 3, namely, GAO-RRT*, and the other one is with all of the other components except the StatisticRecentCollisions (non-SRC, NSRC), namely, GAO-NSRC-RRT*. All the parameters of the two algorithms remain equivalent to ensure a fair comparison.

The comparison results of the two algorithms in the outdoor environment are listed in Table 5. The first column represents the path direction, the second column indicates the compared algorithms, and the following columns display the mean, standard deviation, maximum, and minimum values of three

metrics: Computation time, path cost, and path smoothness.

It can be concluded from Table 5 that: (1) In terms of computation time, GAO-RRT* achieved optimal values for all instances. (2) In terms of path cost, GAO-RRT* obtained the optimal average value for all instances. Although the minimum value is equivalent to GAO-NSRC-RRT*, the standard deviation and maximum value of GAO-RRT* have achieved the best, except for the MD direction. (3) In terms of path smoothness, GAO-RRT* obtained all optimal values in the SD, HA, and VA directions. Therefore, the performance demonstrates that the proposed GAO-RRT* algorithm is better than the GAO-NSRC-RRT* algorithm, and the effectiveness of StatisticRecentCollisions has been verified.

Table 5. Comparison results obtained by GAO-NSRC-RRT* and GAO-RRT* in the outdoor environment.

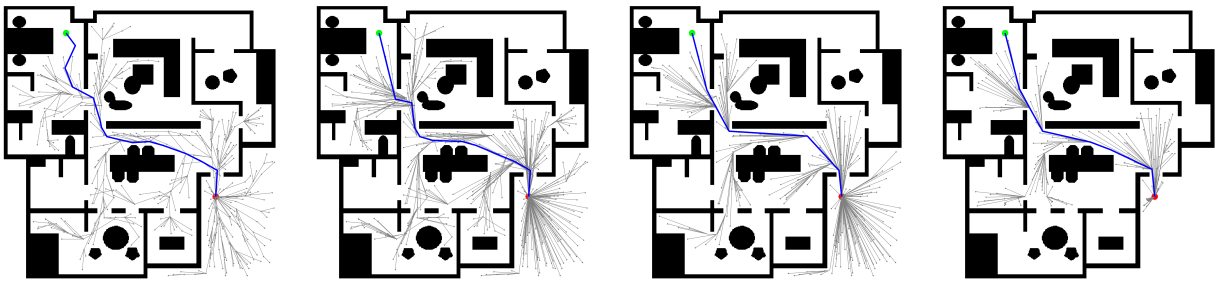
Path direction	Algorithm	Time				Path cost				Path smoothness			
		Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min
MD	GAO-NSRC-RRT*	1.81	1.71	9.68	0.46	617.60	**58.55**	**737.28**	509.65	**3.36**	**1.26**	**6.27**	1.58
	GAO-RRT*	**0.30**	**0.13**	**0.66**	**0.10**	**610.42**	59.92	743.54	**508.35**	3.68	1.45	7.85	**1.43**
SD	GAO-NSRC-RRT*	1.31	0.62	2.44	0.39	645.62	92.78	916.90	**514.60**	4.35	1.59	8.14	2.30
	GAO-RRT*	**0.26**	**0.11**	**0.49**	**0.11**	**609.28**	**60.23**	**703.47**	516.53	**4.09**	**1.38**	**7.38**	**1.92**
HA	GAO-NSRC-RRT*	1.20	0.89	5.25	0.31	554.65	54.53	701.77	**475.89**	3.40	1.89	7.16	0.61
	GAO-RRT*	**0.27**	**0.17**	**0.95**	**0.13**	**541.41**	**24.38**	**629.45**	485.12	**2.12**	**1.54**	**6.63**	**0.51**
VA	GAO-NSRC-RRT*	1.73	1.22	5.09	0.26	523.66	130.37	900.05	413.57	3.95	2.10	9.06	1.29
	GAO-RRT*	**0.35**	**0.15**	**0.98**	**0.14**	**465.28**	**82.12**	**848.05**	**409.48**	**2.88**	**1.71**	**6.93**	**1.28**

4.3. Comparison of initial path generation

To evaluate the performance of GAO-RRT*, each compared algorithm executed 30 independent runs on 16 different instances of four environments. These algorithms are executed under identical conditions with the same sampling sequence. To demonstrate the results, the starting point and the target point are denoted by red and green dots, respectively, in each environment map. The exploration paths, depicted by gray lines, illustrate the expansion of the exploring tree, while the obtained initial path is represented by blue lines. Metrics for evaluating performance include running time, path cost, and path smoothness. Violin charts are used to display detailed statistical data of the path cost simulation results.

4.3.1. Indoor environment

Figure 5 provides the initial paths obtained by the compared algorithms in MD direction of the indoor environment. From this figure, it can be observed that RRT* employs random sampling across the entire state space, resulting in vertices expanding in various directions. Q-RRT* optimizes the structure of the exploring tree to achieve more direct paths. Notably, both F-RRT* and GAO-RRT* exhibit lower path costs than Q-RRT*. In Figure 5(d), GAO-RRT* is shown to generate fewer nodes during path exploration compared to the other three algorithms, highlighting its advantage in terms of path cost.



(a) (b) (c) (d)

Figure 5. Initial paths obtained by the compared algorithms in MD direction of the indoor environment. (a) RRT*: C_{init} = 494.03; (b) Q-RRT*: C_{init} = 483.81; (c) F-RRT*: C_{init} = 471.98; (d) GAO-RRT*: C_{init} = 458.48.

Table 6 presents the simulation results for running time, path cost, and path smoothness metrics obtained by the compared algorithms. The first column indicates path directions, the second column represents the compared algorithms, and the subsequent columns display mean, standard deviation, maximum, and minimum values for the three metrics, respectively. The observations from this table are as follows: (1) GAO-RRT* achieved the lowest average values in running time in most cases, except for the instance in the HA direction. Despite RRT* attaining the minimum time values in all instances, GAO-RRT* demonstrated superiority by achieving the lowest standard deviations and maximum values. It is important to note that the simple steps of RRT* allows it to quickly find an initial solution when encountering a favorable sample distribution, demonstrating its advantages in the time. However, the quality of the solution in terms of path cost and path smoothness may not be guaranteed. (2) In terms of path cost and path smoothness, GAO-RRT* consistently obtained the lowest average values and standard deviations, maintaining an advantage in most of the maximum and minimum values. These findings indicate that GAO-RRT* demonstrates higher stability and robustness than the compared algorithms, suggesting its potential to generate a better initial path.

Table 6. Simulation results for running time, path cost, and path smoothness metrics obtained from the indoor environment.

