Randomized Query Processing in Robot Path Planning

(Extended Abstract)

Lydia E. Kavraki[*] Jean-Claude Latombe[*] Rajeev Motwani[†] Prabhakar Raghavan[‡]

Abstract

The subject of this paper is the analysis of a randomized preprocessing scheme that has been used for query processing in robot path planning. The attractiveness of the scheme stems from its general applicability to virtually any path-planning problem, and its empirically observed success. In this paper we initiate a theoretical basis for explaining this empirical success. Under a simple assumption about the configuration space, we show that it is possible to perform preprocessing following which queries can be answered quickly. En route, we consider related problems on graph connectivity in the evasiveness model, and art-gallery theorems.

1 Introduction

Planning obstacle-avoiding motion for a rigid or articulated robot from a given initial configuration to a goal configuration is an important problem in robotics [3, 8]. Typically, the environment is static and the robot must perform a series of complicated maneuvers to achieve a sequence of goals.

A number of recent papers in the robotics literature [4, 5, 6, 7, 12, 13] have described the success of a class of randomized preprocessing heuristics for query processing in robot path planning. The key idea is the use of random sampling in a preprocessing stage, following which queries of the form "Is configuration B reachable from configuration A?" can be answered quickly. The method is very general and can be applied to virtually any type of holonomic robot. It has proved especially effective for robots with many degrees of freedom, where traditional methods have either failed to yield algorithms or have yielded algorithms that are too slow for normal use. There is another motivation for such a general query processing scheme not bound to the specifics of any particular robot: it is clearly infeasible to invest effort in tailor-made complete algorithms for every robot in existence. While the scheme is general, it is possible to tailor it to any specific type of robot and further enhance its performance [6].

Figure 1 depicts several positions of a robot with 7 revolute joints to which the method has been successfully applied. This paper initiates a theoretical basis for explaining the success of this method.

The configuration of a robot at any instant is described by an ordered tuple of real values, each entry of which is the value of one component of its position. For example, a unit square moving freely in the plane is captured by a triple: the x- and y-coordinates of a designated corner, together with the angle made by the line containing a designated edge with the x-axis. We therefore say that such a square has 3 degrees of freedom, and represent its position by a point in 3-space. The motion of the square forms a trajectory in this space. Given static obstacles in the plane that constrain the motion of the square, we may represent them in the

[*]Robotics Laboratory, Department of Computer Science, Stanford University, Stanford, CA 94305-2140. Partially supported by ARPA grant N00014-92-J-1809 and ONR grant N00014-94-1-0721.

[†]Department of Computer Science, Stanford University, Stanford, CA 94305-2140. Supported by an Alfred P. Sloan Research Fellowship, an IBM Faculty Development Award, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox.

[‡]IBM Research Division, T.J. Watson Research Center, Yorktown Heights, NY 10598.

