Generalizing Informed Sampling for Asymptotically-Optimal Sampling-based Kinodynamic Planning via Markov Chain Monte Carlo

Daqing Yi*[1], Rohan Thakker*[2], Cole Gulino[2], Oren Salzman[2] and Siddhartha Srinivasa[1]

***Abstract*— Asymptotically-optimal motion planners such as RRT* have been shown to incrementally approximate the shortest path between start and goal states. Once an initial solution is found, their performance can be dramatically improved by restricting subsequent samples to regions of the state space that can potentially improve the current solution. When the motion-planning problem lies in a Euclidean space, this region $\mathcal{X}_{\text{inf}}$, called the informed set, can be sampled directly. However, when planning with differential constraints in non-Euclidean state spaces, no analytic solutions exists to sampling $\mathcal{X}_{\text{inf}}$ directly.**

State-of-the-art approaches to sampling $\mathcal{X}_{\text{inf}}$ in such domains such as Hierarchical Rejection Sampling (HRS) may still be slow in high-dimensional state space. This may cause the planning algorithm to spend most of its time trying to produces samples in $\mathcal{X}_{\text{inf}}$ rather than explore it. In this paper, we suggest an alternative approach to produce samples in the informed set $\mathcal{X}_{\text{inf}}$ for a wide range of settings. Our main insight is to recast this problem as one of sampling uniformly within the sub-level-set of an implicit non-convex function. This recasting enables us to apply Monte Carlo sampling methods, used very effectively in the Machine Learning and Optimization communities, to solve our problem. We show for a wide range of scenarios that using our sampler can accelerate the convergence rate to high-quality solutions in high-dimensional problems.

I. Introduction

Sampling-based motion-planning algorithms [1] have proven to be an effective tool at solving motion-planning problems. They search through a continuous state space $\mathcal{X}$ by sampling random states and maintaining a discrete graph G called a *roadmap*. Vertices and edges in G correspond to collision-free states and paths, respectively.

Roughly speaking, these algorithms iteratively sample new states. This is required to ensure that, as the number of samples tends to infinity, (i) a solution will be found and that (ii) given some optimization criteria, the quality of the solution will progressively converge to the quality of the optimal solution.

Initially, when a path has yet to be found, the samples are drawn from the entire state space $\mathcal{X}$. However, once a path γ is produced, algorithms that seek *high-quality paths* can limit their sampling domain to a subset of $\mathcal{X}$ only containing states that may be used to produce higher-quality paths than γ. Following Gammell et al. [2], we call this subset the *informed subset* and denote it $\mathcal{X}_{\text{inf}}$. In this work we address the problem of efficiently producing samples in $\mathcal{X}_{\text{inf}}$ for systems with arbitrary complex costs.

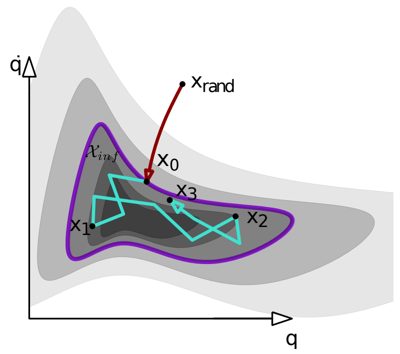


Fig. 1: Algorithmic approach. Cost function is depicted using iso-contours (darker shades reflect lower cost) while the boundary of the informed set is depicted in purple. The root-finding and MCMC algorithms are depicted in red and turquoise, respectively. x_0 lies on the boundary of $\mathcal{X}_{\text{inf}}$

For Euclidean spaces optimizing for path length, $\mathcal{X}_{\text{inf}}$ can be analytically expressed as a prolate hyperspheroid and can be sampled directly using a closed-form solution [2]. Indeed, directly sampling in $\mathcal{X}_{\text{inf}}$ has been shown to dramatically improve computation time when compared to sampling in $\mathcal{X}$, especially in high dimensions.

Unfortunately, in more general settings, it is not clear how to directly sample $\mathcal{X}_{\text{inf}}$. One approach to produce samples in $\mathcal{X}_{\text{inf}}$ is via *rejection sampling*—sampling a state $x \in \mathcal{X}$ and testing if $x \in \mathcal{X}_{\text{inf}}$. However, when the size of the informed space $\mathcal{X}_{\text{inf}}$ is much smaller than entire state space $\mathcal{X}$, this procedure is highly inefficient, dominating the running time of the algorithm [3]. Recently, Kunz et al. [3] showed, under some technical assumptions, how to partially ameliorate this inefficiently by *Hierarchical rejection sampling* (HRS). Here, individual dimensions are sampled recursively and then combined. Rejection sampling is performed for these partial samples until a suitable sample has been produced. Unfortunately, HRS may still produce a large number of rejected samples especially in high-dimensional spaces [3]. This may cause the planning algorithm to spend most of its time trying to produces samples in $\mathcal{X}_{\text{inf}}$ rather than explore it.

In this paper, we suggest an alternative approach to produce samples in the informed set $\mathcal{X}_{\text{inf}}$ for a wide range of settings. **Our main insight is to recast this problem as one of sampling uniformly within the sub-level-set of an implicit non-convex function. This recasting enables us to apply Monte Carlo sampling methods, used very effectively in the Machine Learning and Optimization**

*Daqing Yi and Rohan Thakker contributed equally to this paper.

[1]Daqing Yi and Siddhartha Srinivasa are with Paul G. Allen School of Computer Science & Engineering, University of Washington. {`dqyi, siddh`}`@cs.washington.edu`

[2]Rohan Thakker, Cole Gulino and Oren Salzman are with Robotics Institute, Carnegie Mellon University. {`rthakker, cgulino, osalzman`} `@andrew.cmu.edu`

communities, to solve our problem. Specifically, our approach, depicted in Fig. 1 consists of two stages: in the first, a random sample $x \in \mathcal{X}$ is retracted to the boundary of $\mathcal{X}_{\text{inf}}$ by running a root-finding algorithm; in the second stage, this retracted sample is used to seed a Monte Carlo sampling chain which allows us to produce samples that (approximately) cover $\mathcal{X}_{\text{inf}}$ uniformly.