Path direction	Algorithm	Time				Path cost				Path smoothness			
		Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min
MD	RRT*	0.46	0.57	2.40	**0.01**	495.25	20.02	566.15	464.60	5.48	1.40	9.66	3.70
	Q-RRT*	0.48	0.56	2.29	0.02	486.14	14.70	521.03	461.54	3.62	0.82	5.92	2.13
	F-RRT*	1.04	0.67	2.62	0.22	473.07	16.44	510.76	457.52	2.24	**0.33**	3.62	**1.68**
	GAO-RRT*	**0.33**	**0.28**	**1.26**	0.08	**463.92**	**10.79**	**507.23**	**457.41**	**2.23**	**0.33**	**3.57**	2.05
SD	RRT*	0.49	0.43	1.56	**0.07**	629.32	25.95	711.91	585.15	6.76	1.80	12.67	4.63
	Q-RRT*	0.56	0.47	1.84	0.10	620.36	20.22	665.09	583.33	4.23	2.02	9.16	1.82
	F-RRT*	0.43	0.27	1.48	0.10	602.56	21.31	646.70	**569.10**	3.31	1.49	6.16	**1.56**
	GAO-RRT*	**0.38**	**0.26**	**1.27**	0.11	**601.53**	**17.93**	**621.54**	571.41	**2.83**	**1.44**	**5.74**	**1.56**
HA	RRT*	**0.20**	0.38	2.01	**0.01**	393.66	59.99	559.63	330.79	4.51	1.29	6.38	1.52
	Q-RRT*	0.22	0.37	1.92	0.02	382.58	57.75	533.20	329.85	2.86	1.31	**5.01**	0.87
	F-RRT*	0.67	0.79	4.02	0.09	380.14	53.79	516.36	326.96	2.41	1.21	5.47	**0.50**
	GAO-RRT*	0.46	**0.34**	**1.60**	0.14	**362.04**	**43.34**	**512.73**	**324.73**	**2.15**	**1.15**	5.22	0.75
VA	RRT*	0.47	0.60	2.90	**0.06**	520.01	46.52	608.81	454.82	6.17	1.38	10.51	3.53
	Q-RRT*	0.56	0.67	3.35	0.08	515.71	45.99	603.43	451.92	4.11	0.74	6.10	2.76
	F-RRT*	0.45	0.44	2.03	0.11	482.25	43.88	575.65	**433.31**	3.30	**0.51**	4.56	2.52
	GAO-RRT*	**0.42**	**0.32**	**1.55**	0.10	**480.90**	**39.60**	**549.00**	433.84	**3.09**	**0.51**	**4.27**	**2.48**

Figure 6 illustrates the statistical results of path cost obtained by the compared algorithms after 30 simulations in the indoor environment. The meaning of each violin chart in horizontal direction is the frequency of a special value, i.e., frequency of path cost value. It can be seen that GAO-RRT* exhibits a more stable and compact data distribution in all instances, with mostly smaller outliers compared to the comparison algorithms.

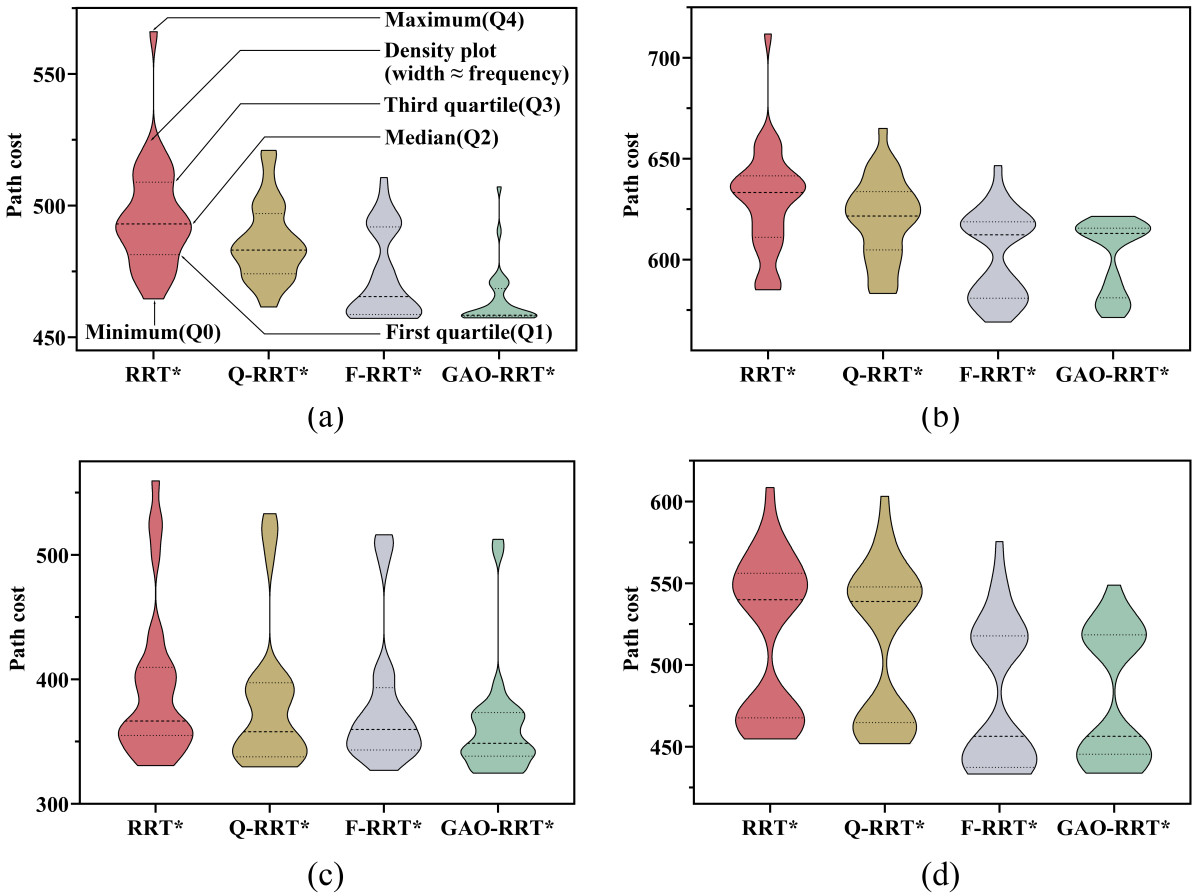
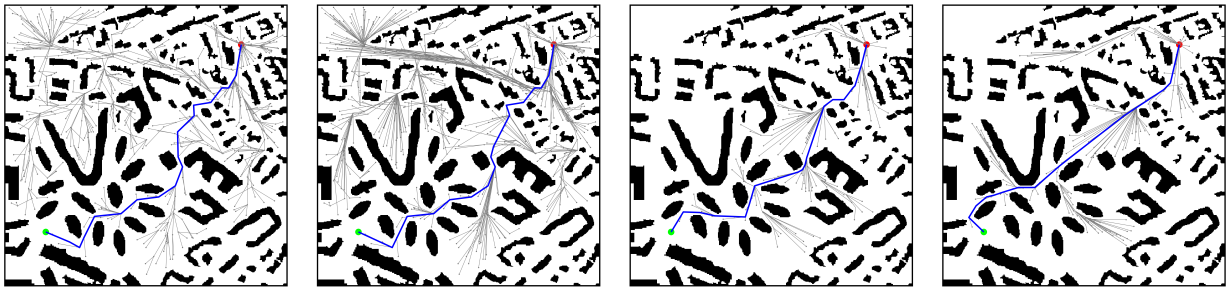


Figure 6. Visualization results of path cost obtained by the compared algorithms in the indoor environment. (a) MD direction; (b) SD direction; (c) HA direction; (d) VA direction.

