2023 IEEE International Conference on Robotics and Automation (ICRA 2023)
May 29 - June 2, 2023. London, UK

LES: Locally Exploitative Sampling for Robot Path Planning

Sagar Suhas Joshi[1] Seth Hutchinson[2] Panagiotis Tsiotras[3]

***Abstract*— Sampling-based algorithms solve the path planning problem by generating random samples in the search-space and incrementally growing a connectivity graph or a tree. Conventionally, the sampling strategy used in these algorithms is biased towards exploration to acquire information about the search-space. In contrast, this work proposes an optimization-based procedure that generates new samples so as to improve the cost-to-come value of vertices in a given neighborhood. The application of the proposed algorithm adds an exploitative-bias to sampling and results in a faster convergence to the optimal solution compared to other state-of-the-art sampling techniques. This is demonstrated using benchmarking experiments performed for 7 DOF Panda and 14 DOF Baxter robots.**

I. INTRODUCTION

Sampling-based motion planning (SBMP) algorithms have become the default choice for solving complex robotic planning tasks due to their scalability to higher dimensional problems. These algorithms do not resort to discretization or explicit construction of the search-space. Instead, popular single-query SBMP algorithms such as RRT [1], [2] and multi-query algorithms such as PRM [3], [4] use a black-box collision checking function to probe a set of random samples and local connections to incrementally build a connectivity graph. These algorithms are *probabilistically complete*, i.e., the probability of finding a feasible solution, if it exists, approaches unity as the number of samples tends to infinity.

Asymptotically optimal variants of RRT, such as RRT* [5], converge to the optimal solution almost-surely. These algorithms comprise of two fundamental modules, namely, graph-growth and graph-processing. The graph-growth module generates random samples, performs nearest neighbor, local steering and collision checking calculations to build a connectivity graph during planning time. The graph-processing module then tries to improve the cost-to-come value of the vertices by performing operations such as edge rewiring. The graph is said to be rewired if the parent of a vertex changes, improving its cost-to-come value. In particular, the "local rewiring" procedure of RRT* first selects the best parent for a newly initialized vertex and then checks if this new vertex can be a better parent for any of the vertices in its neighborhood. The RRT$^{\#}$ [6] algorithm provides an extension to the RRT* procedure by "globally rewiring" the graph using dynamic programming. It uses value-iteration [6] or policy-iteration [7] to optimally connect each vertex in the graph in order to minimize their cost-to-come values. Recently proposed methods such as BIT* [8] and FMT* [9] also use ideas from dynamic programming and heuristics to obtain faster convergence than RRT*.

[1,2,3] Institute for Robotics and Intelligent Machines, Georgia Institute of Technology, USA. Email: {sagarsjoshi94, seth, tsiotras}@gatech.edu

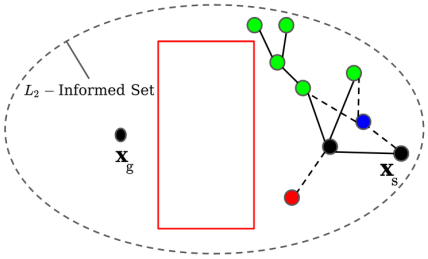

Fig. 1: Schematic motivating the proposed LES algorithm, which leverages local information and considers an optimization problem to generate the blue sample. In contrast to the red sample, the blue sample can initiate rewirings and improve cost-to-come value of (green) vertices in the graph.

Using an intelligent sampling strategy in conjunction with these graph-processing methods is effective for accelerating the convergence of SBMP algorithms. Uniform random sampling, a widely used approach, biases the graph growth towards vertices with larger Voronoi regions in RRT-style methods [1]. This results in a rapid exploration of the search-space and is effective for finding an initial solution in single-query scenarios. However, this strategy, like many others in the literature (e.g., [2], [10], [11]), prioritizes acquisition of new information over the improvement of current paths in the planner's graph. This bias towards exploration can have a detrimental effect on convergence, especially in higher dimensions [12].

The algorithm proposed in this work aims to generate new samples that can improve the cost-to-come value of vertices and initiate rewirings. This is in contrast to exploration-biased techniques. The proposed algorithm first selects a vertex and then generates a new sample in its vicinity. This sample is generated by considering an optimization problem, wherein the objective is to minimize the sum of cost-to-come value of a chosen vertex and its randomly selected descendants. The proposed sampling algorithm thus leverages local information to provide an exploitative bias. The combination of global exploratory and locally exploitative sampling results in faster convergence for SBMP algorithms, as demonstrated by several benchmarking experiments.

II. RELATED WORK

Many approaches have been suggested to address the exploration-exploitation trade-off in SBMP. Authors in [13] generate samples near a randomly selected state on the current solution path. This local biasing technique increases

the probability of improving the current solution at the cost of exploring other homotopy classes. The RRM algorithm [14] adds edges to the current roadmap to balance exploration and refinement. Techniques such as [15], [16], [17], [10], [11], [18] use heuristics and obstacle information to guide search during planning. T-RRT [19] and its variants [20], [21] implement a transition-test to avoid unhindered exploration in high cost regions. These approaches provide a way to focus search during planning. However, they do not directly address the problem of improving the cost-to-come value of vertices through sampling.

Unlike the above approaches, Informed Sampling [12] and its variants [22], [23] avoid redundant exploration after an initial solution is discovered. It focuses search onto a subset of the search-space, called the Informed Set, that contains all the points that can potentially improve the current solution. Generating new samples in the Informed Set is thus a necessary (but not sufficient) condition to improve the current solution. Relevant Region [24], a subset of the Informed Set, leverages cost-to-come information from the planner's graph to further focus search during planning. The combination of Relevant Region and Informed Sampling results in accelerated convergence in uniform and general cost-space environments. However, these techniques do not generate samples to directly improve the cost-to-come value of vertices. Hence, some of the samples may fail to trigger any improvement in the planner's graph. The sampling algorithm proposed in this work also generates new samples in the Relevant Region to avoid redundant exploration. However, it does so by considering an optimization problem aimed towards improving the cost-to-come value of vertices in the graph. Application of the proposed sampling algorithm thus initiates a higher number of rewirings and results in a faster convergence. Please see Fig. 1 for an illustration of this.

