Sparse Methods for Efficient Asymptotically Optimal Kinodynamic Planning

Yanbo Li[⋆], Zakary Littlefield[⋆⋆], and Kostas E. Bekris
Rutgers University, New Jersey, USA,
`yanbo.li@cerner.com`,`{zwl2,kostas.bekris}@cs.rutgers.edu`

Abstract. This work describes `STABLE SPARSE RRT` (`SST`), an algorithm that (a) provably provides asymptotic (near-)optimality for kinodynamic planning without access to a steering function, (b) maintains only a sparse set of samples, (c) converges fast to high-quality paths and (d) achieves competitive running time to `RRT`, which provides only probabilistic completeness. `SST` addresses the limitation of `RRT`*, which requires a steering function for asymptotic optimality. This issue has motivated recent variations of `RRT`*, which either work for a limiting set of systems or exhibit increased computational cost. This paper provides formal arguments for the properties of the proposed algorithm. To the best of the authors' knowledge, this is the first sparse data structure that provides such desirable guarantees for a wide set of systems under a reasonable set of assumptions. Simulations for a variety of benchmarks, including physically simulated ones, confirm the argued properties of the approach.

1 Introduction & Background

Sampling-based motion planners can quickly provide feasible motions for many system types. Tree-based methods, such as `RRT` [16], `EST` [9] and variants [5, 24–26, 29, 30] exhibit good performance in terms of feasibility and have been used to optimize paths over costmaps [11]. Nevertheless, `RRT` converges to suboptimal solutions almost surely [14, 22]. This motivated the development of `RRT`*, which achieves asymptotic optimality, given access to a steering function [14]. A steering function optimally connects two states ignoring obstacles while satisfying motion constraints. Due to `RRT`*'s desirable properties, many efforts focused on applying it in the kinodynamic domain by developing steering functions for specific systems [12] or linearizing the dynamics [8, 32]. Developing a steering function is not always easy and linearization is valid only locally. This motivates methods that rely little on the system dynamics and work even for complex physically simulated systems [7].

The computational cost of tree sampling-based planners methods is asymptotically dominated by the nearest neighbor queries, which depend on the number of vertices. In practice, the cost also depends on the number of propagations per iteration, which may correspond to numerical integration or a physics engine call. These operations are expensive and algorithms need to minimize them. Such considerations have led in methods that aim to speed up the performance of asymptotically optimal solutions [2, 3, 10, 23, 27].

[⋆] Yanbo Li is associated with Cerner Corporation.
[⋆⋆] Zakary Littlefield is supported by a NASA Space Technology Research Fellowship.

RRT	RRT*	SST/SST*
Provably Suboptimal	**Asymp. Optimal**	**Asymp. Near-Opt./Asym. Opt.**
Forward Propagation	Steering Function	**Forward Propagation**
Single Propagation	Many Steering Calls	**Single Propagation**
1 NN Query ($\mathcal{O}(\log N)$)	1 NN + 1 K-Query ($\mathcal{O}(\log N)$)	1 NN + 1 K-Query (**Bounded Time Complexity** / ($\mathcal{O}(\log N)$))
Asymp. All Samples	Asymp. All Samples	**Sparse** / Asymp. All Samples
Minimal	**Minimal**	Desired Clearance / **Minimal**

Table 1. Comparison of RRT, RRT*and SST (SST*). The later methods minimize computation cost and space requirements while providing asymptotic near-optimality. From top to bottom each row compares the following properties: optimality guarantees, propagation method, number of propagations per iteration, type of nearest neighbor query, number of nodes (sparsity), and number of input parameters.

A promising approach to make sampling-based planners more efficient is to maintain a sparse data structure. Many of the existing approaches along this direction focus on sparse roadmaps [6, 20, 28, 31] and provide near-optimality guarantees. Near-optimality has been shown in the context of heuristic search to provide significant computational benefits [18]. Tree data structures can also benefit from sparsity. By maintaining a small set of nodes, the nearest neighbor queries can be performed more efficiently. The authors have recently proposed an RRT variant, called SPARSE RRT, which maintained a sparse tree representation. It was shown empirically - but not formally - that it provides good running time, good quality paths and has low space requirements [19]. Most importantly, it does not require a steering function, but instead relies only on forward propagation. SPARSE RRT provides sparsity by creating regions of a certain radius around high path quality nodes, where only the high-quality node is stored.

This work extends SPARSE RRT [19] so that it is possible to argue formal properties for kinodynamic planning, since this was difficult for the original method. Specifically, nodes are eventually removed almost surely within a region of an optimum path, which makes it difficult to reason about asymptotic properties. A new, modified version of the algorithm is proposed in this work, which is referred to as STABLE SPARSE RRT, or SST. A finite set of witness samples, which corresponds to a "hard-core" point process [21], is built in the state space so as to guarantee that a node of the tree will always exist in the vicinity of every witness and the path cost over time of such nodes improves. The method provides the following properties without access to a steering function:

- Probabilistic δ-robust completeness and asymptotic near-optimality.
- Fast convergence to good quality solutions.
- Low space overhead by building a sparse data structure.
- Lower asymptotic time complexity than RRT.

SST extends to an asymptotically optimal variant, SST*, which gradually relaxes the sparsification to eventually include all samples as nodes in the tree. Table 1 compares the proposed methods relative to RRT and RRT*.

Due to the space limitations, many of the formal arguments regarding the properties of SST are available in an extended version of this work [17].

2 Problem Formulation and Notation

This paper considers time invariant dynamical systems of the form:

$$\dot{x}(t) = f(x(t), u(t)),\ \ where\ x(t) \in \mathbb{X},\ \ and\ u(t) \in \mathbb{U} \tag{1}$$

Let $\mathbb{X}_f \in \mathbb{X}$ denote the obstacle-free space and assume that $\mathbb{X} \subset \mathbb{R}^n$. It should be sufficient if $\mathbb{X}$ is only diffeomorphic to a Euclidean space so that distances can be easily defined locally. Next, define a δ-robust trajectory to be a trajectory π with minimum clearance from obstacles, i.e., $\forall\, x_{obs} = \mathbb{X} \backslash \mathbb{X}_f :\ \ min(||\pi(t) - x_{obs}||) \geq \delta$. This work focuses on the following problem:

Definition 1. *(δ-Robustly Feasible Motion Planning) Given that a δ-robust trajectory exists that connects an initial state $x_0 \in \mathbb{X}_f$ to a goal region $\mathbb{X}_G \in \mathbb{X}_f$ for a dynamical system that follows Eq. 1, find a solution trajectory $\pi : [0, t_\pi] \rightarrow \mathbb{X}_f$, for which $\pi(0) = x_0$ and $\pi(t_\pi) \in \mathbb{X}_G$*

Finding a trajectory π corresponds to to computing controls $u(t)$ that generate π. Trajectory π does not have to be δ-robust. The authors reason first about a variation of the traditional probabilistic completeness property, which explicitly incorporates the clearance value δ.

