Relevant Region Sampling Strategy with Adaptive Heuristic Estimation for Asymptotically Optimal Motion Planning

Chenming Li, Fei Meng, Han Ma, Jiankun Wang, Member, IEEE, Max Q.-H. Meng, Fellow, IEEE

Abstract—The sampling-based motion planning algorithms can solve the motion planning problem in high-dimensional state space efficiently. This article presents a novel approach to sample in the promising region and reduce planning time remarkably. The RRT# defines the Relevant Region according to the cost-to-come provided by the optimal forward-searching tree; however, it takes the cumulative cost of a direct connection between the current state and the goal state as the cost-to-go. We propose a batch sampling method that samples in the refined Relevant Region, which is defined according to the optimal cost-to-come and the adaptive cost-to-go. In our method, the cost-to-come and the cost-to-go of a specific vertex are estimated by the valid optimal forward-searching tree and the lazy reverse-searching tree, respectively. New samples are generated with a direct sampling method, which can take advantage of the heuristic estimation result. We carry on several simulations in both $SE(2)$ and $SE(3)$ state spaces to validate the effectiveness of our method. Simulation results demonstrate that the proposed algorithm can find a better initial solution and consumes less planning time than related work.

Note to Practitioners—This work is motivated by leveraging the lazy reverse-searching method and the batch sampling method to plan efficiently. The sampling-based planning methods have been widely used in robotics to solve the motion planning problem in high-dimensional state space. However, the traditional sampling-based planning methods often produce initial solutions with critically low quality, and the convergence speeds are too slow to meet the real-time operating requirement. This work utilizes the optimal cost-to-come and the adaptive cost-to-go estimating functions to mitigate these drawbacks. First, a reverse-searching tree is constructed upon the batch of samples lazily. Then, the forward tree will grow in terms of the information provided by the lazy reverse-searching tree. The proposed algorithm also includes the priority queue, global replanning, graph pruning, and edge evaluation postponing techniques. Furthermore, this article uses a direct sampling method to generate samples in the promising region, called the Relevant Region.

This project is supported by Shenzhen Key Laboratory of Robotics Perception and Intelligence (ZDYSYS20200810171800001) and the Hong Kong RGC GRF grants # 14200618 awarded to Max Q.-H. Meng. (Corresponding authors: Jiankun Wang, Max Q.-H. Meng.)

Chenning Li, Fei Meng, and Han Ma are with the Department of Electronic Engineering, The Chinese University of Hong Kong, Shatin, N.T., Hong Kong SAR, China (e-mail: licmju@link.cuhk.edu.hk; feimeng@link.cuhk.edu.hk; hannma@link.cuhk.edu.hk).

Jiankun Wang is with the Department of Electronic and Electrical Engineering of the Southern University of Science and Technology, Shenzhen, China (e-mail: wangjk@sustech.edu.cn).

Max Q.-H. Meng is with the Department of Electronic and Electrical Engineering of the Southern University of Science and Technology, Shenzhen, China, and also with the Shenzhen Research Institute of the Chinese University of Hong Kong in Shenzhen, China (e-mail: max.meng@ieee.org).

Index Terms—Asymptotically optimal sampling-based motion planning, Relevant region, Adaptive heuristic.

I. INTRODUCTION

ROBOT motion planning algorithms are developed to find a collision-free trajectory from the source point to the destination for a robot. Some planning algorithms cannot optimize the found solution, such as the Rapidly-exploring Random Trees (RRT) [1] and the Probabilistic Roadmap [2] algorithm. The asymptotically optimal planning algorithms can continuously optimize the current solution until the solution can meet the requirement or the time is up. The optimization metric can be defined as the length of the path, the measurement of the human comfort level in the human-robot coexisting environment, or any metric defined mathematically. The speed of solving the robot planning problem is the key to real-time robot operation. As a critical factor for application, the planning speed is subject to the speed of finding the initial feasible solution, the initial solution quality, and the speed of convergence.

The planning algorithm mainly contains two categories, the searching-based algorithm and the sampling-based algorithm. The searching-based method has a long history of evolution. However, it naturally suffers from the curse of dimensionality. Performing the deterministic searching is impractical in the high-dimensional planning problem. The sampling-based motion planning algorithm is a widely used approach when solving the high-dimensional robot motion planning problem. Typically, the sampling-based motion planning algorithm can be divided into two stages, the learning stage and the searching stage [3]. In the learning stage, the algorithm will sample randomly in the free space. Then, it uses the sample to construct a searching tree or a roadmap in the searching stage. Edge evaluation will ensure that each edge is executable in the searching stage. But the edge evaluation is not cheap in most scenarios, and it is the bottleneck of the computation speed. Therefore, one of the objectives of this article is to make the sampling more efficient and avoid explicitly evaluating every edge. We only check the valuable candidate edges that have the potential to improve the current solution by the adaptive heuristic estimation. The priority queue is used to achieve this aim.

A representative sampling-based motion planning method called the RRT [1] uses the uniformly sampling strategy to generate the sample and utilizes a tree structure to perform
the space searching. The RRT method takes a sample in the learning stage and connects it to the tree in the searching stage. But the RRT is not an asymptotically optimal planner, and the RRT* is proposed to solve this problem \[4\]. The new vertex of the RRT* will choose its parents in terms of the cost. And the rewire function can refine the current tree structure locally. Instead of rewiring the tree locally, the RRT# \[5\] proposes to use a replanning function to propagate changes in the relevant part of the graph. However, the RRT* and RRT# use the uniformly sampling strategy to produce samples in the whole state space, and it suffers from the low sampling efficiency. In their learning stage, uniformly sampling can provide a topology abstraction of the entire state space, though most planning problems do not need to learn the information of the entire state space. This article proposes to sample in a subset of the whole state space with a direct sampling method. Our method can make the learning stage more efficient, and the algorithm will concentrate on abstracting the most promising region.

