Expert Systems With Applications 184 (2021) 115457

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

F-RRT*: An improved path planning algorithm with improved initial solution and convergence rate☆

Bin Liao [a], Fangyi Wan [a,*], Yi Hua [a], Ruirui Ma [a], Shenrui Zhu [a], Xinlin Qing [a]

[a] School of Aeronautics, Northwestern Polytechnical University, Xi'an 710072, China

ARTICLE INFO

Keywords:
Path planning
Sampling-based algorithms
Rapidly-exploring random tree (RRT)
Optimal path planning

ABSTRACT

During the last decades, sampling-based algorithms have been used to solve the problem of motion planning. RRT*, as an optimal variant of RRT, provides asymptotic optimality. However, the slow convergence rate and costly initial solution make it inefficient. To overcome these limitations, this paper proposes a modified RRT* algorithm, F-RRT*, which generates a better initial solution and converges faster than RRT*. F-RRT* optimizes the cost of paths by creating a parent node for the random point, instead of selecting it among the existing vertices. The creation process can be divided into two steps, the *FindReachest*, and *CreatNode* procedures, which require few calculations, and the triangle inequality is used repeatedly throughout the process, thus, resulting in paths with higher performance than those of RRT*. Since the algorithm proposed in this paper is a tree extending algorithm, its performance can be further enhanced when combined with other sampling strategies. The advantages of the proposed algorithm in the initial solution and fast convergence rate are demonstrated by comparing with RRT*, RRT*-Smart, and Q-RRT* through numerical simulations in this paper.

1. Introduction

In recent years, robots are gaining more and more attention since they can increase productivity and provide various conveniences. Motion planning is one of the essential technologies of mobile robots. It is to find a feasible trajectory that connects the starting point with the goal point while avoiding collision with the obstacles.

The research on path planning will become more popular as the number of robots increases. Currently, path planning algorithms mainly include geometric algorithms (Alexopoulos & Griffin, 1992; Maekawa, Noda, Tamura, Ozaki, & Machida, 2010), artificial potential field methods (Khatib, 1986; Hwang et al., 1992), grid-based algorithms (Kanehara, Kagami, Kuffner, Thompson, & Mizoguhi, 2007; Koenig, Likhachev, & Furcy, 2004), and sampling-based algorithms (LaValle & Kuffner, 2001; Sánchez & Latombe, 2002). However, each algorithm has some insurmountable defects and irreplaceable advantages in actual application (González, Pérez, Milanés, & Nashashibi, 2015). The sampling-based algorithm is one of the most popular path planning algorithms because it is a general solving method and not constrained by the complexity of the controlled object. RRT (LaValle & Kuffner, 2001; LaValle, 1998) is one of the typical representatives of Sampling-based algorithms, relying on random sampling and collision detection modules to obtain a set of discrete points in space. After connecting these discrete points, a collision-free path can be formed. The biggest advantage of RRT is to quickly generate a feasible path, however, the cost of the found solution was not considered in RRT so the generated path is often tortuous and optimality cannot be guaranteed. RRT* (Karaman & Frazzoli, 2011) was proposed as an optimal variant of RRT by introducing the *ChooseParent* and *Rewire* procedures when a new node is added to the tree, the optimization modules ensuring finding the optimal solution when the number of samples approaches infinity.

Although RRT* cannot find the optimal solution in a limited time, it is still a milestone in the development of RRT as a new research basis for subsequent algorithms. To reduce the convergence time, Informed-RRT* (Gammell, Srinivasa, & Barfoot, 2014) and RRT*-smart (Islam, Nasir, Malik, Ayaz, & Hasan, 2012) accelerate the convergence rate by introducing a sampling strategy based on the initial solution of traditional RRT*. The former (Gammell et al., 2014) uses the starting and target points to establish an elliptical sampling area, which is continuously reduced with optimization, to quickly obtain the optimal solution. The

☆ This work was supported in part by the National Natural Science Foundation of China (No. 11702216).

* Corresponding author.
E-mail addresses: liaobin@mail.nwpu.edu.cn (B. Liao), fwan@nwpu.edu.cn (F. Wan), yihua0826@163.com (Y. Hua), maruirui@mail.nwpu.edu.cn (R. Ma), zsravarhm@163.com (S. Zhu), xinlinqing@xmu.edu.cn (X. Qing).

https://doi.org/10.1016/j.eswa.2021.115457
Received 9 September 2020; Received in revised form 15 June 2021; Accepted 19 June 2021
Available online 30 June 2021
0957-4174/© 2021 Elsevier Ltd. All rights reserved.

latter (Islam et al., 2012) not only introduces an intelligent sampling strategy into RRT* but also optimizes the path by cutting unnecessary nodes. However, the final path of RRT*-Smart greatly depends on the initial path of RRT* due to that the initial path nodes are used during the intelligent sampling procedure. In addition to improving the convergence rate based on the initial solution of RRT*, the sampling strategy could also be used for the initial extension of the tree. P-RRT* (Qureshi & Ayaz, 2016) and PQ-RRT* (Li, Wei, Gao, Wang, & Fan, 2020) use artificial potential fields (Khatib, 1986) to obtain the new nodes that are closer to expectations, thereby reducing the time to expand to the target area. The Neural RRT* (Wang, Chi, Li, Wang, & Meng, 2020) algorithm uses quantities of successful path planning cases to train a CNN model, which can predict the probability distribution of the optimal path on the map and guide the sampling process. These nodes required for the optimal path will help reduce the time required for convergence.

Abandoning those points that are useless for optimization could reduce the number of collision detections and thus shorten the convergence time. A graph pruning algorithm called Branch-and-Bound is proposed in the paper (Karaman, Walter, Perez, Frazzoli, & Teller, 2011). The algorithm introduces a heuristic function (Hart, Nilsson, & Raphael, 1968; Likhachev, Ferguson, Gordon, Stentz, & Thrun, 2008) to compare the cost of nodes. If the point cannot be used to reduce the cost of a feasible path, then they are deleted from the tree. To reduce the time cost of nodes deletion, any random point that cannot contribute to the current tree, will not be added to the tree (Akgun & Stilman, 2011; Ferguson & Stentz, 2006).

In addition to adopting better sampling strategies and reducing tree nodes, methods to shorten convergence time also include reducing path nodes. As long as the cost calculation method of the path satisfies the triangle inequality, unnecessary nodes in the path can be skipped like RRT*-smart (Islam et al., 2012). This triangular inequality is also used in Q-RRT* (Jeong, Lee, & Kim, 2015; Jeong, Lee, & Kim, 2019) to enlarge the set of possible parent vertices by considering not only a set of vertices contained in a hypersphere as in RRT* (Karaman & Frazzoli, 2011), to reduce path costs and accelerate convergence.

Based on the observation of the convergence results, we found that some nodes of the optimal path may be closer to obstacles than other feasible paths. However, those nodes closer to the obstacle, which is essential to the optimal path, will hardly appear until the number of sampling points approaches infinity. Therefore, we believe that a random tree could be optimized by creating nodes closer to obstacles. This paper proposes a Fast-RRT* (F-RRT*) algorithm to solve the problem of generating nodes near obstacles for the optimal path. Note that these nodes are created by the connection of tree vertexes instead of the shape of obstacles or intelligent algorithms. Although the process of creating is performed during the process of adding each random node to the tree, it is not always successful. Once the node close to the obstacle is successfully created, the random point will be added to the tree at a lower cost.

The remainder of the paper is organized as follows. In Section 2, we discuss RRT*, RRT*-Smart, Q-RRT*, and Informed-RRT*. F-RRT* is presented in Section 3. Section 4 describes the simulation environments, results, and comparative analysis with RRT*, RRT*-Smart, and Q-RRT*. Section 5 concludes the paper and highlights future research avenues.

2. Background

This section formalizes the two motion planning problems and has an in-depth understanding of the optimization modules of RRT* (Karaman & Frazzoli, 2011), Q-RRT* (Jeong et al., 2015), and RRT*-Smart (Islam

Algorithm 1: RRT*

Input: x_{start},X_{goal},Map, n_{repeat}, R_{near}
Output: $G = (V, E)$
$V \leftarrow \{x_{start}\}$,$E \leftarrow \phi$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  if $CollisionFree(x_{rand}, x_{nearest})$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
    $x_{parent} \leftarrow ChooseParent(X_{near}, x_{rand}, x_{nearest})$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup \{(x_{parent}, x_{rand})\}$;
    if $InitialPathFound$ **then**
      Return $G = (V, E)$;
    end
    $G \leftarrow Rewire(G, x_{rand}, X_{near})$;
  end
end
Return $G = (V, E)$;

et al., 2012), which inspired the proposal of the F-RRT* algorithm. In addition, the Informed sampling strategy (Gammell et al., 2014) was introduced to accelerate convergence.