4.3.2. Outdoor environment

Figure 7 depicts the initial paths obtained by the compared algorithms in SD direction of the outdoor environment. These figures distinctly demonstrate that, using the same sampling sequence, F-RRT* and GAO-RRT* are capable of finding paths that closely approach the optimal one. In contrast, owing to the extensive sampling space, RRT* and Q-RRT* encounter challenges in efficiently obtaining valid nodes, leading to a higher path cost compared to the other two algorithms. Additionally, it is noteworthy that GAO-RRT* excels in producing a shorter and smoother path.



(a) (b) (c) (d)

Figure 7. Initial paths obtained by the compared algorithms in SD direction of the outdoor environment. (a) RRT*: C_{init} = 612.04; (b) Q-RRT*: C_{init} = 607.85; (c) F-RRT*: C_{init} = 574.11; (d) GAO-RRT*: C_{init} = 556.69.

The statistical results for running time, path cost, and path smoothness metrics collected by the compared algorithms are presented in Table 7. Key observations from Table 7 include: (1) In MD, SD, and HA directions, GAO-RRT* achieved superior values in average values, standard deviations, and maximum values of all three metrics, respectively. (2) In VA direction, although GAO-RRT* was slightly passive in term of time metric, it exhibited significant advantages in path cost and path smoothness. These results indicate that in an environmental map with local traps, GAO-RRT* still generates better initial paths.

Table 7. Simulation results in running time, path cost, and path smoothness metrics obtained from the outdoor environment.

Path direction	Algorithm	Time				Path cost				Path smoothness			
		Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min
MD	RRT*	0.31	0.54	3.02	**0.03**	661.84	68.85	802.24	542.70	6.98	2.34	10.99	2.91
	Q-RRT*	0.32	0.56	3.12	**0.03**	650.78	67.39	801.04	535.06	4.63	2.06	8.92	2.02
	F-RRT*	0.61	0.55	2.40	0.13	637.36	67.60	778.27	531.32	3.88	1.94	9.95	1.48
	GAO-RRT*	**0.30**	**0.13**	**0.66**	0.10	**610.42**	**59.92**	**743.54**	**508.35**	**3.68**	**1.45**	**7.85**	**1.43**
SD	RRT*	0.31	0.46	2.23	**0.02**	640.87	76.22	777.78	526.39	8.22	2.02	12.90	4.5
	Q-RRT*	0.37	0.53	2.62	0.03	634.70	75.62	766.48	523.55	5.57	2.02	10.63	2.85
	F-RRT*	0.30	0.16	0.83	0.10	629.14	62.74	723.21	**514.00**	4.69	1.39	7.39	2.26
	GAO-RRT*	**0.26**	**0.11**	**0.49**	0.11	**609.28**	**60.23**	**703.47**	516.53	**4.09**	**1.38**	**7.38**	**1.92**
HA	RRT*	**0.27**	0.35	1.67	**0.02**	585.10	74.81	855.20	512.40	6.31	2.74	12.63	3.04
	Q-RRT*	0.33	0.40	1.93	0.03	577.15	74.35	850.39	505.64	4.09	2.84	10.45	0.54
	F-RRT*	0.39	0.39	1.87	0.09	572.18	72.10	738.63	485.19	3.60	2.43	9.41	0.52
	GAO-RRT*	**0.27**	**0.17**	**0.95**	0.13	**541.41**	**24.38**	**629.45**	**485.12**	**2.12**	**1.54**	**6.63**	**0.51**
VA	RRT*	**0.20**	0.16	**0.55**	**0.01**	509.82	124.29	880.77	415.88	5.29	2.29	11.50	3.02
	Q-RRT*	0.25	0.18	0.62	0.02	505.91	124.56	882.57	415.04	3.41	2.04	8.52	1.27
	F-RRT*	0.50	0.48	2.22	0.11	472.74	86.21	**823.19**	410.00	2.92	1.77	7.04	1.29
	GAO-RRT*	0.35	**0.15**	0.98	0.14	**465.28**	**82.12**	848.05	**409.48**	**2.88**	**1.71**	**6.93**	**1.28**

Figure 8 presents the statistical results of the path cost obtained by the compared algorithms after 30 simulations in the outdoor environment. Key observations from Figure 8 are as follows: (1) In Figure 8(a), most of path cost obtained by GAO-RRT* fall below the median line, suggesting that the generated paths have relatively lower costs for the majority. (2) In Figure 8(b), although the median value of path cost obtained by GAO-RRT* is not the best one, Q3 and Q1 values of path cost are better than those achieved by the compared algorithms. Additionally, the difference between the maximum and minimum path cost values is smaller. (3) In Figure 8(c),(d), GAO-RRT* produces fewer or smaller outliers than the other three algorithms. This indicates that GAO-RRT* is stable and less susceptible to generating outliers.

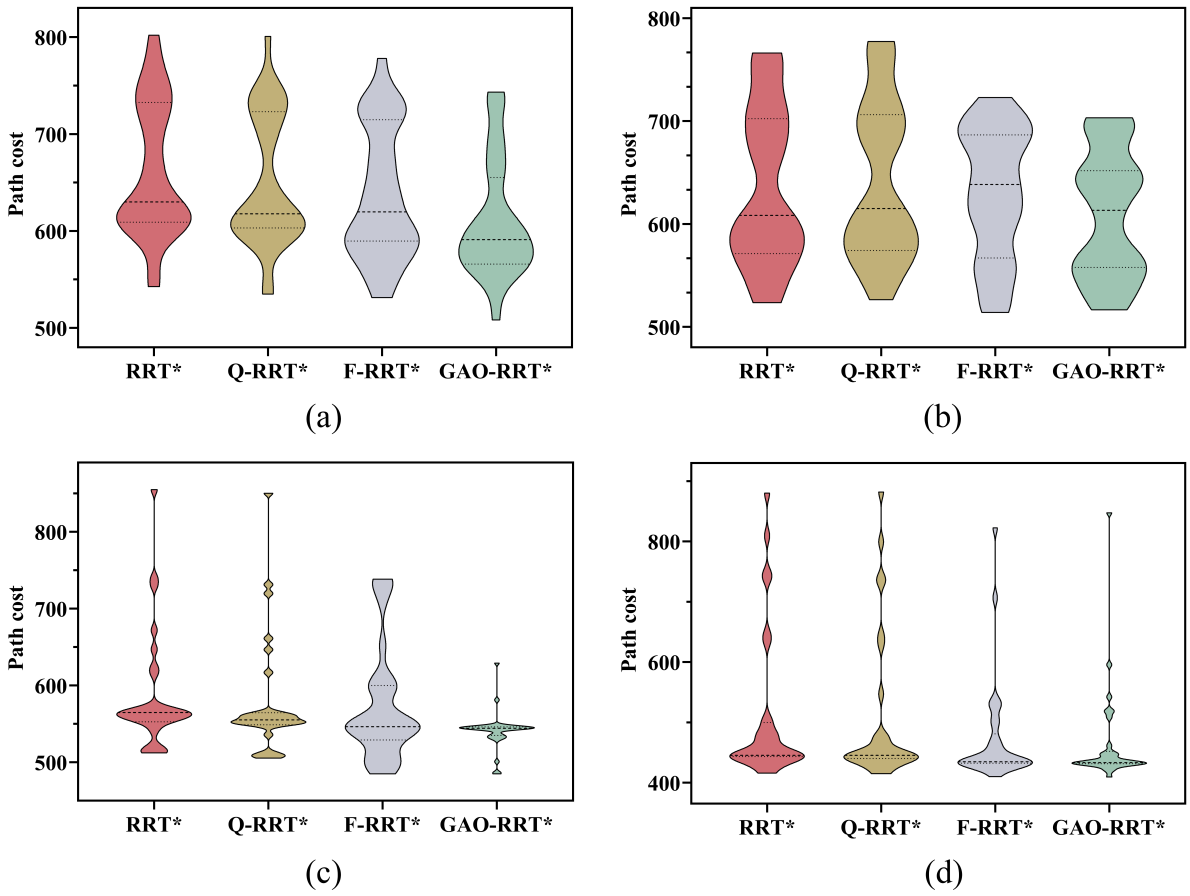
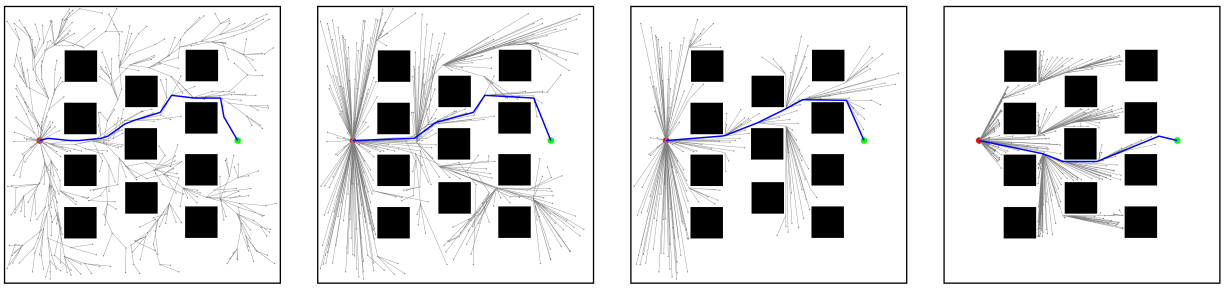