The Informed sampling strategy constrains the sampling stage in a subset of the whole state space, which is a \(d\)-dimensional prolate hyper-spheroid in \(d\)-dimensional space \[6\] \[7\]. The Informed sampling strategy is implemented based on the RRT* algorithm. However, it defines the prolate hyper-spheroid in terms of the estimated cost-to-come and cost-to-go, which are both calculated with the Euclidean metric. Therefore, the Informed sampling strategy is not a promising sampling method in complex environments, such as the maze-like environment. To overcome this, we propose to utilize the optimal cost-to-come and the adaptive cost-to-go estimation methods to guide the sampling. Our method takes samples in the region with the highest chance of improving the current solution (both the approximate solution and the exact solution \[8\]), which is closely relevant to the current query and current tree. That is why we call this sampling method as the Relevant Region sampling strategy. In this work, we follow-up the concept of the Relevant Region defined in \[5\] \[9\], and \[10\], but we refined the cost-to-go estimation method, the definition of the priority queue, and the new vertex generation process.

The speed of finding the initial feasible solution is one of the critical points for speeding up the entire planning procedure. The bi-directional search method is an effective strategy to accelerate the initial feasible solution generation speed \[11\]. Instead of explicitly constructing both the forward-searching and reverse-searching trees, we implicitly construct the reverse-searching tree lazily. We store the samples in a Geometric Near-neighbor Access Tree (GNAT) \[12\], a data structure for nearest neighbor access search, instead of a linear data structure. The lazy reverse-searching tree will provide the adaptive cost-to-go estimation for vertices in our priority queue. In addition, the searching direction for a specific vertex is guided by the lazy reverse-searching tree. The schematic of solving a simple path planning problem in a 2D environment is used to describe the workflow of our method since it is easily visualized, as shown in Fig. \[1\].

Simulations are all carried through the benchmark platform of the Open Motion Planning Library (OMPL) \[8\] \[13\]. Simulation environments include both SE(2) and SE(3) state spaces. The related algorithms are configured with their default parameter settings in the OMPL. The parameters of our planner are chosen as the optimal setting for each simulation through parameter sweeping.

In this article, we introduce the related work in Section II. The problem definition is described in Section III. Then, we present the mathematical details and the proposed algorithm in Section IV. Simulations are carried out to validate the effectiveness of the proposed method, and the results are shown in Section V. Finally, we draw our conclusions in Section VI.

II. RELATED WORK

A. Sampling-based Motion Planning Method

Plenty of modifications are proposed to enhance the performance of the RRT algorithm \[1\] such as the RRT* algorithm \[4\]. The rewiring stage of the RRT* only rewires locally, which means the global optimization of the current tree is ignored. The RRT# \[5\] proposes to find the global optimality in each rewiring stage with dynamic programming. Dynamic programming is also used in the Fast Marching Tree (FMT*) method \[14\] to grow the searching tree.

The Informed sampling strategy \[6\] is proposed to overcome the drawback of uniform sampling. It can accelerate the convergence speed significantly with very little computation consumption. The Informed sampling strategy uses a direct sampling method to generate samples in the \(L_2\)-Informed set. An advanced version of the Informed sampling strategy is proposed in \[7\], which includes the graph pruning stage to keep a relatively constricted tree. A tree with fewer vertices means that the cost reduction in finding the nearest tree vertex. Using the neural network to reinforce the learning stage to

![Fig. 1. The schematic for the proposed method in a simple 2D environment, where the black blocks, the orange point, solid green lines, and the dashed grey lines demonstrate the obstacles, the sampling points, edges in the forward-searching tree, and edges in the reverse-searching trees, respectively. Fig. (a) shows that our method sample a batch of points in the current iteration, Fig. (b) shows the lazy reverse-searching stage while Fig. (c) shows the forward-searching stage. And Fig. (d) shows the current lazy reverse-searching tree is discarded, and a new batch of samples is added in the next iteration.](image-url)
enhance the sampling efficiency [15] [16] [17] is proved as a promising technique.

In the human-robot coexisting environment, the planning problem becomes more complex than that in the static environment [18]. The objective of optimization needs to consider safety, efficiency, and human feelings.

B. Batch Sampling Strategy

The FMT* [14] introduces the thought of batch sampling into the robot motion planning field. The FMT* samples a batch of points and constructs the searching tree on this batch of samples. The asymptotic optimal of the FMT* is guaranteed when the size of the batch goes to infinity. The Batch Informed Trees (BIT*) [19] [20] method is developed based on the Informed RRT*, besides, the BIT* absorbs the thoughts in the FMT* method [14] and the Lifelong Planning A* (LPA*) algorithm [21]. The Regionally Accelerated Batch Informed Trees (RABIT*) [22] aims to solve the difficult-to-sample planning problem, like the narrow passage problem. The RABIT* uses the Covariant Hamiltonian Optimization for Motion Planning (CHOMP) method as its local optimizer, and the local optimizer will exploit the local information. The Fast-BIT* [23] modifies the edge queue and searches the initial solution more aggressively. The Greedy BIT* [24] uses the greedy searching method to generate the initial solution faster and accelerate the convergence speed. But these greedy-based methods often fail to assist the searching procedure without an accurate heuristic estimation method. The Adaptively Informed Trees (AIT*) [25] and the Advanced BIT* (ABIT*) [26] proposed by Strub and Gammell are developed based on the BIT* as well. The AIT* calculate a relatively accurate heuristic estimation with a lazy reverse-searching tree. The ABIT* proposes to utilize inflation and truncation to balance the exploitation and exploration in the increasingly complex Random Geometric Graph (RGG) [27]. Though the AIT* and the ABIT* achieve significant improvements, their sampling regions are not compact enough, and the sampling efficiency will be critically low in the complex environment.

