

Sampling-Based Planning for Discrete Spaces

Stuart Morgan and Michael S. Branicky
Electrical Engineering and Computer Science Dept.
Case Western Reserve University
Cleveland, OH, USA
{sbm5, msb11}@case.edu

Abstract—In this paper, we introduce several discrete-space search algorithms based on continuous-space motion-planning techniques such as Rapidly-exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs). We describe methods for adapting these algorithms for discrete use by replacing distance metrics with cost-to-go heuristic estimates and substituting local planners for straight-line connectivity. Finally, we explore coverage and optimality properties of these algorithms in discrete spaces.

I. INTRODUCTION AND OVERVIEW

Sampling-based planning algorithms, most notably Rapidly-exploring Random Trees (RRTs) and Probabilistic Roadmaps (PRMs), have gained significant popularity in recent years as a tool for solving high-dimensional motion-planning problems. For the most part, their applications have been limited to continuous planning. Recently, however, sampling-based planning algorithms have begun to be applied to a wider category of problems, specifically hybrid and discrete planning.

While the properties and characteristics of sampling-based planning have been studied in continuous spaces, little work has been done to date in exploring these properties in other spaces. Here, we examine the extent to which the properties that make sampling-based planning effective in continuous space are applicable to discrete spaces, and how sampling-based planning might adapt to the issues encountered in discrete space.

Although a variety of optimal and near-optimal solution methods—such as A^* and its variants—exist for solving discrete search problems, they generally become very time- and space-intensive as problem domains become larger and more complex. Sampling-based discrete planners may offer an efficient way of finding feasible paths in complex problems where finding optimal paths is either unnecessary or computationally infeasible.

This paper is organized as follows: First, we give background on sampling-based planning algorithms. Next, we introduce adaptations of sampling-based planners to discrete space. Then, we provide experimental planning results in several sample test problems. Finally, we explore properties of sampling-based planners in a grid world. Full details of this work, including implementation details and more extensive numerical results, are available in [1].

II. SAMPLING-BASED PLANNING BACKGROUND

A. RRTs

RRTs are a probabilistic exploration method developed for searching the high-dimensional continuous spaces en-

countered in motion planning and control problems [2], [3], [4]. They were later adapted for use in hybrid-systems planning and control, and were recently extended preliminarily into discrete space [5], [6], [7].

The RRT algorithm begins with an initial configuration q_{start} as the tree. At each step, it selects a random configuration q_{rand} from the configuration space, then finds the nearest configuration already in the tree, q_{near} , using some definition of nearness (often Euclidean distance). From q_{near} , it moves some distance ϵ toward q_{rand} , and adds that new configuration to the tree. These steps are repeated until some q_{goal} (which may be a specific goal state or an element of a set of goal states) is reached, or until the tree has reached a certain size. RRTs have been shown to be probabilistically complete, and to have good space-filling properties in that their growth is biased toward the largest unexplored regions in the space [2], [8].

The reason for this bias toward unexplored areas is clear when the algorithm is examined in terms of Voronoi diagrams. The Voronoi regions of the nodes in a tree give important indications of the degree to which it has explored the space. First, the size of the largest Voronoi region(s) tell us how far, in the worst case, a point in the space can be from the tree's nodes. Second, the locations of large regions tell us where those worst-case points are, and thus which areas have been least thoroughly explored. Finally, the range of sizes in a Voronoi diagram tell us how evenly the space has been explored: if all of the Voronoi regions are roughly the same size, then the coverage of the tree is essentially uniform, whereas large size disparities indicate that some regions have been explored much more thoroughly than others.

The RRT algorithm leverages these properties of Voronoi regions probabilistically: so long as the random points in the RRT algorithm are selected uniformly from the space, the largest Voronoi regions will be the most likely to contain the target point [2], [8]. Since the selected region's node is expanded, the tree will be biased toward expanding the edges of the tree which border on the largest unexplored regions, dramatically so in the early stages of growth.

B. RRTs in Hybrid Systems

Although RRT research has focused primarily on continuous planning, recent research has shown that the RRT algorithm can be successfully applied to hybrid systems as well [6], [7], [9]. This research adapted the RRT algorithm to solve a variety of hybrid planning problems by extending

the RRT’s state to include discrete state information as well as the continuous configuration, and adding the state-change as an action when the agent reaches a switching boundary. Nearest-neighbor queries incorporated a heuristic evaluation to account for differences in discrete state.

C. PRMs

PRMs constitute another sampling-based method for solving planning problems in high-dimensional spaces, especially multiple query path planning [10], [11].

The PRM algorithm operates by repeatedly selecting a random node q_{rand} and adding it to a set of nodes. The algorithm then attempts to connect q_{rand} to any other nodes in the set that are within some distance δ of q_{rand} . The connections are made by a local planner; in the simple case of holonomic planning, the local planner is often simply a straight-line generator with obstacle checking. This process is repeated until the PRM reaches a set size, or until most or all of the PRM is connected. Path planning queries are then solved by using a local planner to connect q_{start} and q_{goal} to the PRM, then finding a path between the connection points through the pre-planned roadmap.

Like RRTs, PRMs are probabilistically complete using uniform sampling. Unlike RRTs, which expand out from one point, the nodes in a PRM follow the sampling distribution exactly, but do not create a connected graph until some threshold point density is reached.

III. DISCRETE SAMPLING-BASED PLANNING

In general a discrete planning problem exists in a discrete space consisting of a countable set of states \mathbf{S} , and a corresponding set of discrete dynamics that define, for each state q , a transition rule $\Delta : \mathbf{S} \rightarrow 2^{\mathbf{S}}$, where $\Delta(q) = \mathbf{Q}' \subseteq \mathbf{S}$ is the (finite) set of possible successors to q . The planning problem gives us a start state q_{start} and a set of goal states \mathbf{G} , and asks us to find a path from q_{start} to some $q_{\text{goal}} \in \mathbf{G}$. This solution path must consist of a sequence of states $q_{\text{start}} = q_0 \rightarrow q_1 \rightarrow q_2 \rightarrow \dots \rightarrow q_n = q_{\text{goal}}$ with each transition to a new state obeying the transition rule for the previous state (i.e., for each transition $q_i \rightarrow q_{i+1}$, it must be true that $q_{i+1} \in \Delta(q_i)$).