2.1. Problem definition

Let X define the configuration space in which X_{obs} is the obstacle region, $X_{free} = X/X_{obs}$ is the obstacle-free region. (X, x_{start}, X_{goal}) defines a path planning problem, where $x_{start} \in X_{free}$ is the initial state and $X_{goal} \subset X_{free}$ is the goal area. Let a continuous function $\sigma : [0,1] \to X$ of bounded variation be a path. A path σ is collision-free if $\forall \tau \in [0,1], \sigma(\tau) \in X_{free}$.

Definition 1. Feasible Path Planning For the path planning problem (X, x_{start}, X_{goal}), any solution corresponding to a feasible path σ such that σ is collision-free, $\sigma(0) = x_{start}$, and $\sigma(1) \in X_{goal}$. If no solution exists, report failure.

Definition 2. Optimal path planning Given the path planning problem, find a feasible path $\sigma*$ such that $c(\sigma*) = \min\{c(\sigma) : \sigma \text{ is feasible}\}$, where $c(\sigma)$ is the full length of the path σ. If no solution exists, report failure.

2.2. RRT*

Before analyzing the RRT* algorithm, a brief description of RRT will be provided. RRT explores the space through a tree $G = (V, E)$ form x_{start}, where V is a set of vertexes and E is the set of connection relationships between the vertexes. When the tree extends to the goal region X_{goal}, a feasible path will be generated. The extension of the tree is composed of multiple iterations. In each iteration, a sample x_{rand} was chosen randomly from X_{free}, then find the closest vertex $x_{nearest}$ to the sample x_{rand} from V. If the local path σ_{local} from $x_{nearest}$ to x_{rand} is collision-free, the x_{rand} and σ_{local} will be added to the tree. The iterations will continue until a feasible path is found or the stop criteria are met. Next, RRT* will be introduced by the pseudo-code presented in Algorithm 1. The x_{start}, X_{goal} and *Map* are required parameters for each path planning algorithm as a problem description. They are used as global variables by

Algorithm 2: ChooseParent

Input: $X_{near}, x_{rand}, x_{nearest}$

Output: x_{parent}

$x_{parent} \leftarrow x_{nearest}$;
$c_{min} \leftarrow Cost(x_{nearest}) + Distance(x_{rand}, x_{nearest})$;
for *each* $x_{near} \in X_{near}$ **do**
    $c \leftarrow Cost(x_{near}) + Distance(x_{rand}, x_{near})$;
    if $c < c_{min}$ && $CollisionFree(x_{rand}, x_{near})$ **then**
        $x_{parent} \leftarrow x_{near}$;
        $c_{min} \leftarrow c$;
    end
end
Return x_{parent};

Algorithm 3: Rewire

Input: G, x_{rand}, X_{near}

Output: $G = (V, E)$

for *each* $x_{near} \in X_{near}$ **do**
    if $Cost(x_{near}) > Cost(x_{rand}) + Distance(x_{rand}, x_{near})$ **then**
        if $CollisionFree(x_{rand}, x_{near})$ **then**
            $E \leftarrow (E \setminus \{(Parent(x_{near}), x_{near})\}) \cup \{(x_{rand}, x_{near})\}$;
        end
    end
end
Return $G = (V, E)$;

Algorithm 4: Q-RRT*

Input: x_{start},X_{goal},Map, n_{repeat}, R_{near}, $n_{ancestor}$
Output: $G = (V, E)$
$V \leftarrow \{x_{start}\}$,$E \leftarrow \phi$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  if $CollisionFree(x_{rand}, x_{nearest})$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
    $X_{parent} \leftarrow Ancestory(G, X_{near}, n_{ancestor})$;
    $x_{parent \leftarrow ChooseParent(X_{near} \cup X_{parent}, x_{rand}, x_{nearest})}$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup \{(x_{parent}, x_{rand})\}$;
    if $InitialPathFound$ **then**
      Return $G = (V, E)$;
    end
    $G \leftarrow Rewire - Q - RRT*(G, x_{rand}, X_{near}, n_{ancestor})$;
  end
end
Return $G = (V, E)$;

any sub-procedure in the algorithm. Besides, the parameters in RRT* also include n_{repeat} and R_{near}, which are used for the algorithm termination condition and hypersphere radius respectively.

The main difference between RRT* and RRT are the optimization modules, *ChooseParent*, and *Rewire*. The purpose of *ChooseParent*, described in Algorithm 2 and Fig. 1(a), is to make the cost of the path from the x_{start} to x_{rand} lower by searching the X_{near}(a set of vertices in a hypersphere of a specific radius R_{near} centered at x_{rand}). The searching

Algorithm 5: Rewire-Q-RRT*

Input: $G, x_{rand}, X_{near}, n_{ancestor}$
Output: $G = (V, E)$
for *each* $x_{near} \in X_{near}$ **do**
  for *each* $x_{form} \in \{x_{rand}\} \cup Ancestory(G, x_{rand}, n_{ancestor})$ **do**
    if $Cost(x_{near}) > Cost(x_{form}) + Distance(x_{form}, x_{near})$ **then**
      if $CollisionFree(x_{form}, x_{near})$ **then**
        $E \leftarrow (E \setminus \{(Parent(x_{near}), x_{near})\}) \cup \{(x_{form}, x_{near})\}$;
      end
    end
  end
end
Return $G = (V, E)$;

process can find the node x_{parent} from X_{near} to replace the candidate parent node $x_{nearest}$ of x_{rand}, which will reduce the cost of the path from x_{start} to x_{rand}. In the *Rewire* procedure, the cost of the paths from x_{start} to the element of X_{near} is optimized by the x_{rand}. For any point x_{near} in X_{near}, if replacing the parent node of x_{near} by x_{rand} could reduce the cost of the path from x_{start} to x_{near}, then it will be replaced. The *Rewire*, described in Algorithm 3, provides asymptotic optimality because it changes the connection relationships of tree vertexes like Fig. 1(d).

In addition to optimization modules, some primitive procedures involved in RRT* are described as follows.

SampleFree: It returns a sample x_{rand} that selected randomly from X_{free}.

Nearest: It returns the vertex $x_{nearest}$ that is closest to x_{rand} in terms of Euclidean distance.

Near: It returns the set of vertices X_{near} contained in a hypersphere of a specific radius R_{near} centered at x_{rand}.

CollisionFree: Given two nodes, x_{p1} and x_{p2}, it checks whether the local path σ from x_{p1} to x_{p2} is collision-free in *Map*.

Cost: Given a vertex x_p from V. it returns the full length of the path from x_{start} to x_p.

Distance: Given two nodes, x_{p1} and x_{p2}, it returns the distance between x_{p1} and x_{p2} in terms of Euclidean distance.

Parent: Given a vertex x_p from V, it returns the parent vertex of x_p.

InitialPathFound: Determine if the initial solution is found.

2.3. Q-RRT*

In this section, we show Q-RRT*, as shown in Algorithm 4, which is a typical example of fast convergence algorithm (Jeong et al., 2019). Q-RRT* introduces a parameter $n_{ancestor}$ to two optimization modules on the

Algorithm 6: RRT*-Smart

Input: x_{start},X_{goal},Map, n_{repeat}, R_{near}

Output: $G = (V, E)$

$V \leftarrow \{x_{start}\}$,$E \leftarrow \phi$;
for $i = 1$ *to* n_{repeat} **do**
  if $i = n + b, n + 2b...$ **then**
    $x_{rand} \leftarrow SampleFree(i)$;
  else
    $x_{rand} \leftarrow IntelligentSample(Z_{beacons})$;
  end
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  if $CollisionFree(x_{rand}, x_{nearest})$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
    $x_{parent} \leftarrow ChooseParent(X_{near}, x_{rand}, x_{nearest})$;
    $V \leftarrow V \cup \{x_{rand}\}$;
    $E \leftarrow E \cup \{(x_{parent}, x_{rand})\}$;
    $G \leftarrow Rewire(G, x_{rand}, X_{near})$;
    if $InitialPathFound$ **then**
      Return $G = (V, E)$;
      $n \leftarrow i$;
    end
    $(T, directcost) \leftarrow PathOptimization(G, Z_{init}, X_{goal})$;
    if $directcost_{new} < directcost_{old}$ **then**
      $Z_{beacons} \leftarrow PathOptimization(G, Z_{init}, X_{goal})$;
    end
  end
end
Return $G = (V, E)$;