C. Relevant Region Sampling Strategy

The concept of ‘relevant’ is first proposed in the searching-based robot path planning method like the A* [28]. In the A* algorithm, the set of expanded vertices is the whole space. The Dijkstra’s algorithm could expand a smaller set of vertices than the Dijkstras’s algorithm [29]. And it is also not a new idea in the sampling-based planning field. The Relevant Region is formally defined in [5]. The Relevant Region related vertices are the vertices of which the sum of the optimal cost-to-come and the heuristic is less than the cost of the current optimal solution. Since the Relevant Region is the most promising region that could help to improve the solution, so a straightforward modification is to reduce the chance of sampling outside the Relevant Region. Three different metrics are used to achieve this in [9], the modified versions achieve better performance in the convergence speed than the RRT# method. The methods described in [5] and [9] use the rejection method for sampling, which is not efficient since the Relevant Region is a small subset of the whole state space in most scenarios. The direct sampling method is illustrated to overcome this drawback, and the details are described in [10]. But they all use the cumulative cost along the direct connection between current state and the goal state as the cost-to-go, resulting in the inaccurate estimated cost-to-go in most scenarios. Their ordered priority queues are also far from the ground truth.

D. Bi-directional Searching Method

The RRT and the RRT* methods often fail to find the solution in the required period, especially in the narrow passage problem. The RRT-Connect [11] is proposed to find the initial solution faster. It grows two trees from the source point and the goal region simultaneously. It is proved that the RRT-Connect can achieve better performance than the RRT. But the method described in [11] is not an asymptotically optimal method, so the improved version of bi-directional searching RRT is proposed in [30]. The method described in [30] is an asymptotically optimal single-query version of the RRT-Connect, called the RRT*-Connect. The RRT*-Connect provides asymptotically optimal guarantee like the RRT*, and its efficiency and robustness are proved in real-world experiments. In addition, the bi-directional searching method can be used to combine with the kinematic constraints [31], which is essential in generating executable trajectory.

One drawback of the Informed RRT* [6] [7] is that it uses the RRT* to search the whole state space before finding the initial solution. Therefore, the Informed RRT* often fails to find the solution in the required period, same as the RRT*. By combining the advantages of both the Informed and the RRT*-Connect, the Informed RRT*-Connect proposes to use the RRT*-Connect to generate the initial solution and use the Informed sampling strategy to constrain the sampling region after the initial solution is found. It combines the advantages of both the Informed RRT* and the RRT*-Connect. The Informed RRT*-Connect can achieve a much higher success rate in its simulations than the Informed RRT*. Besides, the AIT* [25] can also be viewed as a bi-directional searching method.

III. Problem Definition

Consider the state space $\mathcal{X}$, which is the subset of $\mathbb{R}^d$. $\mathbb{R}^d$ is the whole $d$-dimensional space, and $d$ is a positive integer. $\mathcal{X}_{\text{obs}}$ shows the space occupied by the obstacles, the free space is defined as $\mathcal{X}_{\text{free}} = \mathcal{X} \setminus \mathcal{X}_{\text{obs}}$. The $x \in \mathcal{X}$ represents any state in the state space. The source point $x_{\text{start}}$ is the initial state of the robot. The destination is a region represented by $\mathcal{X}_{\text{goal}}$. The source point and the destination in a valid planning problem must be defined within the free space $x_{\text{start}} \in \mathcal{X}_{\text{free}} \land \mathcal{X}_{\text{goal}} \subseteq \mathcal{X}_{\text{free}}$. The motion planning problem is defined as:

$$\pi \in [0, 1] \rightarrow \mathcal{X}_{\text{free}},$$

subject to $x_{\text{start}} \equiv x_{\text{start}}, \pi(0) \in \mathcal{X}_{\text{goal}}, \pi(s) \in \mathcal{X}_{\text{free}}, \forall s \in [0, 1].$ (1)

The optimization objective can be minimizing the trajectory length, maximizing the minimum clearance, or any objects
that could be mathematically defined. It can also be set as the sum of individual optimization objectives and form a hybrid optimization objective. For simplicity, the optimization objective in the proposed method is set as minimizing the trajectory length. Note that it could be extended to meet the specific planning requirement. The $v \in T$ donates any vertex in the tree. Assume there are two vertices $v_1, v_2 \in T$, where $v_2$ is the descendant of $v_1$. The cost between any two vertices $v_1, v_2$ is calculated with the integral cost along the tree from $v_1$ to $v_2$, donates as $d_T(v_1, v_2)$. Let $\Pi$ denote the set of all feasible solutions. With the definition of trajectory cost, the optimization objective in our method is written as:

$$
\pi^* = \arg \min_{\pi \in \Pi} d_T(v_{\text{start}}, v_{\text{goal}}) \\
\text{s.t. } \pi(0) = v_{\text{start}}, \pi(1) \in X_{\text{goal}}, \pi(s) \in X_{\text{free}}, \forall s \in [0, 1].
$$

(2)

The cost estimation between any two states $x_1, x_2 \in X$ can be set as the cumulative cost along the direct connection. The cost for the unit distance is usually deemed as 1, the heuristic can be calculated with the Euclidean metric: $h(x_1, x_2) = \|x_2 - x_1\|_2$. However, using this metric as the heuristic estimation method between any two states is not promising. The direct connection is highly likely to collide with the obstacle, which can mislead the searching procedure, especially in complex environments.