In discrete planning, it is often useful to consider the transition rules for states as corresponding to a small set of universal or near-universal operators, each of which map $q \rightarrow q'$ in a predictable way. For example, in planning an agent’s motion in a grid world, where each state corresponds to an (x, y) pair, the operators are the four possible moves to adjacent squares: $o_{\text{up}}(x, y) \rightarrow (x, y+1)$, $o_{\text{down}}(x, y) \rightarrow (x, y-1)$, $o_{\text{left}}(x, y) \rightarrow (x-1, y)$, and $o_{\text{right}}(x+1, y) \rightarrow (x, y)$ (with the constraint that an operator cannot be applied to a state if its resultant state would collide with an obstacle).

Although a variety of informed search algorithms exist for such problems, they generally either become infeasible as the problem size grows (as in the case of A^*) or have poor performance in spaces where “obstacles” are not predicted by the heuristic, whether the obstacles are explicit or due to behavior of the system which is not predicted by the simplified heuristic (as in the case of best-first search).

A. Discrete RRTs

In [6] we introduced a discretization of the RRT algorithm, which replaces the distance metric used for determining nearness with a heuristic estimate of the cost-to-go of the same type that is used in general informed search methods, e.g., A^* . As in the original RRT, the discrete algorithm begins with an initial state q_{start} . At each step, we select a random state q_{rand} from the state space (making sure that the state selected is not already in the tree). We find the nearest state in the tree, q_{near} , based on a heuristic estimate of the cost-to-go from each state to q_{rand} . Considering each possible operator on q_{near} , we select the one which yields the successor state q_{new} that is closest to q_{rand} but is not already in the tree, and add q_{new} to the tree with an edge from q_{near} to it. If q_{near} has no successors which are not in the tree, no new node is added during this iteration. In pseudo-code, the RRT construction algorithm is as follows:

```
GrowRRT( $q_{\text{start}}$ )
   $T$ .init( $q_{\text{start}}$ )
  for  $n = 1$  to  $N$ 
     $q_{\text{rand}} = \text{RandomUnexploredState}()$ 
    ExtendRRT( $q_{\text{rand}}, T$ )
```

```
ExtendRRT( $q_{\text{rand}}, T$ )
   $q_{\text{near}} = T$ .nearestTreeNode( $q_{\text{rand}}$ )
  if  $q_{\text{near}}$ .hasUnseenSuccessors()
     $q_{\text{new}} = \text{NearestSuccessor}(q_{\text{rand}}, q_{\text{near}})$ 
     $T$ .addChildNode( $q_{\text{new}}, q_{\text{near}}$ )
```

Since the state q_{new} is chosen from a finite set of successor states, rather than being grown along a straight line toward q_{rand} as in the continuous RRT, the discrete RRT is not guaranteed to take optimal steps toward q_{rand} . In fact, it is possible to encounter situations where q_{new} is a step *away*, leaving q_{rand} heuristically closer to q_{near} than to q_{new} (see Figure 1).

A new algorithm that mitigates the issue of sub-optimal steps is the *Rapidly-Exploring Random Leafy Tree* (RRLT). The RRLT algorithm keeps an open list of all states reachable in one step from the current tree: the “leaves” of the tree nodes. When q_{rand} is selected, the nearest leaf is located directly and added to the tree, and all of its successors are added to the open list (except those that are already tree or leaf nodes). The RRLT algorithm prevents the possibility of a failed ExtendRRT step, since every leaf is a state which does not exist in the tree, and is therefore a valid candidate q_{new} . To obtain the RRLT algorithm, we substitute the call to ExtendRRT with a call to ExtendRRLT:

```
ExtendRRLT( $q_{\text{rand}}, T$ )
   $q_{\text{new}} = T$ .nearestTreeLeaf( $q_{\text{rand}}$ )
   $T$ .changeLeafToNode( $q_{\text{new}}$ )
   $T$ .addNewLeaves( $q_{\text{new}}$ )
```

Since $q_{\text{near}} = q_{\text{new}}$ is selected from the list of leaf states, and not from among the states already in the tree, the algorithm guarantees that at each step the tree grows as far as possible (heuristically) toward q_{rand} . Table I shows the

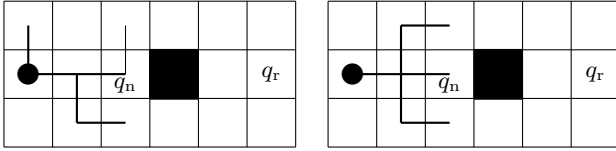


Fig. 1. Non-optimal (left) and failed (right) RRT steps. In each case, growing from a node other than the nearest neighbor, q_n , would yield a new state closer to the random target state, q_r .

results of growing RRTs and RRLTs to fixed percentages of the total space in an 8-Puzzle [12], and measuring the coverage of each tree according to the average distance from every state in the space to the nearest state in the tree (as measured absolutely). Results are averaged over three trials, and although the improvements due to the RRLT algorithm are small, they generally varied less than 0.2% between trials (and never more than 0.4%). Although the RRLT algorithm is more space-intensive than the simple discrete RRT algorithm—by factor of at most b , the branching factor of the space, and generally less since a solution can be found sooner—our experience has been that it is generally faster and finds better solutions [1].

