Informed and Probabilistically Complete Search for Motion Planning under Differential Constraints

Kostas E. Bekris and **Lydia E. Kavraki**
Computer Science Department, Rice University
6100 Main Street, MS 132, Houston, 77005, TX, USA
{bekris,kavraki}@cs.rice.edu

Abstract

Sampling-based search has been shown effective in motion planning, a hard continuous state-space problem. Motion planning is especially challenging when the robotic system obeys differential constraints, such as an acceleration controlled car that cannot move sideways. Methods that expand trajectory trees in the state space produce feasible solutions for such systems. These planners can be viewed as continuous-space analogs of traditional uninformed search as their goal is to explore the entire state space. In many cases, the search can be focused on the part of the state-space necessary to solve a problem by employing heuristics. This paper proposes an informed framework for tree-based planning that successfully balances greedy with methodical search. The framework allows the use of a broad set of heuristics for goal-directed problem solving, while avoiding scaling issues that appear in continuous space heuristic search. It also employs an appropriate discretization technique for continuous state-space problems, based on an adaptive subdivision scheme. Although greedy in nature, the method provides with probabilistic completeness guarantees for a very general class of planning problems. Experiments on dynamic systems simulated with a physics engine show that the technique outperforms uninformed planners and existing informed variants. In many cases, it also produces better quality paths.

I. Introduction

Traditional motion planning computes collision-free, continuous paths for free-floating bodies (Latombe 1991; Choset et al. 2005; LaValle 2006). Although a challenging geometric problem, modern graph-based search algorithms that employ sampling solve difficult and high-dimensional instances (Kavraki et al. 1996; LaValle and Kuffner 2001a; Hsu, Latombe, and Motwani 1999; Sanchez and Latombe 2003). The problem is more complex, however, for systems with differential constraints. Such constraints arise in non-holonomic vehicles that are under-actuated or in general when the control influence of a system is small compared to momentum. The challenge is that not every collision-free path is acceptable; it must also be feasible given the constraints. Additionally, the dynamic model may not be available, but simulated by a software package, like a physics engine (Smith 2006).

Search-based techniques for kinodynamic planning, inspired by a dynamic programming approach (Donald et al. 1993), explore the entire state space for trajectories that respect the differential constraints. A major advantage is that they are applicable to a variety of different systems. They construct a "reachability tree" in the state space through a sampling operation (LaValle and Kuffner 2001b; Hsu, Latombe, and Motwani 1999; Ladd and Kavraki 2005a) that we will describe in more detail later. Although effective in eventually solving hard planning instances, they have high computational requirements as search methods, especially in kinodynamic problems that are typically higher-dimensional compared to geometric ones.

This paper focuses on reducing the solution time of search-based techniques for kinodynamic planning by incorporating heuristics based on workspace and query knowledge. We detail a new method, the Informed Subdivision Tree (`IST`), that balances greedy and methodical search while providing guarantees that eventually every problem can be solved. In simple parts of the state space the exploration is greedily guided by the heuristic. In constrained parts, such as narrow passages, the heuristic may not be beneficial and the algorithm automatically explores alternative routes for a solution. This methodical behavior is a result of an adaptive state-space subdivision scheme that estimates online the algorithm's performance in exploring the entire state space. Experimental comparisons on physically simulated systems between `IST` and uninformed planners (LaValle and Kuffner 2001b; Ladd and Kavraki 2005a) as well as with informed versions (LaValle and Kuffner 2001b; Urmson and Simmons 2003) show that `IST` outperforms the alternatives. `IST` also reports better quality paths in certain complicated workspaces, as in maze-like environments.

A. Problem Definition

Consider a moving system, such as a robot, whose motion is governed by differential equations of the form:

$$\dot{x}(t) = f(x(t), u), \quad g(x(t), \dot{x}(t)) \leq 0 \qquad (1)$$

where f, g are smooth; $x(t)$ is a state and fully describes the system at time t. The set of all states is the state space $\mathcal{X}$. The set of states for which the moving object is not in collision is the free state space $\mathcal{X}_{free}$. The set of all controls u define the control space $\mathcal{U}$. For a given $x(t)$, u is *valid* if it respects Eq. 1.

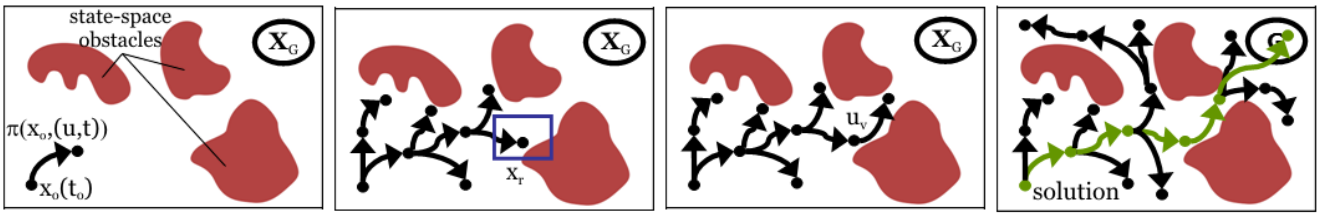

Figure 1: The basic scheme for tree-based planning with sampling. Each iteration of the algorithm uses a selection/propagation step, where a reachable state along the tree is selected first and then a valid control is applied to produce a feasible trajectory.