The RRT algorithm contains two stages: the learning and the searching stage. The learning stage can be viewed as the abstracting process of the state space $X$, and the searching stage will construct the searching tree $T$ based on this abstracted state space. These two phases are performed alternatively in each iteration. We use the $c_{\text{cur}}$ to represent the cost of the current optimal solution. Before finding the initial feasible solution, the $c_{\text{cur}}$ is set to an infinitely large number $+\infty$.

IV. METHODOLOGY

A. Adaptive Heuristic Estimation

We use the optimal forward-searching tree $T_F$ to provide the cost-to-come estimation $h_{T_F}(v)$ of any vertex $v \in T_F$. With the optimization objective, the $h_{T_F}(v) = d_{T_F}(v_{\text{start}}, v)$ is defined as the cumulative cost from the initial state $x_{\text{start}} = v_{\text{start}} \in T_F$ to the vertex $v$ via the current tree $T_F$.

Since calculating the heuristic along the direct connection is inaccurate, we use a lazy reverse-searching tree $T_R$ to provide the cost-to-go estimation $h_{T_R}(x)$ of any state $x$ in the state space. The $T_R$ is constructed without edge evaluation because edge evaluation is the most time-consuming procedure in the majority of motion planning scenarios. To construct the $T_R$, we separate the learning stage and searching stage into two separate modules instead of performing them alternatively in each iteration. In our learning stage, we use our sampling strategy to generate a batch of sampling points in $X_{\text{free}}$. Then, in our searching stage, we construct the $T_R$ and $T_F$ in terms of the current RGG. A conceptual illustration is provided in Fig. 1.

Since the $T_R$ is constructed without edge evaluation, the edges in the $T_R$ may collide with the obstacles or not satisfy the constraints, as the Fig. 1(b) and Fig. 1(c) shows, where an edge in the $T_R$ collides with the obstacle. But in this method, we only use the $T_R$ to provide a relatively accurate cost-to-go estimation and guidance for sampling, which means we do not need edges in the $T_R$ to be executable.

B. Relevant Region with Adaptive Heuristic Estimation

The Relevant Region is defined as a subset of $X_{\text{free}}$ of which the cardinality is smaller than the Informed sampling set. The Relevant Region sampling strategy uses the optimal forward-searching tree $T_F$ to provide the optimal cost-to-come estimation. And using the lazy reverse-searching tree $T_R$ to estimate the cost-to-go of the state $x$ in the RGG, the estimated result can be represented as $h_{T_R}(x) = d_{T_R}(x, x_{\text{goal}})$. With the $h_{T_F}$ and $h_{T_R}$, the definition of relevant vertex is shown in (3), where the $V$ is the vertices set of the optimal forward-searching tree $T_F$ and $v \in V$ is any vertex belongs to the $T_F$.

$$
V_{\text{rel}} = \{ v \in V | h_{T_F}(v) + h_{T_R}(v) < c_{\text{cur}} \}.
$$

We define the ball centered at the vertex $v$ with radius $\epsilon$ as:

$$
B'(v) = \{ x \in X | \|x - v\|_2 < \epsilon, v \in V_{\text{rel}} \}.
$$

(4)

With the $h_{T_F}(v)$ and the $h_{T_R}(x)$, the estimated solution cost that pass by the state $x \in B'(v)$ can be written as $\hat{f}_v(x) = h_{T_F}(v) + d(v, x) + h_{T_R}(x)$, where the $d(v, x)$ denotes cost from vertex $v$ to state $x$. The Relevant Region with the adaptive heuristic estimation of the current forward-searching tree is defined as (5).

$$
X'_{\text{rel}} = \{ x \mid x \in B'(v), \hat{f}_v(x) < c_{\text{cur}} \}.
$$

(5)

C. Relevant Region Sampling Strategy

The ultimate optimization objective is shown in (2). To meet the optimization requirement with fewer sampling points, we propose a novel method to generate samples in the most promising region. Let the new sample be $x_{\text{new}} \in X_{\text{free}}$, and it is related to the relevant point $v_{\text{rel}} \in V_{\text{rel}}$, where $v_{\text{rel}}$ is chosen from the queue $Q$ with the highest priority. The $Q$ is defined in terms of (10). The relationship between $x_{\text{new}}$ and $v_{\text{rel}}$ is $x_{\text{new}} = v_{\text{rel}} + \gamma e$, where $\{ e \in E \mid e \in \mathbb{R}^d, \|e\|_2 = 1 \}$ donates the direction for expanding which points from $v_{\text{rel}}$ to $x_{\text{new}}$, and $\gamma$ is a positive number which indicates the maximum promising cost magnitude to travel along the direction $e$, bounded by $[0, \gamma_{\text{max}}]$. The sampling direction $e$ is generated by utilizing the information provided by the $T_R$, which is the direction along the edge of the $T_R$ pointing to the goal. Then we will use the solution of the following optimization problem to guide the sampling.

$$
\max \gamma, \\
\text{subject to: } \hat{f}_v(x) < c_{\text{cur}}, \\
\gamma \in [0, \gamma_{\text{max}}].
$$

(6)

The inequality in (6) equals to $h_{T_F}(v) + d(v, x) + h_{T_R}(x) < c_{\text{cur}}$. Then we will have:
\[ h_{\mathcal{T}_R}(x) < c_{\text{cur}} - h_{\mathcal{T}_F}(v) - \gamma. \] (7)

Note that the right-hand side of the (7) is always positive, we can obtain another inequality \( \gamma < c_{\text{cur}} - h_{\mathcal{T}_F}(v) \). If the right-hand side \( c_{\text{cur}} - h_{\mathcal{T}_F}(v) \) is smaller than 0, the outgoing edges connecting to \( v \) and its children will be erased from the forward-searching tree \( \mathcal{T}_F \) iteratively.