Approaches combining sampling-based planning and local optimizers have also been explored. RABIT* [25] uses CHOMP [26] to get feasible, high quality edges connecting any two vertices during a global search performed by BIT*. However, RABIT* requires pre-computed domain information, such as an obstacle potential function, which may not be available in many practical problems. Volumetric Tree* [27] addresses this limitation by constructing an approximation of the obstacle-free configuration space on-the-fly. However, it relies on uniform random sampling for graph construction, which may lead to redundant exploration. DRRT [28] employs a gradient-descent based procedure in the graph-processing module. However, this "vertex movement" procedure incurs a very high computational cost due to the extra calls to the nearest-neighbor and collision-checking function to ensure feasibility. The proposed sampler does not require extra collision-checking/nearest-neighbor calculations and can be used in conjunction with any graph-processing module.

Statement of Contributions: The proposed sampling algorithm makes the following specific contributions.

1. Informed Sampling is a necessary condition to improve the current solution. However, it is not sufficient, as an "Informed sample" is not guaranteed to bring about improvements in the current solution or the cost-to-come value of vertices. The proposed algorithm specifically considers an optimization problem to generate new samples that can improve the cost-to-come value of the vertices. Thus, it can be seen as a way of addressing the above limitation of Informed Sampling.
2. The algorithm proposed in [24] only generates random samples in the Relevant Region set. The current work extends this framework by incorporating planner-tree's gradient information while sampling in the Relevant Region. Numerical experiments show that this results in extensive performance gains (in terms of cost convergence) in higher dimensions.
3. The computationally expensive "vertex movement" procedure of DRRT can result in a reduced exploration of the search-space. This can have a detrimental effect on the quality of solution returned by the planner (see Fig. 2). While the proposed sampler is more expensive than Informed or Relevant Region sampler, it is much more efficient computationally compared to DRRT. It can thus provide a right balance between exploration and exploitation.

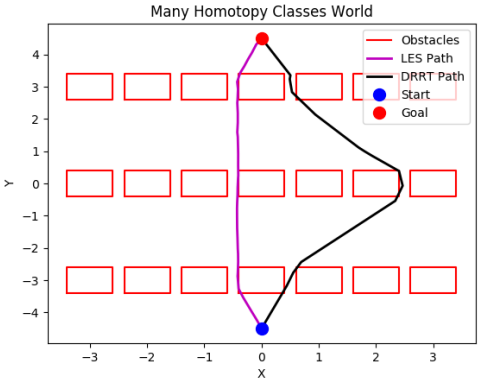

Fig. 2: Planning with the proposed LES-RRT$^{\#}$ and DRRT [28] in a world with multiple homotopy classes. The computationally costlier DRRT (black path) can get "stuck" in a wrong homotopy class. For the above trial run, LES-RRT$^{\#}$ and DRRT sampled 9414 and 573 graph vertices respectively in 1 second of planning time.

III. PROBLEM DEFINITION

A. Path Planning Problem

Consider the search-space $\mathcal{X} \subset \mathbb{R}^d$, with dimension d, $d \geq 2$. Let the obstacle space and free space be denoted by $\mathcal{X}_{\text{obs}}$ and $\mathcal{X}_{\text{free}}$ respectively. Then $\mathcal{X}_{\text{free}} = \text{cl}(\mathcal{X} \setminus \mathcal{X}_{\text{obs}})$, where $\text{cl}(A)$ represents closure of the set $A \subset \mathbb{R}^d$. Let the cost of moving from a point $\mathbf{x}_1 \in \mathcal{X}$ to $\mathbf{x}_2 \in \mathcal{X}$ along a path $\pi : [0, 1] \to \mathcal{X}$, $\pi(0) = \mathbf{x}_1$, $\pi(1) = \mathbf{x}_2$ be denoted by $\text{c}_\pi(\mathbf{x}_1, \mathbf{x}_2)$,

$$\text{c}_\pi(\mathbf{x}_1, \mathbf{x}_2) = \int_0^1 C(\pi(s)) \, \| \frac{\text{d}\pi(s)}{\text{d}s} \|_2 \, \text{d}s. \tag{1}$$

Here, $C : \mathcal{X} \to \mathbb{R}_{\geq 0}$ denotes a continuous state cost function. Note that (1) represents the integral of state-cost (IC) metric as a measure of path quality [19]. The path

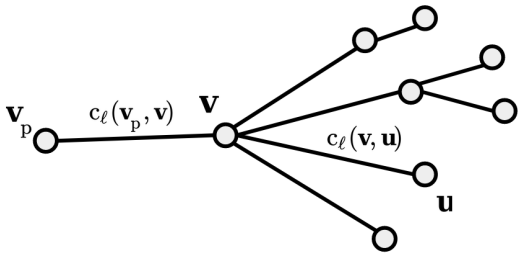

Fig. 3: Neighborhood around a vertex $\mathbf{v}$. Here, $n_\mathbf{v} = 4$ and $\widehat{d}_{\mathbf{v},V_\mathbf{v}} = 4 + (1 + 2) = 7$.