Algorithm 7: F-RRT*

Input: $x_{start}, X_{goal}, Map, n_{repeat}, R_{near}, D_{dichotomy}$

Output: $G = (V, E)$

$V \leftarrow \{x_{start}\}, E \leftarrow \phi$;
for $i = 1$ *to* n_{repeat} **do**
  $x_{rand} \leftarrow SampleFree(i)$;
  $x_{nearest} \leftarrow Nearest(V, x_{rand})$;
  if $CollisionFree(x_{rand}, x_{nearest})$ **then**
    $X_{near} \leftarrow Near(V, x_{rand}, R_{near})$;
    $x_{reachest} \leftarrow FindReachest(G, x_{nearest}, x_{rand})$;
    $x_{create} \leftarrow CreatNode(G, x_{reachest}, x_{rand}, D_{dichotomy})$;
    if $x_{create} \neq \phi$ **then**
      $V \leftarrow V \cup \{x_{create}, x_{rand}\}$;
      $E \leftarrow E \cup \{(Parent(x_{reachest}), x_{create}), (x_{create}, x_{rand})\}$;
    else
      $V \leftarrow V \cup \{x_{rand}\}$;
      $E \leftarrow E \cup \{(x_{reachest}, x_{rand})\}$;
    end
    if $InitialPathFound$ **then**
      Return $G = (V, E)$;
    end
    $G \leftarrow$Rewire(G, x_{rand}, X_{near});
  end
end
Return $G = (V, E)$;

basis of RRT*. The optimization modules of RRT* are adjusted in Q-RRT* based on triangle inequality. In the *ChooseParent* procedure, Q-RRT* increases the number of potential parent nodes for x_{rand} by searching X_{near} as well as X_{parent}. The X_{parent} is the ancestry of X_{near}, any node in X_{parent} being the parent node of x_{rand} will make the cost of the path from x_{start} to x_{rand} lower than X_{near} does. In the *Rewire* procedure, the connections of the X_{near} elements are optimized by the x_{rand} and the ancestory of x_{rand}. The *Rewire* procedure of Q-RRT* is presented in Algorithm 5. According to the results of the Q-RRT* optimization modules, shown in Fig. 1(d) and Fig. 1(e), it is obvious that the initial solution of Q-RRT* is significantly better than RRT* under the same sampling. The following describes the two primitive procedures used in Q-RRT*.

ancestor: Given a graph $G = (V,E)$, a vertex x_p, and a natural number $i \in N$, it returns the i-th parent of x_p.

Ancestry: Given a graph $G = (V,E)$, a vertex x_p, and a natural number $n_{ancestor}$, it returns ϕ if $n_{ancestor}$ is 0, otherwise, it returns $\bigcup_{i=1}^{n_{ancestor}} ancestor(G, x_p, i)$. If a vertex set S is given instead of x_p, then it returns $\bigcup_{x_p \in S} Ancestry(G, x_p, n_{ancestor})$.

2.4. RRT*-Smart

RRT*-Smart (Islam et al., 2012) is proposed to accelerate its rate of convergence, which provides two new techniques for RRT*: *PathOptimization* and *IntelligentSample*. RRT*-Smart works in the same way as RRT* during the initial feasible path solution process. Once the first path is found, the triangle inequality is used in *PathOptimization* procedure to remove redundant nodes of the feasible path. This optimized path yields biasing points for *IntelligentSample* procedure. This process will continue to shorten the length of the shortest path in the random tree, and eventually generate a path close to the obstacle. As shown in Algorithm 6, whenever a shorter path is found, the biasing shifts towards the new path.

2.5. Informed sampling

Essentially, Informed-RRT* (Gammell et al., 2014) is a node rejection technique, but because the passed nodes are limited to a specific area, it is usually used as a sampling strategy. Before finding a solution, Informed-RRT* will not reject any sampling points and works in the

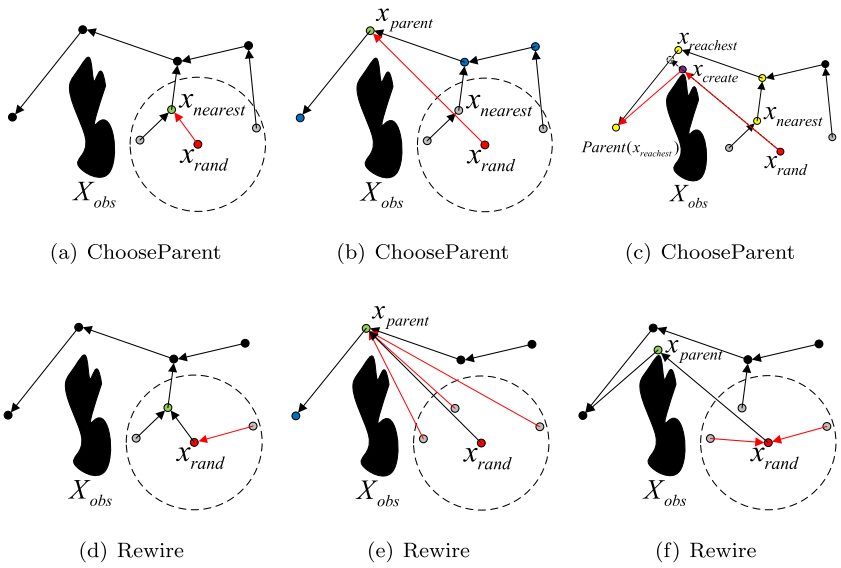


Fig. 1. Tree construction of (a), (d) RRT*,and (b), (e) Q-RRT*, and (c), (f) F-RRT*. (red dot: x_{rand}, green dot: x_{parent}, dashed circle: Near(x_{rand}), gray dots: X_{near}, blue dots: X_{parent}, yellow dots: ancestors of $x_{nearest}$, purple dot: x_{create}).

same manner as RRT* does. Once a solution is found, Informed-RRT* sets a hyper-ellipsoid whose focal points are x_{start} and x_{goal} to limit the range of sampling points. For a 2-d plane environment, a sampling point x_{rand} in the hyper-ellipsoid needs to satisfy Eqn. (1)

$$|x_{rand} - x_{start}| + |x_{rand} - x_{goal}| \leqslant c_{best} \tag{1}$$

where c_{best} is the cost of the best solution at the moment.

3. F-RRT*

This section describes the F-RRT* algorithm and introduces a new way to reduce the cost of the path from x_{start} to x_{rand}.

3.1. Proposed algorithm

The main idea of F-RRT* is based on two observations: the value of the radius parameter R_{near} has a great impact on the computation time and some nodes of the optimal path that are close to obstacles. In the previous algorithms, the quality of the initial solution could be improved by increasing the radius parameter R_{near}, because a large radius R_{near} is easier to reduce the $Cost(x_{rand})$. However, it also increases the number of nearby vertices and the computation time exponentially. To reduce the influence of the radius parameter R_{near} on the computation time, the radius parameter R_{near} is abandoned in F-RRT* when adding x_{rand} to the tree. F-RRT* chooses the parent node for x_{rand} through two steps, the *FindReachest*, and *CreatNode* procedure. The *FindReachest* procedure searches the ancestors of $x_{nearest}$ and finds the vertex $x_{reachest}$ which can be connected to x_{rand} without collision. The *CreatNode* procedure is to create a node x_{create} that can connect the x_{rand} and the parent node of $x_{reachest}$ at the same time like Fig. 1(c). However, the dichotomy was used in the creation of x_{create}, so the F-RRT* algorithm, shown in Algorithm 7, needs to introduce a parameter $D_{dichotomy}$ to determine whether the created point meets the requirements.

Algorithm 8: FindReachest

Input: $G, x_{nearest}, x_{rand}$

Output: $x_{reachest}$

$x_{reachest} \leftarrow x_{nearest}$;
while $x_{reachest} \neq x_{start}$ **do**
  if $CollisionFree(x_{rand}, Parent(x_{reachest}))$ **then**
    $x_{reachest} \leftarrow Parent(x_{reachest})$;
  else
    Return $x_{reachest}$;
  end
end
Return $x_{reachest}$;

Algorithm 9: CreatNode

Input: $G, x_{reachest}, x_{rand}, D_{dichotomy}$

Output: x_{create}

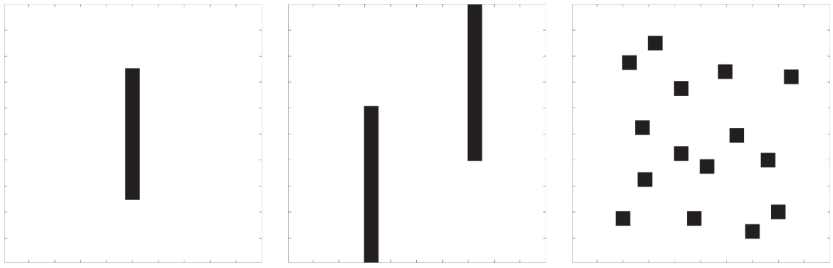
$x_{allow} \leftarrow x_{reachest}$;
if $x_{reachest} \neq x_{start}$ **then**
  $x_{forbid} \leftarrow Parent(x_{reachest})$;
  while $Distance(x_{allow}, x_{forbid}) > D_{dichotomy}$ **do**
    $x_{mid} \leftarrow (x_{allow} + x_{forbid})/2$;
    if $CollisionFree(x_{rand}, x_{mid})$ **then**
      $x_{allow} \leftarrow x_{mid}$;
    else
      $x_{forbid} \leftarrow x_{mid}$;
    end
  end
  $x_{forbid} \leftarrow x_{rand}$;
  while $Distance(x_{allow}, x_{forbid}) > D_{dichotomy}$ **do**
    $x_{mid} \leftarrow (x_{allow} + x_{forbid})/2$;
    if $CollisionFree(x_{mid}, Parent(x_{reachest}))$ **then**
      $x_{allow} \leftarrow x_{mid}$;
    else
      $x_{forbid} \leftarrow x_{mid}$;
    end
  end
end
if $x_{allow} \neq x_{reachest}$ **then**
  $x_{create} \leftarrow x_{allow}$;
else
  $x_{create} \leftarrow \phi$;
end
Return x_{create};