By moving the \( d(v, x) = \gamma \) in (6) to the left-hand side, we can get:

\[ \gamma < c_{\text{cur}} - h_{\mathcal{T}_F}(v) - h_{\mathcal{T}_R}(x). \] (8)

So the (8) is the solution for the best cost magnitude to travel along direction \( e \). After giving the \( \gamma \) an upper bound \( \gamma_{\text{max}} \) for expanding, the final solution is:

\[ \gamma_{\text{rel}} = \min(c_{\text{cur}} - h_{\mathcal{T}_F}(v) - h_{\mathcal{T}_R}(x), \gamma_{\text{max}}). \] (9)

The calculated step size \( \gamma_{\text{rel}} \) may be a negative number since the \( h_{\mathcal{T}_F}(v) \) and \( h_{\mathcal{T}_R}(x) \) always overestimate the cost. In that case, the result will be invalid.

In the implementation, each vertex will be used to calculate random sampling points multiple times with different random distortions in each iteration. We add the random distortions to both the \( e \) and \( \gamma \) because edges in the \( \mathcal{T}_R \) are not valid under the constraints, and direct search along \( e \) is not promising. The searching direction and the expanding distance with random distortion are donated as \( \hat{e} \) and \( \hat{\gamma} \), respectively, and the new sample is \( \hat{x}_{\text{new}} = \psi_{\text{rel}} + \hat{\gamma}\hat{e} \). We illustrate this in Fig. 2. The pale green points circled by the dotted lines in the right-hand side of Fig. 2 indicate the newly added sampling points with our direct sampling method in the current batch.

When processing the learning stage, we sample a batch of points with the proposed sampling strategy. In the searching stage, we perform the lazy reverse-searching and construct the \( \mathcal{T}_R \) implicitly, then expand the \( \mathcal{T}_F \) in terms of the adaptively heuristic estimation provided by the \( \mathcal{T}_R \). The replanning function will guarantee that all promising vertices in \( \mathcal{T}_F \) are optimal under the current topology abstraction. In the next iteration, samples in the previous batch are reused as long as they satisfy the inequation described in (8). The \( \mathcal{T}_R \) will be disposed of by the end of each iteration, we re-construct the \( \mathcal{T}_R \) from the sketch instead of rewiring it in the next iteration, this is due to the concern of the time efficiency. The procedure of constructing the \( \mathcal{T}_R \) can be viewed as a simplified version of the FMT* algorithm without the edge evaluation. Based on the proposed ideas, we designed the basic workflow of our method, as shown in the Algorithm 1.

Algorithm 1 Relevant Sampling Strategy with Adaptive Heuristic Estimation for Asymptotically Optimal Motion Planning

\begin{algorithm}
\caption{Relevant Sampling Strategy with Adaptive Heuristic Estimation for Asymptotically Optimal Motion Planning}
\begin{algorithmic}
\State \textbf{Input:} \texttt{x}^{\text{start}} \in \mathcal{X}_{\text{free}}, \mathcal{X}_{\text{goal}} \subseteq \mathcal{X}_{\text{free}} ;
\State \textbf{Output:} \mathcal{T}_F ;
\State 1: \texttt{Init}(\mathcal{T}_F, \mathcal{T}_R, \mathcal{V}_{\text{sol}});
\While{\texttt{not} \texttt{TerminateCondition}()}
\State 2: \texttt{Sample}(m, \mathcal{P}_{\text{inf}}, \mathcal{T}_F);
\State 3: \texttt{Prune}(\mathcal{X}_{\text{new}}, \mathcal{T}_F, \mathcal{T}_R);
\State 4: \texttt{Sample}(m, \mathcal{P}_{\text{inf}}, \mathcal{T}_F);
\State 5: \texttt{BuildForwardTree}(\mathcal{T}_F, \mathcal{T}_R);
\State 6: \texttt{BuildReverseTree}(\mathcal{T}_F, \mathcal{T}_R);
\EndWhile
\State 7: \texttt{return} \mathcal{T}_F ;
\end{algorithmic}
\end{algorithm}

In the Line 3 of Algorithm 1 we include the graph pruning method to keep the cardinalities of forward and reverse trees as small as possible. Graph pruning can enhance query efficiency. If a sample is not connected to both the \( \mathcal{T}_F \) and the \( \mathcal{T}_R \) in the previous iteration or the sample is outside the promising region, we will prune it.

The details of the sampling method in Line 2 in Algorithm 1 is illustrated in Algorithm 2. The \texttt{m} represents the batch size of the \( \mathcal{P}_{\text{inf}} \) and the \texttt{p} shows the probability we sample in the \texttt{Informed} space. The \( h_{\mathcal{T}_F}(v) \) and \( h_{\mathcal{T}_R}(x) \) always over-estimate the cost, which means the Relevant Region with adaptive heuristic estimation itself can not guarantee the probabilistic completeness. Therefore, the samples in our method are generated in mixed mode. We use the direct sampling method to sample in the \texttt{Informed} space with the probability \( \mathcal{P}_{\text{inf}} \). The function \texttt{WeightVertices} (\( \mathcal{T}_F \)) will weight every vertex \( v \) in the \( \mathcal{T}_F \) and put the weighted vertices into a priority queue \( \mathcal{Q} \). The weight \( \mathcal{W} \) of a particular vertex is designed according to its selected times \( n_v \), outgoing edges \( n_e \), and adaptively heuristic. The weighing method is shown in (10), where the \( \lambda_1 \), \( \lambda_2 \), and \( \lambda_3 \) modulate the behavior of our weighting function [10]. And the adaptively heuristic is normalized by the current optimal solution. When generating the new samples, the construction of \( \mathcal{T}_R \) upon the current batch is not yet started. Therefore, the estimated cost-to-go of state in \( \mathcal{X}_{\text{S-curve}} \) comes from the \( \mathcal{T}_R \) of the last iteration, the newly added sample uses the optimal

---

**Fig. 2.** We add randomness to the \( e \) and \( \gamma \) since the lazy reverse-searching method can not guarantee edges in the \( \mathcal{T}_R \) are valid. In this figure, the black blocks, solid green lines, dashed blue lines, and orange points represent the obstacles, forward tree, reverse tree, and the points in the current RGG, respectively. The right part of the figure is the local zoom of the left part. The purple circular sector shows the region where we may take samples in terms of the vertex \( \psi_{\text{rel}} \). The region with darker purple has a higher probability of being sampled.