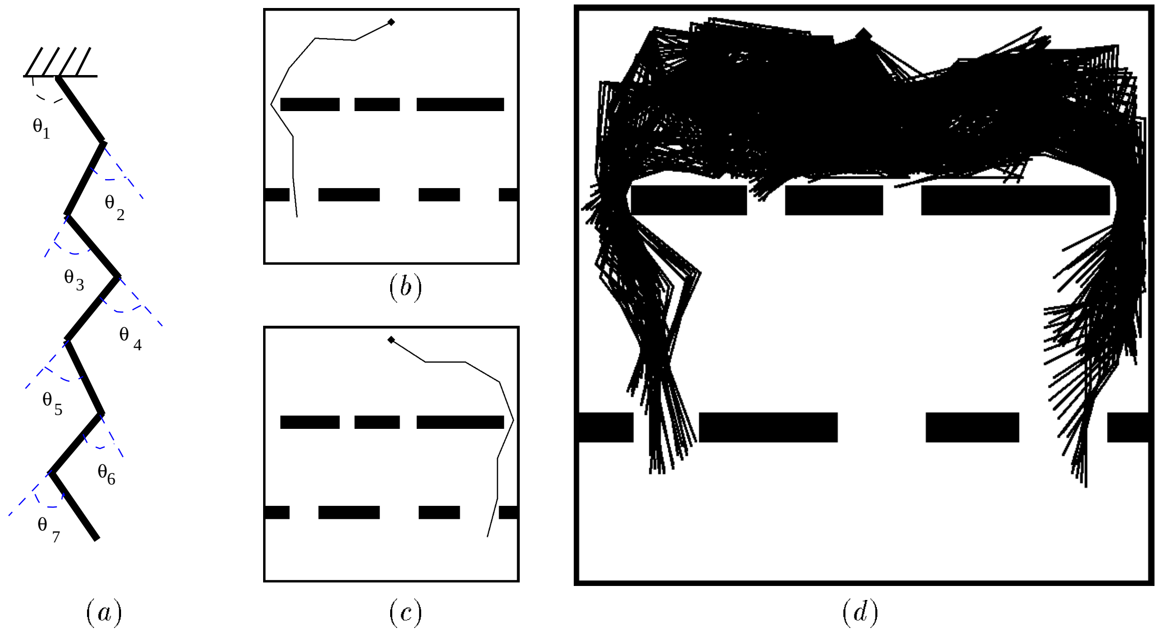

Figure 1: *Several configurations of a robot arm with a fixed base. This arm has 7 revolute joints, and must maneuver through gaps in two walls. (a) a planar arm with 7 revolute joints, (b) and (c) two different configurations of the arm, and (d) a path followed by the arm when it moves between configurations (b) and (c).*

space as a set of forbidden regions that must never be entered by the motion trajectory. The 3-dimensional space representing the position of the square together with these forbidden regions is known as the *configuration space* for this setting. Such a configuration space can be defined for any motion planning problem and, together with a cost measure and possible constraints on the shapes of trajectories, defines the problem completely. For instance, the position of the arm in Figure 1 may be represented in a space with 7 dimensions, with each dimension corresponding to the angular position of one of the revolute joints. Figure 1 (a) depicts the 7 angles giving rise to the seven degrees of freedom. Figure 1 (b) and 1 (c) depict the start and finish configurations, while Figure 1 (d) depicts a sequence of configurations found by the algorithm for going from the start configuration to the finish configuration. We refer to the subset of the configuration space that is *not* forbidden as the *free space*; it may consist of more than one connected component.

More generally, we assume here that the configuration space is the cube $[0,1]^d$, where d is the number of degrees of freedom for the robot. (Our definitions and results can be extended to cases where one or more dimensions of the configuration space — say the angular position of a joint of an arm — can "wrap around", but for simplicity we assume $[0,1]^d$ here.) For the purposes of this abstract, we also assume that the space is *reflexive:* if a point p_1 in free space is reachable from p_2, then p_2 is reachable from p_1. Non-reflexive spaces arise, for instance, when there are moving obstacles so that time becomes one dimension, or if the robot has asymmetric motion capabilities such as when only forward motion is permitted.

A key ingredient of the method is a fast *simple planner* that, given two points p_1 and p_2 in the configuration space, tries to connect them using a fast but simple strategy. For example, one simple planner that has been used for this purpose [6, 7] checks whether the line segment between p_1 and p_2 lies entirely in free space; if not, it reports failure (even though a more complicated path might exist). This is usually implemented by a walk along the line segment (suitably discretized), checking whether each of these discrete points is in free space. In addition we assume that we have access to a *complex planner* that is expensive to run, but is error-free in that it discovers a path between p_1 and p_2 whenever one exists, and reports failure when there is none. One example of such a complex planner for general configuration spaces is due to Barraquand and Latombe [1].

Such an error-free planner may be extremely slow and may not be run to completion in practice. However, if even the complex planner cannot discover a path between two connected configurations, then we may as well assume that these points are disconnected (i.e., we can view connectivity between configurations as being defined by the ability of the complex planner to find connections). Because of its expense, we seek to use this complex planner sparingly. As we will show, with high probability the preprocessing will ensure that only the simple planner is needed for answering queries. Our randomized preprocessing scheme may be summarized as follows:

1. **[Sampling]** Pick a random set of points in the free space. Call these points *milestones*.
2. **[Simple Permeation]** Try to connect all pairs of milestones using the simple planner.
3. **[Resampling]** For any milestones that are connected to relatively few others in this process, pick additional milestones "near" them at random.
4. **[Complex Permeation]** As a last resort, try using the complex planner to connect some pairs of milestones.

Step 4 is seldom used in practice, and would ideally be eliminated. In certain settings in practice this elimination may be possible with resampling and other related techniques.

The result of this preprocessing may be viewed as a graph G each of whose vertices corresponds to a milestone, with an edge signifying that its end-points are in the same component of free space. This graph is sometimes called a *probabilistic roadmap* [7].

Given a query pair of configurations q_1 and q_2 in free space, we detect whether it is possible to move from q_1 to q_2 as follows: we use the simple planner to connect q_1 and q_2 to milestones m_1 and m_2 respectively. We then use a graph search algorithm to determine whether the milestones m_1 and m_2 are in the same connected component of the roadmap G. Queries are never answered incorrectly; with some probability though, the query processing algorithm may fail to give an answer.