We are particularly interested in non-holonomic systems with second-order constraints, such as a car controlled by bounded acceleration and bounded steering velocity. The following notation will be useful in our description:

- A *plan* $p(dt)$ is a time sequence of controls: $p(dt) = \{(u_1, dt_1), \ldots, (u_n, dt_n)\}$, where $dt = \sum_i dt_i$.
- When $p(dt)$ is executed at $x(t)$, a vehicle follows the *trajectory*: $\pi(x(t), p(dt))$.
- If a plan has a single control u applied for duration dt, then $\pi(x(t), (u, dt))$ is called a *primitive* trajectory.
- A *feasible* trajectory is collision-free and respects Eq. 1.
- A state along trajectory $\pi(x(t), p(dt))$ at time $t' \in [t : t + dt]$ is denoted as $x^\pi(\ t'\ )$.
- For *stable* states there exists a control, which retains the system in the same state:
  $x(t)$ is stable iff $\exists\ u\ s.t.\ \int_{dt} f(x(t), u)\ dt = 0$.

Given a state $x_0(t_0)$ and a goal region $\mathcal{X}_G \subset \mathcal{X}$, compute a plan $p(dt)$ so that the resulting trajectory $\pi(x_0(t_0), p(dt))$ is feasible and the end state $x^\pi(\ t_0 + dt\ ) \in \mathcal{X}_G$ is a stable state within the goal region.

B. Search Methods for Planning with Dynamics

We first describe an abstract sampling-based planner for kinodynamic planning. The search operation is initiated at $x_0(t_0)$ and explores $\mathcal{X}_{free}$ by propagating feasible primitive trajectories. These trajectories are stored on a tree data structure $\mathcal{T}$ as edges. An edge that corresponds to trajectory $\pi(x(t), p(dt))$ is rooted at a node, which corresponds to the state $x(t)$. Fig. 1 and Algorithm 1 illustrate the operation of an abstract sampling-based kinodynamic tree planner.

Algorithm 1 BASIC SAMPLING-BASED TREE PLANNER

Set the root of $\mathcal{T}$ to the initial state $x_0(t_0)$
while $\nexists\ x \in \mathcal{T}\ s.t.\ (x \in \mathcal{X}_G$ and x stable) **do**
  Select a reachable state $x_r \in \mathcal{T}$
  Select a valid control u_v for the state x_r
  Propagate the primitive trajectory $\pi(x_r, (u_v, dt))$ for a duration dt and for as long as it is feasible
  Add all the states along $\pi(x_r, (u_v, dt))$ in $\mathcal{T}$
end while

Given the abstraction, we have the following three choices to construct a concrete algorithm. How to select: (i) the state x_r, (ii) the control u_v and (iii) the duration dt. Different algorithms follow different mechanisms for these choices but share the goal of covering the state space quickly and avoiding regression (Kalisiak and Van de Panne 2006), which means propagating paths in already explored space.

RRT (LaValle and Kuffner 2001b) samples a state in $\mathcal{X}$ and then selects the state $x_r \in \mathcal{T}$ closer to the sampled state given a metric. In Euclidean spaces, RRT has higher probability of extending a path inside the largest unexplored Voronoi cell. In kinodynamic planning, however, a good metric may not be available and the Voronoi-bias is not well defined. Similarly, the Expansive Spaces technique (Hsu, Latombe, and Motwani 1999) depends on an ideal sampler to guarantee coverage. PDST (Ladd and Kavraki 2005a; 2005b) avoids the use of a metric by using an adaptive subdivision scheme and provides probabilistic completeness for a general class of problems: if a path exists it will be eventually found (Kavraki, Kolountzakis, and Latombe 1996; Ladd and Kavraki 2004). PDST biases the exploration towards larger cells of the subdivision, which correspond to relatively unexplored parts of the space. Less attention has been given in the literature to the choices regarding the control to be applied and the duration of propagation for kinodynamic problems.

The above planners can be viewed as continuous-space analogs of traditional uninformed search. Some informed variations in the literature exhibit a switching behavior between coverage planning and best-first search. The RRT-"goal bias" variant selects with certain probability to expand from the state which minimizes a distance metric to the goal and the rest of the time uses the RRT Voronoi bias for coverage (LaValle and Kuffner 2001b). An older algorithm, the Randomized Potential Field (RPP) (Barraquand and Latombe 1991), also has multiple modes. It constructs a potential function to execute gradient-descent and then employs random walks and backtracking to exit local minima.

More recent informed variants introduce ideas from traditional AI search and study the effects of heuristics. The RRT* (Urmson and Simmons 2003) merges RRT with the A* algorithm, by using a heuristic in the state selection step. An issue, however, when using heuristics in continuous problems is the scale of the heuristic versus the true path cost. The randomized A* approach (Diankov and Kuffner 2007) uses learning in order to solve this scaling problem. The Anytime RRT algorithm (Ferguson and Stentz 2006) successively constructs RRTs that result in higher quality paths by employing heuristic search. The Exploring/Exploiting Tree (EET) (Rickert, Brock, and Knoll 2007) emphasizes the need to balance the greedy and methodical aspects of search. EET uses potential functions to bias tree-based planners.

C. Contribution

This paper focuses efficient informed search in continuous state spaces and emphasizes the benefits of bringing together ideas from robotics and artificial intelligence. We describe the Informed Subdivision Tree (IST) algorithm that achieves informed but probabilistically complete sampling-based kinodynamic planning. The algorithm does not have multiple modes such as RRT-"goal bias" or RPP, instead it utilizes heuristic information on every iteration and for every algorithmic choice (state, control and duration selection).