3.2. FindReachest procedure

Inspired by Q-RRT*, the *FindReachest* procedure of F-RRT* is to find the reachable vertex $x_{reachest}$ from the ancestors of $x_{nearest}$. The $x_{reachest}$ vertex will become a candidate parent node of x_{rand}. It is obvious that connecting $x_{reachest}$ and x_{rand} will make the cost of the path from the x_{start} to x_{rand} lower than connecting $x_{nearest}$ and x_{rand}. The *FindReachest* process is described in Fig. 1(c) and Algorithm 8, it only searches the ancestors of $x_{nearest}$, not the vertices around x_{rand}. The reduction in the number of search nodes will inevitably lead to F-RRT* gaining an advantage in planning time.

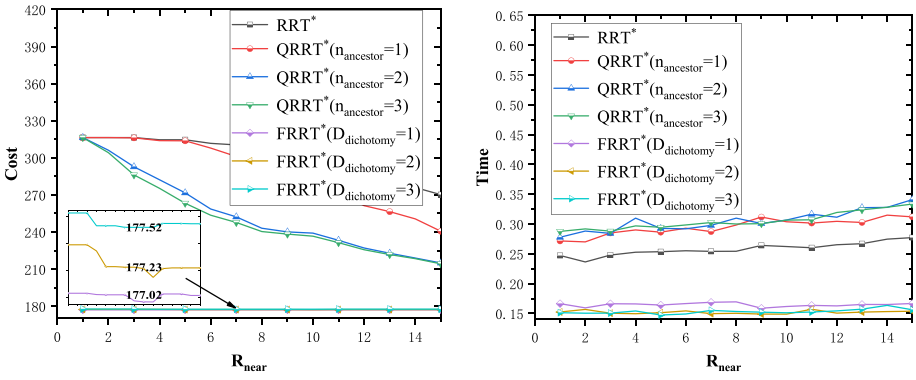
3.3. CreatNode procedure

After observing the relationship between the optimal path and obstacles, we believe that the $Cost(x_{rand})$ of path with connecting $x_{reachest}$ and x_{rand} can be further optimized by creating a new node based on triangle inequality. The node used to optimize the path is created in CreatNode procedure by dichotomy, as shown in Fig. 1(c) and Algorithm 9. A parameter named $D_{dichotomy}$ is used as the termination condition of the dichotomy during the creation of x_{create}. The *CreatNode* procedure creates a node near the obstacle in a limited number of steps, which effectively shortens the time required for optimization and reduces the



(a) Simple (b) Maze (c) Cluttered

Fig. 2. Environments for the simulations. (a) Simple. (b) Maze. (c) Cluttered.



(a) C_{init} (b) T_{init}

Fig. 3. First-solution performance over R_{near} in maze environment.

cost of the initial solution.

3.4. analysis

The main difference between the three extended algorithms (RRT*, Q-RRT*, and F-RRT*) is how to choose the parent for x_{rand} to reduce the cost of the path from x_{start} to x_{rand}. Fig. 1 describes how the optimization procedures of the three algorithms work. RRT* searches the nearby vertices X_{near} (grey dots), Q-RRT* searches the X_{near} and X_{parent} (blue dots), F-RRT* only searches ancestors (yellow dots) of $x_{nearest}$ until finding an unreachable point $Parent(x_{reachest})$, then a node x_{create} is created so that x_{rand} could connect $Parent(x_{reachest})$ through x_{create} without collision as shown in Fig. 1(c). It can be seen from the figure that F-RRT* has advantages in terms of the calculation and $Cost(x_{rand})$.

After the random point x_{rand} is added to the tree, F-RRT* and RRT* only use x_{rand} to optimize the nearby vertices X_{near}, while the ancestry of x_{rand} are used in Q-RRT*. Although F-RRT* uses the same *Rewire* procedure as in RRT*, the result of F-RRT* has a better performance than RRT* due to the different parent of x_{rand}. Perhaps the cost of paths from x_{start} to the vertices of X_{near} could be further optimized in Q-RRT*, but the number of collision detections is also increased, and this number is positively correlated with the number of nodes in X_{near}. This will cause a surge in calculation time during convergence.

4. Simulation results and analysis

In this section, F-RRT* is compared with RRT* (Karaman & Frazzoli, 2011), Q-RRT* (Jeong et al., 2015), and RRT*-Smart (Islam et al., 2012) in 2-dimensional maps shown in Fig. 2. RRT* is chosen to be the baseline, and Q-RRT* and RRT*-Smart are used for comparative analysis, because they both use triangle inequality to optimize paths and are typical examples of fast convergence algorithms. In the process of solving the optimal path, Informed-RRT* (Gammell et al., 2014) is used as a sampling strategy and combined with RRT*, Q-RRT*, and F-RRT* to speed up their convergence rate.

This comparison will be completed through two simulation experiments. In the first simulation, the quality of the initial path generated by four path planning algorithms is compared by three evaluation indicators: C_{init}^{ALG}, which is the cost of the initial solution by algorithm ALG in terms of Euclidean distance; T_{init}^{ALG}, which is the time of initially found solution; S_{init}^{ALG}, which is the sum of the turning angles (Qi, Yang, & Sun, 2020) in the path. In another simulation, the indicator $T_{5\%}^{ALG}$ is utilized to compare the convergence rate of the three algorithms. $T_{5\%}$ is the time to

Table 1
Comparing algorithms with $R_{near} = 5$ on the quality of the initial solution in simple environment.

Indicator	algorithm	Mean	Std	Min	Max
C_{init}	RRT*	314.58	57.86	208.61	447.59
	Q-RRT*	263.22	36.37	198.36	361.93
	F-RRT*	177.23	2.91	171.14	184.99
T_{init}	RRT*	0.2609	0.57	0.0008	3.27
	Q-RRT*	0.2948	0.63	0.0008	3.57
	F-RRT*	0.1624	0.33	0.0014	1.90

Table 2
Comparing algorithms with $R_{near} = 10$ on the quality of the initial solution in simple environment.

Indicator	algorithm	Mean	Std	Min	Max
C_{init}	RRT*	299.34	55.01	208.61	446.87
	Q-RRT*	236.57	28.50	179.65	313.61
	F-RRT*	177.15	2.92	171.14	184.99
T_{init}	RRT*	0.2659	0.59	0.0008	3.36
	Q-RRT*	0.3154	0.67	0.0009	3.81
	F-RRT*	0.1665	0.34	0.0015	1.89

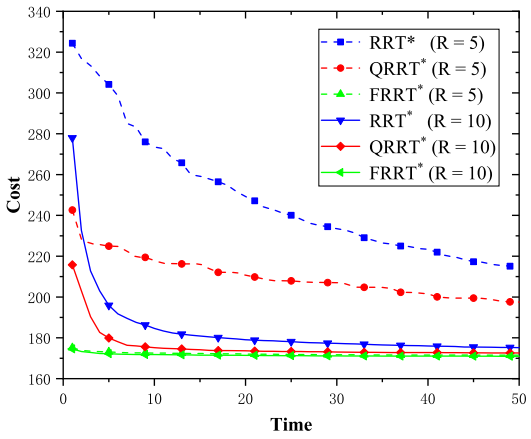


Fig. 4. Solution cost over computation time in the simple environment.

find a sub-optimal solution of $1.05C_{optimal}$, where $C_{optimal}$ is the length of the optimal solution. The simulations are implemented on an Intel Xeon (R) i7-4910 CPU with 32G of RAM. The simulation platform is Matlab R2018a.