Definition 2. *(Probabilistic δ-Robust Completeness) Let Π_n^{ALG} denote the set of trajectories discovered by an algorithm ALG at iteration n. Algorithm ALG is probabilistically δ-robustly complete, if, for any δ-robustly feasible motion planning problem (f, $\mathbb{X}_f$, x_0, $\mathbb{X}_G$, δ) the following holds:*

$$\liminf_{n\to\infty} \mathbb{P}(\ \exists\ \pi \in \Pi_n^{ALG} : \pi\ solution\ to\ (f, \mathbb{X}_f, x_0, \mathbb{X}_G, \delta)) = 1$$

In the above definition, $\mathbb{P}(Z)$ corresponds to the probability of event Z. This paper also argues about the following property relating to path quality:

Definition 3. *(Asymptotic δ-Robust Near-Optimality) Let c^* denote the minimum cost over all solution trajectories for a δ-robust feasible motion planning problem (f, $\mathbb{X}_f$, x_0, $\mathbb{X}_G$, δ). Let Y_n^{ALG} denote a random variable that represents the minimum cost value among all solutions returned by algorithm ALG at iteration n. ALG is asymptotically δ-robustly near-optimal if :*

$$\mathbb{P}\left(\left\{ \limsup_{n\to\infty} Y_n^{ALG} \leq (1 + \alpha \cdot \delta) \cdot c^* \right\}\right) = 1$$

for some known parameter $\alpha > 0$.

Defs. 2 and 3 correspond to weaker versions of probabilistic completeness and asymptotic near-optimality. This work will first describe a method that provides these weaker properties and then leverage the approach so as to achieve the original, more desirable properties. In addition, Def. 2 and 3 make intuitive sense in real-world applications where clearance from obstacles is desirable.

3 Algorithmic Description

Algorithm 1 details `STABLE SPARSE RRT` (`SST`), an adaptation of the previously proposed `SPARSE RRT` so as to achieve formal guarantees [19]. The main idea is that within a neighborhood region only the node with the best path cost from the root is considered in nearest neighbor queries and for expansion. This allows for the removal of nodes that do not contribute to good quality paths.

SST receives as input the typical parameters of kinodynamic planners (state space $\mathbb{X}$, control space $\mathbb{U}$, initial state x_0, propagation time T_{prop}, and number of iterations N). Furthermore, two new parameters are required, δ_v and δ_s, which correspond to radii that are used in different distance metric queries. The authors found that $\delta_v > \delta_s$ worked well in practice. For analysis purposes, these two parameters need to satisfy the constraint $\delta > \delta_v + 2\delta_s$ where δ is the clearance of a δ-robust trajectory that exists in the space.

Algorithm 1: SST($\mathbb{X}$, $\mathbb{U}$, x_0, T_{prop}, N, δ_v, δ_s)

```
1  $i \leftarrow 0$ ;                                              // Iteration counter
2  $\mathbb{V}_{active} \leftarrow \{x_0\}, \mathbb{V}_{inactive} \leftarrow \emptyset, \mathbb{V} \leftarrow \mathbb{V}_{active} \cup \mathbb{V}_{inactive}$ ;   // Node sets
3  $\mathbb{E} \leftarrow \emptyset$, $G = \{V, \mathbb{E}\}$ ;                          // Initialize graph
4  $s_0 \leftarrow x_0$, $s_0.rep = x_0$, $S \leftarrow \{s_0\}$ ;          // Initialize witness set
5  while $i++ < N$ do
6      $s_{sample} \leftarrow$Sample($\mathbb{X}$) ;              // Uniform sampling in state space
7      $x_{nearest} \leftarrow$BestNear($\mathbb{V}_{active}$, $s_{sample}$, $\delta_v$) ;  // Return the BestNear node
8      $x_{new} \leftarrow$ MonteCarlo-Prop($x_{nearest}$, $\mathbb{U}$, $T_{prop}$) ;       // Propagate forward
9      if CollisionFree($\overline{x_{nearest} \rightarrow x_{new}}$) then
10         $s_{new} \leftarrow$ Nearest( $S$, $x_{new}$ ) ;    // Get the nearest witness to $x_{new}$
11         if $dist(x_{new}, s_{new}) > \delta_s$ then
12             $S \leftarrow S \cup \{x_{new}\}$ ;             // Add a new witness that is $x_{new}$
13             $s_{new} \leftarrow x_{new}$;
14             $s_{new}.rep \leftarrow NULL$;
15         $x_{peer} \leftarrow s_{new}.rep$ ;                     // Get current represented node
16         if $x_{peer} == NULL$ or cost($x_{new}$) $<$ cost($x_{peer}$) then
17             $\mathbb{V}_{active} \leftarrow \mathbb{V}_{active} \setminus \{x_{peer}\}$ ;                   // Removing old rep
18             $\mathbb{V}_{inactive} \leftarrow \mathbb{V}_{inactive} \cup \{x_{peer}\}$ ;      // Making old rep inactive
19             $s_{new}.rep \leftarrow x_{new}$ ;                            // Assign the new rep
20             $\mathbb{V}_{active} \leftarrow \mathbb{V}_{active} \cup \{x_{new}\}$, $\mathbb{E} \leftarrow \mathbb{E} \cup \{\overline{x_{nearest} \rightarrow x_{new}}\}$ ; // Grow $G$
21             while IsLeaf ($x_{peer}$) and $x_{peer} \in \mathbb{V}_{inactive}$ do
22                 $x_{parent} \leftarrow$Parent($x_{peer}$);
23                 $\mathbb{E} \leftarrow \mathbb{E} \setminus \{\overline{x_{parent} \rightarrow x_{peer}}\}$ ;                 // Remove from $G$
24                 $\mathbb{V}_{inactive} \leftarrow \mathbb{V}_{inactive} \setminus \{x_{peer}\}$ ; // Remove from inactive set
25                 $x_{peer} \leftarrow x_{parent}$ ;           // Recurse to parent if inactive
26 return $G$;
```

SST begins by initializing two vertex sets $\mathbb{V}_{active}$ and $\mathbb{V}_{inactive}$ (Line 2). The union of these sets corresponds to the set of tree nodes. The two subsets are treated differently by nearest neighbors queries, which will be discussed shortly. Next, the set of witness nodes S is initialized to contain the start node x_0 (Line 4). The algorithm maintains the invariant that within the neighborhood of radius δ_s around any $s \in S$, there is always one state in V_{active}. This state in $\mathbb{V}_{active}$ is called the representative of the witness s. Representatives can change over time but only as long as their cost from the root decreases. To add new nodes to the tree, an approach similar to the framework of sampling-based kinodynamic planning [9, 16] is used, but with some modifications inspired from analysis.

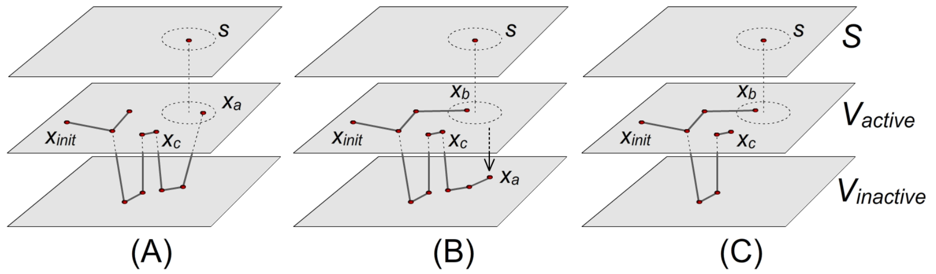

Fig. 1. Relation between S and the V sets. A: A tree and a trajectory $\overline{x_{init} \to x_c \to x_a}$ where x_a is the representative of s; B: The algorithm extends $\overline{x_{init} \to x_b}$ where x_b has better cost than x_a. x_a is moved from V_{active} to $V_{inactive}$. C: The representative of s is now x_b (Lines 21-25 of Alg. 1). The trajectory $\overline{x_c \to x_a}$ in $\mathbb{V}_{inactive}$ is pruned.