Although a wide set of heuristics can be used in IST, we describe here a methodology for constructing heuristics based on configuration space (C-space) information. The approach first computes a roadmap for a kinematic version of the moving system using an approach that is able to capture the connectivity of the C-space, such as PRM (Kavraki et al. 1996). IST uses the kinematic information from the roadmap to bias the tree expansion towards the goal in the state space $\mathcal{X}$.

The use of the C-space bias, however, can easily result in a technique that is greedy and gets stuck in local minima due to state space constraints. IST employs the adaptive subdivision scheme from PDST and an edge penalization scheme. Through these tools it is able to provide with probabilistic completeness, while it lends itself to an efficient implementation.

Furthermore, the integration of the heuristic information with the subdivision allows to separate the heuristic estimate of the cost-to-go from the cost-from-start value. In this way, IST avoids the scaling problems that appear in other continuous space heuristic search algorithms.

II. Informed Subdivision Tree

We will first outline the operation of the various tools employed by IST and then describe how they are combined in the overall algorithm.

A. Adaptive Subdivision for State Space Coverage

IST maintains a subdivision data structure $\mathcal{S}$, that corresponds to a set of cells: $\mathcal{S} = \{c_1, \ldots, c_K\}$. Each $c_i \subset \mathcal{X}$ corresponds to a subset of the state space. Initially this set contains one cell that encompasses the entire $\mathcal{X}$. The subdivision is refined in each iteration of the algorithm as shown in Fig. 2.

Whenever a reachable state x_r is selected, the cell $c' \in \mathcal{S}$ is found so that $x_r \in c'$. The algorithm guarantees that only one cell contains each state. Then c' is removed from $\mathcal{S}$ and two new cells, c'_{left} and c'_{right}, are introduced so that:

$$c'_{left} \cup c'_{right} = c' \text{ and } c'_{left} \cap c'_{right} = 0 \qquad (2)$$

Given that the initial cell is the entire $\mathcal{X}$, Eq. 2 implies that:

$$\cup_{\mathcal{S}} c_i = \mathcal{X} \text{ and } \forall c_i, c_j \in \mathcal{S} : \; c_i \cap c_j = 0 \qquad (3)$$

An implementation of the cell partition can be obtained with a Binary Space Partition Tree. Although not necessary, cell c' is typically split into two equal size cells along a dimension that maintains the subdivision balanced.

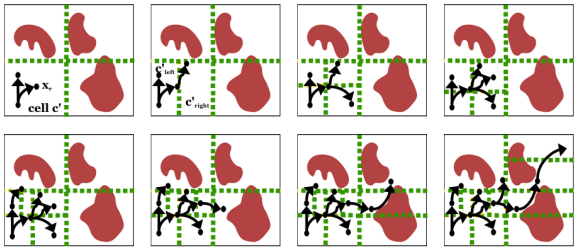

Figure 2: The interaction between state selection and the subdivision data structure.

The purpose of $\mathcal{S}$ is to provide an online estimate of the coverage performance of the state space exploration. Large cells in $\mathcal{S}$ represent parts of the space that have not been explored as much as smaller cells, in the sense that the algorithm has not expanded often new trajectories from states within them. Consequently, the subdivision level of a cell $c \in \mathcal{S}$ (the number of subdivisions that occurred to create c) is used as an estimator $\mu(c)$ of the exploration value of the cell c. Cells that do not contain any reachable states in $\mathcal{T}$ have by definition infinite estimator value: $\mu(c) = \infty$ if $\forall x \in \mathcal{T} : \nexists \, x \in c$. IST promotes the expansion of new trajectories from states that belong to cells with small estimator value $\mu(c)$ (large explored cells). This behavior promotes the exploration of unexplored parts of the space.

Although we have defined the subdivision to operate in the entire state space, this is not necessary for probabilistic completeness. It is sufficient to define a projection of the state space into a lower dimensional space and attempt to cover the projected space. Defining this projection depends on the application, allowing us to define the important projection of the state space that the algorithm must cover. For example, in the case of a second-order control car that has a five dimensional space: $\mathcal{X}_{car} = (x, y, \theta, V, s)$, ($(x, y)$: the planar coordinates, θ: orientation, V: velocity and s: steering direction) we typically subdivide only along the first three parameters of the state space.

B. C-space Heuristic

A heuristic function $h(q)$ maps any state to a value that expresses how good is this state in promoting the solution of the planning problem: $h : \mathcal{X} \rightarrow \mathbb{R}$. IST is able to incorporate any heuristic information. The only requirement is that the heuristic is upper and lower bounded by finite positive values. For computational efficiency, it is also desirable that the computation of the heuristic to be fast.

During a preprocessing step, we construct a C-space roadmap by following the Probabilistic Roadmap Method (PRM) (Kavraki et al. 1996), so as to gather the necessary information for the online computation of the heuristic. The roadmap contains nodes that correspond to configurations and edges that correspond to kinematic paths. We follow an incremental approach for the construction of the roadmap. Initially there are two nodes, the start configuration q_0 and a configuration q_G in the goal region $\mathcal{X}_G$. At each iteration we sample a new collision-free configuration q_{new}. q_{new} is considered for addition in the roadmap depending on the distance to the closest existing node. The smaller the distance,