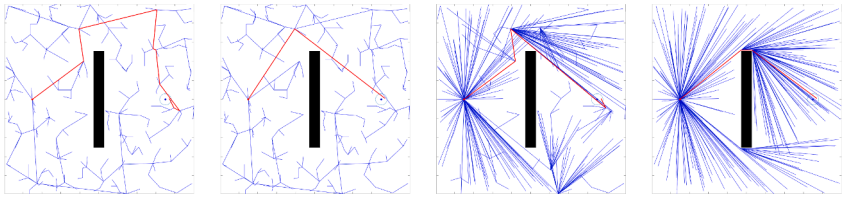
4.1. Choosing parameter

Before comparing algorithms performance, the path planning problem shown in Fig. 5 is simulated to choose the proper parameter value. It is important to set an appropriate value for R_{near}, $D_{dichotomy}$, and $n_{ancestor}$, because they are closely related to the cost and time required to generate a path. For example, a lower $D_{dichotomy}$ could make F-RRT* generate a path with a lower cost, but this will increases the number of calls to the *CollisionFree* procedure which strongly affects the computation time.

Fig. 3 shows the relationship between the initial solution performance of the three algorithms in the maze environment and the R_{near} value. The value of the curve is the average value obtained from 40 simulations. It can be seen from the curves that when the radius R_{near} increases, C_{init} decreases, while T_{init} increases simultaneously. As R_{near} gradually increases from 1 to 15, the performance of Q-RRT* changes the most, while F-RRT* is almost unaffected.

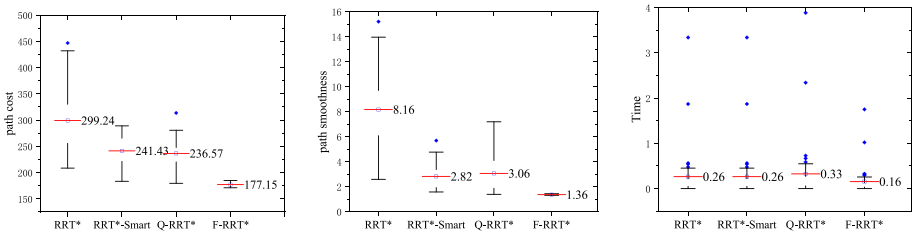
To avoid the increase in planning time due to the low value of $D_{dichotomy}$, F-RRT* with $D_{dichotomy} = 2$ is recommended for comparative analysis with other algorithms. For Q-RRT*, although increasing $n_{ancestor}$ will increase T_{init}^{Q-RRT*}, the significant improvement in C_{init}^{Q-RRT*} makes $n_{ancestor} = 3$ worthwhile. Tables 1 and 2 provide statistical information of C_{init} and T_{init} when $R_{near} = 5$ and $R_{near} = 10$, respectively. As R_{near} increases from 5 to 10, C_{init}^{Q-RRT*} decreases by 10%, but the C_{init}^{Q-RRT*} at $R_{near} = 10$ is still 33% higher than the C_{init}^{F-RRT*} at $R_{near} = 5$. Simultaneously, T_{init}^{Q-RRT*} increased by 7%, while T_{init}^{F-RRT*} changed by less than 2.5%.

Fig. 4 shows the average of solutions cost over computation time in simple environment about the planning problem defined in Fig. 5. It reveals that the three algorithms perform much better at $R_{near} = 10$ than at 5. Therefore, $R_{near} = 10$ is recommended for comparative analysis in



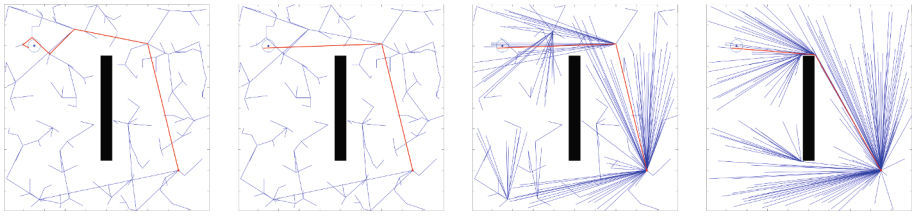
(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 5. One of the experimental results of the first planning problem defined in a simple environment. (a) $C_{init} = 319$, (b) $C_{init} = 208$, (c) $C_{init} = 243$, (d) $C_{init} = 179$.



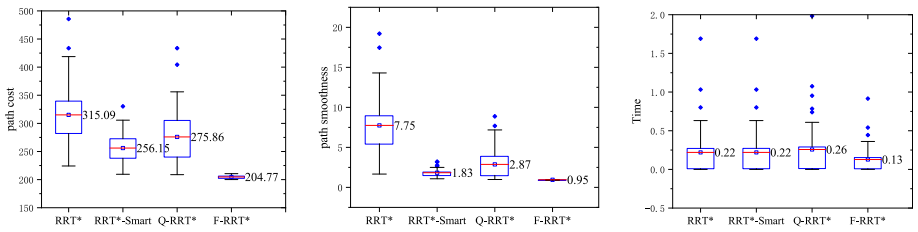
(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 6. The performance of the four algorithms on the first planning problem defined in a simple environment. (red line: the average value).



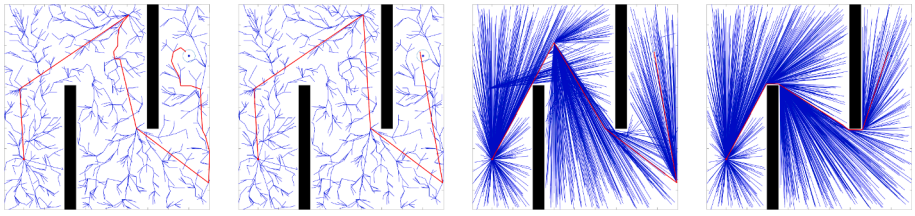
(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 7. One of the experimental results of the second planning problem defined in a simple environment. (a) $C_{init} = 273$, (b) $C_{init} = 241$, (c) $C_{init} = 241$, (d) $C_{init} = 210$.



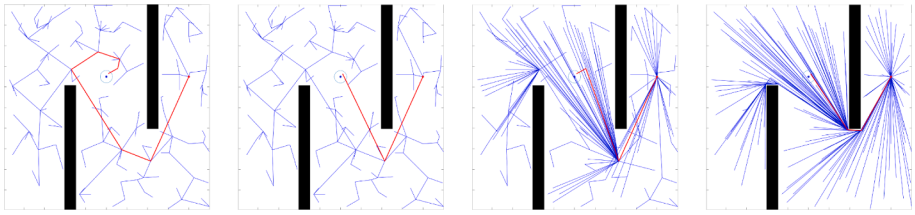
(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 8. The performance of the four algorithms on the second planning problem defined in the simple environment. (red line: the average value).



(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 9. One of the experimental results of the first planning problem defined in a maze environment. (a) $C_{init} = 567$, (b) $C_{init} = 523$, (c) $C_{init} = 441$, (d) $C_{init} = 269$.



(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 11. One of the experimental results of the second planning problem defined in a maze environment. (a) $C_{init} = 285$, (b) $C_{init} = 184$, (c) $C_{init} = 195$, (d) $C_{init} = 138$.

all algorithms.

4.2. Comparison of initial path generation

The first comparative simulation is performed to verify the effectiveness of the proposed algorithm in initial path generation. The three maps shown in Fig. 2 were selected, all of which are 200×200 in size. Besides, we set two sets of starting/goal points in each of the three maps to define two path planning problems for each environment. Due to the randomness of sampling-based algorithms, each algorithm was run 40 times with 40 sets of sampling points. In other words, all algorithms used the same sampling sequence in each comparison experiment. The parameter settings in the simulation are as follows: $n_{ancestor} = 3$ was selected in Q-RRT*, $D_{dichotomy} = 2$ in F-RRT*, $b = 10$ in RRT*-Smart, $R_{near} = 10$ for all algorithms, and $n_{repeat} = Inf$ is to keep the random tree expanding. The stopping condition of the program is to find the initial solution or reach the predetermined time.

4.2.1. Simple environment

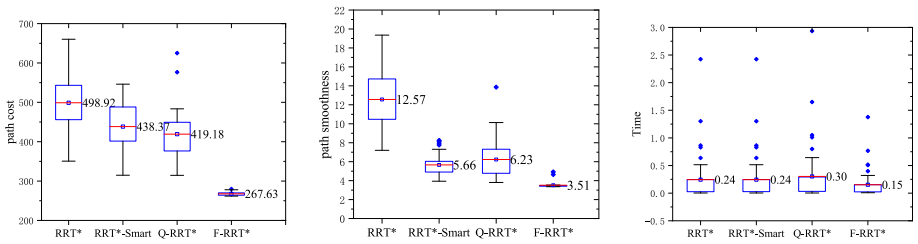
By defining different starting points/goal points for the simple environment, two path planning problems can be obtained, as shown in Figs. 5 and 7. Figs. 6 and 8 show the performance of the four algorithms about C_{init}, S_{init} and T_{init}. What can be clearly seen in Figs. 5 and 7 is that the path generated by F-RRT* is closer to obstacles, which will make the path have a lower length. The simulation result statistics also illustrate that F-RRT* has an advantage on the length of initial solution in simple environment, because the average of C_{init}^{F-RRT*} is the lowest. And its cabinet height is also the lowest, which proves that F-RRT* has higher stability than Q-RRT* and RRT*. Analyzing in the same way, it is not difficult to conclude that the F-RRT* algorithm also has significant advantages over S_{init} and T_{init}. In general, these results show that F-RRT* is more successful in finding a high-quality initial path in simple environment.