First, a state is sampled in the state space (Line 6). Then an operation called `BestNear` [19, 30] determines which existing node in $\mathbb{V}_{active}$ will be selected for propagation (Line 7). To achieve this, a neighborhood of size δ_v is explored around the randomly sampled point s_{sample}. For every node that exists in that neighborhood, path cost values from the root are compared and the best path cost node $x_{nearest}$ is returned. If no nodes exist within the neighborhood, the nearest node is returned. This has been shown to have good properties for path quality, and will be more formally explored in the analysis section.

After selecting $x_{nearest}$, the method calls `MonteCarlo-Prop` to generate x_{new} (Line 8), which forward simulates the system using a random piecewise-constant control for a random duration. It can be shown that this random propagation has good asymptotic properties, argued in the analysis section. Given the newly propagated trajectory is collision-free, the method determines the closest witness s_{new} to node x_{new} (Lines 9-10). If the closest witness is outside the δ_s radius, a new witness in the set S is created that corresponds to x_{new} (Lines 11-14). This computation requires access to a distance function $dist(\cdot, \cdot)$ in the state space. In practice, this distance can be computed in a lower-dimensional task space $\mathbb{T}$.

Finally, after the closest witness s_{new} has been found, the representative x_{peer} of s_{new} and the new node x_{new} are compared (Line 16). The comparison is performed using the function `cost`$(\cdot)$, which is the cost of the trajectory from x_0 to that node in the tree. If x_{new} has a better cost or x_{peer} is $NULL$ (which is the case when x_{new} is outside the δ_s radius of the closest witness), the witness s_{new} will forget about its old representative x_{peer} and now it will be represented by x_{new} (Lines 17-20). Subsequently, the old node x_{peer} will be removed from $\mathbb{V}_{active}$ and added to $\mathbb{V}_{inactive}$, thereby removing it from any nearest neighbor queries. One can think of the $\mathbb{V}_{inactive}$ set as consisting of nodes that no longer themselves provide good paths, but may provide connectivity to children nodes in $\mathbb{V}_{active}$ that may be the best in their respective neighborhoods. After manipulating the vertex sets, an optimization step can be taken that involves removing needless nodes in the tree (Lines 21-25). These correspond to leaf nodes that are in the $\mathbb{V}_{inactive}$. They can be removed in a recursive manner. The addition and removal of nodes in the two vertex sets $\mathbb{V}_{active}$ and $\mathbb{V}_{inactive}$ is an important part of making `SST` computationally efficient, and is illustrated in Fig. 1.

SST is a modification of SPARSE RRT that makes use of the "witness" set of nodes. Including the set of witnesses allows for regions in the state space to have a representative node, regardless of the pruning process of the tree. While SPARSE RRT performed well experimentally, when exploring the theoretical guarantees that the algorithm could provide, difficulties arose when reasoning about probabilistic completeness and asymptotic optimality. The main issue was that in execution, nodes are potentially deleted frequently and unpredictably, which meant that some asymptotic behaviors are difficult to determine.

4 Analysis

This section discusses the properties of SSTand describes a schedule for reducing parameters δ_s and δ_v over time to achieve asymptotic optimality.

Assumption 1 *The assumptions used by the analysis include the following:*

- *The dynamics of Eq. 1 are Lipschitz continuous in states and controls, have bounded second derivatives, and the system is Small-Time Locally Accessible (STLA) [4].*
- *The cost function is considered by this work to be the duration of a trajectory. Thus, the cost function is Lipschitz continuous w.r.t. states, additive, and monotonic.*
- *The robustly feasible motion planning problem admits robustly feasible trajectories that are generated by piecewise constant control functions.*

This set of assumptions define the widest set of systems for which asymptotic optimality has been shown without access to a BVP*solver.*

A key part of the analysis is concerned with examining a δ-robust optimal path. To facilitate this, a covering ball sequence is defined over such a path.

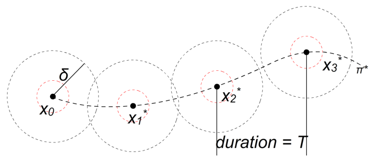

Fig. 2. A set of covering balls $\mathbb{B}(\pi^*(t),\ \delta,\ T)$ around the optimal path π^*.

Definition 4. *(Covering Balls) Given a trajectory $\pi(t)$: $[0, T_{end}] \to \mathbb{X}_f$, clearance $\delta \in R^+$ and time step T, the set of covering balls $\mathbb{B}(\pi(t),\ \delta,\ T)$ is defined as a set of $M+1$ balls $\{\mathcal{B}_\delta(x_0),\ \mathcal{B}_\delta(x_1),\ ...,\ \mathcal{B}_\delta(x_M)\}$ of radius δ, such that $\mathcal{B}_\delta(x_M)$ is centered at $x_i = \pi(iT)\ \forall\ i \in [0, M]$, where $M = \frac{T_{end}}{T}$.*

For an example of a covering ball sequence, see Fig. 2. This construction gives rise to the following definition:

Definition 5. *(δ-Similar Trajectories) Trajectories π,π' are δ-similar if for a continuous scaling function $\sigma : [0, t] \to [0, t']$, it is true: $\pi'(\sigma(t)) \in \mathcal{B}_\delta(\pi(t))$.*

Lemma 1 *(Existence of δ-Similar Trajectories) Let there be a trajectory π satisfying Eq. 1. Then there exists a positive value δ_0, such that: $\forall\ \delta \in (0, \delta_0]$, $\forall\ x'_0 \in \mathcal{B}_\delta(\pi(0))$, and $\forall\ x'_1 \in \mathcal{B}_\delta(\pi(t))$, there exists a δ-similar trajectory π', so that: (i) $\pi'(0) = x'_0$ and $\pi'(t') = x'_1$*

Lem. 1, which can be argued given the assumptions, helps to show that a δ-similar trajectory to a δ-robust optimal one can be generated. If such a δ-similar trajectory is found, then from the assumptions of Lipschitz continuity and the cost function characteristics, a bound on the path quality can also be drawn.

4.1 Probabilistic δ-Robust Completeness

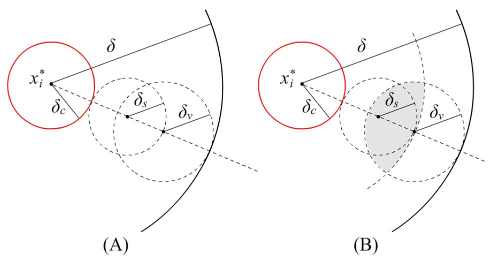

Fig. 3. A visualization of the relationship between the different radii used in the analysis of `SST`.

The proof begins by constructing a sequence of balls $\mathbb{B}(\pi^*, \delta, T)$ that cover the δ-robust optimal path π^* (see Fig. 2), which is guaranteed to exist by the problem definition. Let $\mathcal{B}_\delta(x_i^*)$ denote the i^{th} ball in the sequence centered around state x_i^* on π^*. The first thing to show is that if a trajectory reaches one of these balls, there will always be a node in the ball with equal or better cost in future iterations. Lem. 2 explains this result.