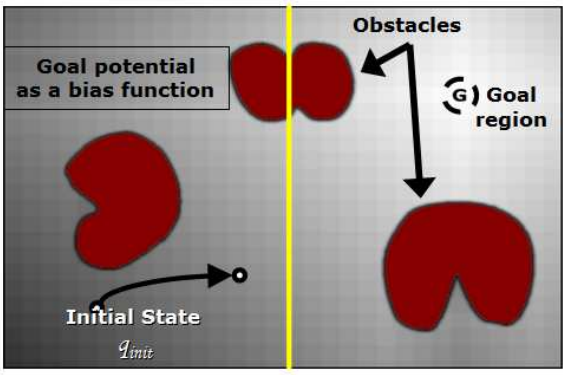

Figure 3: An illustration of an abstract potential function computed for a given goal.

the smaller the probability of inserting the node. Then we try to connect the node q_{new} through local paths to its K closest neighbors q_{cl} in the roadmap according to a C-space metric $d(q_{new}, q_{cl})$. The connection will be attempted if the neighbor q_{cl} satisfies any of the following two criteria:

1. q_{cl} is on a different connected component than q_{new}.
2. Or the current roadmap distance $path(q_{new}, q_{cl})$ is considerably longer than $d(q_{new}, q_{cl})$.

This PRM version has low collision-checking overhead because it does not consider redundant edges for collision due to the connected components heuristic (criterion 1). If only criterion 1 was used, however, the resulting roadmap would have no loops. The second criterion allows edges that create loops in the graph only when they appear beneficial in terms of C-space connectivity.

During the online computation of the heuristic for a state $x \in \mathcal{T}$, we get the projection $q(x)$ in the C-space of the state x and compute the closest node on the roadmap q_{cl}. Then the heuristic is computed as follows:

$$h(x) = d(q(x), q_{cl}) + path(q_{cl}, q_{goal}) + \frac{1.0}{obs(q(x))} \quad (4)$$

where $obs(q(x))$ corresponds to the distance between the moving body and the closest obstacle. Various collision checking packages provide such information. If it is not available, this factor can be dropped. For the nearest neighbor queries we use an efficient technique for approximate nearest neighbor search (Yershova and LaValle 2007; Plaku and Kavraki 2006).

The important advantage of this heuristic over simpler distance metrics is that it considers C-space obstacles. Nevertheless, it is more expensive to compute. In our experiments we have seen improvements even if we use very simple heuristics that consider workspace obstacles (i.e. a wavefront function on a grid for point approximations of the true system). The advantage of the PRM-based approach is that it computes paths that are truly collision-free in the C-space. Moreover, on all of our experiments, the computation of the roadmap, which occurs before the construction of $\mathcal{T}$, is orders of magnitude faster than the computation of $\mathcal{T}$ itself, since the kinematic reduction of a kinodynamic problem is considerably less constrained.

C. Edge Penalization Scheme for Completeness

We have been very relaxed on the requirements that the subdivision and the potential function have to satisfy and argued that the algorithm guarantees the solution of any planning problem because it is probabilistically complete.

To achieve this property we penalize edges of $\mathcal{T}$ when a state along the edge is selected as x_r. IST maintains a penalty factor $p(e)$ along each edge $e \in \mathcal{T}$. The first edge in the tree starts with a penalty of 1. Fig. 4 shows how the penalty factors are updated by the algorithm. When a state along e is selected then the penalty of the edge increases exponentially: $p(e) = 2 \cdot p(e)$. The penalty for the new edge being created will be: $p(e') = p(e) + 1$. IST promotes the selection of edges that have low penalty value. The objective of the penalization is to avoid selecting the same edges for expansion, which would result in regression.

Given the subdivision, the heuristic and the penalty scheme we can describe IST's state selection step. Figure 5 illustrates two applications of the state selection step.

1. Select the cell $c_{min} \in \mathcal{S}$ that minimizes a score function:
$$score(c) = \mu(c) \cdot h(c), \quad (5)$$
where $\mu(c)$ is the cell's subdivision level and the cell heuristic is: $h(c) = min_{\forall x \in \mathcal{T} \ s.t. \ x \in c} h(x)$.
2. Select an edge $e_{min} \in c_{min}$ that minimizes:
$$\texttt{score}(e) = p(e) \cdot cost(e), \quad (6)$$
where $p(e)$ is the edge penalty and $cost(e)$ is the duration of the trajectory from x_0 until the last state on e.
3. Along the edge e_{min} select a random state $x_r \in e_{min}$.
4. Execute bookkeeping operations:
   (a) Split c_{min} according to the subdivision rules.
   (b) Split all the edges that belonged to c_{min} so that the invariant that every edge belongs to only one cell holds.

The important characteristic of the state selection is that it separates the heuristic $h(c)$ from the true path cost $cost(e)$. The heuristic is used to select a cell in the subdivision in a depth-first manner. The true path cost is used to select the edge within a cell in a breadth-first manner. In this way, we do not need to scale $h(c)$ vs. $cost(e)$. At the same time, the approach promotes the exploration of alternative parts of the state space by promoting the selection of large unexplored cells. The penalization scheme allows the eventual selection of every edge on the tree. Moreover, this algorithm still allows for an efficient implementation. The cell selection can be achieved with a binary heap and most of the information necessary to make decisions can be updated in constant time at each iteration of the algorithm.

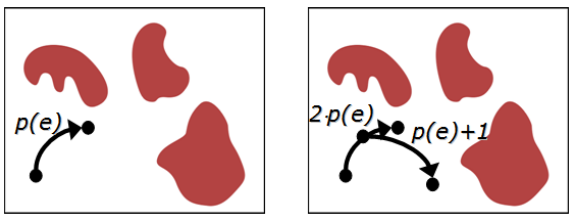

Figure 4: The edge penalization scheme.