π in (1) is assumed to be collision free. The path-cost is infinite otherwise. The optimal path planning problem can be formally defined as the search for minimum cost path π^* from the set of feasible paths Π connecting the start state $\mathbf{x}_\mathrm{s} \in \mathcal{X}_\mathrm{free}$ to the goal region $\mathcal{X}_\mathrm{goal} \subset \mathcal{X}_\mathrm{free}$,

$$\begin{aligned} &\min_{\pi \in \Pi} \mathrm{c}_\pi(\mathbf{x}_\mathrm{s}, \mathbf{x}_\mathrm{g}), \\ &\text{subject to: } \pi(0) = \mathbf{x}_\mathrm{s},\ \pi(1) = \mathbf{x}_\mathrm{g} \in \mathcal{X}_\mathrm{goal}, \\ &\qquad \pi(s) \in \mathcal{X}_\mathrm{free}, \quad s \in [0, 1]. \end{aligned} \tag{2}$$

SBMP algorithms solve the above problem (2) by constructing a connectivity graph $\mathcal{G} = (V, E)$ with a finite set of vertices $V \subset \mathcal{X}_\mathrm{free}$ and a set of edges $E \subseteq V \times V$. The "geometric" versions of SBMP algorithms ignore the kino-dynamic constraints of the robot. Conventionally, these planners construct an edge $(\mathbf{u}, \mathbf{v}) \in E$ using a straight line path $\pi(s) = \mathbf{u} + (\mathbf{v} - \mathbf{u})s$, $s \in [0, 1]$ connecting $\mathbf{u}$ and $\mathbf{v}$. Using (1), the edge-cost can be denoted as

$$\mathrm{c}_\ell(\mathbf{u}, \mathbf{v}) = \|\mathbf{u} - \mathbf{v}\|_2 \int_0^1 C(\mathbf{u} + (\mathbf{v} - \mathbf{u})s)\ \mathrm{d}s. \tag{3}$$

SBMP algorithms can perform numerical integration to calculate the edge-cost $\mathrm{c}_\ell(\mathbf{u}, \mathbf{v})$ for any edge $(\mathbf{u}, \mathbf{v}) \in E$. The graph $\mathcal{G}$ embeds a spanning tree $\mathcal{T} = (V_t, E_t)$ with $V_t = V$ and $E_t = \{(\mathbf{u}, \mathbf{v}) \in E \mid \mathbf{v} = \mathsf{parent}(\mathbf{u})\}$. Here, $\mathsf{parent} : V \to V$ denotes the function mapping a vertex to its unique parent in the tree. By definition, we have $\mathsf{parent}(\mathbf{x}_\mathrm{s}) = \mathbf{x}_\mathrm{s}$. The cost-to-come value $\mathrm{g}_\mathcal{T}(\mathbf{v})$ for a vertex $\mathbf{v}$ denotes the sum of edge-costs along the path from $\mathbf{v}$ to the root $\mathbf{x}_\mathrm{s}$ in $\mathcal{T}$. The function $\mathrm{g}_\mathcal{T} : V \to \mathbb{R}_{\geq 0}$ can be written recursively as

$$\mathrm{g}_\mathcal{T}(\mathbf{v}) = \mathrm{g}_\mathcal{T}(\mathbf{v}_\mathrm{p}) + \mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}), \tag{4}$$

where $\mathbf{v}_\mathrm{p} = \mathsf{parent}(\mathbf{v})$. By definition, the recursion ends at $\mathbf{x}_\mathrm{s}$ with $\mathrm{g}_\mathcal{T}(\mathbf{x}_\mathrm{s}) = 0$. Let the set of children for vertex $\mathbf{v}$ be denoted by $V_\mathbf{v} = \{\mathbf{u} \in V \mid \mathbf{v} = \mathsf{parent}(\mathbf{u})\}$ and the number of children by $n_\mathbf{v} = |V_\mathbf{v}|$. Descendants of a vertex $\mathbf{v}$ are all the vertices $\mathbf{u} \in V$ whose path from $\mathbf{u}$ to the root $\mathbf{x}_\mathrm{s}$ in $\mathcal{T}$ contains $\mathbf{v}$. Let $D_\mathbf{v}$ denote the set of vertices that are descendants of $\mathbf{v}$ and let $d_\mathbf{v} = |D_\mathbf{v}|$. Then,

$$d_\mathbf{v} = n_\mathbf{v} + \sum_{\mathbf{u} \in V_\mathbf{v}} d_\mathbf{u}. \tag{5}$$

Note that for a leaf vertex $\mathbf{v} \in V$, we have $n_\mathbf{v} = d_\mathbf{v} = 0$. Let $\mathrm{h} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}_{\geq 0}$ denote a consistent heuristic function. This function obeys the triangle inequality and gives an under-estimate of the path-cost $\mathrm{c}_\pi(\mathbf{x}_1, \mathbf{x}_2)$ between any two points $\mathbf{x}_1, \mathbf{x}_2 \in \mathcal{X}$. An example of function h is the L_2-norm (Euclidean distance).

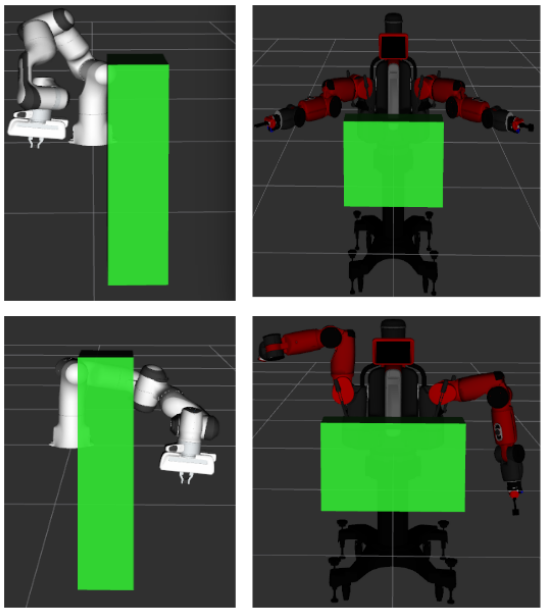

Fig. 4: Planning in the joint space of Panda and Baxter manipulator arms. The start and goal positions for both robots are indicated in the top and bottom figures respectively. A video of full robot plan can be found here: `https://youtu.be/J4B5_L2Ghrs`

Let $\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o})$ denote an ϵ-ball around $\mathbf{x}_\mathrm{o} \in \mathcal{X}$, given by $\mathcal{B}^\epsilon(\mathbf{x}_\mathrm{o}) = \{\mathbf{x} \in \mathcal{X} \mid \|\mathbf{x} - \mathbf{x}_\mathrm{o}\|_2 < \epsilon\}$, for $\epsilon > 0$. Finally, let $\mu(A)$ denote the Lebesgue measure of the set $A \subset \mathbb{R}^d$.