While our approach can be used with any Markov Chain Monte Carlo (MCMC) method, it is especially suited to be used with Hit-and-Run [4]. Roughly speaking, this is because Hit-and-Run (detailed in Sec. V) produces a series of one-dimensional rejection samples which are extremely fast to compute, even in high-dimensional spaces.

Our approach requires that the system has a solution to the two-point boundary value problem (2pBVP) [1] and that a gradient can be defined over the cost function. Indeed, we demonstrate the efficiency of our approach in several systems and show that it has the potential of reducing the planning time by several orders of magnitude when compared to algorithms using rejection sampling or HRS.

The rest of the paper is structured as follows: after describing related work in Sec. II, we formally define our problem in Sec. III. We then provide in Sec. IV an intuitive description of the challenges faced in sampling within the informed set for our planning domains. We continue in Sec. V with a description of our algorithm and present experimental evaluations in Sec. VI. Finally, we conclude with a discussion in Sec. VII.

II. Related work

We start in Sec. II-A by giving an overview of relevant sampling-based motion-planning algorithms. We then continue in Sec. II-B to describe different approaches that can be used by these algorithms to sample $\mathcal{X}$. We conclude our literature review in Sec. II-C with a brief overview of Markov Chain Monte Carlo methods.

A. Sampling-based motion-planning algorithms

Initial sampling-based algorithms such as RRT [5] and PRM [6] did not take into account the *quality* of a path, given some optimization criteria, and only guaranteed to asymptotically return *a* solution, if one exists. Karaman and Frazzoli [7], presented variants of PRM and RRT, named PRM* and RRT*, respectively that were shown to produce paths who's cost converges asymptotically to the minimal-cost path. This was done by recognizing the underlying connections between stochastic sampling-based motion planning and the theory of random geometric graphs (see also [8]). Additional algorithms followed, increasing the converges rate by various techniques such as lazy dynamic programming [9], [10], relaxing optimality to near-optimality [11], [12] and more.

Many of the algorithms mentioned require solving a two-point boundary value problem (2pBVP) to perform exact and optimal connections between vertices in the roadmap. For holonomic robots, these are simply straight lines in the configuration space, but for kinodynamic sytems with arbitrary cost functions, computing an optimal trajectory between two states is non-trivial in general.

Xie et al. [13] use a variant of sequential quadratic programming (SQP) to solve 2pBVP and integrate it with BIT* [9]. Webb and van den Berg [14] use a fixed-final-state-free-final-time controller to solve the 2pBVP with respect to a cost function that allows for balancing between the duration of the trajectory and the expended control effort. Perez et al. [15] propose a variant of RRT* that automatically defines a distance metric and node extension method by locally linearizing the domain dynamics and applying linear quadratic regulation (LQR).

Finally, we note that we are not the first to integrate Monte Carlo sampling into planning algorithms. T-RRT [16] and its variants [17] are inspired by Monte Carlo optimization techniques and use notions such as the Metropolis criterion [18] to guide the exploration of the configuration space.

B. State-space sampling

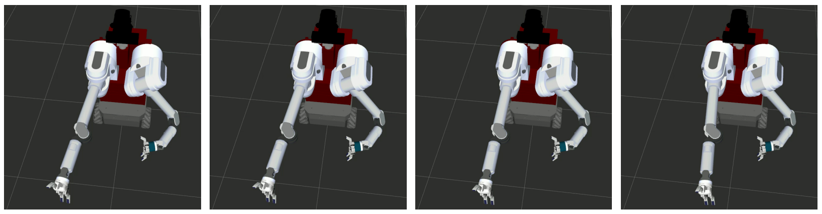
There is a rich body of literature on how to produce samples that increase the efficiency of a planner in terms of finding a solution or producing high-quality solutions. Heuristic approaches include sampling on the medial axis [19], [20], sampling near the boundary of the obstacles [21], resampling along a given trajectory [22] and more [23], [24]. For planning under the differential constraints, reachability-guided sampling [25] focuses on sampling regions of the state space that are most likely to promote expansion for the given constraints.

Of specific interest to our work are approaches that produce samples in the informed set $\mathcal{X}_{\text{inf}}$. As mentioned in Sec. I Gammel et al. [2] describe an approach to sample uniformly in $\mathcal{X}_{\text{inf}}$ for the specific case where $\mathcal{X} = \mathbb{R}^d$ and when optimizing for path length. To the best of our knowledge, the only method to produce samples in non-Euclidean spaces that can be applied to motion planning problems (other than rejection sampling) is HRS by Kunz et al. [3].

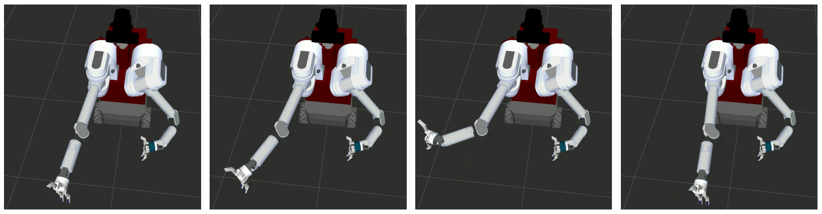
C. Markov Chain Monte Carlo (MCMC)

Monte Carlo simulation is a general sampling framework widely used in various domains. Roughly speaking, Monte Carlo simulation repeatedly samples a domain at random to approximate some value or function. One specific domain where Monte Carlo simulation is used which is relevant to this work is generating draws from a desired distribution which is hard to sample directly.