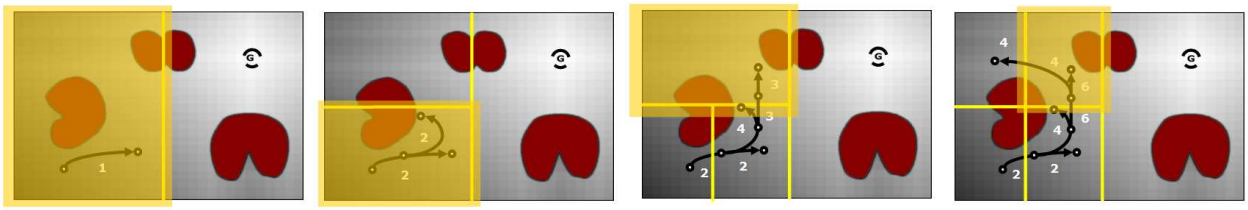

Figure 5: Four consecutive steps from the operation of the proposed state selection technique. The highlighted cell corresponds to the selected cell c_{min} at each iteration. The numbers corresponds to the edge penalty values $p(e)$.

D. Propagation Scheme

Given a selected x_r, the control selection step's objective is to estimate the best possible control to apply from x_r while allowing eventually the selection of any possible controls.

IST employs an offline procedure to create a database of trajectories for various dynamic parameters of the system's state. The offline procedure samples a discrete set of states $\hat{X}$ for the system. All these states are set to zero Cartesian coordinates and zero orientation. For each state $\hat{x} \in \hat{X}$ we sample various controls $u_{\hat{x}}$ and propagate a trajectory in an obstacle-free environment. We then store a discretized version of the resulting primitive trajectory $\pi(\hat{x}, (u_{\hat{x}}, dt_{max})))$ for a maximum duration dt_{max}.

During the online operation, if an edge is selected for the first time as e_{min} we make use of the database of trajectories with the objective of selecting a control that will bring the system to a state with a smaller heuristic cost. We find the prestored state $\hat{x}$ that is closer to the state x_r in terms of the dynamics parameters. Then for each control $u_{\hat{x}}$ we transform the stored trajectory $\pi(\hat{x}, (u_{\hat{x}}, dt_{max}))$ so that the initial state is x_r without doing any collision checking. Out of all trajectories we find the one with the best heuristic value $h(x)$ and choose the corresponding control $u_{\hat{x}}$ as the u_v control selection for this iteration.

Every other time that an edge e is selected for expansion, we attempt to increase the variety of controls that have been expanded from e. IST stores on each edge the controls that have been expanded in the past from states along this edge and builds a discretized probability distribution. The probability distribution is used so as to bias towards the selection of controls that have not been selected in the past. It retains a non-zero probability, however, for every possible control.

The last point related to the propagation scheme is the duration dt of the resulting trajectory $\pi(x_r, (u_v, dt))$. By default, the duration of the trajectory has to be such so that the trajectory remains feasible throughout its execution. Nevertheless, IST imposes an additional constraint that aims towards improving the quality of the resulting paths. If the propagated trajectory $\pi(x_r, (u_v, dt))$ enters a new cell of the subdivision, other than the cell that x_r belongs to, where there is already another path with a better cost, then the propagation of $\pi(x_r, (u_v, dt))$ is stopped. The idea is that there is already a better path that reaches this part of the space, so this new trajectory might be unnecessary. Nevertheless, for probabilistic completeness purposes we do propagate a single state from the trajectory into this cell so that it can be potentially revisited by the algorithm in subsequent iterations.

E. Algorithm

Algorithm 2 summarizes the overall operation of IST as described in the previous sections. The algorithm assumes that the database of trajectories is available.

Algorithm 2 INFORMED SUBDIVISION TREE

(Initialization)
Set the root of $\mathcal{T}$ to the initial state $x_0(t_0)$
Create a cell c_0 that corresponds to the entire state space $\mathcal{X}$
Initiate the set of subdivision cells $S = \{c_0\}$
Execute a PRM for a kinematic version of the system

while $\nexists\, x \in \mathcal{T}$ so that $x \in \mathcal{X}_G$ (goal region) **do**
  (State Selection)
  Select the cell $c_{min} = argmin_{\forall c \in \mathcal{S}} \{score(c)\}$
  Select the edge $e_{min} = argmin_{\forall e \in c_{min}} \{score(e)\}$
  Update the penalty value: $p(e_{min}) = 2 \cdot p(e_{min})$
  Select a random state $x_r \in e_{min}$
  Split the cell c_{min} according to the subdivision rules
  Split the edges in c_{min} to the corresponding cells

  (Control Selection)
  if first time that e_{min} is selected **then**
    Find u_v that brings to a state with good heuristic value $h(\cdot)$ based on the trajectory database (no collision-checking)
  else
    Compute the valid control u_v based on the probability distribution of e_{min}
  end if
  Given u_v, update probability distribution of e_{min}

  (Duration Selection)
  while The resulting trajectory is feasible and there is no better path from x_0 to the current cell **do**
    Propagating for time dt
  end while

  Add the primitive trajectory $e_{new} = \pi(x_r, (u_v, dt))$ in $\mathcal{T}$
  Set the penalty parameter of $p(e_{new}) = p(e_{min}) + 1$
end while

III. Probabilistic Completeness

Due to space limitations we will provide only an outline of the arguments for the algorithm's probabilistic completeness. In order to prove that a sampling-based planner is

probabilistic complete it is sufficient to show that it can eventually produce any random walk in $\mathcal{X}_{free}$ (Ladd and Kavraki 2004). This criterion can be achieved if we can show that all the states along the tree data structure $\mathcal{T}$ are selected infinitely often and that all possible controls can be eventually propagated from each state.