TABLE I
DIFFERENCES IN 8-PUZZLE COVERAGE BY RRT AND RRLT

Percent Filled	Average RRT Distance	Average RRLT Distance	RRLT Percent Improvement
5%	3.94	3.90	1.0%
10%	3.23	3.18	1.5%
15%	2.79	2.74	1.9%
25%	2.13	2.06	3.3%
50%	1.01	0.93	8.4%
75%	0.36	0.31	13.4%

B. RRTs with Local Planners

In continuous space, the step-size ϵ of the RRT algorithm can be varied to change the characteristics of the search. In general, a very small ϵ results in a tree which explores slowly, but fills the space more completely and gives shorter solution paths. Although we are constrained in discrete space to take steps between states according to the transition rule in the underlying dynamics, it is possible to achieve a similar effect to varying ϵ by using an RRT or RRLT as a global planner, with a different planner used for local planning: At each step, instead of picking the nearest leaf to the randomly selected node q_{rand} , we perform a search with a local planner (e.g., A^* , best-first search, or even another RRT or RRLT)—limited by depth, size, and/or time—from the nearest neighbor, q_{near} , toward q_{rand} . When the limit of the local search is reached, the node closest to q_{rand} is added to the tree, along with the nodes along the path to reach it. Here, the degree to which the local search is limited plays a similar role to the parameter ϵ in the continuous RRT. The most significant difference is that the limit on the local planner has a significant impact on the running time of each step of the algorithm. Whereas a large ϵ has only a linear effect

on the continuous RRT’s iterative running time (assuming incremental collision-checking), relaxing the limitations on the local search could cause the running time of a discrete RRT to increase exponentially.

Because the distance heuristic is based on discrete states, it is not uncommon to find multiple neighbors with equal heuristic distance from q_{rand} . Since the goal of using a local planner is to maximize exploration at each step, we chose the node from among the ties for nearest by selecting for maximal cost from q_{near} . Although this approach may result in higher solution-path lengths than would result from breaking ties by selecting for minimal cost, or selecting randomly, intuition suggests that the maximal cost should be the maximal distance from q_{near} (assuming a relatively optimal local search). Since the cost from q_{near} is an actual cost, whereas the heuristic distance to q_{rand} is estimated, it seems reasonable to assume that considering the actual cost as well as the heuristic cost would give a more reliable estimate of distance toward q_{rand} than considering the heuristic alone.

We can further improve the parallel between the global planner and the continuous ϵ value by building a Meta-RRT or Meta-RRLT as a global planner that considers the path to the new node as a unit rather than as a sequence of nodes in the tree. Thus, instead of adding all the nodes along the path, we add the node at the end of the path as a child of q_{near} , storing information about the intermediate nodes as meta-data of the edge between q_{near} and the new node. Whereas the continuous tree always adds exactly one new node, regardless of ϵ , the non-meta global planner described above adds a sequence of nodes, all of which are close together in the space. This degrades the nearest-neighbor query time, without adding significantly to the coverage of the tree, especially in cases where we are primarily interested in exploring quickly. The meta-tree improves nearest-neighbor query running time by discarding these less-useful intermediate nodes.

C. Discrete PRMs

An approach closely related to the Meta-RRT is to combine a PRM as a global planner (that uses a heuristic to define the connection radius) with a discrete local planner for connecting nodes. The discrete PRM is constructed using very similar techniques to those described for the Meta-RRT: At each step of growth, a random node is created, and a neighborhood is defined which encompasses all nodes within some distance δ , as defined by a heuristic cost-to-go estimate (just as in RRT nearest-neighbor querying). For each node in the neighborhood, an attempt is made to connect that node to the new node. Since each PRM connection attempt is all-or-nothing, unlike the incremental growth of the Meta-RRT, bi-directional search can be used to allow significantly faster connection search. A continuous version of this idea, using local planners such as bi-directional RRT search to connect nodes in a PRM, has been shown to be effective in high-dimensional motion planning [10], [11].

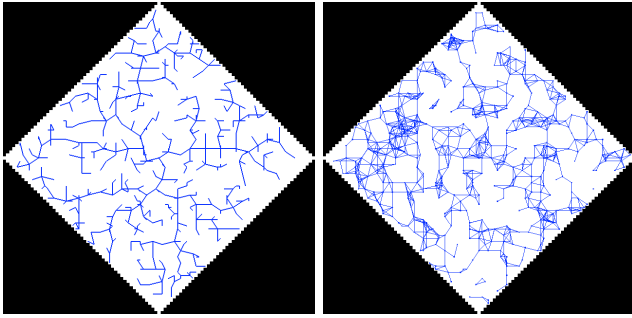


Fig. 2. A Meta-RRT (left) with a local search depth of 5 and PRM (right) with a connection radius of 5, both grown in a discrete grid world using the L_1 (Manhattan distance) heuristic.

D. Optimal-Path-Biased RRTs

Although the RRT algorithm is primarily concerned with finding solutions quickly, we generally prefer low-cost solutions over high-cost solutions when all other factors are equal. In many cases, even a slight reduction in solution speed is acceptable if it results in a noticeable improvement in solution optimality.

Achieving a speed–optimality trade-off in the RRT and RRLT algorithms is possible by modifying the nearness heuristic to consider the cost from the starting configuration in addition to the estimated cost-to-go. In this method, we define the nearness of a state q_{tree} in the RRT to the random node q_{rand} as $f(q_{\text{tree}}, q_{\text{rand}}) = \alpha g(q_{\text{tree}}) + h(q_{\text{tree}}, q_{\text{rand}})$, where $g(q_{\text{tree}})$ is the path cost from the root of the tree to q_{tree} , and $h(q_{\text{tree}}, q_{\text{rand}})$ is the heuristically estimated cost-to-go from q_{tree} to q_{rand} . A non-zero value of α allows the RRT’s exploration to be biased toward low-cost solutions by encouraging growth that moves out from the center of the tree, rather than back toward the interior of the tree.