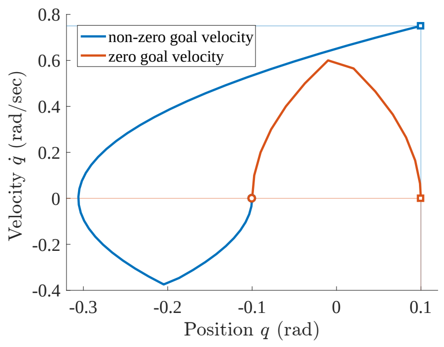
One of the popular classes of Monte Carlo simulation is *Markov Chain Monte Carlo* (MCMC) [26]. Here, the samples are drawn by generating a Markov chain such that the distribution of points on the chain converges to the desired distribution. One variant, which is of special interest to us is Hit-and-Run [4]. Here, given the current point x_i the next point x_{i+1} in the Markov chain is produced by sampling a random direction θ on the surface of the unit sphere centered at x_{i+1}. This defines a ray r_i rooted at x_i and passing through θ. The point x_{i+1} is chosen by randomly sampling a point on r_i. This algorithm is considered to



(a) Both start velocity and goal velocity are zero.



(b) Start velocity is zero but goal velocity is non-zero.



(c) Phase plot of two trajectories of one of the joints.

Fig. 2: HERB moves right arm from a start configuration to a goal configuration, which are in close proximity. When the goal velocity is non-zero, HERB needs to move right arm further away to accelerate.

be one of the most efficient algorithms for generating an asymptotically uniform distribution if the set under consideration is convex [27] and it can also be extended to sample points that converge to an arbitrary target distribution in total variation [28].

The attractiveness of Hit-and-Run for our problem domain stems from the fact that it performs a series of one-dimensional rejection samples which are extremely fast to compute, even in high-dimensional spaces. Finally, it is worth noting that we are not the first to apply Hit-and-Run for motion-planning problems. Recently [29] it was used as an alternative to RRT to produce *feasible motions* (and not high-quality paths). Interestingly the paper concludes with the statement "*One drawback is that the sample paths for Hit-and-Run have no pruning and are therefore longer than the RRT paths. Hybrid approaches that yield short paths but also explore quickly are a promising future direction.*" Our paper can be seen as a hybrid approach marrying sampling-based planning with MCMC-based approaches.

III. Problem definition

Let $\mathcal{X}$,$\mathcal{U}$ denote the state and controls spaces, respectively and set $\mathcal{X}_{\text{free}} \subset \mathcal{X}$ to be the set of states where the robot is collision free. A *trajectory* γ is a timed path through $\mathcal{X}$ obtained by applying at time t control $u(t) \in \mathcal{U}$ and satisfying the system dynamics $\dot{x}(t) = f(x(t), u(t))$. A trajectory is collision free if $\forall t,\ \gamma(t) \in \mathcal{X}_{\text{free}}$.

Given a cost function $c : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, the cost of a trajectory γ is the accumulated cost along the path $c(\gamma) = \int_0^T c(x(t), u(t))|\dot{\gamma}(t)|dt$, where T is the duration of γ.

Given start and target states $x_s, x_g \in \mathcal{X}$, we wish to find a collision-free trajectory γ^* connecting x_s to x_g such that $c(\gamma^*) = \min_{\gamma \in \Gamma} c(\gamma)$, where Γ is the set of all collision-free trajectories.

Given a trajectory γ_{best} with cost $c_{\text{best}} = c(\gamma_{\text{best}})$ the *informed set* $\mathcal{X}_{\text{inf}}$ is defined to be all states x which may be on trajectories with lower cost than c_{best}. Specifically, $\mathcal{X}_{\text{inf}} = \{x \in \mathcal{X} \mid c(\gamma^*(x)) < c_{\text{best}}\}$ [2]. Here $\gamma^*(x)$ denotes the optimal trajectory from x_s to x_g constrained to pass through x. Notice that we do not require that $\gamma^*(x)$ is collision free.

In this work we consider the problem of efficiently producing samples within $\mathcal{X}_{\text{inf}}$. These samples will be used within the informed RRT* framework to efficiently and incrementally compute trajectories of decreasing cost, converging to the optimal trajectory.

IV. Motivation—$\mathcal{X}_{\text{inf}}$ in kinodynamic state spaces

In this section we properly motivate our work. Specifically, we start by describing the differences in between planning in Euclidean configuration spaces (also called geometric planning) and non-Euclidean state spaces.

A. Geometric vs. Kinodynamic planning

Consider the problem depicted in Fig. 2 where HERB is required to reach a goal position with a high velocity of its end effector. One approach to address this problem is to first plan in the geometric configuration space and then re-scale the trajectory in time. However, when the start and goal are in close proximity, a geometric planner will simply connect the two states (Fig. 2a). On re-scaling this trajectory in time, reaching the goal velocity in such short distance will require large acceleration, which will not be feasible. Hence, it is required to move the arm back and then reach the goal, i.e. the trajectory returned by the kinodynamic planner shown in (Fig. 2b). The difference between the two motions are shown in a phase plot in Fig. 2c.

B. Minimal Time Double Integrator

To understand why we resort to optimization-based methods and do not attempt to provide a closed-form solution to sample $\mathcal{X}_{\text{inf}}$ we study the structure of the informed set for a simple yet important dynamical system—the double integrator minimizing time (MTDI). Here, we are given a one-dimensional point robot with bounded acceleration moving amid obstacles. We wish to compute the minimal-time trajectory between two states x_s, x_g. A state $x \in \mathcal{X}$ in