In our analysis, we assume that the configuration space is available as a membership oracle: given a point p in the configuration space, we can decide whether or not the point is in free space. This is reasonable in implementations [6, 8]: such a membership test corresponds to checking whether a configuration violates any of the constraints in the input, and this can be done rather efficiently. We treat the simple planner (denoted B_S) and the complex planner (B_C) as black-boxes. We assume without loss of generality that both planners are *reflexive:* i.e., if a planner succeeds in connecting p_1 to p_2, it can also connect p_2 to p_1.

A word about the random sampling in Step 1 of the preprocessing: in the experimental work [5, 6, 7, 13] this is done simply by choosing a point at random from $[0,1]^d$. If the chosen point is in the free space, it is retained; else it is discarded and the process repeated. Clearly a point chosen at random in this fashion is uniformly distributed in the free space, but in order for the number of repetitions to be reasonably small we need the free space to constitute a good fraction of the configuration space. We assume this is the case based on empirical evidence (else no analysis is possible). Choosing a random sample has a minuscule cost in practice compared with the other operations, and can be repeated a very large number of times if necessary (see also Section 5).

Our main thesis is that the empirically observed success of the scheme stems from a property we call ϵ-goodness which we now define. Let $\mathcal{F}$ denote the free space. For a point $p \in \mathcal{F}$, let $S(p)$ consist of those points of $\mathcal{F}$ that can be connected to p by the simple planner B_S. For a subset $\mathcal{X}$ of the configuration space, let $\mu(\mathcal{X})$ denote its volume.

Definition 1.1 *Let ϵ be a positive real. We say that a point p in the free space $\mathcal{F}$ is ϵ-good if $\mu(S(p)) \geq \epsilon\mu(\mathcal{F})$. We say that the free space $\mathcal{F}$ is ϵ-good if for all points $p \in \mathcal{F}$ we have $\mu(S(p)) \geq \epsilon\mu(\mathcal{F})$.*

While any non-degenerate configuration space is ϵ-good for some positive ϵ, the intent in this definition is that the space be ϵ-good for a "reasonably large" value of ϵ. Many configuration spaces arising in practice do not have the ϵ-good property; for example, consider a crescent-shaped region or one where a circular obstacle is tangential to a rectangular obstacle. However, in these cases, our definition applies to the subset of free space obtained by removing a small neighborhood of the cusp or tangency points from the configuration space. (See Section 5.1 for a more rigorous treatment of this issue.)

Contributions and Organization

The first contribution of this paper is a model of computation appropriate for the analysis of the probabilistic roadmap scheme, taking into account the realities of the problem at hand. In Section 2 we define a concrete algorithm based on the high-level outline given above. This algorithm and its analysis do not make use of resampling (Step 3 above); we present this simplified version first because it succinctly outlines the main ideas using only the simple notion of ϵ-goodness. We argue in Section 3 that if the free space is ϵ-good then every point of the free space $\mathcal{F}$ can, with high probability, be connected to a milestone using only B_S. In Section 4 we give a bound on the number of invocations of the complex planner B_C in constructing the probabilistic roadmap; this involves a new randomized algorithm for determining connected components in a model related to the decision tree model used in the study of *evasive* graph properties [10], and may be of independent interest. We complement this with tight bounds for deterministic algorithms. These results imply bounds on the work done in preprocessing and in query processing, in terms of the running times of B_C and B_S; in particular, the complex planner is not used for answering queries. Section 5 summarizes results from experiments with the robot arm of Figure 1; these suggest that most but not all points in the corresponding free space are ϵ-good for a reasonably large value of ϵ. Interestingly, the resampling step seems to be helpful for settings such as this arm. We therefore extend (Section 5.1) the definition of ϵ-goodness and use it to explain these observations: assuming the configuration space satisfies a weaker condition we call (ϵ, t)-goodness for a small integer t, we give an explanation for the resampling step similar to the analysis in Sections 3 and 4. Finally, our work is related to classic problems in art-gallery theorems. In Section 6 we establish this connection, give some new results related to our work, and mention some resulting open problems in art-gallery theorems.

2 Algorithms and Results

For the remainder of the paper, we say that two points $p_1, p_2 \in \mathcal{F}$ are mutually *visible* when B_S can connect p_1 and p_2. We use this terminology primarily for brevity, and our usage is inspired by a commonly used simple planner [7, 6] that checks whether the straight line segment joining p_1 and p_2 is in $\mathcal{F}$ (equivalently, p_1 and p_2 are mutually visible in $\mathcal{F}$); however, our entire analysis works for any simple planner B_S.