It is easy to show that the propagation step allows for probabilistic completeness. Although the control selection process is biased we retain a non-zero probability for every control. Another issue is related to the duration of the propagation. Although we stop the propagation early if it enters a new cell with a better trajectory, we still propagate at least one state within this cell. As we argued, this allows the edge to be propagated again later and does not effect the proof.

The argument that the state selection step allows every state to be selected infinitely often follows the structure of the algorithm. We must first show that every cell will be selected and then that this also holds true for every edge:

(i) *Every cell will be selected in finite time:* The heuristic value must be upper and lower bounded by finite positive values $h_{max} > h_{min}$ and $h_{min} > 0$. Assume a cell $c \in \mathcal{S}_M$, where $\mathcal{S}_M$ is the state of the subdivision at iteration M. Its score $score(c)$ depends on its heuristic value $h(c)$ and its subdivision level $\mu(c)$. As long as it is not selected, $\mu(c)$ does not change, while $h(c)$ can only improve by newly propagated edges inside c.

Lets assume the worst case, where $score(c)$ does not improve on consecutive iterations and every other cell $c' \in \mathcal{S}_M$ has the best possible heuristic:

$$h(c') = h_{min},\ \forall c' \in \mathcal{S}_M \text{ and } c' \neq c.$$

Even in this case, however, there is a finite iteration $N > M$ where the score of every other cell will be worse than the score of cell c: $score(c') > score(c),\ \forall c' \in \mathcal{S}_N$. At some point all the children of a cell $c' \in \mathcal{S}_M$ will have scores worse than $score(c)$ because their subdivision level will be increasing and their heuristic is lower bounded by a positive value. Furthermore, the number of children cells of all c' with better score than c will be finite. Consequently, the cell c at some finite iteration $N > M$ will be selected.

(ii) *Every edge in a cell will be eventually selected:* Similar is the argument for an edge e, only this time the proof is based on the edge penalties $p(e)$. Since edges that are selected get penalized and newly propagated edges get increasing penalties, at some point the penalties of all other edges will be so high that edge e will be selected even if the path cost of e is very high. The number of newly propagated edges in the cell with better score is finite.

The combination of all the above arguments results in a proof that the algorithm is probabilistic complete as it has been described. We did not have to resort to simplistic fixes such as for a certain percentage of iterations to select a random state. Other greedy algorithms typically refer to this approach as a way to provide with probabilistic completeness. Our state selection process tries to make informed choices given the heuristic at every iteration of the algorithm.

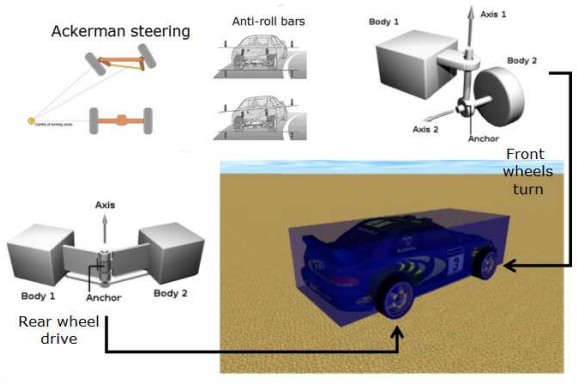

Figure 7: We use a car-like system as our testbed in this work. The car is modeled as five rigid bodies, the chassis and the four wheels, connected through four joints. The front joints allow the wheels to be steerable, while the back joints allow the car to accelerate. In order to have a car that does not flip over often in the physics-based simulation we have to: (a) apply controls to the wheels that follow the Ackerman steering model and (b) to simulate the effect of anti-roll bars that real cars have.

IV. Experiments

In order to evaluate the efficiency of the proposed approach we have executed experiments using a physically simulated car in a variety of scenes. The simulation environment is based on the Open Dynamics Engine (ODE) (Smith 2006). Fig. 7 provides details about the simulation. The high-level controls for the car are acceleration and steering velocity, which are appropriately translated into control parameters to the joints that connect the wheels with the chassis.

The workspaces for which we provide comparisons in this paper are displayed in Fig. 8. The first is an approximation of a known benchmark for motion planning, the bug-trap problem. The second problem, referred to as the iso-test problem. It requires the car to swerve between obstacles in a road-like environment, and it has been reported in the literature as a challenging case for sampling-based kinodynamic planners (Boyer and Lamiraux 2006). The last problem is the most challenging in terms of the workspace constraints since it is a maze-like environment.

The algorithms that we compare against include the three uninformed sampling-based kinodynamic planners described in the introduction section, RRT (LaValle and Kuffner 2001b), Expansive Spaces (Hsu, Latombe, and Motwani 1999) and PDST (Ladd and Kavraki 2005a). We have also experimented with the informed variant of RRT, called RRT-"goal bias", which 5% of iterations selects for expansion the state along the tree closer to the goal. The last alternative we tested is the RRT* (Urmson and Simmons 2003) that integrates an A* like heuristic in the RRT algorithm.

The results of our experiments in terms of computation time are shown in Fig. 6. The cost for the computation of the probabilistic roadmap is included in the computation time of the IST. The proposed algorithm is able to outperform all the alternative techniques in all workspaces. In some cases, the speedup is close to one order of magnitude.