---

**D. Proposed Algorithm**

The planning problem is defined by the state space \( \mathcal{X} \), together with the start point \( x^{\text{start}} \) and goal region \( \mathcal{X}_{\text{goal}} \). The \( \mathcal{T}_F \), \( \mathcal{T}_R \), and \( \mathcal{V}_{\text{sol}} \) are initialized as the \( \emptyset \). Our planner will try to find the optimal solution to the problem described in (2).

In the proposed method, we separate the learning stage and the searching stage into two explicitly different procedures instead of performing them alternatively in each iteration.
value of \( h_{\text{agg}}(x_{\text{near}}) + d(x, x_{\text{near}}) \), where \( h_{\text{agg}}(x_{\text{near}}) \) is the cost-to-go of its neighbors. All samples in \( \mathcal{X}_S \) are stored in the GNAT.

**Algorithm 2** Sample\((m, P_{\text{Inf}}, T_F)\)

1. \( \mathcal{X}_{S_{\text{new}}} \leftarrow \emptyset; \)
2. \( Q = \text{WeightVertices}(T_F); \)
3. \( \mathcal{X}_{S_{\text{new}}} \leftarrow \mathcal{X}_{S_{\text{new}}} \cup \text{SampleRel}(Q); \)
4. \( \mathcal{X}_{S_{\text{new}}} \leftarrow \mathcal{X}_{S_{\text{new}}} \cup \text{SampleInf}(m - \mathcal{X}_{S_{\text{new}}}.\text{size}()); \)
5. return \( \mathcal{X}_{S_{\text{new}}}; \)

\[ W = \lambda_1 n_s + \lambda_2 n_o + \lambda_3 \frac{h_{\text{agg}}(v) + h_{\text{agg}}(v)}{c_{\text{cur}}}. \tag{10} \]

The function \( \text{SampleRel}(Q) \) in Line.\[ Algorithm.2 \] is the method we use to generate samples with the proposed Relevant Region sampling strategy. Details are shown in Algorithm.\[ Algorithm.3 \] The variable \( \text{ExpandTimes} \) shows the number of samples generated in terms of vertex \( v_{\text{elem}} \) with different distortions. The function \( \text{SampleInf}() \) will take \( m - \mathcal{X}_{S_{\text{new}}}.\text{size}() \) samples in the Informed space \[\mathcal{F}].\]

**Algorithm 3** SampleRel\((Q)\)

1. \( \text{ExpandTimes} = \text{int}((1 - P_{\text{Inf}})m/Q.\text{size}()); \)
2. for \( v_{\text{elem}} \) in \( Q \) do
3. \( \gamma, e = \text{EstimateStepSizeAndDirection}(v_{\text{elem}}); \)
4. for iter in iter = 1, 2, 3, \ldots, \text{ExpandTimes} do
5. \( \hat{\gamma}, \hat{e} = \text{GaussianNoise}(\gamma, e); \)
6. \( \mathcal{X}_{S_{\text{new}}}.\text{append}(v_{\text{elem}} + \hat{\gamma}\hat{e}); \)
7. end for
8. end for
9. return \( \mathcal{X}_{S_{\text{new}}}; \)

We utilize the idea of avoiding evaluating every edge in the forward-searching, which comes from the BIT* algorithm \[\cite{20}.\] We illustrate the details of the function \( \text{BuildForwardTree}(T_F, T_R) \) (Line.\[ Algorithm.1 \] Algorithm.\[ Algorithm.7 \] in Algorithm.\[ Algorithm.4 \]) The vertices in the \( T_F \) will be put in an ordered queue \( Q_V \), the \( Q_V \) is sorted in terms of \( f(v) = h_{\text{agg}}(v) + h_{\text{agg}}(v) \). The \( Q_E \) is an ordered queue for the promising edges, which is sorted in terms of \( h_{\text{agg}}(v) + d(v, \hat{x}) + h_{\text{agg}}(\hat{x}) \). Function \( \hat{c}(v_{\text{min}}, x_{\text{min}}) \) and \( c(v_{\text{min}}, x_{\text{min}}) \) calculate the estimated cost and the actual cost between vertex \( v_{\text{min}} \) and state \( x_{\text{min}} \). The function \( \text{Replan}(T_F) \) guarantees the \( T_F \) is optimal under current space abstraction \[\cite{5}].

**V. SIMULATIONS**

The simulations are all carried through the benchmark platform of the OMPL \[\cite{8} \cite{13}.\] To validate the generalization ability of our method, we solve the planning problem in both the \( SE(2) \) and \( SE(3) \) state spaces with our method and several state-of-art algorithms. All the state spaces of simulation environments are continuous spaces, and all tested algorithms take samples in these continuous state spaces without space discretization. The robot is defined as a collection of convex polyhedrons and occupies a certain volume.