B. Optimization Problem for Sampling

Given $\mathcal{G}$, the objective of the graph-processing module is to minimize the cost-to-come value of all vertices. This objective can be written as

$$J_\mathcal{T} = \sum_{\mathbf{u} \in V} \mathrm{g}_\mathcal{T}(\mathbf{u}). \tag{6}$$

Let $J_\mathcal{T}(\mathbf{v})$ denote the terms of $J_\mathcal{T}$ that are dependent only on a particular vertex $\mathbf{v} \in V$. The position of vertex $\mathbf{v}$ impacts the cost-to-come value of itself and its descendants. Then,

$$J_\mathcal{T}(\mathbf{v}) = \mathrm{g}_\mathcal{T}(\mathbf{v}) + \sum_{\mathbf{w} \in D_\mathbf{v}} \mathrm{g}_\mathcal{T}(\mathbf{w}). \tag{7}$$

Using (4), the above equation for $J_\mathcal{T}(\mathbf{v})$ can be written in terms of the edge-costs $\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v})$ and $\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$. Here, $\mathbf{v}_\mathrm{p} = \mathsf{parent}(\mathbf{v})$ and $\mathbf{u}$ is any child of $\mathbf{v}$. The edge-cost $\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v})$ will appear $1 + d_\mathbf{v}$ times in total, to calculate the cost-to-come value of $\mathbf{v}$ and its descendants. Similarly, the edge-cost $\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$ will appear $1 + d_\mathbf{u}$ times in total, to calculate the cost-to-come value of $\mathbf{u}$ and its descendants. Then,

$$J_\mathcal{T}(\mathbf{v}) = k_1 + (1 + d_\mathbf{v})\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}) + \sum_{\mathbf{u} \in V_\mathbf{v}} (1 + d_\mathbf{u})\mathrm{c}_\ell(\mathbf{v}, \mathbf{u}). \tag{8}$$

Note that equation (8) for $J_\mathcal{T}(\mathbf{v})$ only contains terms dependent on $\mathbf{v}$. Other terms are incorporated in the constant k_1.

Algorithm 1: LES Algorithm Flow

1 $V \leftarrow \{\mathbf{x}_\mathrm{s}\}$; $E \leftarrow \phi$; $\mathcal{G} \leftarrow (V, E)$;
2 **for** $i = 1 : N$ **do**
3 $\quad c_i \leftarrow \mathsf{getBestSolutionCost}()$;
4 $\quad u_\mathrm{rand} \sim \mathcal{U}(0, 1)$;
5 $\quad$ **if** $u_\mathrm{rand} < p_\mathrm{LES}$ and $c_i < \infty$ **then**
6 $\quad\quad \mathbf{v} \leftarrow \mathsf{chooseVertex}(V_\mathrm{rel})$;
7 $\quad\quad \hat{\mathbf{e}} \leftarrow \mathsf{getGradientDirection}(\mathbf{v})$;
8 $\quad\quad \gamma \leftarrow \mathsf{getStepSize}(\mathbf{v}, \hat{\mathbf{e}})$;
9 $\quad\quad \mathbf{x}_\mathrm{rand} \leftarrow \mathbf{v} - \gamma\hat{\mathbf{e}}$;
10 $\quad$ **else**
11 $\quad\quad \mathbf{x}_\mathrm{rand} \leftarrow \mathsf{InformedSampling}(\mathsf{c_i})$
12 $\quad \mathsf{Extend}(\mathbf{x}_\mathrm{rand})$;
13 $\quad \mathsf{GraphProcessing}(\mathcal{G})$;
14 **return** $\mathcal{G}$

Also, $d_\mathbf{v}$ and $d_\mathbf{u}$ in (8) are linked by equation (5). A new sample can be generated by first selecting a vertex $\mathbf{v}$ and then finding a "better" position for it by optimizing $J_\mathcal{T}$ with respect to $\mathbf{v}$. Note that $\arg\min_\mathbf{v} J_\mathcal{T} = \arg\min_\mathbf{v} J_\mathcal{T}(\mathbf{v})$.

However, calculating the values of the coefficients $d_\mathbf{v}, d_\mathbf{u}$ in (8) requires a depth-first search with time complexity of $O(|V_t|)$. This may get computationally cumbersome, especially as the planner tree grows larger with the number of iterations. Hence, the following objective function is considered instead,

$$\widehat{J}_{\mathcal{T},V_\mathbf{v}}(\mathbf{v}) = k_2 + (1 + \widehat{d}_{\mathbf{v},V_\mathbf{v}})\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}) + \sum_{\mathbf{u}\in V_\mathbf{v}} (1 + n_\mathbf{u})\mathrm{c}_\ell(\mathbf{v}, \mathbf{u}),$$
$$\widehat{d}_{\mathbf{v},V_\mathbf{v}} = n_\mathbf{v} + \sum_{\mathbf{u}\in V_\mathbf{v}} n_\mathbf{u}. \tag{9}$$

Please see Fig. 3. Note that minimizing $\widehat{J}_{\mathcal{T},V_\mathbf{v}}(\mathbf{v})$ in (9) with respect to $\mathbf{v}$ is equivalent to minimizing the cost-to-come values of $\mathbf{v}$, the set of children $V_\mathbf{v}$ and their children. The objective $\widehat{J}_{\mathcal{T},V_\mathbf{v}}(\mathbf{v})$ can be calculated efficiently with the information contained in the data structure of vertex $\mathbf{v}$, without recursing deeper down the tree. Effectively, $\widehat{J}_{\mathcal{T},V_\mathbf{v}}(\mathbf{v})$ considers descendants of $\mathbf{v}$ up to a depth of 2, whereas $J_\mathcal{T}(\mathbf{v})$ considers full depth. This can be generalized to depth-k descendants.