4.2.2. Maze environment

Figs. 9 and 11 respectively show an example of the simulation results of two path planning problems in a maze environment. Figs. 10 and 12 provide the detailed statistics of the simulation results. From the analysis of the statistical results of 40 experiments, we can conclude that the F-RRT* algorithm has obvious advantages in terms of the length and smoothness of the initial path, whether it is evaluated by using the average, or dispersion.

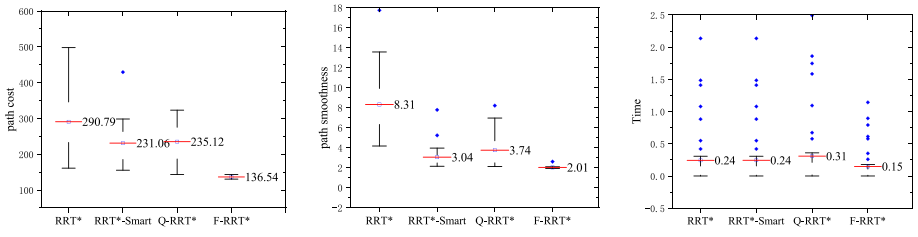
4.2.3. Cluttered environment

Figs. 13 and 15 respectively show an example of the simulation results of two path planning problems in a maze environment. Figs. 14 and 16 provide the detailed statistics of the simulation results. According to the statistical results, the three evaluation indicators of F-RRT* are the lowest, indicating that the performance of F-RRT* is better than the other three algorithms in the cluttered environment.



(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 10. The performance of the four algorithms on the first planning problem defined in the maze environment. (red line: the average value).



(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 12. The performance of the four algorithms on the second planning problem defined in the maze environment. (red line: the average value).

4.3. Comparing algorithms on the convergence rate

To compare the convergence of the four algorithms, the pseudo-code of the algorithms needs to be partially modified: lines 10–12 in Algorithm 1, line 16 in Algorithm 6, lines 11–13 in Algorithm 4, and lines 16–18 in Algorithm 7 are deleted. The parameters in the simulation are the same as in Section 4.2. To avoid the disturbance of the sample sequence and keep unique variables, the random points used in each trial are the same. Besides, the Informed sampling strategy will be used in combination with three expansion algorithms (RRT*, Q-RRT*, and F-RRT*) to accelerate convergence. And due to the randomness of sampling, the simulation will be conducted 40 times to compare the convergence rate of the three algorithms for each path planning problem. In each simulation, algorithms will be performed for 50 s, and the random tree will be recorded once a second, and then the shortest path contained in the random tree will be considered as the best path generated by the algorithm at that moment. The best path obtained found within 50 s runtime is considered the optimal path in this paper.

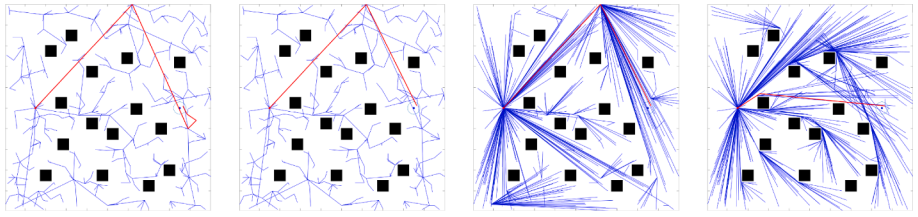
4.3.1. Simple environment

Fig. 17 shows the average of solutions cost over computation time in simple environment about the planning problem defined in Fig. 5. Fig. 18 shows the statistical results of the $T_{5\%}$. In this statistical figure, 'R' is the abbreviation of 'RRT*', 'Q' denotes 'Q-RRT*', 'RS' denotes 'RRT*-Smart', 'F' denotes 'F-RRT*'. And 'IR', 'IQ', and 'IF' denote 'Informed + RRT*', 'Informed + Q-RRT*', and 'Informed + F-RRT*', respectively.

It can be seen from Fig. 17 that all algorithms have the property of optimal convergence, among which F-RRT* uses the least time to plan a sub-optimal path. The statistical results show that F-RRT* takes an average of 1.25s to plan a sub-optimal path, which only accounts for 20% of $T_{5\%}^{Q-RRT*}$ and 5% of $T_{5\%}^{RRT*}$. And, according to its box height, it can be known that the performance of F-RRT* is the most stable. The advantages of F-RRT* in a simple environment can also be verified in the simulation experiment of the second path planning problem. Figs. 19 and 20 contain the statistical results of the planning problem defined in Fig. 7. What can be clearly seen from the statistical results is that F-RRT* algorithm can generate the near-optimal path faster than other algorithms, which makes the proposed algorithm more popular in simple environment.

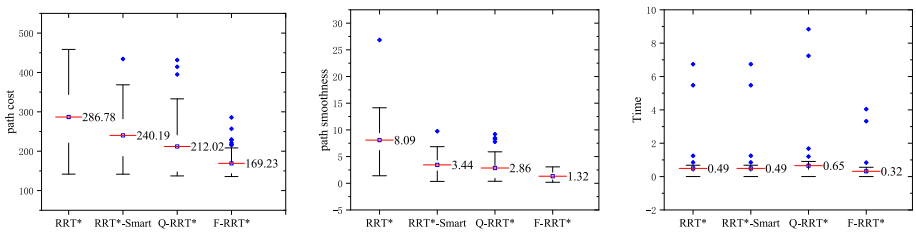
4.3.2. Maze environment

Fig. 21 shows the average of solutions cost over computation time in maze environment about the planning problem defined in Fig. 9. Fig. 22 shows the statistical results of the $T_{5\%}$. Fig. 23 and 24 contain the statistical results of the planning problem defined in Fig. 11. The figures show that the proposed algorithm can generate a relatively better solution with a shorter time under the same conditions, which causes the



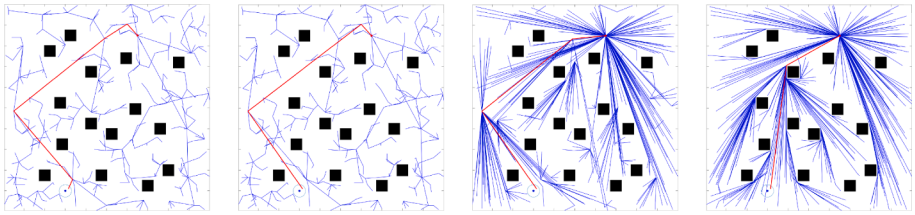
(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 13. One of the experimental results of the first planning problem defined in a cluttered environment. (a) C_{init} = 300, (b) C_{init} = 247, (c) C_{init} = 247, (d) C_{init} = 148.



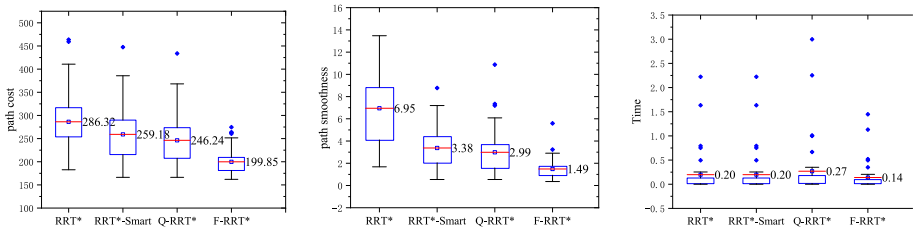
(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 14. The performance of the four algorithms on the first planning problem defined in the cluttered environment. (red line: the average value).



(a) RRT* (b) RRT*-Smart (c) Q-RRT* (d) F-RRT*

Fig. 15. One of the experimental results of the second planning problem defined in a cluttered environment. (a) $C_{init} = 250$, (b) $C_{init} = 245$, (c) $C_{init} = 236$, (d) $C_{init} = 179$.



(a) C_{init} (b) S_{init} (c) T_{init}

Fig. 16. The performance of the four algorithms on the second planning problem defined in the cluttered environment. (red line: the average value).

time required for F-RRT* to converge to the sub-optimal path is much lower than that of RRT* and Q-RRT* algorithms. In addition, even when combined with sampling strategies such as Informed sampling and RRT*-Smart, the time required for RRT* and Q-RRT* to plan sub-optimal solutions is much higher than F-RRT*. From the simulation results, we can conclude that the proposed algorithm, F-RRT*, has better convergence performance in the maze environment than the other algorithms.