Figure 8. Visualization results of path cost obtained by the compared algorithms in the outdoor environment. (a) MD direction; (b) SD direction; (c) HA direction; (d) VA direction.

4.3.3. Regular environment

Figure 9 displays the initial paths obtained by the compared algorithms in HA direction of the regular environment. As seen in Figure 9, the exploring trees generated by RRT* and Q-RRT* have numerous branches, and the obtained paths are more redundant. In contrast, F-RRT* and GAO-RRT* obtain shorter paths with fewer nodes, thanks to the procedure of creating nodes. Benefitting from the dual-weighted sample strategy, GAO-RRT* produces a shorter initial path compared to F-RRT*.



(a) (b) (c) (d)

Figure 9. Initial paths obtained by the compared algorithms in HA direction of the regular environment. (a) RRT*: C_{init} = 441.99; (b) Q-RRT*: C_{init} = 440.60; (c) F-RRT*: C_{init} = 422.59; (d) GAO-RRT*: C_{init} = 374.99.

After 30 independent runs, the statistical results for running time, path cost, and path smoothness metrics are collected as shown in Table 8. From Table 8, it can be observed that: (1) GAO-RRT* obtained all optimal values in both path cost and path smoothness metrics from all four instances. (2) In terms of time, GAO-RRT* failed to achieve the best values, while RRT* actually obtained the best results. This phenomenon can be attributed to the relatively simple execution steps of RRT*, making it particularly suitable for environments with reasonable regular layouts. F-RRT* and GAO-RRT* require more time to create nodes and optimize path cost in comparison to RRT* or Q-RRT*. These findings confirm that GAO-RRT* has better stability in finding a better initial solution with path cost and path smoothness.

Table 8. Simulation results in running time, path cost, and path smoothness metrics obtained from the regular environment.

Path direction	Algorithm	Time				Path cost				Path smoothness			
		Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min
MD	RRT*	**0.06**	**0.07**	**0.38**	**0.01**	542.59	69.63	695.24	463.80	5.78	1.28	8.74	4.22
	Q-RRT*	0.08	0.08	0.44	0.02	528.06	68.66	684.17	451.06	2.49	1.07	5.67	0.18
	F-RRT*	0.77	0.42	1.85	0.25	517.10	65.78	661.15	439.71	1.75	0.98	3.95	**0.14**
	GAO-RRT*	0.52	0.45	1.84	0.14	**456.97**	**14.72**	**499.27**	**439.41**	**1.15**	**0.67**	**3.24**	**0.14**
SD	RRT*	**0.06**	**0.06**	**0.26**	**0.01**	502.85	62.43	651.02	438.23	5.76	1.07	9.01	3.98
	Q-RRT*	0.08	0.07	0.30	0.02	489.86	61.65	629.66	423.58	2.69	1.11	4.62	0.49
	F-RRT*	0.90	0.64	3.01	0.24	486.98	64.88	625.23	417.89	1.71	0.91	3.77	0.20
	GAO-RRT*	0.19	0.13	0.80	0.08	**432.21**	**9.45**	**447.68**	**417.09**	**1.31**	**0.60**	**2.41**	**0.18**
HA	RRT*	**0.06**	**0.07**	**0.36**	**0.01**	492.43	88.21	700.87	379.69	4.91	1.53	8.08	2.56
	Q-RRT*	0.08	**0.07**	0.38	**0.01**	473.32	78.12	682.50	377.26	2.13	1.01	4.68	0.63
	F-RRT*	0.64	0.36	1.54	0.22	458.62	81.81	654.53	370.34	1.54	0.70	3.61	0.57
	GAO-RRT*	0.40	0.36	1.63	0.10	**386.08**	**22.27**	**434.74**	**367.82**	**1.35**	**0.54**	**2.55**	**0.55**
VA	RRT*	**0.04**	**0.04**	**0.18**	**0.01**	449.87	71.29	691.04	384.27	4.01	1.49	8.88	1.88
	Q-RRT*	0.06	0.05	0.22	**0.01**	433.79	72.26	678.33	381.57	1.53	1.33	4.24	0.49
	F-RRT*	0.46	0.29	1.42	0.08	434.50	67.25	636.96	372.41	1.77	1.36	4.97	0.40
	GAO-RRT*	0.23	0.09	0.55	0.13	**396.56**	**17.99**	**452.60**	**372.18**	**0.75**	**0.74**	**2.80**	**0.36**

Figure 10 illustrates the statistical results of the path cost obtained by the four algorithms after 30 simulations in the regular environment. Observing Figure 10, it is evident that GAO-RRT* maintains a significant advantage in terms of path cost. Additionally, from the height of the violin plots, GAO-RRT* produces the lowest cost and exhibits the most stability.