Lemma 2 *Let $\delta_c = \delta - \delta_v - 2\delta_s > 0$. If a state $x \in V_{active}$ is generated at iteration n s.t. $x \in \mathcal{B}_{\delta_c}(x_i^*)$, then for every iteration $n' \geq n$, there is a state $x' \in V_{active}$ so that $x' \in \mathcal{B}_{(\delta-\delta_v)}(x_i^*)$ and* `cost`*$(x') \leq$*`cost`*(x).*

Proof. Given x is a node, there is a witness point s located near x. As in Fig. 3 A, the witness s can be located, in the worst case, at distance δ_s away from the boundary of $\mathcal{B}_{\delta_c}(x_i^*)$ if $x \in \mathcal{B}_{\delta_c}(x_i^*)$. Note that x can be removed from $\mathbb{V}_{active}$ by `SST` in later iterations. In fact, x almost surely will be removed, if $x \neq x_0$. When x is removed, there could be no state in the ball $\mathcal{B}_{\delta_c}(x_i^*)$. In this case, the selection procedure has no chance to return any state within this ball. The sample s will not be deleted, however. A node x' representing s will always exist in $\mathbb{V}_{active}$ and x' will not leave the ball $\mathcal{B}_{\delta_s}(s)$. `SST` guarantees that the cost of x' will never increase, i.e. `cost`$(x') \leq$`cost`(x). In addition, x' has to exist inside $\mathcal{B}_{\delta-\delta_v}(x_i^*) = \mathcal{B}_{\delta_c+2\delta_s}(x_i^*)$. □

Lem. 3 lower bounds the probability of selecting $x' \in \mathcal{B}_{\delta-\delta_v}(x_i^*)$, which exists.

Lemma 3 *Assuming uniform sampling in the* `Sample` *function of Alg. 1, if $\delta_v + 2\delta_s < \delta$ and if $\exists\ x \in \mathbb{V}_{active}$ s.t. $x \in \mathcal{B}_{\delta_c}(x_i^*)$ at iteration n, then the probability that* `BestNear` *selects for propagation a node $x' \in \mathcal{B}_\delta(x_i^*)$ can be lower bounded by a positive constant γ for every $n' > n$.*

Proof. `BestNear` performs uniform random sampling in $\mathbb{X}$ to generate s_{sample} and examines the ball $\mathcal{B}_{\delta_v}(s_{sample})$ to find the node with the best path. In order for a node in $\mathcal{B}_\delta(x_i^*)$ to be returned, the sample needs to be in $\mathcal{B}_{\delta-\delta_v}(x_i^*)$. If the sample is outside this ball, then a node not in $\mathcal{B}_\delta(x_i^*)$ can be considered, and therefore may be selected. See Fig. 3 A. Next, consider the size of the intersection of $\mathcal{B}_{\delta-\delta_v}(x_i^*)$ and a ball of radius δ_v that is entirely enclosed in $\mathcal{B}_\delta(x_i^*)$. Let x_v denote the center of this ball. This intersection, shown in Fig. 3 B, represents the area that a sample can be generated to return a state from ball $\mathcal{B}_{\delta-\delta_v}(x_i^*)$. In the worst case, the center of ball $\mathcal{B}_{\delta_v}(x_v)$ could be on the border of $\mathcal{B}_{\delta-\delta_v}(x_i^*)$ as in Fig. 3 B. Then, the probability of sampling a state in this region is: $\gamma = \inf \mathbb{P}\Big(\big\{x' \text{ returned by } \texttt{BestNear} : x' \in \mathcal{B}_\delta(x_i^*)\big\}\Big) = \frac{\mu(\mathcal{B}_{\delta-\delta_v}(x_i^*) \cap \mathcal{B}_{\delta_v}(x_v))}{\mu(\mathbb{X}_f)}$. This is the smallest region that guarantees selection of a node in $\mathcal{B}_\delta(x_i)$. □

Given the assumptions of *STLA*, the *Lipschitz continuity* of $\mathbb{X}$, $\mathbb{U}$, and bounded second order derivatives of the system equation, it can be shown that the probability of propagating from one ball to another using `MonteCarlo-Prop` is positive.

Lemma 4 *Given a trajectory π of duration T, the success probability for function* `MonteCarlo-Prop` *to generate a δ-similar trajectory to π when called from an input state $x_{prop} \in \mathcal{B}_\delta(\pi(x^*_{i-1}))$ and for a propagation duration $T_{prop} > T$ to the ball $\mathcal{B}_{\delta_c}(\pi(x^*_i))$ is lower bounded by a positive value $\rho_{\delta\to\delta_c} > 0$.*

At this point, lower bounds on both the probability of selecting a node and the probability of generating a trajectory that ends in the next ball of the sequence have been argued. Based on these lower bounds, the following can be shown:

Theorem 1 *If $\delta_v + 2\delta_s < \delta$, then* `STABLE SPARSE RRT` *is probabilistically δ-robustly complete.*

Proof. As in Fig. 2, consider the sequence $\mathbb{B}(\pi^*,T,\delta)$ over the optimal path π^* for $\delta > \delta_v + 2\delta_s$. A specific ball $\mathcal{B}_\delta(x^*_i)$ can be seen in Fig. 3 A. Lem. 2 shows that nodes will continue to exist in $\mathcal{B}_{\delta_c}(x^*_i)$, if one was generated. Lem. 3 shows there is a positive probability that nodes in $\mathcal{B}_{\delta-\delta_v}(x^*_i)$ can be selected. Lem. 4 argues that `MonteCarlo-Prop` has a positive probability $\rho_{\delta\to\delta_c}$ of generating a trajectory into the next ball $\mathcal{B}_{\delta_c}(x^*_{i+1})$.

Let $A_i^{(n)}$ denote the event that at the n^{th} iteration, the algorithm generates one trajectory π such that $\pi(0) \in \mathcal{B}_\delta(x^*_{i-1})$ and $\pi(T_{end}) \in \mathcal{B}_{\delta_c}(x_i)$, meaning π is δ-similar to $\overline{x^*_{i-1}, x^*_i}$. Let $E_i^{(n)}$ denote the event that from iteration 1 to n, the algorithm generates at least one such trajectory. Then, the event $\neg E_i^{(n)}$ is the event the algorithm fails to generate any near-optimal trajectory inside $\mathcal{B}_{\delta-\delta_v}(x^*_i)$ after n iterations, which only happens when all n iterations fail, i.e.,

$$\mathbb{P}(\neg E_i^{(n)}) = \mathbb{P}(\neg A_i^{(1)}) \cdot \mathbb{P}(\neg A_i^{(2)} | \neg A_i^{(1)}) \cdot \ \dots \ \cdot \mathbb{P}(\neg A_i^{(n)} | \bigcap_{j=1}^{n-1} \neg A_i^{(j)}) \qquad (2)$$

The probability that $\neg A_i^{(n)}$ happens given $\bigcap_{j=1}^{n-1} \neg A_i^{(j)}$ is equivalent to the probability of failing to generating a trajectory to the $\mathcal{B}_{\delta_c}(x^*_{i-1})$ plus the probability that a trajectory has been generated to $\mathcal{B}_{\delta_c}(x^*_{i-1})$, but fails to generate a new trajectory to $\mathcal{B}_{\delta_c}(x^*_i)$, i.e.,

$$\begin{aligned}\mathbb{P}(\neg A_i^{(n)} | \bigcap_{j=1}^{n-1} \neg A_i^{(j)}) &= \mathbb{P}(\neg E_{i-1}^{(n)}) + \mathbb{P}(E_{i-1}^{(n)}) \cdot \mathbb{P}(\{\text{fails to propagate to } \mathcal{B}_{\delta_c}(x^*_i)\}) \\ &\leq \mathbb{P}(\neg E_{i-1}^{(n)}) + \mathbb{P}(E_{i-1}^{(n)})(1 - \gamma\rho_{\delta\to\delta_c}) \leq 1 - \mathbb{P}(E_{i-1}^{(n)}) \cdot \gamma\rho_{\delta\to\delta_c} \qquad (3)\end{aligned}$$