Although any positive value of α is possible in creating optimal-path biased RRTs, in practice only small values are useful. For example, adopting $\alpha = 1$ will create an algorithm which generates optimal solutions, but does so in a vastly inefficient way when compared with A* or breadth-first search due to the need to perform nearest-neighbor queries at each iteration. However, even a very small value of α can be useful. For example, an α value very close to 0, while too small to have a significant impact on nearest-neighbor selection, will serve to break ties in nearest-neighbor queries in favor of lower-cost nodes.

IV. PLANNING RESULTS

A. Single-Query Search

We tested the single-query path planning performance of our sampling-based discrete planners on the Knight-Swapping Puzzle, which entails swapping the positions of Knights on a $k \times k$ chess board (where k must be odd), using only valid knight moves. The set-up consists of filling all but the center square with knights, divided between black and white along a diagonal (see Figure 3). For this puzzle, all experiments were performed using the heuristic value obtained by summing the number of required knight moves (ignoring the presence of other

knights) for each out of place knight to reach a destination position for its color that is not already occupied by another knight of the same color. This heuristic is *admissible* [12], but not a metric, violating both symmetry and, in some cases, the triangle inequality [1]. Although it could be made symmetrical by defining a new heuristic $h'(a, b) = h'(b, a) = \max(h(a, b), h(b, a))$, doing so would double the heuristic computation time, which is already the most costly part of RRT construction.

In order to create “obstacles” and dead-ends in our trials, we added two additional constraints to the Knight-Swapping Puzzle (see Figure 4):

- A knight cannot move to the empty square if there is not already an ally (a knight of the same color) in one of the four squares adjacent to the square.
- A knight cannot move if it would leave behind a knight without an ally in one of the four squares adjacent to it.

We solve the Knight-Swapping Puzzle by searching bi-directionally, with one tree rooted at the start state and another at the goal state. At each step, we grow each tree first randomly as described in the original RRT algorithm, then toward a node randomly selected from the other tree, in order to bias the trees toward connection [4], [5]. Once the trees connect, a solution path is extracted from the intersection of the trees. Table II shows the results of several of the algorithms described above in solving the constrained Knight-Swapping Puzzle on our test machine, a 2 GHz G5 with 2 GB of RAM. The results show that the Meta-RRLT is, in some cases, over twice as fast as the simple RRLT, although the solution paths are significantly longer. The Meta-RRLT also uses significantly less memory than either the simple RRLT or the non-meta RRLT with a local planner.

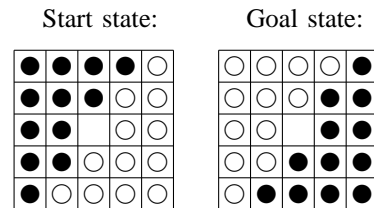


Fig. 3. The 5×5 Knight-Swapping Puzzle

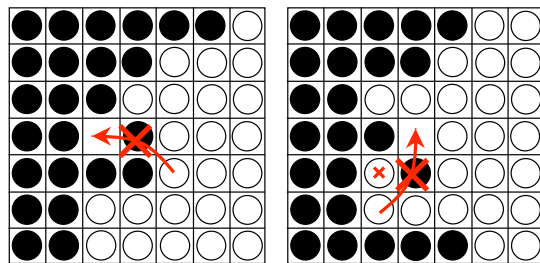


Fig. 4. Constraints on the Knight-Swapping Puzzle. The move on the left is not allowed since there is no white knight adjacent to the destination. The move on the right is not allowed since it leaves the white knight marked with the small ‘x’ without an adjacent white knight.

These results show only a small set of possible permutations of the algorithms we presented here. The choice of RRT or RRLT as a global planner, the type and details of the local planner (e.g., simple or optimal-path-biased RRT or RRLT, A*, best-first search, intermediate heuristic searches), and the methods of exploration (e.g., bi-directional or single-directional, goal-biased or unbiased, method of connecting trees in bi-directional search), all interact in ways that have significant impact on solution quality, running time, and memory requirements.

TABLE II
RESULTS OF BI-DIRECTIONAL SEARCH WITH VARIOUS ALGORITHMS
IN THE 9×9 KNIGHT-SWAPPING PUZZLE

Algorithm	Nodes	Leaves	Solution Length	Time (s)
RRLT	4580	17500	304	133
RRLT: A* Local				
100-Node Local Limit	7390	28100	433	90
500-Node Local Limit	6360	23600	425	124
Meta-RRLT: A* Local				
100-Node Local Limit	1340	5800	490	62
500-Node Local Limit	460	2370	462	101

B. Multi-Agent Planning

In order to explore prioritized multi-agent planning using discrete sampling-based planners, we discretized a version of the air traffic control problem described in [5] by placing it in a grid world. In this problem, we consider a number of airports located in a two-dimensional grid-world, with each airplane taking off from one airport and traveling to its destination airport.

The agents' paths are planned using prioritized planning: once an agent's path is planned, that path is immutable. The tree used to plan the agent's path is discarded, except for the solution path. That path becomes an obstacle in the space-time (x, y, t) for all future agents. In our trials, we extend the path obstacles to include the next and previous location of an agent at each time step, in addition to its current location, in order to define a safety buffer around each airplane. (Note that this buffer does not apply to airport cells or the 4 cells around each airport, where different rules would be used to prevent collisions.) In order to explore the added effects of static obstacles, we further allow the definition of "no-fly" zones, which act as obstacles for all agents, at all time steps.

We were able to plan the paths of 300 airplanes, with several hundred in the air at any given time, in 2–3 seconds on our test machine, using simple single-directional RRLT search (see Figure 5).

V. PROPERTIES OF SAMPLING-BASED PLANNING IN DISCRETE SPACE

In order to better understand the adaptation of sampling-based planning to discrete space, we directly examined the Voronoi regions that determine their properties, as well as the resulting coverage and optimality effects.