Finally, a random subset of the children, denoted by $\widehat{V}_\mathbf{v} \subseteq V_\mathbf{v}$, can be selected and a new sample generated by minimizing $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{v})$. This serves two purposes. First, it promotes a desirable randomness in the sampling process. Second, focusing on the subset $\widehat{V}_\mathbf{v}$ effectively assigns a weight of zero for the terms corresponding to the vertices $V_\mathbf{v} \setminus \widehat{V}_\mathbf{v}$ in the objective (8), (9). This can lead to a better improvement in the cost-to-come value of vertices corresponding to $\widehat{V}_\mathbf{v}$.

IV. LOCALLY EXPLOITATIVE SAMPLING

The proposed "Locally Exploitative Sampling (LES)" procedure first selects a vertex $\mathbf{v}$ and then generates a new sample considering $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{v})$. Expansive Space Trees (EST) [29] and its variants, such as [16], [17], also proceed by selecting a vertex and generating a random sample in its vicinity. However, the probability of generating a "good" sample (that can improve $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{v})$) with such random search decreases rapidly in higher dimensions. This is illustrated in the supplementary material[1] by considering the problem of minimizing a quadratic function $J_q(\mathbf{x}) = \mathbf{x}^\mathsf{T}\mathbf{x}$ with random local search. The probability of generating a sample that can improve J_q diminishes exponentially with the dimension d.

Algorithm 2: Calculate Gradient Direction

1 **getGradientDirection** ($\mathbf{v}$) **:**
2 $\quad \widehat{V}_\mathbf{v} \leftarrow \mathsf{getRandomSubset}(V_\mathbf{v})$;
3 $\quad \mathbf{e} \leftarrow (1 + \widehat{d}_{\mathbf{v},\widehat{V}_\mathbf{v}})\frac{\partial}{\partial \mathbf{v}}\mathrm{c}_\ell(\mathbf{v}_\mathrm{p}, \mathbf{v}) + \sum_{\mathbf{u}\in\widehat{V}_\mathbf{v}}(1 + n_\mathbf{u})\frac{\partial}{\partial \mathbf{v}}\mathrm{c}_\ell(\mathbf{v}, \mathbf{u})$;
4 $\quad \hat{\mathbf{e}} \leftarrow \mathbf{e}/\|\mathbf{e}\|_2$
5 **return** $\hat{\mathbf{e}}$

Algorithm 3: Calculate Step-size

1 **getStepSize** ($\mathbf{v}, \hat{\mathbf{e}}$) **:**
2 $\quad \gamma_\mathrm{rel} \leftarrow \mathsf{getMaxStepSize}(\mathbf{v}, \hat{\mathbf{e}})$;
3 $\quad \gamma_\mathrm{max} \leftarrow \gamma_\mathrm{rel}$;
4 $\quad$ **while** $\gamma_\mathrm{max} > \delta$ **do**
5 $\quad\quad u_\mathrm{rand} \sim \mathcal{U}(0, 1)$; $\gamma \leftarrow (u_\mathrm{rand})^{1/d}\gamma_\mathrm{max}$;
6 $\quad\quad \mathbf{x}_\mathrm{rand} \leftarrow \mathbf{v} - \gamma\hat{\mathbf{e}}$;
7 $\quad\quad$ **if** $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{x}_\mathrm{rand}) < \widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{v})$ **then**
8 $\quad\quad\quad$ break;
9 $\quad\quad$ **else**
10 $\quad\quad\quad \gamma_\mathrm{max} \leftarrow \gamma$;
11 $\quad$ **if** $\gamma_\mathrm{max} < \delta$ **then**
12 $\quad\quad u_\mathrm{rand} \sim \mathcal{U}(0, 1)$; $\gamma \leftarrow (u_\mathrm{rand})^{1/d}\gamma_\mathrm{rel}$;
13 **return** γ

This motivates the LES procedure, given in Algorithm 1. With probability p_LES, LES is used to generate a new sample $\mathbf{x}_\mathrm{rand}$ (Algorithm 1, line 6-8). Otherwise, a new sample is generated using the conventional Informed Sampling technique given in [12]. This ensures a balance between exploration-exploitation (controlled by the parameter p_LES) and graph growth in all the relevant homotopy classes. The Extend function takes this random sample and performs relevant procedures (nearest-neighbor, local steering and collision checking) to incorporate a new vertex in the graph. Finally, the graph-processing module operates on $\mathcal{G}$ considering the addition of a new vertex.

After an initial solution with cost c_i is discovered after i iterations, redundant exploration can be avoided by sampling the Informed or Relevant Region set. Due to its focusing properties, LES generates new samples in the Relevant Region $\mathcal{X}^\epsilon_\mathrm{rel}$ [24], defined as follows

$$\mathcal{X}^\epsilon_\mathrm{rel} = \bigcup_{\mathbf{v}\in V_\mathrm{rel}} \mathcal{B}^\epsilon_\mathrm{rel}(\mathbf{v}), \tag{10}$$

where,

$$\mathcal{B}^\epsilon_\mathrm{rel}(\mathbf{v}) = \{\mathbf{x} \in \mathcal{B}^\epsilon(\mathbf{v}) \mid \hat{f}_\mathbf{v}(\mathbf{x}) < c_i\},$$
$$\hat{f}_\mathbf{v}(\mathbf{x}) = \mathrm{c}_\ell(\mathbf{v},\mathbf{x}) + \mathrm{g}_\mathcal{T}(\mathbf{v}) + \mathrm{h}(\mathbf{x}, \mathbf{x}_\mathrm{g}), \tag{11}$$

[1] https://tinyurl.com/msuh7j44

and V_{rel} denotes the set of "relevant vertices",

$$V_{\text{rel}} = \{\mathbf{v} \in V \mid \mathrm{g}_{\mathcal{T}}(\mathbf{v}) + \mathrm{h}(\mathbf{v}, \mathbf{x}_{\mathrm{g}}) < c_i\}. \tag{12}$$

The value of ϵ in (10) is set to $\epsilon = 1.5\eta$, where η is the range parameter in SBMP algorithms [30], which controls the maximum edge-length in $\mathcal{G}$. The procedure for selecting a vertex (chooseVertex), is same as the implementation in [24]. It assigns a weight $q_{\mathbf{v}}$ for each $\mathbf{v} \in V_{\text{rel}}$ and uses a binary heap data-structure for sorting. Start, goal and leaf vertices (vertices with no children) are ignored by the chooseVertex function.