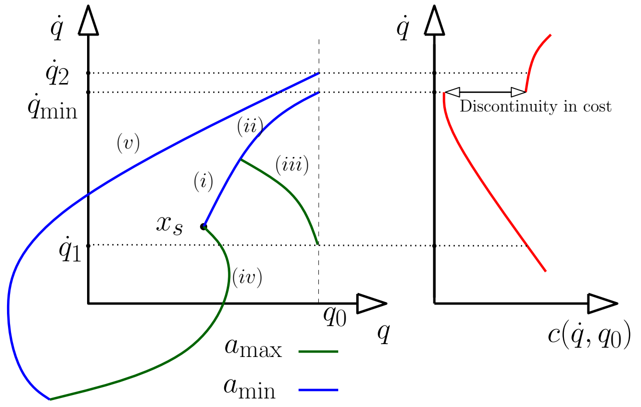


Fig. 3: Visualization of the discontinuity in the cost function of MTDI (right) related to the types of controls applied (left). Given state x_s and fixed position q_0, we depict the cost (time) as a function of the velocity $\dot{q}$. The minimal cost is attained at $\dot{q}_{\min}$ by applying maximal acceleration (blue curves $(i), (ii)$). To reach states such as $\dot{q}_1$, where $\dot{q}_1 < \dot{q}_{\min}$ we need to apply maximal acceleration (curve (i)) followed by minimal acceleration (green curve (iii)), which result in a continuous increase in cost. However, for states such as $\dot{q}_2$, where $\dot{q}_2 > \dot{q}_{\min}$, we need to apply minimal acceleration followed by maximal acceleration (curves $(iv), (v)$), which result in the discontinuity.

this model is defined by the position $q \in \mathbb{R}$ and the velocity $\dot{q} \in \mathbb{R}$ of the robot. The system dynamics are described by:

$$\begin{bmatrix} \dot{q} \\ \ddot{q} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} q \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} u. \tag{1}$$

Here, the control $u \in [\underline{u}, \overline{u}]$ is the (bounded) acceleration.

Notice that (i) this is model can be seen as a simplified one-dimensional instance of a robot manipulator with many degrees of freedom and that (ii) closed-form solutions exist to the 2pBVP for this specific case (as well as the multi-dimensional setting) [30], [31].

Recall that for Euclidean spaces minimizing path length, the informed set $\mathcal{X}_{\text{inf}}$ is a prolate hyperspheroid [2]. Moreover, the size and shape of the hyperspheroid is defined only be the cost c_{best} of the current best solution and not by the location of the start x_s and goal x_g.

For the case of a MTDI, this is not the case. Specifically, we have that (i) the structure of $\mathcal{X}_{\text{inf}}$ changes not only with c_{best} but also according to the specific values of x_s and x_g and that (ii) the cost map that implicitly defines $\mathcal{X}_{\text{inf}}$ can contain discontinuities (in contrast to Euclidean spaces minimizing path length where the cost map is continuous and differentiable at every point).

To understand the differences recall that optimal trajectories for MTDI follow a "bang-bang" controller [30], [31]. Namely, we first apply maximal (or minimal) acceleration for some duration and then switch to applying minimal (or maximal, respectively) acceleration. It is straightforward to see that both the type and the amount of acceleration applied (and hence the structure of $\mathcal{X}_{\text{inf}}$) depend on the specific values of x_s and x_g. Fig. 3 depicts a simple example where the cost map is discontinuous.

To summarize, the structure of $\mathcal{X}_{\text{inf}}$ can change given different start and goal states. Furthermore, its boundary may not be differentiable due to the aforementioned discontinuity.

V. MCMC-BASED INFORMED SAMPLING

In this section we describe our approach to efficiently produce new samples in an informed set $\mathcal{X}_{\text{inf}}$ given a specific cost c_{best} of trajectory $\gamma_{\text{best}}(t)$. The samples follow a Markov Chain Monte Carlo, in which a new sample candidate is produced from a previous sample that also lies in the same informed set. Furthermore, the value c_{best} can decrease between consecutive iterations in the planning process of an informed RRT* planner. This will occur if the search algorithm that uses the sampler finds a path to the goal whose cost is lower than c_{best}.

The idea behind applying MCMC for informed sampling is to define a target distribution π that has $\Pr(x_{\text{sample}} \in \mathcal{X}_{\text{inf}}) \neq 0$ and $\Pr(x_{\text{sample}}) \notin \mathcal{X}_{\text{inf}} = 0$ (here, $\Pr(\cdot)$ is the probability that an event will occur). This is specially useful if we want to bias the samples based on our knowledge of the environment. However, we make no such assumption about the environment and use a uniform distribution over all points in $\mathcal{X}_{\text{inf}}$. Our approach consists of two stages,

1) finding an initial sample $x_0 \in \mathcal{X}_{\text{inf}}$ which will serve as the start of a Markov chain. This is implemented using the function `sample_in_informed_space()`, and
2) sampling a new sample $x_i \in \mathcal{X}_{\text{inf}}$ given a previous sample x_{i-1}. This is implemented using the function `MCMC_sample`$(x_{i-1}, c_{\text{best}})$.

Our framework is described in Algorithm 1 and visualized in Fig. 1. We now continue to detail each of the algorithm's stages.

A. Finding an initial sample in $\mathcal{X}_{\text{inf}}$

In theory, MCMC methods converge to the desired distribution regardless of the initial sample used to seed the chain. In our setting, the probability distribution π is defined by having all points in $\mathcal{X}_{\text{inf}}$ distributed uniformly while the probability of sampling any configuration $x \in \mathcal{X} \setminus \mathcal{X}_{\text{inf}}$ is zero. A common practice to avoid starting biases in MCMC-type algorithm is to discard an initial set of samples (a process referred to as "burn-in") [26].

In our setting, we are only interested in points in $\mathcal{X}_{\text{inf}}$, thus we suggest to start the Markov Chain in $\mathcal{X}_{\text{inf}}$ and avoid this burn-in stage. We restart our process and generate a new Markov chain when (i) the cost of c_{best} is updated (i.e. a new solution is found by the planner) or (ii) the new sample on the existing Markov chain is outside $\mathcal{X}_{\text{inf}}$. We suggest several methods to produce an initial sample $x_0 \in \mathcal{X}_{\text{inf}}$

- randomly returning either the start state or the goal state,
- randomly sampling a state $x_{\text{rand}} \in \mathcal{X}$ and using a gradient descent algorithm (e.g. Newton-Raphson Method [32]) to find a sample in $\mathcal{X}_{\text{inf}}$
- sampling from a pool of previous samples that are in the informed set $\mathcal{X}_{\text{inf}}$ and
- applying rejection sampling until a sample in the informed set is found.