Using Eq. 2 and Eq. 3,

$$\mathbb{P}(E_i^{(n)}) \geq 1 - \prod_{j=1}^{n} (1 - \mathbb{P}(E_{i-1}^{(j)}) \cdot \gamma\rho_{\delta\to\delta_c}) \qquad (4)$$

For the base case, $\mathbb{P}(E_0^{(j)}) = 1$ because x_0 is always in $\mathcal{B}_{\delta_c}(x_0)$. Then, consider event E_1 from iteration 1 to n using Eq. 4,

$$\mathbb{P}(E_1^{(n)}) \geq 1 - \textstyle\prod_{j=1}^{n}(1 - \gamma\rho_{\delta\to\delta_c}) = 1 - (1 - \gamma\rho_{\delta\to\delta_c})^n \implies$$

$$\lim_{n\to\infty} \mathbb{P}(E_1^{(n)}) \geq 1 - \lim_{n\to\infty} (1 - \gamma\rho_{\delta\to\delta_c})^n = 1 - 0 = 1$$

The same result needs to be shown for $E_{i+1}^{(n)}$. Set $y_i^{(n)} = \prod_{j=1}^{n}(1 - \mathbb{P}(E_{i-1}^{(j)}) \cdot \gamma\rho_{\delta\to\delta_c})$ The logarithm of $y_i^{(n)}$ behaves as follows,

$$\log y_i^{(n)} = \log \prod_{j=1}^{n}(1 - \mathbb{P}(E_{i-1}^{(j)}) \cdot \gamma\rho_{\delta\to\delta_c}) = \sum_{j=1}^{n} \log(1 - \mathbb{P}(E_{i-1}^{(j)}) \cdot \gamma\rho_{\delta\to\delta_c}))$$

$$< \sum_{j=1}^{n} -\mathbb{P}(E_{i-1}^{(j)}) \cdot \gamma\rho_{\delta\to\delta_c} = -\gamma\rho_{\delta\to\delta_c} \cdot \sum_{j=1}^{n} \mathbb{P}(E_{i-1}^{(j)}) \tag{5}$$

From the inductive assumption that, $\mathbb{P}(E_i^{(j)})$ converges to 1 as $j \to \infty$, then $\lim_{n\to\infty} \sum_{j=1}^{n} \mathbb{P}(E_i^{(j)}) = \infty$. Then,

$$\lim_{n\to\infty} \log y_{i+1}^{(n)} < -\gamma\rho_{\delta\to\delta_c} \cdot \lim_{n\to\infty} \sum_{j=1}^{n} \mathbb{P}(E_i^{(j)}) = -\infty \iff \lim_{n\to\infty} y_{i+1}^{(n)} = 0$$

Using Eq. (4), with $\lim_{n\to\infty} y_{i+1}^{(n)} = 0$, it can be shown that:

$$\lim_{n\to\infty} \mathbb{P}(E_{i+1}^{(n)}) = 1 - \lim_{n\to\infty} y_{i+1}^{(n)} = 1 - 0 = 1.$$

4.2 Asympotic Near-Optimality

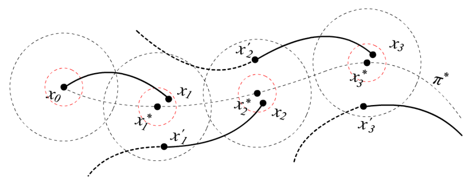

Fig. 4. Sequence of covering balls over an optimal trajectory π^* and nodes/edges generated by `SST`.

The proof of asymptotic δ-robust near-optimality follows directly from Thm. 1, the *Lipschitz continuity*, *additivity*, and *monotonicity* of the cost function (Assumption 1). The completeness proof is already examining the generation of a near optimal trajectory, but the bound on the cost needs to be calculated.

Theorem 2 *If* $\delta_v + 2\delta_s < \delta$, *then* `STABLE SPARSE RRT` *is asymptotically* δ*-robustly near-optimal.*

Proof. Let $\overline{x'_{i-1}, x_i}$ denote the δ-similar trajectory segment generated by `SST` where $x'_{i-1} \in \mathcal{B}_{\delta_c}(x^*_{i-1})$ of the optimal path and $x_i \in \mathcal{B}_{\delta-\delta_v}(x^*_i)$. Lem. 4 guarantees that the probability of generating it by `MonteCarlo-Prop` can be lower bounded as $\rho_{\delta\to\delta_c}$. Then from the definition of δ-similar trajectories and Lipschitz continuity of the cost function (K_x is the Lipschitz constant for $\mathbb{X}$):

$$\texttt{cost}(\overline{x'_{i-1} \to x_i}) \leq \texttt{cost}(\overline{x^*_{i-1} \to x^*_i}) + K_x \cdot \delta \tag{6}$$

Lem. 3 guarantees that when x_i exists in $\mathcal{B}_{\delta-\delta_v}(x^*_i)$, then x'_i, returned by the `BestNear` function with least bound γ, must have equal or less cost, i.e x'_i can be the same state as x_i or a different state with smaller or equal cost:

$$\texttt{cost}(x'_i) \leq \texttt{cost}(x_i) \tag{7}$$

Consider $\mathcal{B}_\delta(x^*_1)$, as illustrated in Fig. 4, according to (6) and (7),

$$\texttt{cost}(\overline{x_0 \to x'_1}) \leq \texttt{cost}(\overline{x_0 \to x_1}) \leq \texttt{cost}(\overline{x_0 \to x^*_1}) + K_x \cdot \delta$$

Assume this is true for k segments, then: $\texttt{cost}(\overline{x_0 \to x'_k}) \leq \texttt{cost}(\overline{x_0 \to x^*_k}) + k \cdot K_x \cdot \delta$. Consider the cost of the trajectory with $k+1$ segments:

$$\begin{aligned}
\texttt{cost}(\overline{x_0 \to x'_{k+1}}) &\leq \texttt{cost}(\overline{x_0 \to x_{k+1}}) = \texttt{cost}(\overline{x_0 \to x'_k}) + \texttt{cost}(\overline{x'_k \to x_{k+1}}) \\
&\leq \texttt{cost}(\overline{x_0 \to x^*_k}) + k \cdot K_x \cdot \delta + \texttt{cost}(\overline{x'_k \to x_{k+1}}) \\
&\leq \texttt{cost}(\overline{x_0 \to x^*_k}) + k \cdot K_x \cdot \delta + \texttt{cost}(\overline{x^*_k \to x^*_{k+1}}) + K_x \cdot \delta \\
&= \texttt{cost}(\overline{x_0 \to x^*_{k+1}}) + (k+1) \cdot K_x \cdot \delta
\end{aligned}$$

By induction, this holds for all k.
Since the largest $k = \frac{T^*_k}{T}$, and the cost of the trajectory is its duration,

$$\texttt{cost}(\overline{x_0 \to x'_k}) \leq \texttt{cost}(\overline{x_0 \to x^*_k}) + \frac{\texttt{cost}(\overline{x_0 \to x^*_k})}{T} \cdot K_x \cdot \delta = (1 + \frac{K_x \delta}{T}) \cdot c^*_k$$

Recall from Theorem 1, event E_k implies event $\{Y^{SST} \leq (1 + \alpha\delta) c^*_k\}$.

$$\mathbb{P}\Big(E^{(n)}_k\Big) = \mathbb{P}\Big(\big\{Y^{SST}_n \leq (1 + \frac{K_x \delta}{T}) c^*_k\big\}\Big)$$

As $n \to \infty$, Thm. 1 guarantees that if $\delta > \delta_v + 2\delta_s$, $E^{(\infty)}_k$ almost surely happens.

$$\mathbb{P}\Big(\big\{\limsup_{n \to \infty} Y^{SST}_n \leq (1 + \frac{K_x \delta}{T}) c^*_k\big\}\Big) = \lim_{n \to \infty} \mathbb{P}\Big(E^{(n)}_k\Big) = 1$$

□

4.3 Time Complexity Arguments

Now consider the convergence rate for `SST`, i.e. the iterations needed to return a near-optimal trajectory with a certain probability. Specifically, the convergence rate depends on the difficulty level of the kinodynamic planning problem which is measured by the probability $\rho_{\delta \to \delta_c}$ of successfully generating a δ-similar trajectory segment connecting two covering balls.