Note that, a closed-form solution to the optimization objective $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ cannot be obtained in general. Hence, LES proceeds by numerically calculating the gradient of $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ and moving an appropriate step-size in the direction of the gradient. The procedure to calculate the gradient direction $\hat{\mathbf{e}}$ is given in Algorithm 2. First, a random subset of children $\widehat{V}_{\mathbf{v}}$ is obtained. The gradient $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v})$ with respect to $\mathbf{v}$ is calculated numerically using the symmetric difference formula (Algorithm 2, line 3). Having obtained the gradient direction $\hat{\mathbf{e}}$, the algorithm to calculate the step-size is given in Algorithm 3. As finding the optimal step-size γ^* by solving $\arg\min_{\gamma} \widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}(\mathbf{v} - \gamma\hat{\mathbf{e}})$ is intractable, approaches such as backtracking line search [31] have been suggested. However, executing backtracking line search is computationally not viable for the current application, as it requires a higher number of expensive calls to calculate $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}$. Instead, LES uses a procedure given in Algorithm 3, which is similar to the Hit-and-Run Sampler implemented in [32]. First, given a vertex $\mathbf{v}$ and the travel direction $-\hat{\mathbf{e}}$, the procedure in [24] is used to calculate the maximum step-size γ_{rel}. This ensures that a candidate $\mathbf{v} - \gamma\hat{\mathbf{e}} \in \mathcal{X}^{\epsilon}_{\text{rel}}$ for any $\gamma \in (0, \gamma_{\text{rel}})$. Variable $\gamma_{\max}$ is set to γ_{rel}. Next, a random step-size γ is sampled from the interval $(0, \gamma_{\max})$. The exponent of $1/d$ in Algorithm 3, line 5 biases γ towards $\gamma_{\max}$. If the candidate $\mathbf{v} - \gamma\hat{\mathbf{e}}$ results in an improvement for $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}$, step-size γ is returned. Else, $\gamma_{\max}$ is updated to γ. Thus, the search interval is sequentially reduced until a suitable step-size is discovered. Theoretically, a travel of infinitesimal magnitude in the direction of the gradient always results in an improvement. However, if $\gamma_{\max}$ is less than a small quantity $\delta \ll \eta$, then a random γ in the interval $(0, \gamma_{\text{rel}})$ is returned (Algorithm. 3, line 11-12) to avoid clumping of new vertices around $\mathbf{v}$.

V. NUMERICAL EXPERIMENTS

Benchmarking was performed against Informed sampler and Relevant Region sampler described in [12] and [24] respectively. Note that LES and the Relevant Region sampler share a similar chooseVertex procedure. However, the Relevant Region sampler only generates random samples in $\mathcal{X}^{\epsilon}_{\text{rel}}$ and does not consider the optimization problem corresponding to (8) or (9). All three sampling strategies, namely, LES, Relevant Region and Informed used a goal bias of 5% and were paired with RRT$^{\#}$'s global rewiring for graph-processing. All the algorithms were implemented using C++/OMPL [30]. Data was gathered over 100 trials for each experiment using the standardized OMPL benchmarking tools [33]. All experiments were performed on a 64 bit laptop running Ubuntu 18.04 O.S, with 16 GB RAM and an Intel i7 processor. The parameter p_{LES} and an analogous parameter p_{rel} for Relevant Region sampler were both set to 0.5. Please see [24] for brief discussion about this. The parameter δ was set to 10^{-4}. A description of the different benchmarking environments is given below.

Many Homotopy Classes World: This case, similar to one studied in [12], is illustrated in Fig. 2. The objective is to find a length-optimal path in this cluttered environment. This world has multiple non-optimal homotopy classes, but only two optimal ones. Experiments were performed for a 4D and 6D version of this environment with the range parameter η set to 1.5 and 2.5 respectively. Benchmarking for this case also considered two variants of the DRRT planner, namely DRRTd and DRRT0.3 [28]. The DRRTd algorithm delays the computationally expensive, vertex optimization procedure until an initial solution is discovered, whereas DRRT0.3 calls that procedure only with a probability of 0.3.

Robot Manipulator: A planning problem for a 7 DOF Panda and a 14 DOF Baxter arm is illustrated in Fig. 4. The objective was to find the minimum length path ($C(\mathbf{x}) = 1$ for all $\mathbf{x} \in \mathcal{X}$) in the configuration-space with strict joint limits ($\mathcal{X} \subset \mathbb{R}^7$ for Panda, $\mathcal{X} \subset \mathbb{R}^{14}$ for Baxter). These joint limits and collision checking calculations were implemented with the help of MoveIt! [34]. The range parameter η was set to 1.2 and 2 for the Panda and Baxter experiments respectively. These experiments were performed with three variations of the proposed LES algorithm, namely LES-2, LES-8 and LES-∞. LES-2 and LES-8 consider descendants up to a maximum depth of 2 and 8 respectively, whereas LES-∞ considers full depth up to the leaf nodes.

Results from the numerical experiments are illustrated in Fig. 5. The proposed LES variations outperform Informed and Relevant Region samplers in higher dimensional settings (6D, Panda, Baxter) in terms of cost convergence. While the performance of all three LES variants is similar, LES-∞ and LES-8 perform slightly better than LES-2 in case of Panda. LES also initiates a larger number of rewirings in $\mathcal{T}$. However, similar performance gains are not seen in the lower dimensional environments (see 4D). In the 4D case of Fig. 2, the Relevant Region sampler with its focusing properties, performs better than Informed sampling and LES in terms of cost convergence. LES incurs a higher computational cost due to the numerical gradient calculations in Algorithm 2 and expensive function evaluations of $\widehat{J}_{\mathcal{T},\widehat{V}_{\mathbf{v}}}$ in Algorithm 3. Thus, the application of LES leads to a lesser number of iterations executed in a given time period compared to the other two methods. This might slow down convergence in lower dimensions. However, random search techniques are affected by the "curse of dimensionality", as illustrated in the supplementary material. This justifies the computationally more costly procedures of LES which lead to an accelerated convergence in higher dimensions.