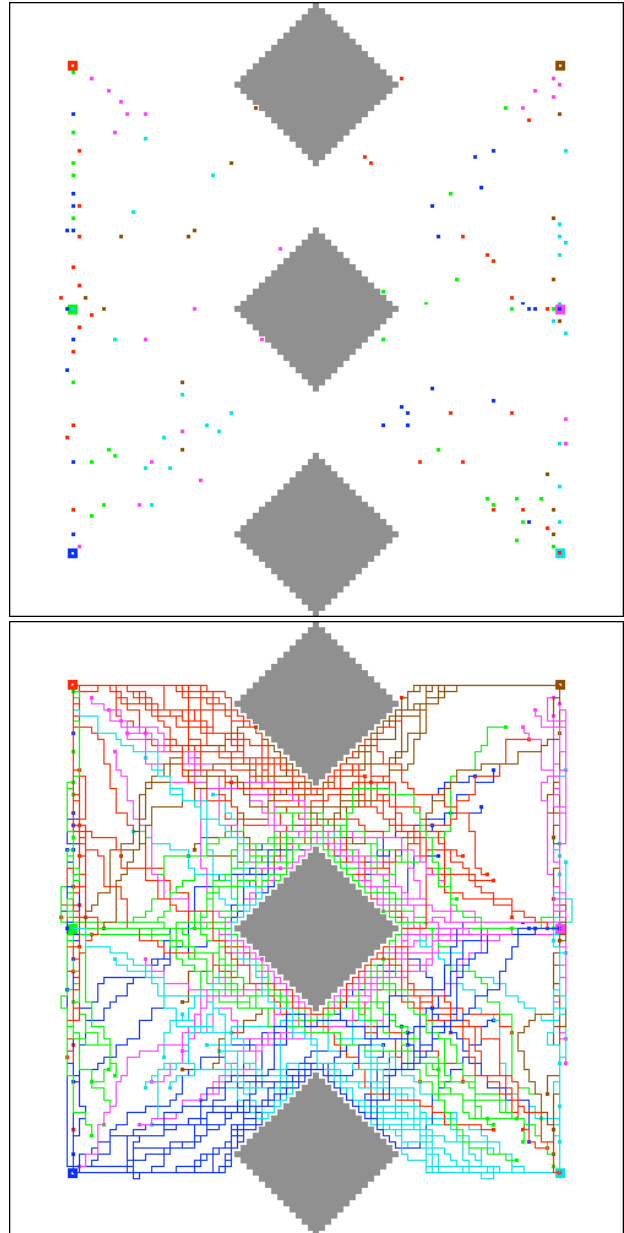


Fig. 5. Air traffic control in a grid world. The six airports are denoted by empty squares (at the four corners and the middles of the vertical sides), and each airplane is color-coded by its destination airport. The grey diamond-shaped regions indicate no-fly zones. Top: A snapshot in time showing all airplanes currently in the air. Bottom: Complete paths for all airplanes during the same snapshot.

A. Voronoi Region Overlap and Tie-Breaking

The fundamental difference between Voronoi regions in discrete and continuous spaces is the introduction of overlap in discrete space. Although the definition of a Voronoi region guarantees non-overlapping regions in continuous space, discrete space heuristics generally cannot guarantee that there will not be ties. It is of course possible to modify any heuristic to prevent ties by introducing arbitrary tie-breaking rules, but doing so will introduce an equally arbitrary bias in the exploration of the RRT algorithm, which is generally undesirable. Thus, the discrete RRT algorithm should expect the occurrence of ties.

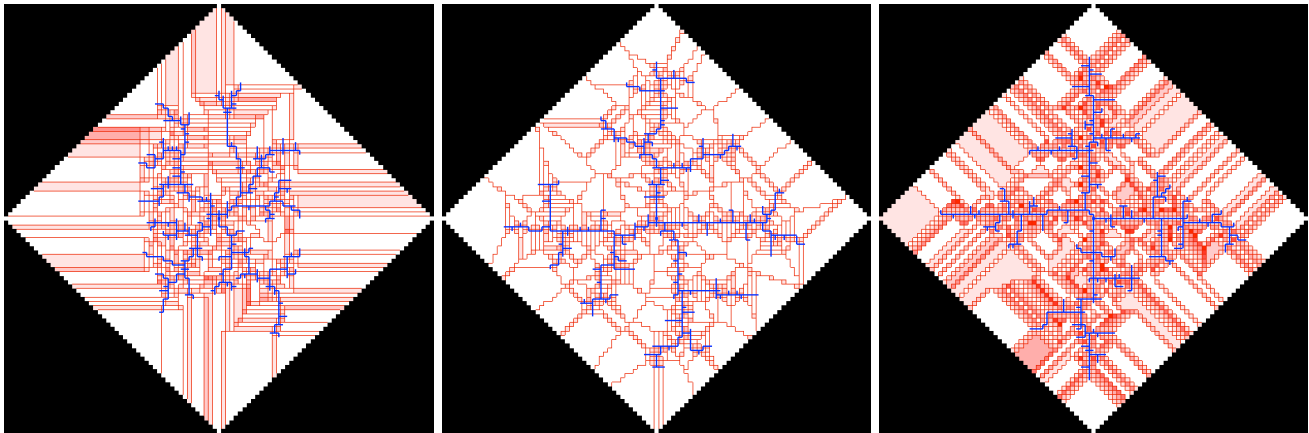


Fig. 6. Voronoi regions of RRTs in the grid world, using the L_1 (left), L_2 (center), and L_∞ (right) heuristics.