Each of the methods proposed has its own pros and cons. For example, a gradient-descent algorithm is usually efficient in finding a solution, but subject to only convex problems.

Algorithm 1 MCMC-based Informed Sampling $(x_{i-1}, c_{\text{best}})$

```
1: loop
2:     if $i = 0$ then
3:         $x_0 \leftarrow$ sample_in_informed_space()
4:     $x_i \leftarrow$ MCMC_sample$(x_{i-1}, c_{\text{best}})$
5:     if $x_i \notin \mathcal{X}_{\text{inf}}$ then
6:         $i \leftarrow 0$
7:         Goto line 2
8:     return $x_i$
```

Algorithm 2 Metropolis-Hastings MCMC $(x_{i-1}, c_{\text{best}})$

```
1: $x'_i \leftarrow$ sample_normal$(q(x \mid x_{i-1}, \Sigma))$
2: $\alpha \leftarrow \frac{q(x_{i-1}|x'_i,\Sigma)\pi(x'_i)}{q(x'_i|x_{i-1},\Sigma)\pi(x_{i-1})}$
3: if sample_random$(0.0, 1.0) < \alpha$ then
4:     return $x'_i$
5: return $x_{i-1}$
```

Algorithm 3 Hit-and-Run MCMC $(x_{i-1}, c_{\text{best}})$

```
1: $d_i \leftarrow$ sample_random_direction()
2: $L(\lambda) = \{\lambda \in \mathbb{R} \mid c(x_{i-1} + \lambda d_i) \leq c_{\text{best}}\}$
3: $\lambda^+ \leftarrow \sup L(\lambda)$;     $\lambda^- \leftarrow \inf L(\lambda)$
4: loop
5:     $\lambda' \leftarrow$ sample_random$(\lambda^-, \lambda^+)$
6:     $x_i \leftarrow x_{i-1} + \lambda'_i d_i$
7:     if $c(\gamma^*(x_i)) < c_{\text{best}}$ then
8:         return $x_i$
9:     if $\lambda' > 0$ then
10:        $\lambda^+ \leftarrow \lambda'$
11:    else
12:        $\lambda^- \leftarrow \lambda'$
```

Sampling from a pool of samples is algorithmic-free but biases new samples to be near previous samples.

B. Generating a new sample in a Markov chain

Our approach is general and can be applied to any MCMC algorithm (see Sec. II). The process is demonstrated in Algorithm 1. At the beginning of a Markov chain, `sample_in_informed_space()` is called to generate the first sample in an informed set. `MCMC_sample()` is called to generate a new sample based on a previous sample x_{i-1} and a cost c_{best} that defines an informed set. We demonstrate how to instantiate it with two different algorithms *Metropolis-Hastings* and *Hit-and-Run*, which will be described in later subsections. If a generated new sample candidate is in the informed set, this candidate will be returned as a new sample (line 4). But if a generated new sample candidate is not in the informed set (line 5), a new Markov chain will be initiated by calling `sample_in_informed_space()` to generate a new sample x_0 (lines 7 and 3).

1) Metropolis-Hastings sampler: The Metropolis-Hastings algorithm is one of the most popular MCMC samplers [18] because it provides a simple framework that guarantees the convergence of Markov chains to a target distribution. Our work adopts the general Metropolis-Hastings algorithm, as described in Algorithm 2, We generate a new sample x_i around the previous sample x_{i-1} using a Gaussian distribution (line 1). An acceptance ratio α is used to keep the reversibility even if the target probability π is asymmetric, which is needed to guarantee the convergence [18]. The calculation of the acceptance ratio α is given in line 2 in Algorithm 2. A new sample will be accepted if a generated random number from a uniform distribution $[0.0, 1.0]$ is less than the acceptance ratio (lines 3-5).

In our implementation, we use the Newton-Raphson method as a gradient descent with random restart to find $x_0 \in \mathcal{X}_{\text{inf}}$ as the start of a Markov chain.

2) Hit-and-Run sampler: The Hit-and-Run [33] sampler is known to efficiently generate uniform samples. Specifically, we use the Accelerated Hit-and-Run variant [4] of the algorithm which is described in Algorithm 3. It allows for uniform sampling in both convex and non-convex state spaces [4]. Given the previous sample x_{i-1} it first samples a random direction on a unit sphere (line 1). This induces a line $L(\lambda)$ passing through x_{i-1} in the direction sampled and parametrized by a scalar λ (line 2). We obtain upper and lower bounds on λ (line 3) that are problem dependent. For example, if we have box constraints on the joint limits of the robot and on the maximum velocity, then the bounds are given by $\lambda^+ = -\lambda^- = l_{diag}$; where l_{diag} is the length of the longest diagonal of the box. We then sample a point along $L(\lambda)$ by sampling a scalar λ' within our bounds (line 5). This defines a point x_i which is a candidate for the next sample of the Markov Chain (line 6). We then check if the point lies in the informed set (line 7) and if it does, we return it. If not, we update our bounds (lines 9-12) and repeat the process. The algorithm can be viewed as an efficient method that performs rejection sampling along a one-dimensional line passing through the previous sample parametrized by λ.

For this algorithm, we continue sampling along a Markov Chain until either (i) the difference between the lower and upper bounds (λ^- and λ^+) that define our sampling domain is below a predefined threshold or (ii) a predefined number of samples was exceeded. We want to point out that a Hit-and-Run sampler only requires that a Markov chain starts in an informed set, and will not produce a sample outside of the informed set. Also, in our implementation, we pick the start or the goal state to find $x_0 \in \mathcal{X}_{\text{inf}}$ as the start of a Markov chain.

C. Asymptotic optimality

We note that our approach produces samples that cover the informed space. Namely, there is a non-zero probability to sample in any region of $\mathcal{X}_{\text{inf}}$. A direct implication of the proof of optimality presented in [7] is that our algorithm is asymptotic optimal:

Proposition 1: Informed RRT* [2] running with MCMC-based informed sampling is asymptotic optimal.