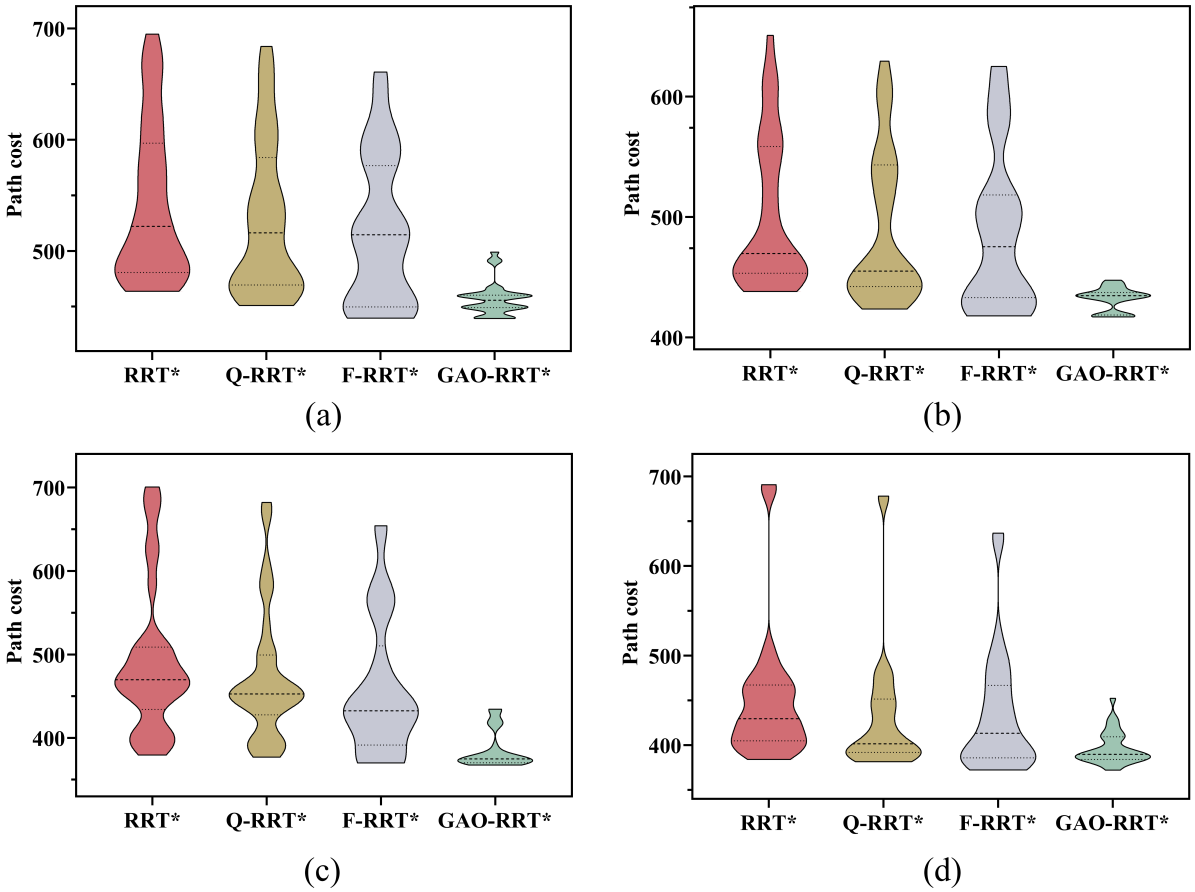
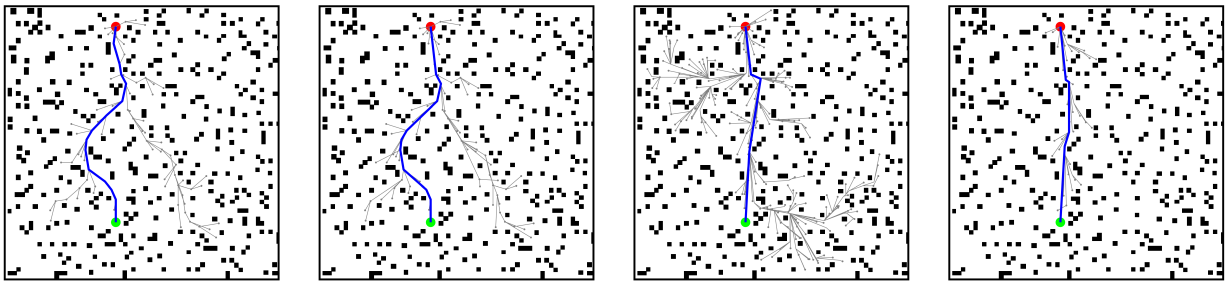


Figure 10. The path cost performance of the four algorithms in the regular environment. (a) MD direction; (b) SD direction; (c) HA direction; (d) VA direction.

4.3.4. Cluttered environment

Figure 11 presents the initial paths obtained by four compared algorithms in VA direction of the cluttered environment. The results reveal that the RRT* and Q-RRT* hardly produce effective nodes, resulting in a higher path cost compared to the other algorithms. Simultaneously, it can be observed that GAO-RRT* performs the best in yielding an initial solution.



(a) (b) (c) (d)

Figure 11. Initial paths obtained by the compared algorithms in VA direction of the cluttered environment. (a) RRT*: C_{init} = 413.32; (b) Q-RRT*: C_{init} = 411.35; (c) F-RRT*: C_{init} = 374.09; (d) GAO-RRT*: C_{init} = 365.15.

The statistical results for running time, path cost, and path smoothness metrics collected by the compared algorithms are given in Table 9. Analyzing Table 9, it can be concluded that: (1) In terms of time, GAO-RRT* achieved the lowest average, standard deviation, and maximum values in all four instances. (2) Regarding path cost and path smoothness, GAO-RRT* obtained almost all the best values, except for the minimum value of path cost in the SD direction. These findings indicate that GAO-RRT* is a good candidate for solving path planning problems in the cluttered environment.

Table 9. Simulation results in running time, path cost, and path smoothness metrics obtained from the cluttered environment.

Path direction	Algorithm	Time				Path cost				Path smoothness			
		Mean	Std	Max	Min	Mean	Std	Max	Min	Mean	Std	Max	Min
MD	RRT*	0.60	1.16	4.30	**0.02**	622.60	44.71	745.88	552.44	9.70	2.42	16.38	5.92
	Q-RRT*	0.73	1.35	4.98	0.04	610.91	40.14	698.44	544.43	7.71	2.19	14.01	4.17
	F-RRT*	0.70	1.14	6.23	0.10	597.58	43.78	686.67	528.64	5.83	1.79	10.07	2.74
	GAO-RRT*	**0.45**	**0.63**	**3.05**	3.05	**588.94**	**36.10**	**649.18**	**526.99**	**4.87**	**1.46**	**8.65**	**1.77**
SD	RRT*	0.31	0.41	1.47	**0.01**	601.03	72.78	806.04	506.44	9.10	2.63	15.41	4.53
	Q-RRT*	0.39	0.49	1.82	0.02	589.03	65.10	739.75	495.39	7.26	2.86	16.20	3.78
	F-RRT*	0.31	0.33	1.41	0.04	577.07	64.73	730.23	**483.41**	5.90	1.99	11.03	**2.15**
	GAO-RRT*	**0.30**	**0.22**	**1.05**	0.08	**575.99**	**48.47**	**668.74**	495.74	**5.64**	**1.82**	**10.87**	3.11
HA	RRT*	0.45	0.67	3.33	**0.03**	515.24	71.31	719.86	423.45	7.48	2.52	11.98	3.06
	Q-RRT*	0.55	0.76	3.67	0.04	508.07	71.45	723.18	723.18	5.81	2.66	14.01	1.64
	F-RRT*	0.54	0.93	5.07	0.05	501.11	53.11	624.68	424.38	5.10	1.80	8.05	1.73
	GAO-RRT*	**0.36**	**0.33**	**1.60**	0.06	**496.44**	**51.48**	**597.71**	**421.30**	**4.51**	**1.68**	**7.55**	**1.58**
VA	RRT*	0.40	0.97	4.32	**0.01**	455.04	75.09	693.25	368.23	6.67	2.95	17.17	2.33
	Q-RRT*	0.49	1.15	5.12	**0.01**	449.53	71.87	666.84	364.70	4.84	2.42	11.29	1.33
	F-RRT*	0.19	0.22	1.17	0.03	449.48	57.55	613.98	363.65	4.52	1.67	8.71	1.51
	GAO-RRT*	**0.18**	**0.21**	**1.11**	0.04	**438.16**	**51.00**	**570.84**	**362.82**	**4.18**	**1.40**	**7.53**	**1.01**