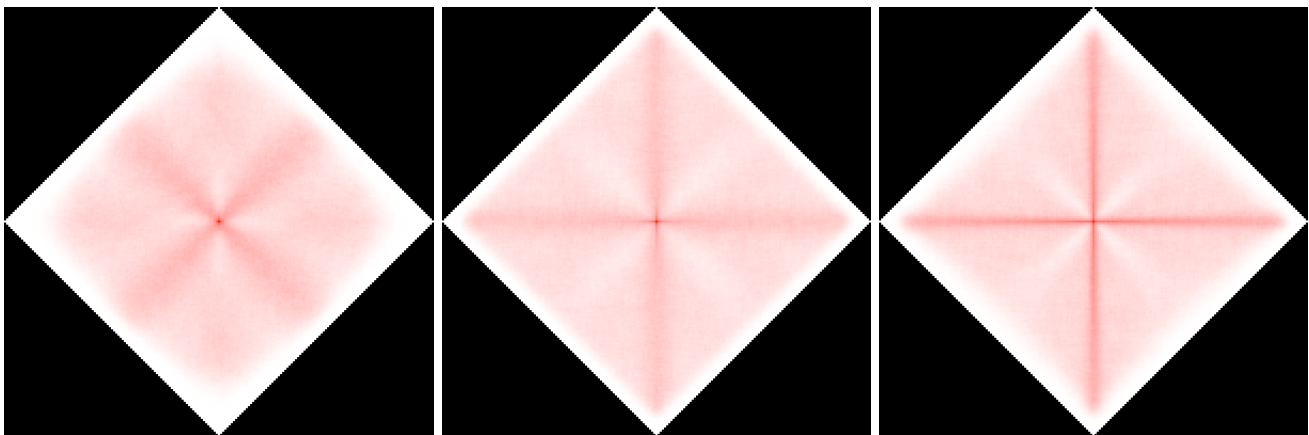


Fig. 7. RRT coverage in a grid world, using the L_1 (left), L_2 (center), and L_∞ (right) heuristics.

The extent to which Voronoi regions overlap depends strongly on the space, the heuristic used, and the extent to which the tree has explored the space, as we will show below. Since such overlap is highly likely to exist in some form, however, some decision must be made as to how to handle ties in the nearest-neighbor search. The simplest solution, which is the one we used in our experiments (except where noted), is to break ties by selecting one of the candidate neighbors at random. However, depending on the problem domain and the desired behavior of the algorithm, a variety of other solutions are possible. Using the cost-so-far of ties as a tie breaker can either emphasize optimality (by favoring low-cost nodes) or exploration (by favoring high-cost nodes, as in the case of our local planners for Meta-RRTs). Another possibility is a second-pass heuristic evaluation of tied nodes: if we can assume that the number of ties for nearest-neighbor is generally a small number regardless of tree size, then we can use a much more costly heuristic to break ties with minimal impact on the overall complexity of the algorithm.

B. Grid-World Coverage and Optimality

In order to explore the effect of different heuristics on the behavior of RRTs, we conducted trials of RRT growth in a two-dimensional grid world—where an agent

can move left, right, up, or down at each step—using the L_1 (Manhattan), L_2 (Euclidean), and L_∞ (max) metrics as distance estimates. We created visualizations of the trees and their discrete Voronoi regions (see Figure 6, which is highly typical of the patterns generated by each heuristic function), with the tree shown using thick blue (dark grey) lines, and the edges of Voronoi regions with thin red (light-grey) lines. Cells with red (grey) shading denote areas of Voronoi region overlap, with darker shading indicating more ties (especially evident in the L_∞ metric). The differences in Voronoi patterns are striking, and have a significant impact on the behavior of the tree.

1) *Coverage*: The effects of these Voronoi patterns can be clearly seen in Figure 7, which shows histograms of node location in 1000 trials of RRTs grown to 3000 nodes, using each of the three heuristics; cells with darker shading are those which are covered most often. Slower exploration in the L_1 tree is clearly evident (as seen by the lack of samples near the edges of the space), as are \times - and $+$ -shaped biases in the L_1 and L_∞ metrics. These biases are due to the distinctive shapes of the L_1 and L_∞ Voronoi regions: the L_1 metric forms large triangular regions at the edges of the space, which results in growth toward the middle of each edge, whereas the L_∞ metric creates large diamond-shaped regions in the corners, which results

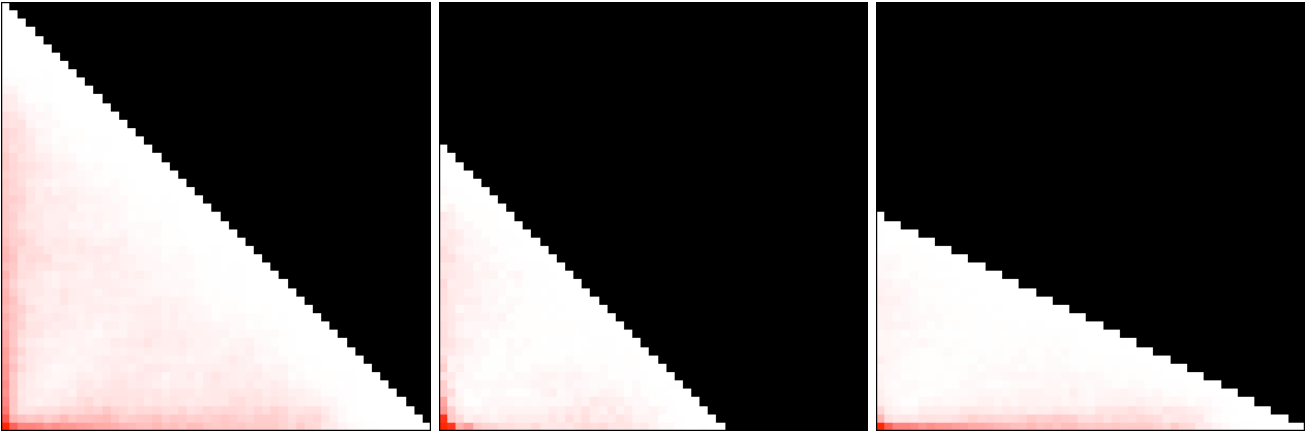


Fig. 8. RRT coverage in a 3-D grid world using the L_∞ heuristic. The panels show the slices $z = 0$ (left), $z = y$ (middle), and $z = \frac{x+y}{2}$ (right).