The DRRT planner can show tremendous performance gains, in terms of cost-convergence, for higher dimensional,

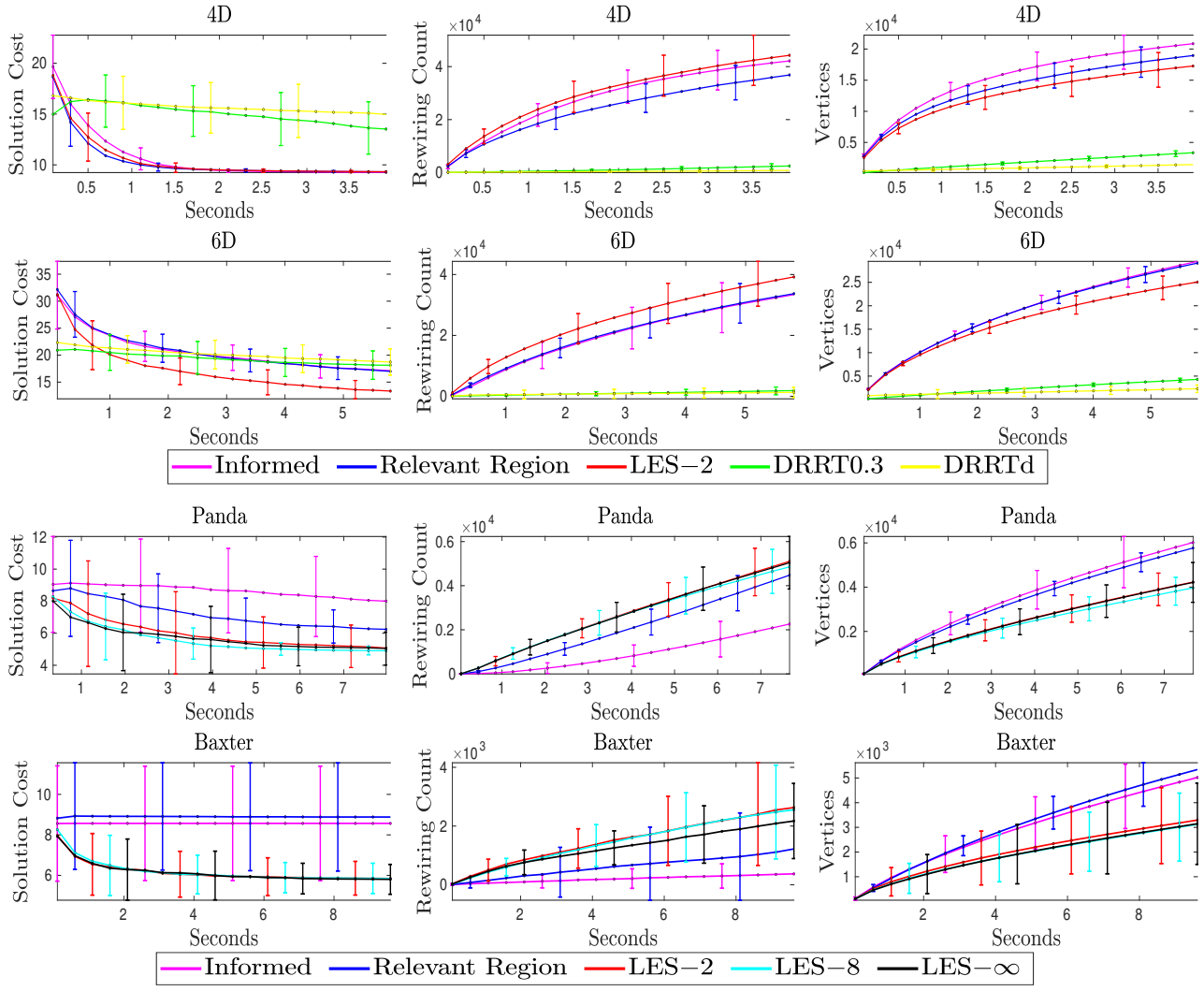

Fig. 5: Benchmarking plots for the numerical experiments. Solid lines indicate the value averaged over 100 trials and the error bars represent standard deviation. Application of the proposed LES algorithm leads to a faster convergence and a larger number of tree rewirings in higher dimensions. However, it incurs a higher computational cost and hence executes a lesser number of iterations compared to Informed (magenta) and Relevant Region (blue) sampling.

relatively less cluttered environments. However, for cases such as Fig. 2, the DRRT planner can find an initial solution in a non-optimal homotopy class. Its computationally expensive, vertex movement procedure requires additional collision-checking and nearest-neighbor calculations to maintain feasibility and graph neighborhoods. As a consequence, it generates significantly lesser number of vertices compared to the other methods (see Fig. 5). Hence, optimizing the graph comes at a steep cost of reduced exploration and finding solution paths in other homotopy classes for DRRT. It can thus get "stuck" in the wrong homotopy class. While LES also leverages the gradient information, it is much more computationally tractable than DRRT. It generates enough vertices to explore all relevant homotopy classes while also initiating higher number of rewirings to improve the cost-to-come values. Thus, LES strikes the right balance between exploration and exploitation and outperforms all other methods in a higher dimensional case such as 6D in Fig. 5.

VI. CONCLUSION

This work proposes a "Locally Exploitative Sampling" algorithm, that generates new samples to improve the cost-to-come value of vertices in a neighborhood. LES numerically calculates the gradient of $\widehat{J}_{\mathcal{T},\widehat{V}_\mathbf{v}}(\mathbf{v})$ and decides an appropriate step-size to obtain a new sample. Although computationally costlier than other sampling methods, LES adds an "exploitative-bias" that can initiate higher number of tree rewirings. This can accelerate convergence of SBMP algorithms, especially in higher dimensions.