VI. Evaluation

We evaluate the performance of proposed MCMC methods by comparing four types of samplers, which are Rejection Sampler (RS), Hierarchical Rejection Sampler (HRS), Metropolis-Hastings Sampler (MH), and the Hit-and-Run Sampler (HNR). We use different samplers to generate a fixed number of samples in different informed sets to check the sampling efficiency. We then compare the quality of the samplers by how the samplers work with informed RRT* [2].

A. Sampling Efficiency

Fig. 4a shows how the informed set volume ratio decreases as the informed set cost c_{best} becomes smaller in problems of different dimensions. In higher dimensions, the informed set volume ratio decreases much more quickly with decrease in the informed set cost c_{best}, as new cheaper trajectories are found in the planning process.

We define the *informed set volume ratio* as the ratio of the volume of informed space to the volume of entire state space. Fig. 4 shows the plot of the average time taken to generate one sample in the informed space vs. informed set volume ratio for 5000 samples. The informed set volume ratio is estimated by the acceptance rate of rejection sampler. The informed set volume ratio is approximated by the ratio of the number of accepted to the total number of samples obtained while running rejection sampling. Fig. 4 shows that MH and HNR have a better sampling efficiency compared to HRS and RS with decrease in informed set volume ratio or increase in dimensions.

Metropolis-Hastings shows consistent sampling time when problems get harder. It takes the advantage of sampling a near state that generate samples in an informed set. However this does not reflect the quality of the samples, though all the samples are in the informed set. Recall in Algorithm 2, a new sample candidate is obtained from a Gaussian distribution $q(x \mid x_{i-1}, \Sigma)$. The best covariance Σ that generates faster convergence differs with problem setting. A small covariance tends to generate more samples near previous samples, while a large covariance has better exploration but is more likely to drive a Markov chain outside the informed set. In our next planning experiment setting, we use the same covariance for different problems.

When informed set volume ratio is relatively high, it is easy to generate samples in the informed set. All the samplers have close performances. It actually implies rejection sampler is the best because of its simplicity in implementation and minimum correlation between successive samples. The sampling time of all samplers except MH, increases as problems gets harder. Notice that the sampling efficiency of HNR scales better than HRS, and HRS scales better than RS. Moving from a 4 dimension problem in Fig. 4b to a 12 dimension problem in Fig. 4c, sampling in an informed set becomes even harder, because the informed set volume ratio becomes smaller. Here, HNR and MH samplers show much better efficiency over the others.

We want to point out that efficiently sampling in an informed set is not sufficient for determining the performance of a sampler. For example, a sampler that constantly returns the same sample in an informed set might show the best sampling efficiency, however it is the worst sampler in a path planning problem. Ideally, we want generated samples to be uniformly distributed in an informed set to get the best exploration.

B. Planning Efficiency

The quality of samples determines the efficiency of resulting planning algorithms. If a sampler could provide samples with same quality as others but generate samples in a much efficiency way, we would expect that an informed RRT* with this sampler would show two properties.

- It shall converge faster in finding the optimal solution. Sampling in an informed set is gradually becoming harder as new better solution reduces c_{best} which reduces the informed set volume ratio.
- Its performance should not degrade significantly in high dimensional problems. As shown in Fig. 4a, the informed set volume ratio decrease more significantly in a high-dimension state space. The advantage of a good informed sampler becomes evident.

To evaluate the planning efficiency of the samplers, we run them with the informed RRT* planner [2] in position-velocity space with MTDI as steering function, on three different problems described below and shown in Fig. 5. For each problem the start and goal states (positions and velocities) are known in the joint space. Joint velocities at start and goal are calculated from desired end-effector velocities using inverse kinematics before starting the planning. Table I shows the parameters used in the problems and Fig. 5 depicts the planned path for the three problems.

P1 *6 Dimensional 3 DoF planar manipulator*—The objective is to move the arm from a start to a goal state, both with zero velocities.

P2 *12 Dimensional 6 DoF snake arm* —The objective is to hammer the end-effector into the wall while starting with zero velocity.

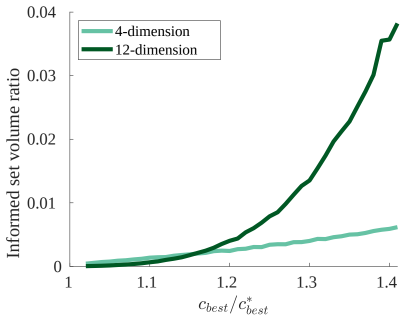
P3 *14 Dimensional 7 DoF WAM arm*—The objective is to to quickly swing away a glass on a table using the right arm.

As shown in in Fig. 6, MH has the worst performance in all three problems, especially when the dimension increases. Though theoretically samples converge to a target distribution only in the limit of infinite time. However, in practice the samples are to close to each other and don't explore the entire informed space. If the variance of transition distribution is too high, it will tend to move out of the informed set too frequently, and takes longer to converge as the rejection rate is too high.

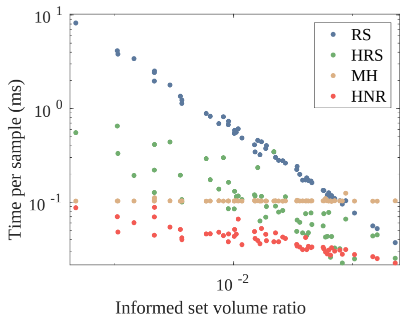
HNR shows close performance with RS and HRS in a 6-dimension problem, as in Fig. 6a. As shown in Fig. 6b and 6c, the advantages of HNR are clearly evident in higher dimensional problems. The cost of best solutions generated by planner with HNR sampler converges significantly faster to a cheaper to trajectory compared to others.

VII. Conclusion

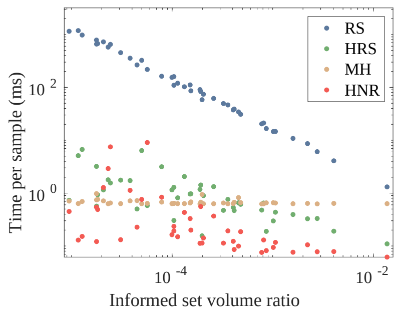
In this work we demonstrated the effectiveness of using MCMC algorithms to efficiently produce samples for



(a) The informed set volume ratio decrease as c_{best} decreases in the planning process in different dimensions.