Let $\beta \in (0, 1]$ be a positive real constant which represents the failure probability we can tolerate in the preprocessing (this will become clear in the statements of Theorems 2.1, 2.2 and 2.3). Let c be a fixed positive constant large enough that for any $x \in (0, 1]$, $(1 - x)^{(c/x \ln 1/x)} \leq x\beta/4$. Let $s = (c/\epsilon)(\ln 1/\epsilon)$. The algorithm for preprocessing is listed in Figure 2.

As we will see in Section 4, Step 4 probes the "edge-slots" of the roadmap, trying to determine the structure of the connected components without expending too many calls to B_C. Note that the algorithm in Figure 2 does not make use of resampling; we will get to this in Section 5. In practice Step 4 is a last resort; much if not all of the connectivity information should have been discovered before this step.

The query processing algorithm is listed in Figure 3. Given the query points q_1 and q_2, we connect them to milestones m_1 and m_2 using B_S as in Figure 3. Here $\gamma \in (0, 1]$ is the allowable failure probability for a query. For each i, Step 1a can be implemented using s invocations of B_S, one for each milestone. Each trial of Step 1b can be implemented using s invocations of B_S.

For an ϵ-good free space $\mathcal{F}$ call a set of milestones M *adequate* if the volume of the subset of $\mathcal{F}$ not visible from any milestone of M is at most $(\epsilon/2)\mu(\mathcal{F})$. Intuitively, if we were to place a point source of light at each milestone, we would like a fraction at least $1 - \epsilon/2$ of $\mathcal{F}$ to be illuminated. Note that as ϵ increases, the requirement for adequacy grows weaker but the number of milestones needed becomes smaller.

Theorem 2.1 *The preprocessing stage will generate an adequate set of milestones with probability at least* $1 - \beta$.

Theorem 2.1 only says that most of $\mathcal{F}$ is likely to be visible from some milestone in M; using this property alone, we can show that queries can be answered *quickly.* However, we need a stronger property — which we may think of as *permeation* — to guarantee that queries can be answered *correctly.* Permeation is essentially the following: for any two milestones in the same connected region of $\mathcal{F}$, we can infer this connectedness from the preprocessing algorithm. Theoretically, we cannot hope to show that the use of B_S alone will provide such permeation: if $\mathcal{F}$ consists of two spheres each of diameter $1/2$ and the spheres touch at a single point p, we have a free space that is ϵ-good for $\epsilon = 0.5$. Yet it is extremely unlikely that B_S can yield permeation in this case (if for instance

```
1. Pick $s$ points in $\mathcal{F}$ at random, and call these milestones.
2. Invoke $B_S$ on every pair of milestones.
3. Pick a representative milestone from each component that results. Let $V$ be the set of these
   representatives and $|V| = n$.
4. Invoke the **Randomized Permeation** algorithm (Figure 5) on these representatives.
```

Figure 2: *The Preprocessing Algorithm*

```
1. For $i = 1, 2$ do:
   (a) If $q_i$ can see a milestone $v$, set $m_i = v$.
   (b) Else Repeat $\log(2/\gamma)$ times:
       i. Choose $v_i$ uniformly at random from $S(q_i)$;
       ii. If a milestone is visible from $v_i$ then set $m_i$ to be that milestone.
   (c) If all $\log(2/\gamma)$ trials fail then declare FAILURE and halt.
2. If $m_1$ and $m_2$ are in the same component of $G$ then output YES else output NO.
```

Figure 3: *The Query Processing Algorithm*

B_S simply checks visibility between milestones). In such configuration spaces, the use of the complex planner B_C in Step 4 is inevitable to ensure a good overall success probability. Define a function $g()$ on an ordered k-tuple of positive integers $n_1, n_2, \ldots, n_k$ by $g(n_1, n_2, \ldots, n_k) = \sum_{i=1}^{k} i n_i$.

Theorem 2.2 *Let S be a set of n milestones lying in k connected components denoted $S_1, \ldots, S_k$ such that $|S_1| \geq |S_2| \geq \ldots \geq |S_k|$. The preprocessing stage will determine the partition correctly and the expected number of invocations of B_C is at most*

$$2g(|S_1|, |S_2|, \ldots, |S_k|).$$

Theorem 2.3 *Suppose that the set of milestones chosen during preprocessing is adequate. Then the probability that the query processing algorithm outputs* FAILURE *is at most γ. When the query processing algorithm does not output* FAILURE, *it correctly answers the query by either producing a path or declaring that none exists.*

In fact, our analysis will imply that the expected number of executions of Step 1b in the query processing algorithm (Figure 3) is at most 2.