Theorem 3 *For a δ-robust optimal trajectory consisting of $k > 0$ segments, and a fixed $\rho_{\delta \to \delta_c} > 0$, the iterations $N_{\rho_{\delta \to \delta_c}}$ for* `SST` *to generate a near-optimal solution with probability greater than $1 - e^{-1}$ can be bounded by: $\frac{1}{1-e^{-1}} \cdot \frac{k}{\gamma \rho_{\delta \to \delta_c}}$*

The above theorem argues that in order to achieve at least $1 - e^{-1} \approx 63.21\%$ probability for `SST` to generate a near-optimal trajectory, the needed iterations can be upper bounded. This bound is in the same order as the expected number of iterations for `RRT` to return a solution [16] [1].

[1] In [16], the iteration bound for `RRT` to return a feasible solution with probability of at least $1 - e^{-1}$ is shown as $\frac{k}{p}$ in theorem 1 of the paper. k is the number of trajectory segments of the solution. And p is the minimum probability to select a vertex in the "attraction sequence". p corresponds to the same concept of γ_{rrt} in this paper. `RRT` models the Extend procedure with an additional assumption such that generating a connection edge between consecutive "attraction wells" shall succeed in one shot. Here, the Extend function corresponds to `MonteCarlo-Prop`, which generates connection edges with probability at least $\rho_{\delta \to \delta}$. Therefore, the expected iteration bound for `RRT` is in the form of $\mathcal{O}(\frac{k}{\gamma_{rrt} \cdot \rho_{\delta \to \delta}})$.

In contrast to `RRT`* which employs a steering function, the proposed algorithm involves no such functions. All operations for the proposed algorithm are well understood. Therefore, it is possible to evaluate the overall computational cost needed for `SST`. The proof for Lemma 5 and Lemma 6 are included in an extended version of this work [17].

Lemma 5 *For a k segment optimal trajectory with δ clearance, the expected running time for* `SST` *to return a near-optimal solution with $1 - e^{-1}$ probability can be evaluated as,* $\mathcal{O}\Big(\delta^{-d} \cdot \frac{k}{\gamma \rho_{\delta \to \delta_c}}\Big)$

The benefit of `SST` is that the per iteration cost ends up being lower than that of `RRT`, while a certain form of optimality guarantees can be provided.

Lemma 6 *For a k segments trajectory with δ clearance, the expected running time for the* `RRT` *algorithm to return a solution with $1 - e^{-1}$ probability can be evaluated as* $\mathcal{O}\Big((\frac{k}{\gamma_{rrt}\rho_{\delta \to \delta}}) \cdot (\log(\frac{k}{\gamma_{rrt}\rho_{\delta \to \delta}}))\Big)$

Comparing Lem. 5 and Lem. 6 by quotient, $\mathcal{O}\Big(\frac{k/\gamma\rho_{\delta \to \delta_c}}{k/\gamma_{rrt}\rho_{\delta \to \delta} \cdot \log(k/\gamma_{rrt}\rho_{\delta \to \delta})}\Big) = \mathcal{O}\Big(\frac{\gamma_{rrt}\rho_{\delta \to \delta}}{\gamma\rho_{\delta \to \delta_c}} \cdot \frac{1}{-\log \gamma_{rrt}\rho_{\delta \to \delta}}\Big)$. The first term $\frac{\gamma_{rrt}\rho_{\delta \to \delta}}{\gamma\rho_{\delta \to \delta_c}}$ is a finite value which is shown in [17]. And the second term converges to 0 as $\rho_{\delta \to \delta}$ decreases to 0. Therefore, the expected time complexity of `SST` is indeed smaller than the expected time complexity of `RRT` for sufficiently difficult kinodynamic problems. This is mainly because `SST` keeps a sparse data structure so that the cost of all near neighbor queries, which is asymptotically the most expensive operation in these algorithms, can be bounded by a constant. But this is noticeable only for difficult problems where $\rho_{\delta \to \delta}$ is sufficiently small. Practically, `RRT`, perhaps, is still the fastest algorithm to return the first feasible trajectory.

4.4 Space Requirements Arguments

A fairly simple fact is stated formally in Lem. 7.

Lemma 7 *For any two distinct witnesses of* `SST` *$s_1, s_2 \in S$, where $s_1 \neq s_2$, the distance between them is at least δ_s, e.g., $\forall s_1, s_2 \in S : ||s_1 - s_2|| > \delta_s$*

It can then be shown that S can be bounded if $\mathbb{X}_f$ is bounded.

Corollary 1 *If $\mathbb{X}_f$ is bounded, the number of points of the set S and nodes in $\mathbb{V}_{active}$ is always finite, i.e. $\exists M \in \mathcal{O}(\delta^{-d}) : |S| = |V_{active}| \leq M$*

The size of $\mathbb{V}_{inactive}$ cannot be easily bounded, but if pruning is performed as in the algorithm, the size of $\mathbb{V}_{inactive}$ is manageable.

Generating the set S corresponds to a variant of `Poisson Disk Sampling`, a.k.a. `Naive Dart-Throwing` with the difference that the sampling does not strictly follow a `Poisson Distribution`. Related research refers to such processes as `Matérn Type III` point processes [21]. This literature can be utilized to improve the distribution of S, i.e., improve its *discrepancy* and *dispersion.* In kinematic planning, there have been demonstrations of quasi-random sampling for generating low discrepancy points [15]. The requirement for S is that it has to be evenly distributed such that each Voronoi cell can be bounded by a hyper ball. Therefore, `SST` can take advantage of deterministic sampling. In addition, "hard-core" point process contributions can be employed for kinodynamic planning by generating S offline, and then running `SST`.

4.5 Asymptotically Optimal Variant

Now consider the SST* algorithm shown in Alg. 2. It provides a schedule to shrink the parameters of SST. It appropriately merges solving an infinite sequence of δ-robust motion planning problems. It can be proven that SST* is probabilistically complete and asymptotically optimal. This is done by leveraging decreasing δ_s and δ_v values determined by the scaling parameter $\xi \in (0,1)$) and the decreasing δ of the δ-robust trajectories that are admitted by SST.

Algorithm 2: SST*($\mathbb{X}$,$\mathbb{U}$,x_0,T_{prop},$\delta_{s,0}$,$\delta_{v,0}$,ξ)

```
1 $j \leftarrow 0$; $K \leftarrow k_0$;
2 $\delta_s \leftarrow \delta_{s,0}$; $\delta_v \leftarrow \delta_{v,0}$;
3 while true do
4     $SST(\mathbb{X}, \mathbb{U}, x_0, T_{prop}, K, \delta_v, \delta_s)$;
5     $\delta_s \leftarrow \xi \cdot \delta_s$; $\delta_v \leftarrow \xi \cdot \delta_v$; $j \leftarrow j+1$;
6     $K \leftarrow (1+\log j) \cdot \xi^{-(d+w+1)j} \cdot k_0$;
```

Theorem 4 SST* *is probabilistically complete and is asymptotically optimal.*

When the δ clearance is arbitrarily small, the arguments outlined in Thms. 1 and 2 still hold. The drawback with starting with this arbitrarily small δ is that SST will not be able to take advantage of sparsity. SST* is able to take advantage of intermediate results, returning near-optimal results quickly, and progressively increasing the number of nodes allowed for nearest neighbor queries, and thereby providing an asymptotically optimal solution.