Figure 12 illustrates the statistical results of the path cost obtained by the four algorithms after 30

simulations in the cluttered environment. From Figure 12, the following observations can be made: (1) GAO-RRT* obtains a more stable and compact data distribution in all instances, as indicated by the height of the violin plots. (2) GAO-RRT* almost has no outliers, except for the results in VA direction. These results further emphasize the reliability and stability of GAO-RRT*.

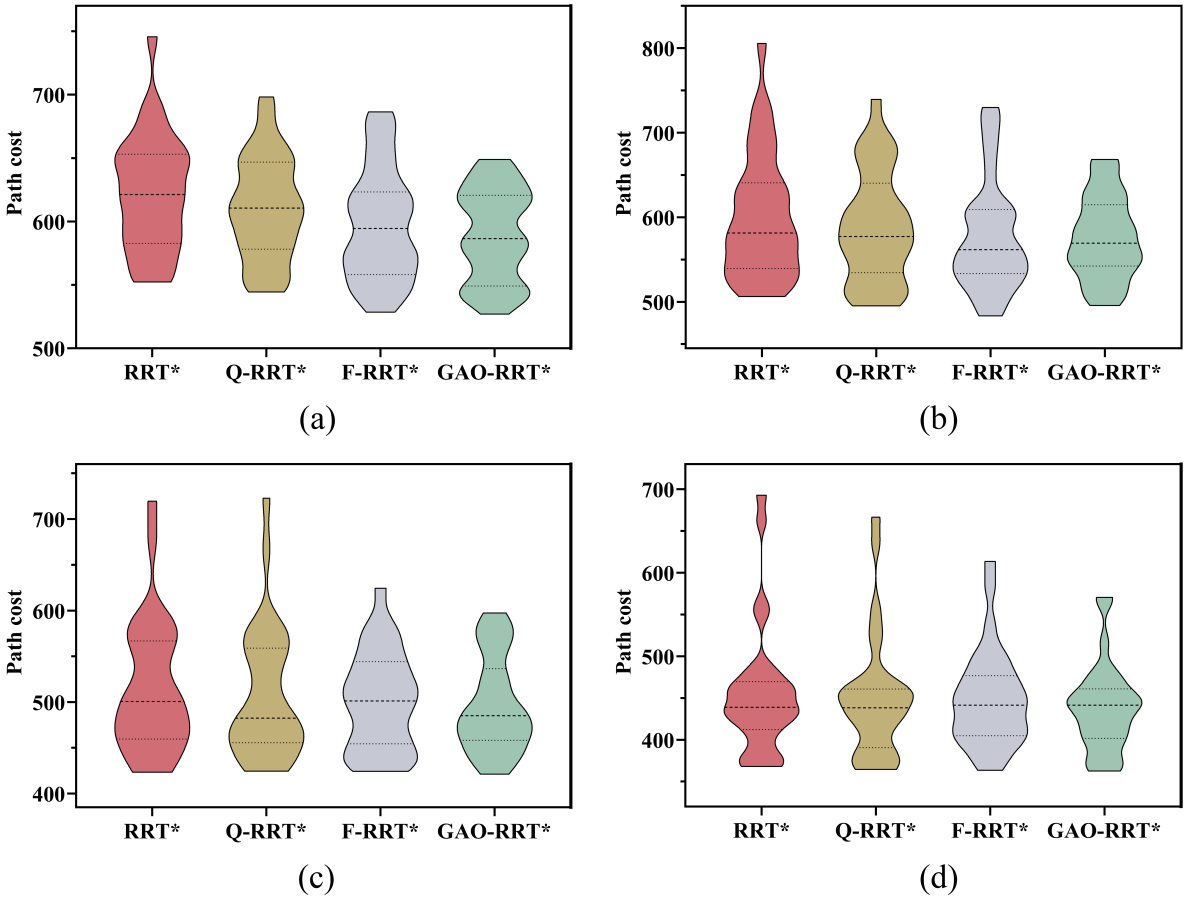


Figure 12. Visualization results of path cost obtained by the compared algorithms in the cluttered environment. (a) MD direction; (b) SD direction; (c) HA direction; (d) VA direction.

4.4. Comparing algorithms on the converge

To compare the convergence of four algorithms, the parameters in simulation and the sample sequences used in each trial are the same. Each algorithm in the simulation is performed for 60 seconds, and the path cost is recorded once per second. The optimal path obtained within 60 seconds of runtime is determined by the lowest path cost generated by the algorithms. Figure 13 provides path cost over computation time achieved by the compared algorithms on four selected instances, i.e., the MD direction of indoor environment, the SD direction of outdoor environment, the HA direction of regular environment, and the VA direction of cluttered environment. It can be observed that GAO-RRT* generates relatively better solutions in less time under the same conditions; hence, the time required for GAO-RRT* to converge to the suboptimal path is much shorter than that of the compared algorithms. From these convergence curves, we can conclude that the proposed algorithm, GAO-RRT*,

has remarkable convergence ability than the other algorithms.