(b) 4-dimensional state space.

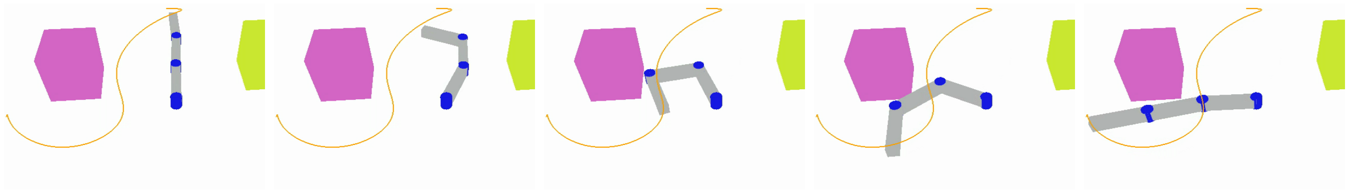


(c) 12-dimensional state space.

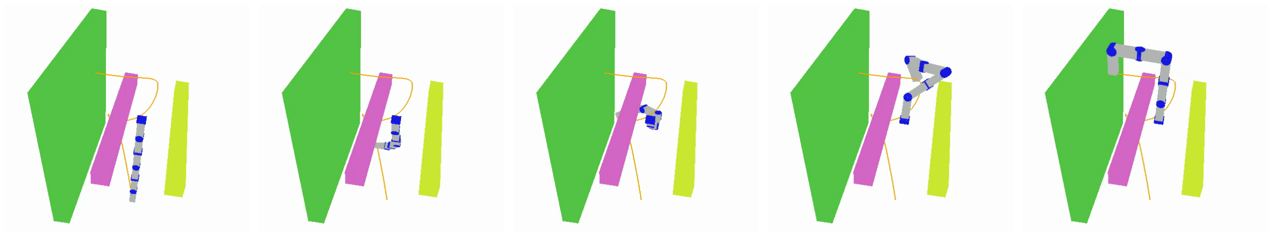
Fig. 4: Average sampling time vs informed set volume ratio of four samplers (RS, HRS, MH and HNR) in state spaces of different dimensions. The x-axis is the ratio of the volume of informed set and the volume of the entire state space. The y-axis is the average time per sample.

HERB Joint	1	2	3	4	5	6	7	3D Arm	6D Snake
Joint limits (rad)	[0.54, 5.74]	[-2.00, 2.00]	[-2.80, 2.80]	[-0.90, 3.10]	[-4.76, 1.24]	[-1.60, 1.60]	[-3.00, 3.00]	$[-\pi, \pi]$	$[-\pi, \pi]$
$\lvert v_{max}\rvert$ (rad/s)	0.75	0.75	2.00	2.50	2.50	2.50	2.00	10	10

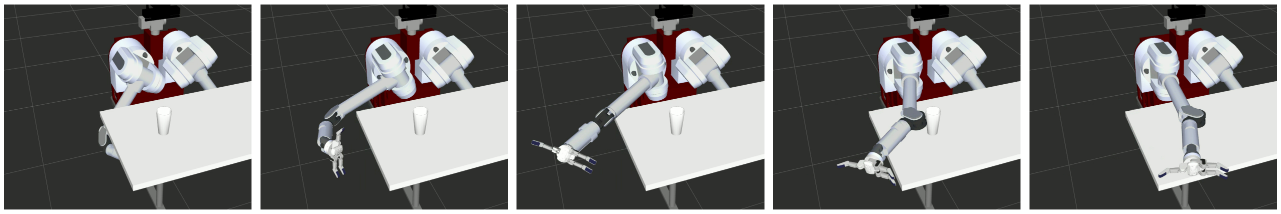
TABLE I: Parameters for the problems. All robots have $|a_{max}| = 1.0\ rad/s^2$
.



(a) Problem **P1** : 3DOF planar arm moving from a start state to a goal state, both with zero velocities.



(b) Problem **P2** : 6DOF snake hammers the end-effector into the wall while starting with zero velocity.



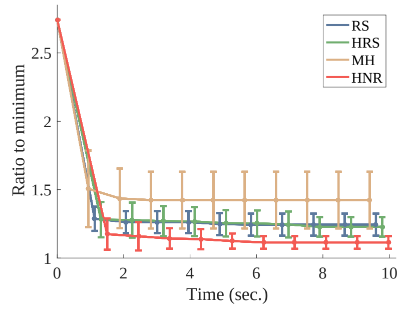
(c) Problem **P3** : HERB sweeps a cup on a table, in which the right arm starts with zero velocity and ends with non-zero velocity.

Fig. 5: Three problems in different state spaces and subject to different dynamic constraints used to evaluate planning efficiency.

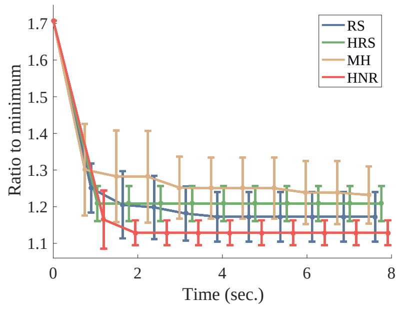
asymptotically-optimal motion planning algorithms. Clearly, there are multiple other MCMC algorithms that can be used and it is interesting to see if alternative algorithms may produce better results. One drawback of these approaches is that they usually incur parameters that have to be tuned. Indeed, in this work we did not spend effort in tuning the parameters and did not change them across the range of scenarios we tested. There is a wealth of literature in the optimization community regarding this topic and integrating such tools is left for future work. Finally, we are interested in using this framework with alternative sampling-based algorithms such as BIT* [9] or LBT-RRT [12] and with alternative state spaces.