5 Evaluation

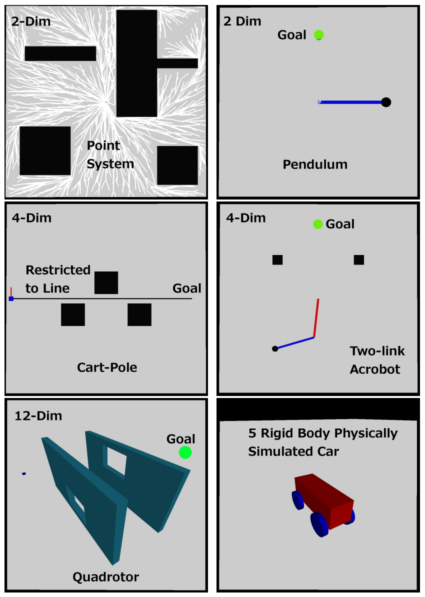

Fig. 5. The benchmarks considered in the experiments. Each experiment is averaged over 50 runs for each algorithm.

In order to evaluate the proposed method, a set of experiments involving several different systems have been conducted. The proposed algorithm, SST, is compared against RRT as a baseline and also with another common algorithm: (a) if a steering function is available, a comparison with RRT* is conducted, (b) if RRT* cannot be used, a comparison with a heuristic alternative based on a "shooting" function is utilized [13].
The shooting function is numerically approximating a steering function but doesn't connect two states exactly. To alleviate this problem, when a rewire is performed, entire subtrees are resimulated with the new end state that is close to the original state. The overall results show that SST can provide consistently improving path quality given more iterations as RRT* does for kinematic systems, achieve running time

equivalent (if not better) than RRT, and maintain a small number of nodes, all while using a very simple random propagation primitive.

Fig. 5 details the various setups that the algorithms have been evaluated on. As a baseline, a kinematic point system is used. This allows a direct comparison of results with RRT* given that a steering function is easily created. SST still makes use of random propagation in this case, but good behavior is shown in the following sections.

Evaluation was also conducted on pendulum-like systems which include a single link pendulum, a two-link passive-active acrobot system, and a cart-pole system. In addition, a quadrotor system is considered, where distances are taken in a task space. These systems have simple state update equations, but are nonlinear. No steering function is used in these experiments

One of the more interesting applications of SST is in the domain of planning for physically-simulated systems [1]. SST is able to provide improving path quality given enough time and keeps the number of forward propagations to one per iteration. In this setup, the computational cost of propagation overtakes the cost of nearest neighbor queries. Nearest neighbor queries become the bottleneck in problems like the kinematic point where propagation and collision checking are cheap. In this respect, SST is specially suited to plan for physically-simulated systems.

5.1 Quality of Solution Trajectories

In Fig. 6, the average solution quality to nodes in each tree is shown. This average is a measure of the quality of trajectories that have explored the space being searched. In every case, SST is able to improve quality over time, even in the case of the physically-simulated car. RRT will increase this average over time because it chooses suboptimal nodes and further propagates them.

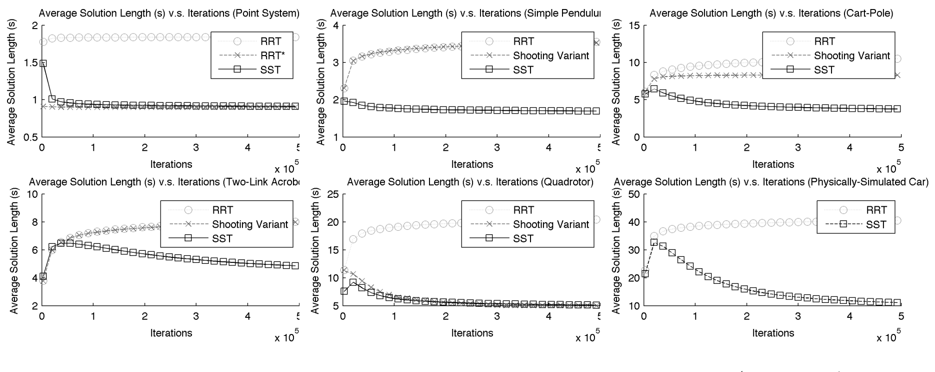

Fig. 6. The average cost to each node in the tree for each algorithm (RRT, RRT* or the shooting approach, and SST).

5.2 Time Efficiency

Fig. 7 shows time vs. iterations plots for each of the systems. The graphs show the amount of time it took to achieve a number of iterations. The running time of SST is always comparable or better than RRT. RRT* has a higher time cost per iteration as expected. Initially SST is slightly slower than RRT for the kinematic point, but becomes increasingly more efficient later on. This is explained by Lem. 5 and Lem. 6, since SST has better running time than RRT given the sparse data structure. For physically-simulated systems, the computational cost is dominated by the forward propagation, where both RRT and SST perform the same amount.

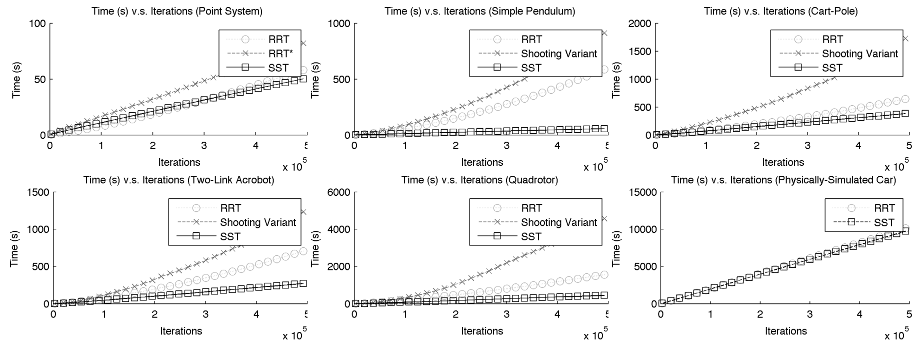

Fig. 7. The amount of time needed for each algorithm (RRT, RRT* or the shooting approach, and SST).

5.3 Space Efficiency

One of the major gains of using SST is in the smaller number of nodes that are needed in the data structure. Fig. 8 shows the number of nodes stored by each of the algorithms. The number of nodes is significantly lower in SST, even when also considering the witness set S. The sparse data structure of SST makes the memory requirements quite small, in contrast to RRT and RRT* which don't perform any pruning operations. In the case using shooting, sometimes the inaccuracy of shooting will cause collisions to occur in resimulated trees, pruning them from the tree. This can lead to losing solution trajectories however.

6 Discussion

Recently, the focus in sampling-based motion planning has been on providing optimality guarantees. Achieving this objective for systems with dynamics has generally required specialized steering functions. SST is a method that does not require such processes and still provides trajectory quality guarantees. Experiments and analysis also find that the running time and space requirements of SST are better even than RRT, which can quickly provide feasible trajectories.