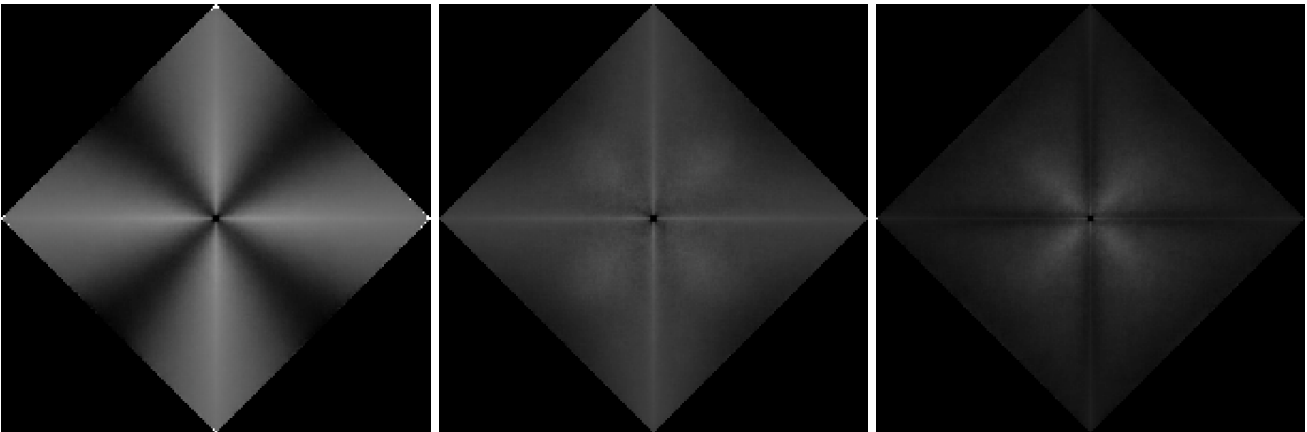


Fig. 9. Optimality of RRT paths in the grid world, using the L_1 (left), L_2 (center), and L_∞ (right) heuristics.

in growth toward the corners of the space. Other, more subtle effects are visible as well, such as the L_2 tree’s better coverage at the edge of the region and more uniform coverage of the space in general. Also noticeable are the off-bias regions near the center of the tree which are poorly sampled, especially by the L_∞ metric. Similar patterns are visible in three-dimensional grid worlds (see Figure 8).

Figure 7 also demonstrates the importance of the interaction between heuristic biases and the dynamics of the agent itself. The slight +-shaped bias in the L_2 tree is largely due to the fact that moves by the agent are in a +-shaped pattern, thereby seeding the tree in a slightly biased way during the initial moves—if diagonal moves are allowed [1], or a meta-tree or PRM is used to mask the low-level agent dynamics (see Figure 10, also averaged over 1000 trials with 3000 nodes each), the +-shape does not appear. In the case of the L_∞ tree, the agent dynamics and the Voronoi bias magnify each other, resulting in the sharp, well-defined lines along the bias axes. The L_1 tree shows the opposite effect; growth in the bias directions is approximated by selecting one of several moves at a 45-degree angle at each step. The resulting randomized zigzagging gives noticeably fuzzier regions along the bias axes. Introduction of diagonal moves, however, causes these lines to be sharply defined [1].

2) *Optimality*: The effects of growth biases introduced by interactions between heuristics and the dynamics of the discrete system are not limited to the smoothness of tree coverage. Figure 9 shows visualizations of the optimality of the paths in the tree to each point they have reached, averaged over the same 1000 trials, with darker values indicating paths closer to optimal. Specifically, the brightness of each cell on a scale of 0 to 1 is computed as $1 - d/\bar{p}$, where d is the optimal Manhattan distance and \bar{p} is the path length averaged over every trial out of the 1000 where that cell was included in the tree.

Immediately noticeable in the figures is that the L_1 metric is significantly less uniform in the optimality of its paths. This can again be explained by the interaction between the bias in the tree and the dynamics of the space. First, we must notice that even without considering any bias in the tree, not every cell has an equal chance of being reached in an optimal way by any randomized algorithm. A cell n steps from the starting point and positioned along one of the four diagonals can be reached by any one of $n!/(\frac{n}{2}!\frac{n}{2}!)$ optimal paths, whereas a cell the same distance out but positioned on the line $x = 0$ or $y = 0$ can be reached by exactly one optimal path. This creates an underlying bias toward an \times -shaped optimality histogram. In the case of the L_1 metric, that bias is magnified

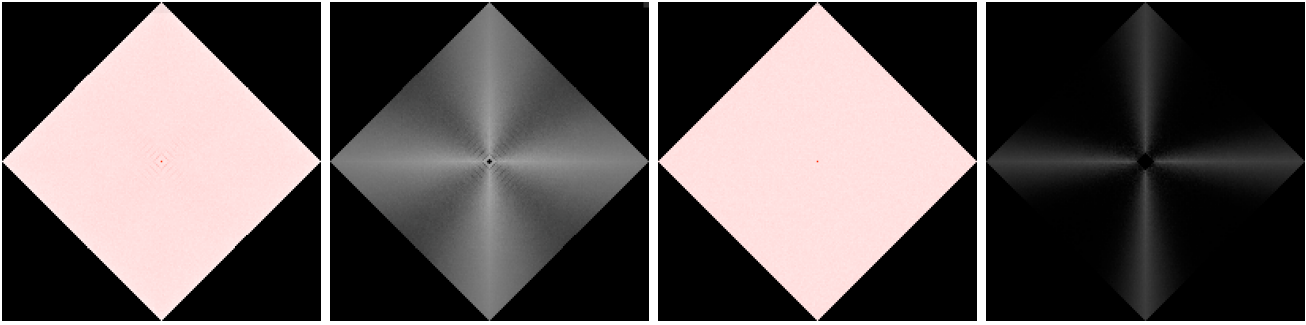


Fig. 10. Coverage and optimality of a Meta-RRT (left) and PRM (right), both with a local search radius of 5 (L_1 heuristic).