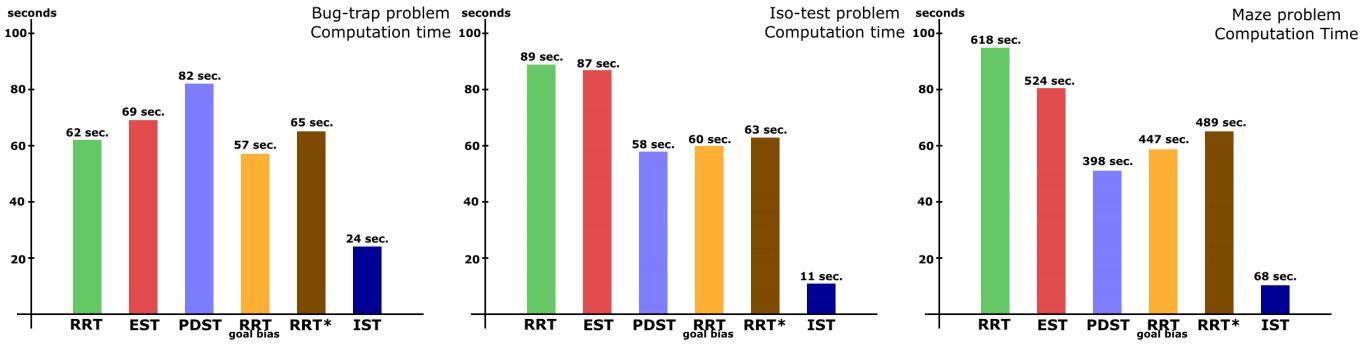

Figure 6: Comparison of computation time on the bug trap, iso-test and maze workspaces. Averages of 50 experiments.

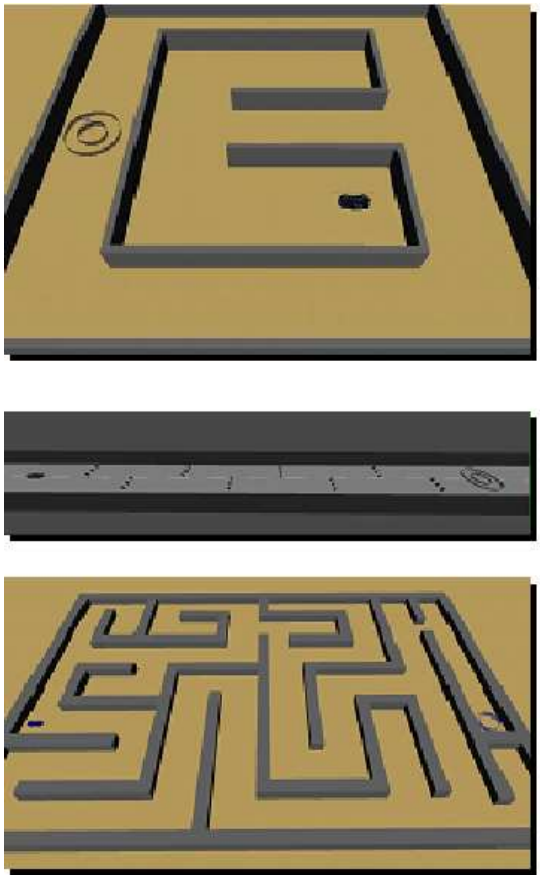

Figure 8: The bug trap, iso-test and maze workspaces.

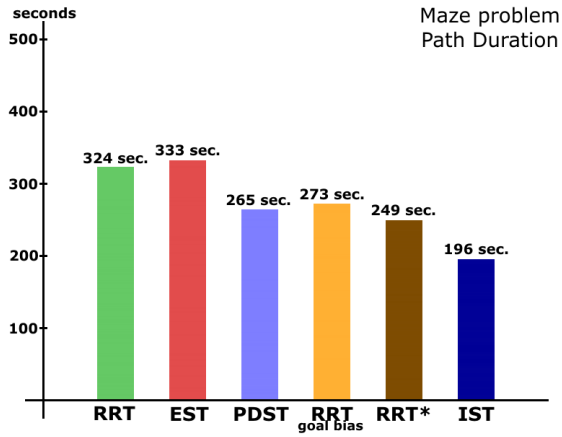

Figure 9: Comparison of path duration on the maze scene. Averages over 50 experiments.

IST not only outperforms the uninformed planners but is also able to perform better than RRT-"goal bias" and RRT*. The best results are achieved for the maze-like environment. In this workspace a heuristic that uses workspace or C-space knowledge is considerably advantageous.

Fig. 9 shows a different statistic. It provides the average duration of a path computed by the algorithms on the maze. It is noteworthy, that although IST computes the solution faster, it is also able to compute a better path by taking advantage of the heuristic. The maze is large enough to allow us to see this difference. For the other two scenes, almost all of the algorithms computed trajectories of similar duration.

It must be noted that further improvement in path quality is to be expected by implementing an anytime approach as the Anytime RRT algorithm (Ferguson and Stentz 2006). Anytime planners continue searching after a solution has been already found with the objective of improving path quality. The time to compute the first solution path with Anytime RRT is the same as with RRT, since the first algorithm uses the second as an initialization procedure. The hope is that since IST appears to perform better than RRT in problems with physically-simulated dynamics, an anytime version of IST will have similar advantages. We are planning to investigate the properties of an anytime version of IST in future work.