VIII. Acknowledgements

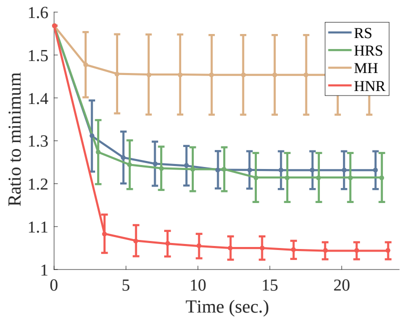
This work was (partially) funded by the National Science Foundation IIS (#1409003), and the Office of Naval Research.



(a) Problem **P1** - 6 dimensions.



(b) Problem **P2** - 12 dimensions.



(c) Problem **P3** - 14 dimensions.

Fig. 6: Planning Efficiency of four different samplers (RS, HRS, MH and HNR) in three problems. The x-axis is the planning time. The y-axis is the ratio of the current best and the optimal $c_{\text{best}}/c^*_{\text{best}}$.

References

[1] S. M. LaValle, *Planning Algorithms*. Cambridge University Press, 2006.

[2] J. D. Gammell, S. S. Srinivasa, and T. D. Barfoot, "Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2014, pp. 2997–3004.

[3] T. Kunz, A. Thomaz, and H. Christensen, "Hierarchical rejection sampling for informed kinodynamic planning in high-dimensional spaces," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2016, pp. 89–96.

[4] S. Kiatsupaibul, R. L. Smith, and Z. B. Zabinsky, "An analysis of a variation of hit-and-run for uniform sampling from general regions," *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, vol. 21, no. 3, p. 16, 2011.

[5] S. M. LaValle and J. J. K. Jr., "Randomized kinodynamic planning," *I. J. Robotics Res.*, vol. 20, no. 5, pp. 378–400, 2001.

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

[7] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *I. J. Robotics Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[8] K. Solovey, O. Salzman, and D. Halperin, "New perspective on sampling-based motion planning via random geometric graphs," in *Robotics: Science and Systems (RSS)*, 2016.

[9] 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 Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 3067–3074.

[10] O. Salzman and D. Halperin, "Asymptotically-optimal motion planning using lower bounds on cost," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 4167–4172.

[11] A. Dobson and K. E. Bekris, "Sparse roadmap spanners for asymptotically near-optimal motion planning," *I. J. Robotics Res.*, vol. 33, no. 1, pp. 18–47, 2014.

[12] O. Salzman and D. Halperin, "Asymptotically near-optimal RRT for fast, high-quality motion planning," *IEEE Trans. Robotics*, vol. 32, no. 3, pp. 473–483, 2016.

[13] C. Xie, J. P. van den Berg, S. Patil, and P. Abbeel, "Toward asymptotically optimal motion planning for kinodynamic systems using a two-point boundary value problem solver," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2015, pp. 4187–4194.

[14] D. J. Webb and J. van den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 5054–5061.

[15] A. Perez, R. Platt, G. Konidaris, L. P. Kaelbling, and T. Lozano-Pérez, "LQR-RRT*: Optimal sampling-based motion planning with automatically derived extension heuristics," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2012, pp. 2537–2542.

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

[17] D. Devaurs, T. Siméon, and J. Cortés, "Enhancing the transition-based RRT to deal with complex cost spaces," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013, pp. 4120–4125.

[18] S. Chib and E. Greenberg, "Understanding the Metropolis-Hastings algorithm," *The American Statistician*, vol. 49, no. 4, pp. 327–335, 1995.

[19] S. A. Wilmarth, N. M. Amato, and P. F. Stiller, "MAPRM: A probabilistic roadmap planner with sampling on the medial axis of the free space," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 1999, pp. 1024–1031.

[20] H. C. Yeh, J. Denny, A. Lindsey, S. L. Thomas, and N. M. Amato, "UMAPRM: uniformly sampling the medial axis," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2014, pp. 5798–5803.

[21] H. Yeh, S. L. Thomas, D. Eppstein, and N. M. Amato, "UOBPRM: A uniformly distributed obstacle-based PRM," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2012, pp. 2655–2662.

[22] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*. IEEE, 2011, pp. 2640–2645.

[23] C. Urmson and R. Simmons, "Approaches for heuristically biasing RRT growth," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2003, pp. 1178–1183.

[24] A. C. Shkolnik, M. R. Walter, and R. Tedrake, "Reachability-guided sampling for planning under differential constraints," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2009, pp. 2859–2865.

[25] S. D. Pendleton, W. Liu, H. Andersen, Y. H. Eng, E. Frazzoli, D. Rus, and M. H. Ang, "Numerical approach to reachability guided sampling-based motion planning under differential constraints," *IEEE Robotics and Automation Letters*, 2017.

[26] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine learning*, vol. 50, no. 1-2, pp. 5–43, 2003.

[27] L. Lovász and S. Vempala, "Hit-and-run from a corner," *SIAM Journal on Computing*, vol. 35, no. 4, pp. 985–1005, 2006.

[28] H. E. Romeijn and R. L. Smith, "Simulated annealing and adaptive search in global optimization," *Probability in the Engineering and Informational Sciences*, vol. 8, no. 4, pp. 571–590, 1994.

[29] Y. Abbasi-Yadkori, P. Bartlett, V. Gabillon, and A. Malek, "Hit-and-Run for Sampling and Planning in Non-Convex Spaces," in *International Conference on Artificial Intelligence and Statistics*, vol. 54, 2017, pp. 888–895.

[30] K. K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *IEEE Int. Conf. Robotics and Automation (ICRA)*, 2010, pp. 2493–2498.

[31] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, 2014, pp. 3713–3719.

[32] V. S. Ryaben'kii and S. V. Tsynkov, *A theoretical introduction to numerical analysis*. CRC Press, 2006.

[33] R. L. Smith, "Efficient Monte Carlo procedures for generating points uniformly distributed over bounded regions," *Operations Research*, vol. 32, no. 6, pp. 1296–1308, 1984.