significantly by the bias in tree growth: the straight lines from the tree are predominantly along the \times , with cells furthest from the \times most likely to be reached by some amount of doubling-back in the tree in either the x or y direction. The L_2 and L_∞ metrics, on the other hand, give much more even results since the two biases serve to counteract each other. In both cases, however, there are still clearly visible lines of lower optimality along $x = 0$ and $y = 0$ due to the existence of only one optimal path.

Figure 10 shows that although optimality of a meta-tree or PRM is somewhat more uniform than that of the simple RRT, the bias of the space is still significant. The PRM's advantage in finding an optimal path is clearly evident: due to the increased number of inter-node connections—assuming that the point density is high enough to create a well-connected graph—PRM paths are significantly more optimal than those of the meta-tree.

VI. CONCLUSIONS AND FUTURE WORK

We have demonstrated that several sampling-based planning algorithms can successfully be applied and adapted to discrete space search with relatively little modification, and we explored the properties of these algorithms in simple discrete spaces. Furthermore, we explored variations on the direct discrete RRT adaptation, such as meta-trees and optimal-path biasing, as well as others not discussed here (see [1] for details).

It remains to be seen whether these algorithms will scale to very difficult discrete search and planning as well as their corresponding continuous versions have scaled to high-dimensional continuous planning, and what variations of these algorithms will prove most effective in other problem domains. However, trends in our results suggest that sampling-based discrete search is more competitive with existing methods in more complex problems [1].

The results given here are based on preliminary implementations of the sampling-based planning algorithms we presented. Further improvements and optimizations of them would be useful in determining the extent to which they are competitive with existing discrete search techniques.

Although many of the issues we have discussed were in relation to adapting these algorithms for use in discrete space, their solutions may be applicable to other planning problems as well. Highly constrained non-holonomic

agents, for example, share the issue of potentially non-optimal updates at each step of the RRT algorithm, as do some aspects of hybrid planning. As we continue to explore the properties of discrete sampling-based planning, we hope to find ways to use our results to improve sampling-based planners in a wide range of problems.

ACKNOWLEDGMENT

This work was supported by the NSF Embedded and Hybrid Systems program, under the direction of Dr. Helen Gill (grant CCR-0208919, with Frazzoli and LaValle); it does not necessarily reflect the views of the NSF.

REFERENCES

- [1] S. Morgan, "Sampling-based planning for discrete spaces," Master's thesis, Electrical Engineering and Computer Science Dept., Case Western Reserve Univ., Cleveland, OH, May 2004. <http://dora.cwru.edu/msb/pubs/sbmMS.pdf>
- [2] S.M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," Computer Science Dept., Iowa State Univ., Tech. Rep. 98-11, Oct. 1998. (<http://msl.cs.uiuc.edu/~lavalle/papers/Lav98c.ps.gz>)
- [3] S.M. LaValle and J.J. Kuffner, "Rapidly-exploring random trees: Progress and prospects," in *Algorithmic and Computational Robotics: New Directions*, B.R. Donald, K.M. Lynch, and D. Rus, Eds. Wellesley, Massachusetts: A.K. Peters, 2001, pp. 293–308.
- [4] J.J. Kuffner and S.M. LaValle, "RRT-connect: An efficient approach to single-query path planning," in *IEEE Intl. Conf. on Robotics and Automation*, San Francisco, CA, 2000, pp. 995–1001.
- [5] M.M. Curtiss, "Motion planning and control using RRTs," Master's project report, Electrical Engineering and Computer Science Dept., Case Western Reserve Univ., Cleveland, OH, May 2002. <http://dora.cwru.edu/msb/pubs/mmcMS.pdf>
- [6] M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan, "RRTs for nonlinear, discrete, and hybrid planning and control," in *IEEE Conf. on Decision and Control*, Lahaina, HI, Dec. 9–12, 2003.
- [7] J.A. Levine, "Sampling-based planning for hybrid systems," Master's thesis, Electrical Engineering and Computer Science Dept., Case Western Reserve Univ., Cleveland, OH, Jan. 2004. <http://dora.cwru.edu/msb/pubs/jalMS.pdf>
- [8] S.R. Lindemann and S.M. LaValle, "Incrementally reducing dispersion by increasing voronoi bias in RRTs," in *IEEE Intl. Conf. on Robotics and Automation*, New Orleans, LA, 2004, pp. 3251–3257.
- [9] M.S. Branicky, M.M. Curtiss, J. Levine, and S. Morgan, "Sampling-based planning and control," in *Twelfth Yale Workshop on Adaptive and Learning Systems*, New Haven, CT, May 28–30, 2003.
- [10] K.E. Bekris, B.Y. Chen, A.M. Ladd, E. Plaku, and L.E. Kavraki, "Multiple query probabilistic roadmap planning using single query planning primitives," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Las Vegas, NV, Oct. 27–31 2003, pp. 656–661.
- [11] P. Ito, "Constructing probabilistic roadmaps with powerful local planning and path optimization," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, Sept. 30–Oct. 4 2002, pp. 2323–2328.
- [12] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 2nd ed. Upper Saddle River, NJ: Prentice Hall, 2003.