LES presents many openings for future research. For instance, LES can be extended to kino-dynamic settings and be used with planners such SST [35]. Ideas from [10] can also be used to have an "obstacle-aware" version of LES.

Acknowledgements: This work has been supported by NSF awards IIS-1617630 and IIS-2008686.

REFERENCES

[1] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE International Conference on Robotics and Automation.*, vol. 2, San Franciso, CA, April 24–28 2000, pp. 995–1001.

[3] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *Transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[4] R. Bohlin and L. E. Kavraki, "Path planning using lazy PRM," in *IEEE International Conference on Robotics and Automation*, vol. 1, San Franciso, CA, April 24–28 2000, pp. 521–528.

[5] 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, June 2011.

[6] O. Arslan and P. Tsiotras, "Use of relaxation methods in sampling-based algorithms for optimal motion planning," in *IEEE International Conference on Robotics and Automation*, Karlsruhe, Germany, May 6–10 2013, pp. 2421–2428.

[7] ——, "Incremental sampling-based motion planners using policy iteration methods," in *IEEE 55th Conference on Decision and Control*, Las Vegas, NV, Dec. 12–15 2016, pp. 5004–5009.

[8] 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 *IEEE International Conference on Robotics and Automation*, Seattle, WA, May, 25–30 2015, pp. 3067–3074.

[9] 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, May 2015.

[10] T. Lai, P. Morere, F. Ramos, and G. Francis, "Bayesian local sampling-based planning," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1954–1961, 2020.

[11] S. Rodriguez, X. Tang, J.-M. Lien, and N. M. Amato, "An obstacle-based rapidly-exploring random tree," in *IEEE International Conference on Robotics and Automation*, Orlando, FL, May 15–19 2006, pp. 895–900.

[12] 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, Aug. 2018.

[13] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, Sept. 25–30 2011, pp. 2640–2645.

[14] R. Alterovitz, S. Patil, and A. Derbakova, "Rapidly-exploring roadmaps: Weighing exploration vs. refinement in optimal motion planning," in *IEEE International Conference on Robotics and Automation*, Shanghai, China, 2011, pp. 3706–3712.

[15] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *IEEE/RSJ International Conference on Intelligent Robots and Systems.*, vol. 2, Las Vegas, NV, Oct. 27–31 2003, pp. 1178–1183.

[16] J. M. Phillips, N. Bedrossian, and L. E. Kavraki, "Guided expansive spaces trees: a search strategy for motion-and cost-constrained state spaces," in *IEEE International Conference on Robotics and Automation*, New Orleans, LA, April 26–30 2004, pp. 3968–3973.

[17] S. M. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *The International Journal of Robotics Research*, vol. 33, no. 13, pp. 1683–1708, 10 2014.

[18] B. Burns and O. Brock, "Single-query motion planning with utility-guided random trees," in *IEEE International Conference on Robotics and Automation*, Rome, Italy, April 10–14 2007, pp. 3307–3312.

[19] L. Jaillet, J. Cortés, and T. Siméon, "Sampling-based path planning on configuration-space costmaps," *IEEE Transactions on Robotics*, vol. 26, no. 4, pp. 635–646, 8 2010.

[20] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces," in *IEEE International Conference on Robotics and Automation*, Karlsrühe, Germany, May 6–10 2013, pp. 4120–4125.

[21] ——, "Optimal path planning in complex cost spaces with sampling-based algorithms," *IEEE Transactions on Automation Science and Engineering*, vol. 13, no. 2, pp. 415–424, 2015.

[22] S. S. Joshi and T. Panagiotis, "Non-parametric informed exploration for sampling-based motion planning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 5915–5921.

[23] S. S. Joshi, S. Hutchinson, and P. Tsiotras, "TIE: Time-informed exploration for robot motion planning," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3585–3591, 2021.

[24] S. S. Joshi and P. Tsiotras, "Relevant region exploration on general cost-maps for sampling-based motion planning," in *International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV: IEEE/RSJ, Oct. 25–29 2020.

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

[26] M. Zucker, N. Ratliff, A. D. Dragan, M. Pivtoraiko, M. Klingensmith, C. M. Dellin, J. A. Bagnell, and S. S. Srinivasa, "CHOMP: Covariant Hamiltonian optimization for motion planning," *The International Journal of Robotics Research*, vol. 32, no. 9-10, pp. 1164–1193, 2013.

[27] D. Kim, M. Kang, and S.-E. Yoon, "Volumetric tree*: Adaptive sparse graph for effective exploration of homotopy classes," in *International Conference on Intelligent Robots and Systems (IROS)*. IEEE/RSJ, 2019, pp. 1496–1503.

[28] F. Hauer and P. Tsiotras, "Deformable rapidly-exploring random trees." in *Robotics: Science and Systems*, Cambridge, MA, July 12–16 2017.

[29] D. Hsu, J.-C. Latombe, and R. Motwani, "Path planning in expansive configuration spaces," in *IEEE International Conference on Robotics and Automation*, vol. 3, Albuquerque, NM, April 25–29 1997, pp. 2719–2726.

[30] 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, Dec. 2012.

[31] S. Boyd and L. Vandenberghe, *Convex Optimization*. Cambridge University Press, 2004.

[32] D. Yi, R. Thakker, C. Gulino, O. Salzman, and S. Srinivasa, "Generalizing informed sampling for asymptotically-optimal sampling-based kinodynamic planning via Markov Chain Monte Carlo," in *IEEE International Conference on Robotics and Automation (ICRA)*, Brisbane, Australia, May 21–25 2018, pp. 7063–7070.

[33] 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.

[34] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ROS topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[35] Y. Li, Z. Littlefield, and K. E. Bekris, "Sparse methods for efficient asymptotically optimal kinodynamic planning," in *Algorithmic Foundations of Robotics XI*. Springer, 2015, pp. 263–282.