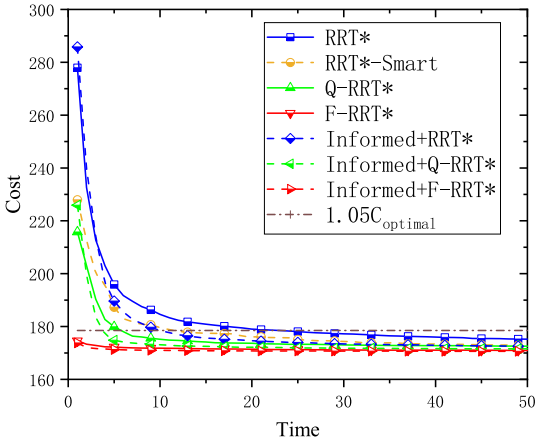


Fig. 17. Solution cost over computation time of the first planning problem defined in the simple environment.

4.3.3. Cluttered environment

Fig. 25 shows the average of solutions cost over computation time in simple environment about the planning problem defined in Fig. 13. Fig. 26 shows the statistical results of the $T_{5\%}$. Fig. 27 and 28 contain the statistical results of the planning problem defined in Fig. 15. It can be seen from Fig. 25 and 27 that F-RRT* has a clear advantage at the beginning and final stages of the simulation. Although F-RRT* does not dominate in the middle of the simulation, it may be because F-RRT* only searches for one branch of the tree, which leads to a weaker local dilemma escape than the other two algorithms. Despite this, F-RRT* can still quickly converge to the optimum and regain its advantages after a period of simulation.

Statistics also show that compared with the Q-RRT* algorithm, F-RRT* spends more than 13% of the time in a second path planning problem to generate the sub-optimal path. However, if combined with the Informed sampling strategy, the average time required for F-RRT* to plan a sub-optimal path is still less than Q-RRT*.

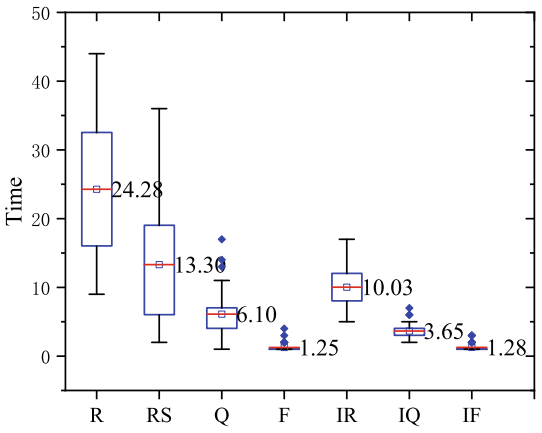


Fig. 18. $T_{5\%}$ of the first planning problem defined in the simple environment. (red line: the average value of $T_{5\%}$).

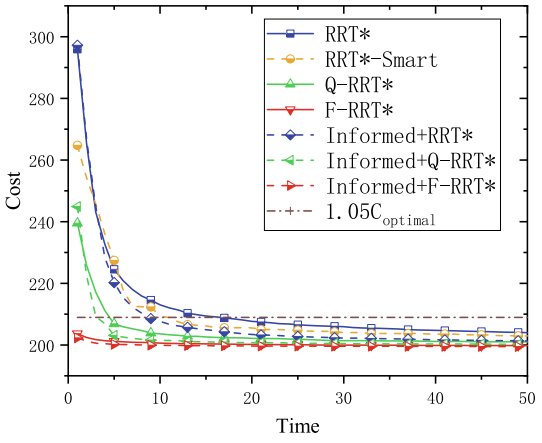


Fig. 19. Solution cost over computation time of the second planning problem defined in the simple environment.

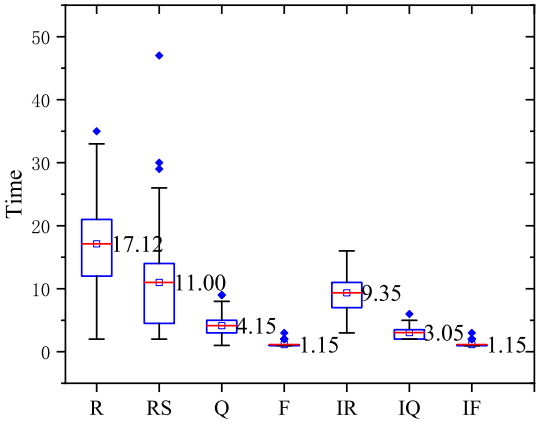


Fig. 20. $T_{5\%}$ of the second planning problem defined in the simple environment. (red line: the average value of $T_{5\%}$).

4.4. Result analysis

From these experimental results, the statistics show that the F-RRT* algorithm outperforms the other two algorithms. The advantages may come from the following reasons.

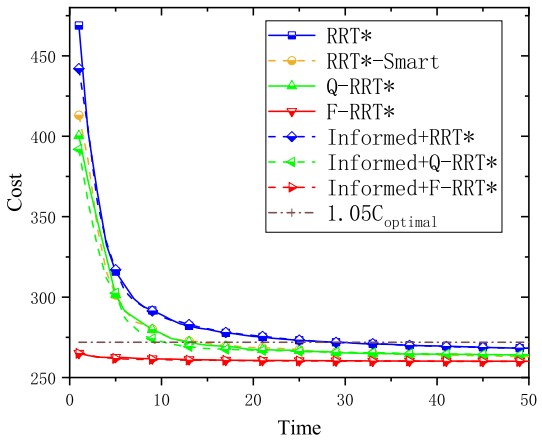


Fig. 21. Solution cost over computation time of the first planning problem defined in the maze environment.

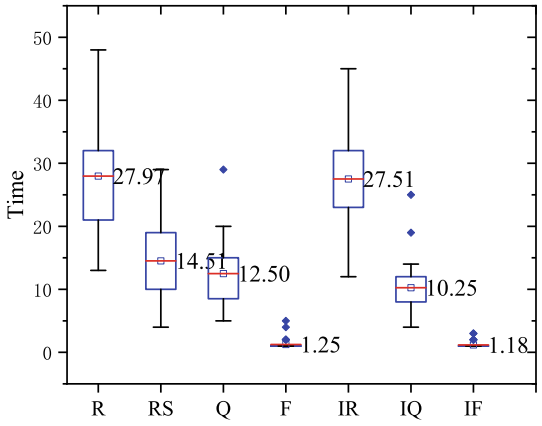


Fig. 22. $T_{5\%}$ of the first planning problem defined in the maze environment. (red line: the average value of $T_{5\%}$).

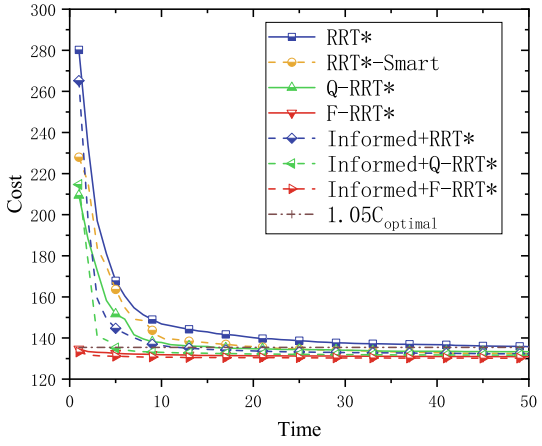


Fig. 23. Solution cost over computation time of the second planning problem defined in the maze environment.

(1) The use of triangle inequality makes the initial path generated by F-RRT* shorter and smoother than Q-RRT* and RRT*. This superiority is more obvious in maze environment.
(2) F-RRT* occupies an advantage in the time of the initial path planning, because the number of nodes searched during the tree extension process is much smaller than the other two algorithms.

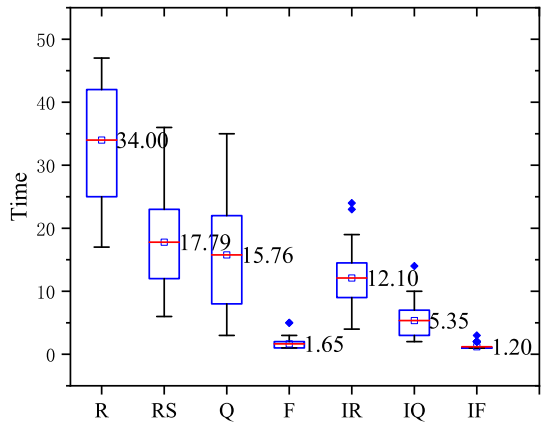


Fig. 24. $T_{5\%}$ of the second planning problem defined in the maze environment. (red line: the average value of $T_{5\%}$).