In regard to space, SST builds a sparse data structure, in contrast to other tree-based methods. Instead of asymptotically requiring an infinite number of states to be stored, SST keeps an exponential number of witnesses for any positive

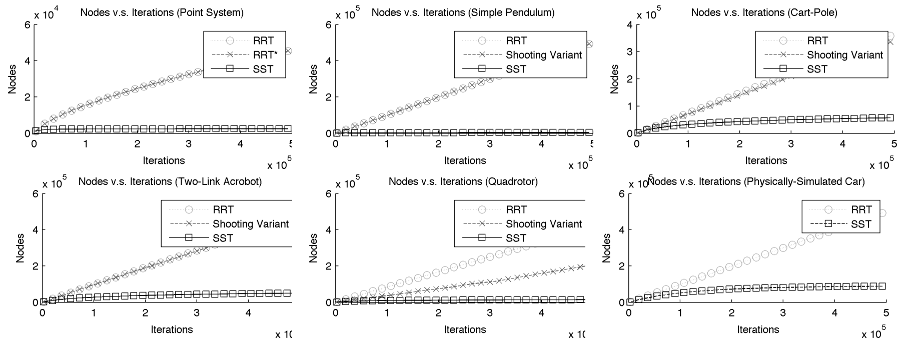

Fig. 8. Number of nodes in the tree (RRT, RRT* or the shooting approach, and SST).

δ_v, δ_s values, making it more akin to grid-based approaches. SST still provides a benefit over grid-based approaches however because the exponential number of states that SST has to store is asymptotic, meaning that a solution could be found before the exponential bound is reached. Other techniques such as branch-and-bound can mitigate the space required. This is in contrast to the grid based approaches where the grid representation limits feasibilities and optimizations. In addition, in grid-based methods, the grid points are usually constrained by the resolution of the grid, making the discrepancy and dispersion unfavorable [15]. While in SST, since it is built on sampling-based solutions, its points aren't constrained in that way.

Also, by removing the requirement of the steering function, SST is well suited to research in other areas where steering functions are difficult to construct. One of these areas is planning under uncertainty, where planning is performed in belief space. It is impossible to find a steering function in this domain, but forward propagation can update the probability distributions. It is also important to evaluate the effectiveness of the approach on real systems with significant dynamics, high-dimensional state spaces, and in cluttered spaces.

References

1. Bullet Physics Engine. http://bulletphysics.org
2. Akgun, B., Stilman, M.: Sampling Heuristics for Optimal Motion Planning in High Dimensions. In: IROS (2011)
3. Arslan, O., Tsiotras, P.: Use of Relaxation Methods in Sampling-Based Algorithms for Optimal Motion Planning. In: ICRA (2013)
4. Choset, H., Lynch, K.M., Hutchinson, S., Kantor, G., Burgard, W., Kavraki, L.E., Thrun, S.: Principles of Robot Motion. The MIT Press (2005)
5. Denny, J., Morales, M.M., Rodriguez, S., Amato, N.M.: Adapting RRT Growth for Heterogeneous Environments. In: IROS. Tokyo, Japan (2013)
6. Dobson, A., Bekris, K.: Sparse Roadmap Spanners for Asymptotically Near-Optimal Motion Planning (2013)
7. Gayle, R., Segars, W., Lin, M.C., Manocha, D.: Path Planning for Deformable Robots in Complex Environments. In: Robotics: Science and Systems (2005)
8. Goretkin, G., Perez, A., Platt, R., Konidaris, G.: Optimal Sampling-based Planning for Linear-Quadratic Kinodynamic Systems. In: ICRA (2013)

9. Hsu, D., Kindel, R., Latombe, J.C., Rock, S.: Randomized Kinodynamic Motion Planning with Moving Obstacles. IJRR **21**(3), 233–255 (2002)
10. Islam, F., Nasir, J., Malik, U., Ayaz, Y., Hasan, O.: RRT*-Smart: Rapid convergence implementation of RRT* towards optimal solution. In: ICMA (2012)
11. Jaillet, L., Cortés, J., Siméon, T.: Sampling-based Path Planning on Configuration-Space Costmaps. IEEE TRO **26**(4), 635–646 (2010)
12. Jeon, J.H., Cowlagi, R., Peters, S., Karaman, S., Frazzoli, E., Tsiotras, P., Iagnemma, K.: Optimal Motion Planning with the Half-Car Dynamical Model for Autonomous High-Speed Driving. In: ACC (2013)
13. Jeon, J.H., Karaman, S., Frazzoli, E.: Anytime Computation of Time-Optimal Off-Road Vehicle Maneuvers using the RRT*. In: CDC (2011)
14. Karaman, S., Frazzoli, E.: Sampling-based Algorithms for Optimal Motion Planning. IJRR **30**(7), 846–894 (2011)
15. LaValle, S.M., Branicky, M.S.: On the relationship between classical grid search and probabilistic roadmaps. In: WAFR. Nice, France (2002)
16. LaValle, S.M., Kuffner, J.J.: Randomized Kinodynamic Planning. IJRR **20**(5), 378–400 (2001)
17. Li, Y., Littlefield, Z., Bekris, K.: Asymptotically Optimal Sampling-based Kinodynamic Planning ((submitted: 10 July 2014)). Http://arxiv.org/abs/1407.2896
18. Likhachev, M., Gordon, G.J., Thrun, S.: ARA*: Anytime A* with Provable Bounds on Sub-Optimality. In: NIPS (2004)
19. Littlefield, Z., Li, Y., Bekris, K.: Efficient Sampling-based Motion Planning with Asymptotic Near-Optimality Guarantees with Dynamics. In: IROS (2013)
20. Marble, J.D., Bekris, K.: Asymptotically Near-Optimal Planning With Probabilistic Roadmap Spanners (2013)
21. Matérn, B.: *Spatial Variation* second ed. vol. 36 of *Lecture Notes in Statistics.* Springer-Verlag, New York, NY. (1986)
22. Nechushtan, O., Raveh, B., Halperin, D.: Sampling-Diagrams Automata : a Tool for Analyzing Path Quality in Tree Planners. In: WAFR (2010)
23. Papadopoulos, G., Kurniawati, H., Patrikalakis, N.M.: Asymptotically Optimal Inspection Planning using Systems with Differential Constraints. In: ICRA (2013)
24. Plaku, E., Kavraki, L.E., Vardi, M.Y.: Motion Planning with Dynamics by a Synergistic Combination of Layers of Planning. IEEE TRO **26**(3), 469–482 (2010)
25. Rickert, M., Brock, O., Knoll, A.: Balancing Exploration and Exploitation in Motion Planning. In: ICRA (2008)
26. Rodriguez, S., Tang, X., Lien, J.M., Amato, N.M.: An Obstacle-Based Rapidly-Exploring Random Tree. In: ICRA (2005)
27. Salzman, O., Halperin, D.: Asymptotically near-optimal RRT for fast, high-quality, motion planning. Tech. rep., Tel Aviv University (2013)
28. Shaharabani, D., Salzman, O., Agarwal, P., Halperin, D.: Sparsification of motion-planning roadmaps by edge contraction. In: ICRA (2013)
29. Shkolnik, A., Walter, M., Tedrake, R.: Reachability-Guided Sampling for Planning under Differential Constraints. In: ICRA (2009)
30. Urmson, C., Simmons, R.: Approaches for Heuristically Biasing RRT Growth. In: IROS, pp. 1178–1183 (2003)
31. Wang, W., Balkcom, D., Chakrabarti, A.: A fast streaming spanner algorithm for incrementally constructing sparse roadmaps. In: IROS (2013)
32. Webb, D., van Den Berg, J.: Kinodynamic RRT*: Asymptotically Optimal Motion Planning for Robots with Linear Differential Contstraints. In: ICRA (2013)