To provide a further explanation of our method, we use the RRT\# \[\cite{5} \], the AIT* \[\cite{25} \], and our method to solve the motion planning problem in an OMPL benchmark environment called the ‘BugTrap’. The state space of the ‘BugTrap’ environment is the \( SE(2) \) state space, which is composed of two floating numbers \( x, y \) indicate the position and one floating number \( w \) shows the orientation; The planning procedures of the RRT\# \[\cite{5} \], the AIT* \[\cite{25} \], and our method are illustrated in Fig.\[ Fig.3 \] in Fig.\[ Fig.3 \] obstacles, the free space, the start state, the goal region, and vertices are indicated with black, ivory white, pale blue, wine, and orange color, respectively. We use the dark green lines and violet lines to show the forward tree and the current optimal solution, respectively. The reverse trees are shown as the grey lines in the figure of the AIT* \[\cite{25} \] and our method.

In the simulation shown in Fig.\[ Fig.3 \] the planning problem contains two optimal solutions, one is to pass through the region upper the obstacle, and the other one is to pass through the lower part. Fig.\[ Fig.3 \] shows that all the methods in Fig.\[ Fig.3 \] can acquire the global asymptotic optimal. Both the AIT* and our method use the lazy reverse-searching tree to guide the sampling and have the graph pruning method to constraint the RGG in the Informed Region and Relevant Region, respectively. From the (j)-(m) in Fig.\[ Fig.3 \] it can find that both the forward and reverse trees of our method are optimal under current state space abstraction. In addition, our method concentrates on taking samples in the region with a higher potential to improve the current solution, which can be seen in (m) of Fig.\[ Fig.3 \] our method pays more attention to the turning points with our direct sampling method.
The RRT# algorithm.

The AIT* algorithm.

Our method.

Fig. 3. Planning procedures of the RRT# algorithm, the AIT* algorithm, and our method. The planning problem is set in the 'BugTrap' environment provided by the OMPL benchmark platform. We choose its start state and goal region to make there exists two different optimal solutions, which pass through different zones of the space and have the same solution cost. The optimization objective is set as minimizing the path length. It can be seen that our method output a better path than the other two algorithms.

B. Simulations in $SE(2)$ State Space

We choose the $SE(2)$ environments shown in Fig. 5 to verify our method, and they are called the 'Bug Trap', the 'BarrierEasy', the 'Maze', and the 'RandomPolygons' in the OMPL benchmark platform. To give the reader an intuitional understanding of our 2D planning simulations, we show the trajectories found by our method in Fig. 6. The trajectories are interpolated in terms of time.

In our 2D simulations, the state space definition contains the position $x, y$ and orientation $w$. To manifest the superiority of our method, we compared with seven different state-of-art algorithms, they are the RRT*, the BIT*, the AIT*, the ABIT*, the Informed RRT*, the RRT#, and the Informed+Relevant sampling method proposed in [10]. In these simulations, we use the trajectory length as the cost metric. The optimization objective is set as $0.92 \times c_{opt}$, where the $c_{opt}$ is the cost of the optimal solution. The $c_{opt}$ is the solution cost of the RRT* method after 300 seconds’ execution. All the state-of-art planners are configured with their default parameter sets. To reduce the randomness, each planner runs 100 times in each environment.

The simulation results in the $SE(2)$ state spaces are shown in Fig. 4. The left parts of the pictures are the time each planner spending to generate the required trajectory. The right parts of the pictures show the cost distribution in terms of time. We start drawing the line charts when 50 percent of all runs find the solution and stop drawing when 95 percent have finished the problem-solving. In addition, we provide the error bars for all bar charts and line charts.
Fig. 4. (a), (b), and (c) show the 2D simulation result in 'BugTrap', the 'Maze', and the 'RandomPolygons' environments, where the left pictures are the time each planner spent to meet the optimization objective and the right pictures are the cost variations over time. Planners try to meet the optimization objective, dashed lines in the right pictures show the cost value of the optimization objective.

Fig. 5. The 2D simulation environments. They are the 'BugTrap', the 'Maze', and the 'RandomPolygons' in the OMPL benchmark platform, respectively. The red cuboid shows the start state and the state in the goal region.

Fig. 6. The trajectories found by our planner in the 'BugTrap', the 'Maze', and the 'RandomPolygons' in the OMPL benchmark platform.

C. Simulations in $SE(3)$ State Space

Besides the 2D simulation introduced previously, we also carried on the simulation in the $SE(3)$ state space. In the 3D simulations, we include the '3D_Apartment' planning problem from the OMPL benchmark platform [13], which is a 'piano movers' problem, as the left environment in Fig. 8 shows. The other simulation is set as a planning problem in 3D narrow passage environment. In the 3D simulation, planners and their parameter sets are the same as the planners we choose in 2D simulations. Each planner will solve each planning problem 100 times. The 3D simulation results are shown in Fig. 7.

VI. CONCLUSIONS

In this paper, a novel asymptotic optimal planning algorithm is proposed. The core of the proposed method is the Relevant Region sampling strategy. Besides, some other techniques are included to improve the performance further. We propose to use the lazy reverse-searching tree to guide the sampling procedure, and a 'replanning' method is used to guarantee the optimality of the forward-searching tree under current topology abstraction. In addition, the new sampling points are generated with our direct sampling method. Simulation results in $SE(2)$ and $SE(3)$ spaces show that our method can achieve a better initial solution and faster planning speed than other work.
Fig. 7. (a) and (b) show the 2D simulation result in ‘BugTrap’, the ‘Maze’, and the ‘RandomPolygons’ environments, where the left pictures are the time each planner spent to meet the optimization objective and the right pictures are the cost variations over time. Planners try to meet the optimization objective, dashed lines in the right pictures show the cost value of the optimization objective.

Fig. 8. The 3D simulation environments.

In the future, we will try to take more advantages of the reverse-searching method to accelerate the planning speed further. Although we try to simplify the reverse-searching procedure, we find the reverse-searching stage consumes the majority of computation resource, and only using it to provide the adaptive cost-to-go estimation is a kind of waste.