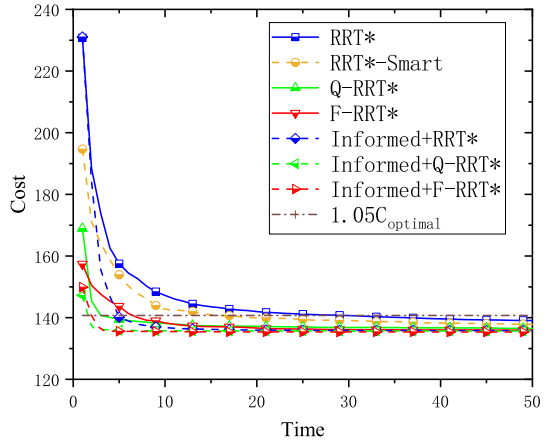


Fig. 25. Solution cost over computation time of the first planning problem defined in the cluttered environment.

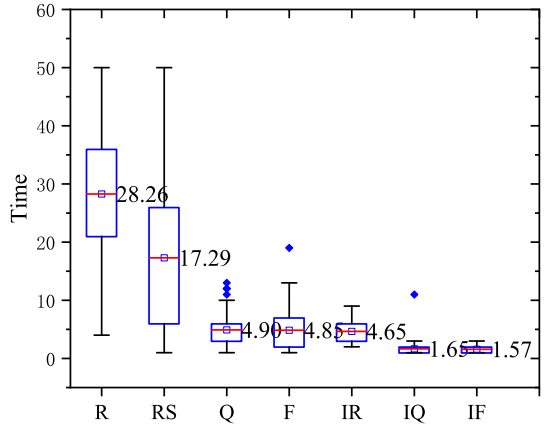


Fig. 26. $T_{5\%}$ of the first planning problem defined in the cluttered environment. (red line: the average value of $T_{5\%}$).

(3) Compared with sampling, creation makes it easier to obtain the nodes required for the optimal path. It causes an advantage in convergence for F-RRT*, especially in a maze environment.

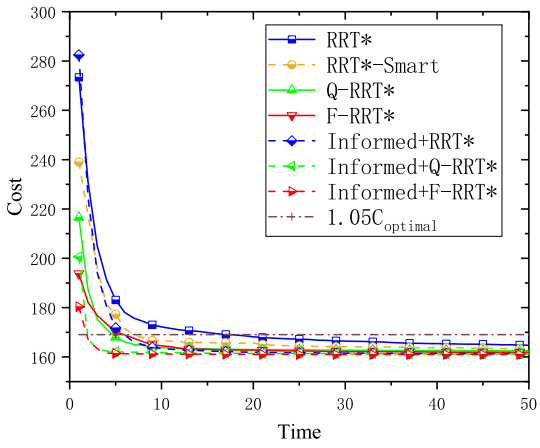


Fig. 27. Solution cost over computation time of the second planning problem defined in the cluttered environment.

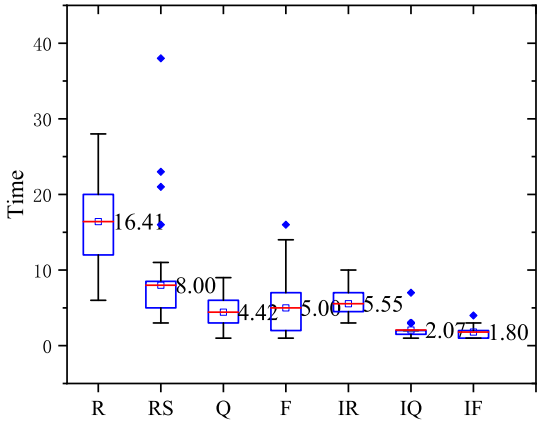


Fig. 28. $T_{5\%}$ of the second planning problem defined in the cluttered environment. (red line: the average value of $T_{5\%}$).

5. Conclusion

This paper proposed an RRT*-based optimal path planning algorithm, F-RRT*, which performs better than the other state-of-the-art algorithms in the initial solution and fast convergence rate. The main idea of F-RRT* is to create nodes based on the triangle inequality to optimize the cost of each sampling point. Instead of selecting from the existing vertices, it searches only one branch of the random tree to create a parent node close to the obstacle for each sampling point. Thereby obtaining a better initial solution and a faster convergence rate than Q-RRT* and RRT* under the same condition. In addition to this, F-RRT*, as a tree-extending algorithm, can be combined with any sampling strategy to further enhance the performance. The effectiveness of the proposed algorithm F-RRT* was demonstrated by comparing it with other algorithms in the numerical simulation.

Although the F-RRT* algorithm can obtain a lower-cost initial solution and converge faster compared to the current sampling-based planning algorithms, the algorithm needs to be improved or discarded if the triangular inequality it based on is not satisfied.

CRediT authorship contribution statement

Bin Liao: Conceptualization, Methodology, Software, Writing - original draft, Writing - review & editing. **Fangyi Wan:** Validation, Supervision, Funding acquisition, Methodology. **Yi Hua:** Formal analysis, Data curation, Project administration, Writing - original draft, Writing - review & editing. **Ruirui Ma:** Investigation, Validation.

Shenrui Zhu: Writing - original draft, Visualization. **Xinlin Qing:** Supervision, Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

References

Akgun, B., & Stilman, M. (2011). Sampling heuristics for optimal motion planning in high dimensions. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 2640–2645). IEEE.

Alexopoulos, C., & Griffin, P. M. (1992). Path planning for a mobile robot. *IEEE Transactions on Systems, Man, and Cybernetics, 22*, 318–322.

Ferguson, D., & Stentz, A. (2006). Anytime rrts. In *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems* (pp. 5369–5375). IEEE.

Gammell, J. D., Srinivasa, S. S., & Barfoot, T. D. (2014). 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* (pp. 2997–3004). IEEE.

González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems, 17*, 1135–1145.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics, 4*, 100–107.

Hwang, Y. K., Ahuja, N., et al. (1992). A potential field approach to path planning. *IEEE Transactions on Robotics and Automation, 8*, 23–32.

Islam, F., Nasir, J., Malik, U., Ayaz, Y., & Hasan, O. (2012). Rrt*-smart: Rapid convergence implementation of rrt* towards optimal solution. In *2012 IEEE International Conference on Mechatronics and Automation* (pp. 1651–1656). IEEE.

Jeong, I.-B., Lee, S.-J., & Kim, J.-H. (2015). Rrt*-quick: A motion planning algorithm with faster convergence rate. In *Robot Intelligence Technology and Applications 3* (pp. 67–76). Springer.

Jeong, I.-B., Lee, S.-J., & Kim, J.-H. (2019). Quick-rrt*: Triangular inequality-based implementation of rrt* with improved initial solution and convergence rate. *Expert Systems with Applications, 123*, 82–90.

Kanehara, M., Kagami, S., Kuffner, J. J., Thompson, S., & Mizoguhi, H. (2007). Path shortening and smoothing of grid-based path planning with consideration of obstacles. In *2007 IEEE International Conference on Systems, Man and Cybernetics* (pp. 991–996). IEEE.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research, 30*, 846–894.

Karaman, S., Walter, M. R., Perez, A., Frazzoli, E., & Teller, S. (2011). Anytime motion planning using the rrt. In *2011 IEEE International Conference on Robotics and Automation* (pp. 1478–1483). IEEE.

Khatib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots. In *Autonomous Robot Vehicles* (pp. 396–404). Springer.

Koenig, S., Likhachev, M., & Furcy, D. (2004). Lifelong planning a*. *Artificial Intelligence, 155*, 93–146.

LaValle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning*.

LaValle, S.M., & Kuffner, J.J. (2001). Rapidly-exploring random trees: Progress and prospects. Algorithmic and computational robotics: new directions, (pp. 293–308).

Li, Y., Wei, W., Gao, Y., Wang, D., & Fan, Z. (2020). Pq-rrt*: An improved path planning algorithm for mobile robots. *Expert Systems with Applications*, 113425.

Likhachev, M., Ferguson, D., Gordon, G., Stentz, A., & Thrun, S. (2008). Anytime search in dynamic graphs. *Artificial Intelligence, 172*, 1613–1643.

Maekawa, T., Noda, T., Tamura, S., Ozaki, T., & Machida, K.-I. (2010). Curvature continuous path generation for autonomous vehicle using b-spline curves. *Computer-Aided Design, 42*, 350–359.

Qi, J., Yang, H., & Sun, H. (2020). Mod-rrt*: A sampling-based algorithm for robot path planning in dynamic environment. *IEEE Transactions on Industrial Electronics.*

Qureshi, A. H., & Ayaz, Y. (2016). Potential functions based sampling heuristic for optimal path planning. *Autonomous Robots, 40*, 1079–1093.

Sánchez, G., & Latombe, J.-C. (2002). On delaying collision checking in prm planning: Application to multi-robot coordination. *The International Journal of Robotics Research, 21*, 5–26.

Wang, J., Chi, W., Li, C., Wang, C., & Meng, M. Q.-H. (2020). Neural rrt*: Learning-based optimal path planning. In *IEEE Transactions on Automation Science and Engineering*.