V. Discussion

Sampling-based kinodynamic planners are able to solve some hard kinodynamic problems but they often face the problem of regression as they propagate trajectories in already explored space. Heuristic search can be very beneficial in guiding the operation of sampling-based planners in such challenging problems so as to avoid regression and focus the search in the part of the state space that is beneficial to the solution of a problem. Nevertheless, heuristic search in continuous spaces is not as straightforward as in traditional discrete AI problems. For example, it is not obvious how to scale the heuristic parameter against the true path

cost. Furthermore, it is still very important that eventually the entire state space will be covered in order to be able to provide with the guarantee of probabilistic completeness.

This work emphasizes the importance of heuristic search in sampling-based kinodynamic planning and describes an algorithm, the Informed Subdivision Tree (`IST`), that achieves informed search, while still providing probabilistic completeness guarantees. The algorithm integrates the heuristic information in an adaptive subdivision scheme. The subdivision is used to appropriately discretize the state space and estimate in an online fashion the algorithm's performance in state space exploration. The heuristic information is used for the selection of the next cell in the subdivision from where the tree data structure of the sampling-based planner will be expanded from. At the cell level the algorithm operates in a depth-first manner. Then the algorithm moves on to select an edge within the cell given the best path cost from the starting state and a penalty factor. This means that at the level of selecting an edge within a cell the algorithm operates in a breadth-first manner. In this way, there is no need to scale the heuristic with the true path cost, while we still achieve an A^* like behavior overall. Heuristic information is also employed in the propagation step of the algorithm. Experiments on various workspaces for a physically simulated system show that `IST` outperforms uninformed sampling-based kinodynamic planners as well as some existing informed variants.

Acknowledgments

Work on this paper by K. Bekris and L. Kavraki has been supported in part by NSF IIS 0713623 and NSF IIS 0308237. The computational experiments were run on equipment obtained by CNS 0454333, and CNS 0421109 in partnership with Rice University, AMD and Cray.

References

Barraquand, J., and Latombe, J. C. 1991. Robot motion planning: A distributed representation approach. *International Journal of Robotics Research* 10(6):628–649.

Boyer, F., and Lamiraux, F. 2006. Trajectory optimization applied to kinodynamic motion planning for a realistic car model. In *IEEE Intl. Conf. on Robotics and Automation*, 487–492.

Choset, H.; Lynch, K. M.; Hutchinson, S.; Kantor, G.; Burgard, W.; Kavraki, L. E.; and Thrun, S. 2005. *Principles of Robot Motion: Theory, Algorithms and Implementation*. Boston: MIT Press.

Diankov, R., and Kuffner, J. J. 2007. Randomized statistical path planning. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'07)*.

Donald, B.; Xavier, P.; Canny, J.; and Reif, J. 1993. Kinodynamic motion planning. In *Journal of the ACM*, volume 40, 1048–1066.

Ferguson, D., and Stentz, A. 2006. Anytime RRTs. In *IEEE/RSJ Intelligent Robots and Systems (IROS-06)*, 5369–5375.

Hsu, D.; Latombe, J.-C.; and Motwani, R. 1999. Path planning in expansive configuration spaces. *International Journal of Computational Geometry and Applications* 9(405):495–512.

Kalisiak, M., and Van de Panne, M. 2006. RRT-Blossom: RRT with A Local Flood-Fill Behavior. In *IEEE Intl. Conf. on Robotics and Automation*.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE TRA* 12(4):566–580.

Kavraki, L. E.; Kolountzakis, M.; and Latombe, J.-C. 1996. Analysis of probabilistic roadmaps for path planning. In *Proc. of the Intl. Conf. on Robotics and Automation (ICRA '96)*, 3020–3026.

Ladd, A. M., and Kavraki, L. E. 2004. Measure theoretic analysis of probabilistic path planning. *IEEE TRA* 20(2):229–242.

Ladd, A. M., and Kavraki, L. E. 2005a. Fast tree-based exploration of state space for robots with dynamics. In *WAFR*, 297–312.

Ladd, A. M., and Kavraki, L. E. 2005b. Motion planning in the presence of drift, underactuation and discrete system changes. In *Robotics Science and Systems-2005*.

Latombe, J. C. 1991. *Robot Motion Planning*. Boston, MA: Kluwer Academic Publishers.

LaValle, S., and Kuffner, J. 2001a. Rapidly exploring random trees: Progress and prospects. In *WAFR*, 293–308.

LaValle, S. M., and Kuffner, J. J. 2001b. Randomized kinodynamic planning. *IJRR* 20(5):378–400.

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

Plaku, E., and Kavraki, L. E. 2006. Quantitative analysis of nearest-neighbors search in high-dimensional sampling-based motion planning. In *Intl. Workshop on Algorithmic Foundations of Robotics (WAFR)*.

Rickert, M.; Brock, O.; and Knoll, A. 2007. Balancing exploration and exploitation in motion planning. In *Proc. of the International Conference on Robotics and Automation, ICRA-07*.

Sanchez, G., and Latombe, J.-C. 2003. A single-query bi-directional probabilistic roadmap planner with lazy collision checking. *Robotics Research, STAR 6* 403–407.

Smith, R. 2006. *Open Dynamics Engine: v05. User Guide*.

Urmson, C., and Simmons, R. 2003. Approaches for Heuristically Biasing RRT Growth. In *Intl. Conf. on Intelligent Robots and Systems*, volume 2, 1178–1183.

Yershova, A., and LaValle, S. M. 2007. Improving motion planning algorithms by efficient nearest-neighbor searching. *IEEE Transactions on Robotics* 23(1):151–157.