REFERENCES

[1] S. M. LaValle et al., “Rapidly-exploring random trees: A new tool for path planning,” 1998.
[2] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” IEEE transactions on Robotics and Automation, vol. 12, no. 4, pp. 566–580, 1996.
[3] W. Khaksar, K. S. M. Sahari, and T. S. Hong, “Application of sampling-based motion planning algorithms in autonomous vehicle navigation,” Autonomous Vehicle, vol. 735, 2016.
[4] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” The international journal of robotics research, vol. 30, no. 7, pp. 846–894, 2011.
[5] O. Arslan and P. Tsiotras, “Use of relaxation methods in sampling-based algorithms for optimal motion planning,” in 2013 IEEE International Conference on Robotics and Automation. IEEE, 2013, pp. 2421–2428.
[6] J. D. Gammell, S. S. Srinivasa, and 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. IEEE, 2014, pp. 2997–3004.
[7] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Informed sampling for asymptotically optimal path planning,” IEEE Transactions on Robotics, vol. 34, no. 4, pp. 966–984, 2018.
[8] I. A. Sucan, M. Moll, and L. E. Kavraki, “The open motion planning library,” IEEE Robotics & Automation Magazine, vol. 19, no. 4, pp. 72–82, 2012.
[9] O. Arslan and P. Tsiotras, “Dynamic programming guided exploration for sampling-based motion planning algorithms,” in 2015 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2015, pp. 4819–4826.
[10] S. S. Joshi and P. Tsiotras, “Relevant region exploration on general cost-maps for sampling-based motion planning,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 6689–6695.
[11] J. J. Kuffner and S. M. LaValle, “Rrt-connect: An efficient approach to single-query path planning,” in Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065), vol. 2. IEEE, 2000, pp. 995–1001.
[12] S. Brin, “Near neighbor search in large metric spaces,” in 21th International Conference on Very Large Data Bases (VLDB 1995), 1995.
[13] M. Moll, I. A. Sucan, and L. E. Kavraki, “Benchmarking motion planning algorithms: An extensible infrastructure for analysis and visualization,” IEEE Robotics & Automation Magazine, vol. 22, no. 3, pp. 96–102, 2015.
[14] L. Janson, E. Schmerling, A. Clark, and M. Pavone, “Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions,” The International Journal of Robotics Research, vol. 34, no. 7, pp. 883–921, 2015.
[15] J. Wang, W. Chi, C. Li, C. Wang, and M. Q.-H. Meng, “Neural rrt*: Learning-based optimal path planning,” IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 1748–1758, 2020.
[16] Z. Li, J. Wang, and M. Q.-H. Meng, “Efficient heuristic generation for robot path planning with recurrent generative model,” in 2021 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2021, pp. 7386–7392.
[17] A. H. Qureshi, A. Simeonov, M. J. Bency, and M. C. Yip, “Motion planning networks,” in 2019 International Conference on Robotics and Automation (ICRA). IEEE, 2019, pp. 2118–2124.
[18] J. Wang, M. Q.-H. Meng, and O. Khatib, “Eb-rrt: Optimal motion planning for mobile robots,” IEEE Transactions on Automation Science and Engineering, vol. 17, no. 4, pp. 2063–2073, 2020.
[19] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, “Batch informed trees (bit*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs,” in 2015 IEEE international
conference on robotics and automation (ICRA). IEEE, 2015, pp. 3067–3074.

[20] J. D. Gammell, T. D. Barfoot, and S. S. Srinivasa, “Batch informed trees (bit*): Informed asymptotically optimal anytime search,” The International Journal of Robotics Research, vol. 39, no. 5, pp. 543–567, 2020.

[21] S. Koenig, M. Likhachev, and D. Furcy, “Lifelong planning a*,” Artificial Intelligence, vol. 155, no. 1-2, pp. 93–146, 2004.

[22] S. Choudhury, J. D. Gammell, T. D. Barfoot, S. S. Srinivasa, and S. Scherer, “Regionally accelerated batch informed trees (rabbit*): A framework to integrate local information into optimal path planning,” in 2016 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2016, pp. 4207–4214.

[23] A. C. Holston, D.-H. Kim, and J.-H. Kim, “Fast-bit*: Modified heuristic for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs,” in 2017 IEEE International Conference on Robotics and Biomimetics (ROBIO). IEEE, 2017, pp. 1892–1899.

[24] L. Chen, L. Yu, S. Libin, and Z. Jiwen, “Greedy bit*(gbit*): Greedy search policy for sampling-based optimal planning with a faster initial solution and convergence,” in 2021 International Conference on Computer, Control and Robotics (ICCCR). IEEE, 2021, pp. 30–36.

[25] M. P. Strub and J. D. Gammell, “Adaptively informed trees (ait*): Fast asymptotically optimal path planning through adaptive heuristics,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 3191–3198.

[26] Strub, Marlin P and Gammell, Jonathan D, “Advanced bit*(abit*): Sampling-based planning with advanced graph-search techniques,” in 2020 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2020, pp. 130–136.

[27] M. Penrose et al., Random geometric graphs. Oxford university press, 2003, vol. 5.

[28] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” IEEE transactions on Systems Science and Cybernetics, vol. 4, no. 2, pp. 100–107, 1968.

[29] E. W. Dijkstra et al., “A note on two problems in connexion with graphs,” Numerische mathematik, vol. 1, no. 1, pp. 269–271, 1959.

[30] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, “Rrt*-connect: Faster, asymptotically optimal motion planning,” in 2015 IEEE international conference on robotics and biomimetics (ROBIO). IEEE, 2015, pp. 1670–1677.

[31] J. Wang, B. Li, and M. Q.-H. Meng, “Kinematic constrained bidirectional rrt with efficient branch pruning for robot path planning,” Expert Systems with Applications, vol. 170, p. 114541, 2021.