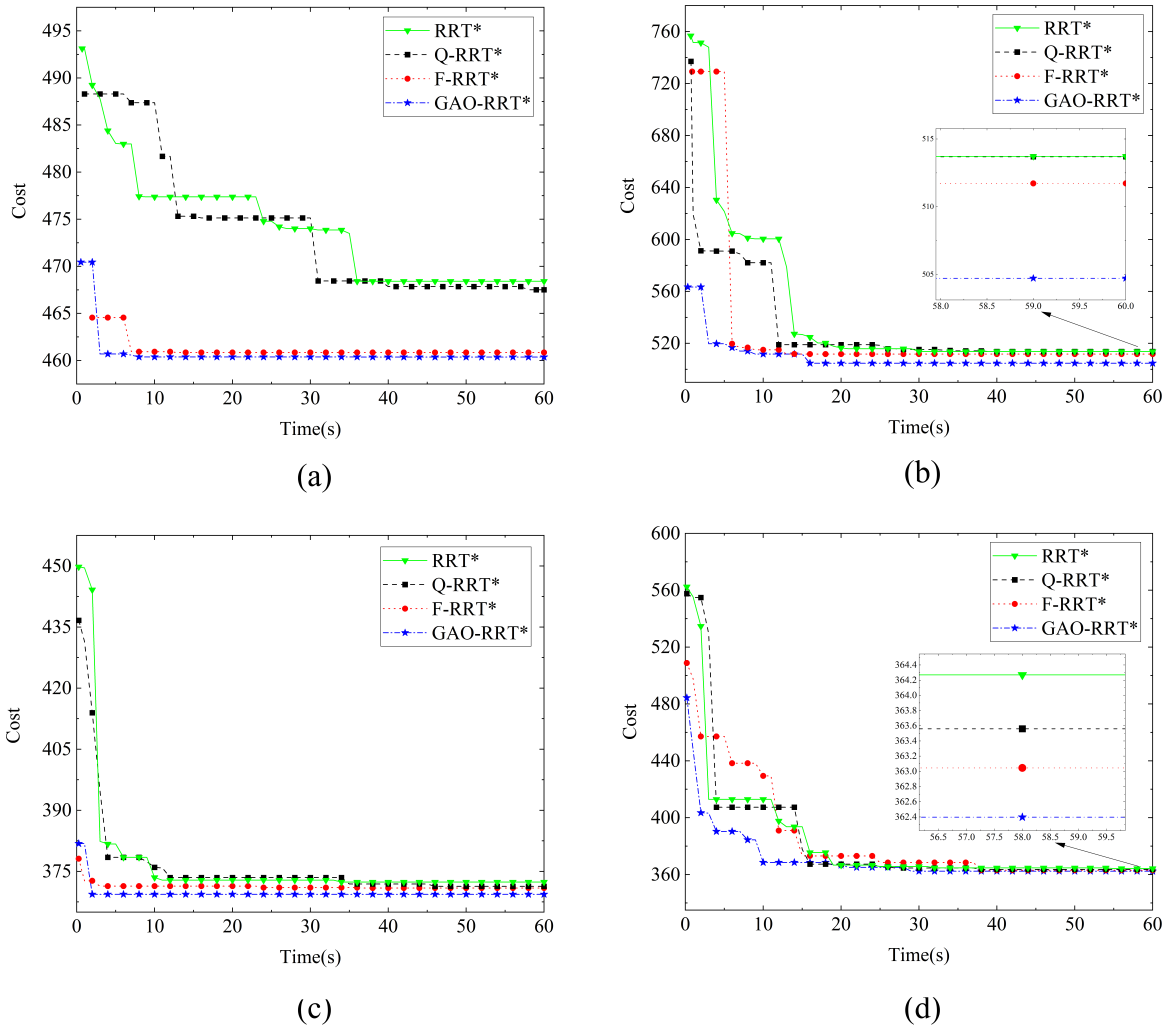


Figure 13. Path cost over computation time achieved by the compared algorithms in four environments. (a) Indoor environment (MD direction); (b) Outdoor environment (SD direction); (c) Regular environment (HA direction); (d) Cluttered environment (VA direction).

5. Conclusions

This paper proposes an enhanced version of the RRT* algorithm to address the challenge of slow convergence rate. The proposed GAO-RRT* employs a dual-weighted sample strategy to optimize the search direction of the exploring tree. By monitoring the growth status of nodes and combining with a reverse growth strategy, GAO-RRT* facilitates the tree's escape from regions with local traps. Four representative maps with 16 test instances are employed to evaluate the capabilities of GAO-RRT*. Experimental results show that the average deviation of running time for GAO-RRT* in finding the initial path is 0.53, 0.47, and 0.44 times that of RRT*, Q-RRT*, and F-RRT*, respectively. In addition,

compared with these three algorithms, the results of GAO-RRT* have reduced path cost by 38.32%, 29.69%, and 20.44% respectively, and reduced path smoothness by 54.57%, 30.07%, and 13.82%, respectively. The excellent performance of GAO-RRT* is verified in terms of computation time, path cost, and path smoothness metrics. In short, it can be concluded that the proposed GAO-RRT* is a good candidate when solving path planning problems of mobile robot.

The limitations of this study are as follows: 1) This study focuses on solving the shortest path and accelerating convergence speed, while ignoring other path indicators such as path safety and dynamic constraints of the mobile robot. 2) The proposed algorithm mainly concentrates on solving path planning problems in static environments, without considering the interference of dynamic targets.

In future work, the following directions will be explored: 1) We will integrate path safety and kinematic constraints into path planning and apply the proposed algorithm to the navigation of a home service robot. 2) The proposed path planning algorithm will consider environmental constraints, especially the challenges posed by dynamic obstacles. 3) The problem of multi-robot collaborative path planning will also be a considered research topic in the coming years.

Use of AI tools declaration

The authors declare they have not used Artificial Intelligence (AI) tools in the creation of this article.

Acknowledgments

This work was supported by the Opening Fund of Shandong Provincial Key Laboratory of Network based Intelligent Computing, the National Natural Science Foundation of China (52205529, 61803192), the Natural Science Foundation of Shandong Province (ZR2021QE195), the Youth Innovation Team Program of Shandong Higher Education Institution (2023KJ206), and the Guangyue Youth Scholar Innovation Talent Program support received from Liaocheng University (LCUGYTD2022-03).

Conflict of interest

The authors declare that there is no conflict of interest.

References

1. L. Liu, X. Wang, X. Yang, H. Liu, J. Li, P. Wang, Path planning techniques for mobile robots: Review and prospect, *Expert Syst. Appl.,* **227** (2023), 1–30. https://doi.org/10.1016/j.eswa.2023.120254
2. Z. Zhou, L. Li, A. Fürsterling, H. J. Durocher, J. Mouridsen, X. Zhang, Learning-based object detection and localization for a mobile robot manipulator in SME production, *Robot. Comput. Integr. Manuf.,* **73** (2022), 1–12. https://doi.org/10.1016/j.rcim.2021.102229
3. M. Pantscharowitsch, B. Kromoser, Automated subtractive timber manufacturing-joinery machines versus industrial robots, *J. Manuf. Sci. Eng.,* **145** (2023), 1–15. https://doi.org/10.1115/1.4056924

4. K. Hu, Z. Chen, H. Kang, Y. Tang, 3D vision technologies for a self-developed structural external crack damage recognition robot, *Autom. Constr.,* **159** (2024), 1–19. https://doi.org/10.1016/j.autcon.2023.105262
5. J. Holthöwer, J. van Doorn, Robots do not judge: Service robots can alleviate embarrassment in service encounters, *J. Acad. Mark. Sci.,* **51** (2023), 767–784. https://doi.org/10.1007/s11747-022-00862-x
6. M. Zhang, G. Tian, Y. Zhang, P. Duan, Service skill improvement for home robots: Autonomous generation of action sequence based on reinforcement learning, *Knowl. Based Syst.,* **212** (2021), 1–15. https://doi.org/10.1016/j.knosys.2020.106605
7. J. Cheng, L. Zhang, Q. Chen, X. Hu, J. Cai, A review of visual SLAM methods for autonomous driving vehicles, *Eng. Appl. Artif. Intell.,* **114** (2022), 1–17. https://doi.org/10.1016/j.engappai.2022.104992
8. H. Kim, Y. Choi, Development of autonomous driving patrol robot for improving underground mine safety, *Appl. Sci.,* **13** (2023), 1–19. http://doi.org/10.3390/app13063717
9. Z. Jiang, S. E. Salcudean, N. Navab, Robotic ultrasound imaging: State-of-the-art and future perspectives, *Med. Image Anal.,* **89** (2023), 1–26. https://doi.org/10.1016/j.media.2023.102878
10. B. Song, Z. Wang, L. Zou, An improved PSO algorithm for smooth path planning of mobile robots using continuous high-degree Bezier curve, *Appl. Soft Comput.,* **100** (2021), 1–11. http://doi.org/10.1016/j.asoc.2020.106960
11. O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. Robot. Res.,* **5** (1986), 90–98. http://dx.doi.org/10.1177/027836498600500106
12. W. Deng, X. Zhang, Y. Zhou, Y. Liu, X. Zhou, H. Chen, et al., An enhanced fast non-dominated solution sorting genetic algorithm for multi-objective problems, *Inf. Sci.,* **585** (2022), 441–453. https://doi.org/10.1016/j.ins.2021.11.052
13. L. Wu, X. Huang, J. Cui, C. Liu, W. Xiao, Modified adaptive ant colony optimization algorithm and its application for solving path planning of mobile robot, *Expert Syst. Appl.,* **215** (2023), 1–37. https://doi.org/10.1016/j.eswa.2022.119410
14. C. Pozna, R. E. Precup, E. Horváth, E. M. Petriu, Hybrid particle filter-particle swarm optimization algorithm and application to fuzzy controlled servo systems, *Inst. Electr. Electron. Eng. Trans. Fuzzy Syst.,* **30** (2022), 4286–4297. http://dx.doi.org/10.1109/TFUZZ.2022.3146986
15. Z. Zhang, J. Jiang, J. Wu, X. Zhu, Efficient and optimal penetration path planning for stealth unmanned aerial vehicle using minimal radar cross-section tactics and modified A-Star algorithm, *Int. Soc. Autom. Trans.,* **134** (2023), 42–57. https://doi.org/10.1016/j.isatra.2022.07.032
16. Y. Zhang, G. Tian, X. Shao, S. Liu, M. Zhang, P. Duan, Building metric-topological map to efficient object search for mobile robot, *Inst. Electr. Electron. Eng. Trans. Ind. Electron.,* **69** (2022), 7076–7087. https://doi.org/10.1109/TIE.2021.3095812
17. S. M. LaValle, J. J. Kuffner Jr, Randomized kinodynamic planning, *Int. J. Robot. Res.,* **20** (2001), 378–400. http://dx.doi.org/10.1177/02783640122067453

18. J. Rao, C. Xiang, J. Xi, J. Chen, J. Lei, W. Giernacki, et al., Path planning for dual UAVs cooperative suspension transport based on artificial potential field-A* algorithm, *Knowl. Based Syst.,* **277** (2023), 1–20. https://doi.org/10.1016/j.knosys.2023.110797

19. Y. Tang, S. Qi, L. Zhu, X. Zhuo, Y. Zhang, F. Meng, Obstacle avoidance motion in mobile robotics, *J. Syst. Simul.,* **36** (2024), 1–26. http://dx.doi.org/10.16182/j.issn1004731x.joss.23-1297E

20. Y. Guo, X. Liu, Q. Jia, X. Liu, W. Zhang, HPO-RRT*: A sampling-based algorithm for UAV real-time path planning in a dynamic environment, *Complex Intell. Syst.,* **9** (2023), 7133–7153. https://doi.org/10.1007/s40747-023-01115-2

21. S. Karaman, E. Frazzoli, Sampling-based algorithms for optimal motion planning, *Int. J. Robot. Res.,* **30** (2011), 846–894. http://dx.doi.org/10.1177/0278364911406761

22. W. Zhang, L. Shan, L. Chang, Y. Dai, SVF-RRT*: A stream-based VF-RRT* for USVs path planning considering ocean currents, *Inst. Electr. Electron. Eng. Robot. Autom. Lett.,* **8** (2023), 2413–2420. http://dx.doi.org/10.1109/LRA.2023.3245409

23. J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, et al., RRT*-SMART: A rapid convergence implementation of RRT, *Int. J. Adv. Robot. Syst.,* **10** (2013), 1–12. http://dx.doi.org/10.5772/56718

24. J. Fan, X. Chen, Y. Wang, X. Chen, UAV trajectory planning in cluttered environments based on PF-RRT* algorithm with goal-biased strategy, *Eng. Appl. Artif. Intell.,* **114** (2022), 1–12. http://dx.doi.org/10.1016/j.engappai.2022.105182

25. J. D. Gammell, S. S. Srinivasa, T. D. Barfoot, *Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic*, In: 2014 IEEE/RSJ international conference on intelligent robots and systems, Chicago, IL, USA, IEEE, 2014, 2997–3004.

26. A. H. Qureshi, Y. Ayaz, Potential functions based sampling heuristic for optimal path planning, *Auton. Robot.,* **40** (2016), 1079–1093. http://dx.doi.org/10.1007/s10514-015-9518-0

27. J. Fan, X. Chen, X. Liang, UAV trajectory planning based on bi-directional APF-RRT* algorithm with goal-biased, *Expert Syst. Appl.,* **213** (2023), 1–12. http://dx.doi.org/10.1016/j.eswa.2022.119137

28. L. Ye, F. Wu, X. Zou, J. Li, Path planning for mobile robots in unstructured orchard environments: An improved kinematically constrained bi-directional RRT approach, *Comput. Electron. Agric.,* **215** (2023), 1–17. http://doi.org/10.1016/j.compag.2023.108453

29. I. B. Jeong, S. J. Lee, J. H. Kim, Quick-RRT*: Triangular inequality-based implementation of RRT* with improved initial solution and convergence rate, *Expert Syst. Appl.,* **123** (2019), 82–90. http://dx.doi.org/10.1016/j.eswa.2019.01.032

30. J. Ding, Y. Zhou, X. Huang, K. Song, S. Lu, An improved RRT* algorithm for robot path planning based on path expansion heuristic sampling, *J. Comput. Sci.,* **67** (2023), 1–12. http://dx.doi.org/10.1016/j.jocs.2022.101937

31. B. Liao, F. Wan, Y. Hua, R. Ma, S. Zhu, X. Qing, F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate, *Expert Syst. Appl.,* **184** (2021), 1–12. http://dx.doi.org/10.1016/j.eswa.2021.115457

32. J. Ou, S. H. Hong, P. Ziehl, Y. Wang, GPU-based global path planning using genetic algorithm with near corner initialization, *J. Intell. Robot. Syst.,* **104** (2022), 1–17. http://doi.org/10.1007/s10846-022-01576-6

33. A. Hidalgo-Paniagua, M. A. Vega-Rodríguez, J. Ferruz, N. Pavón, Mosfla-mrpp: Multi-objective shuffled frog-leaping algorithm applied to mobile robot path planning, *Eng. Appl. Artif. Intell.,* **44** (2015), 123–136. http://dx.doi.org/10.1016/j.engappai.2015.05.011

34. Z. Yu, P. Duan, L. Meng, Y. Han, F. Ye, Multi-objective path planning for mobile robot with an improved artificial bee colony algorithm, *Math. Biosci. Eng.,* **20** (2023), 2501–2529. http://dx.doi.org/10.3934/mbe.2023117

35. X. He, Q. K. Pan, L. Gao, J. S. Neufeld, J. N. D. Gupta, Historical information based iterated greedy algorithm for distributed flowshop group scheduling problem with sequence-dependent setup times, *Omega,* **123** (2024), 1–19. https://doi.org/10.1016/j.omega.2023.102997

AIMS Press