3 Nearly Complete Coverage

This section establishes Theorems 2.1 and 2.3. The expectation of the volume of points not visible from any of the s randomly chosen milestones in M is

$$\mathbf{E}[\mu(\{p \in \mathcal{F} \mid p \notin \cup_{m \in M} S(m)\})] = \int_{p \in \mathcal{F}} \mathbf{Pr}[p \notin \cup_{m \in M} S(m)].$$

The probability that a fixed point is not visible from any of the s milestones is at most $(1-\epsilon)^s$. Thus, the above is bounded by

$$\begin{aligned} \int_{p \in \mathcal{F}} (1-\epsilon)^s &= \mu(\mathcal{F})(1-\epsilon)^s \\ &\leq \mu(\mathcal{F})\epsilon\beta/4, \end{aligned}$$

By the Markov inequality, it follows that

$$\mathbf{Pr}[\mu(\{p \in \mathcal{F} \mid p \notin \cup_{m \in M} S(m)\}) > \mu(\mathcal{F})\epsilon/2]$$

is at most $\beta/2$. Thus with probability $1 - \beta/2$ the "shadow region" not visible from any $m \in M$ has volume at most $\mu(\mathcal{F})\epsilon/2$, in which case it follows that for any $p \in \mathcal{F}$, the volume of the subset of $S(p)$ visible from some $m \in M$ is at least $\mu(S(p)) - \mu(\mathcal{F})\epsilon/2 \geq \mu(\mathcal{F})\epsilon/2$.

This establishes Theorem 2.1 and leads to Theorem 2.3: for either query point q_i, the probability that a random point chosen from $S(q_i)$ is not visible from any $m \in M$ is $(\epsilon/2)/S(q_i) < 1/2$. The probability that we fail on $\log(2/\gamma)$ trials is less than $\gamma/2$. Since we do this for the two query points, the overall failure probability is at most γ.

4 Permeation

This section establishes Theorem 2.2. En route, we connect our problem to the decision tree model used to study evasive graph properties, and prove some related results. The permeation problem is the following: given a free space $\mathcal{F}$ containing $n \leq (c/\epsilon)\ln 1/\epsilon$ milestones, determine which milestones are reachable from each other. (Note that because of Step 2 in the Preprocessing Algorithm of Section 2, n may be much smaller than $(c/\epsilon)\ln 1/\epsilon$.) Given any pair of milestones the complex planner B_C will decide whether they are connected. The graph G can be computed with $O(n^2)$ invocations of B_C by trying it on every pair of points, but we show that far fewer invocations may suffice.

We work with the following abstract version of the permeation problem. The input is a graph $G(V,E)$ with n vertices, consisting of k disjoint cliques. The goal is to determine this clique partition of G. The cost of an algorithm is measured by the number of entries it examines in the adjacency matrix of G. This is the *edge probe model* used in the study of *evasive graph properties* [10]. Let $N(n,K)$ denote the non-deterministic complexity of this problem.

Theorem 4.1 $N(n,k) = \Theta(n + k^2)$.

We now characterize the worst-case *deterministic* complexity of this problem, denoted $T(n,k)$. Consider the following deterministic algorithm: by probing all edge slots incident on an arbitrary vertex x, determine the neighborhood of x, say $\Gamma(x)$; let $C_x = \{x\} \cup \Gamma(x)$, and output C_x; then, recur on the vertex-induced subgraph $G[V \setminus C_x]$. The proof of correctness is obvious, and we sketch only the analysis of the running time. The number of levels in the recursion is k, since one of the k cliques is removed from G prior to each recursive call. The number of probes made in the process of determining each such clique is at most n. The total number of probes is $O(nk)$. In Figure 4, we illustrate the Deterministic Permeation Algorithm, which is an iterative version of the recursive algorithm. The iterative version will prove useful when describing a randomized algorithm. By the preceding discussion, we have:

Theorem 4.2 *The Deterministic Permeation Algorithm correctly solves the permeation problem using $O(nk)$ probes.*

The following lower bound establishes that the Deterministic Permeation Algorithm is optimal.

Theorem 4.3 *For $1 \leq k \leq n$, $T(n,k) = \Omega(nk)$.*

Proof Sketch: We sketch an adversary argument in terms of the complementary problem: given a graph $\overline{G}$ which is a complete k-partite graph for some k, determine the k-partition of the vertices of $\overline{G}$ into independent sets. The adversary responds to each probe for an edge by some deterministic algorithm, and its strategy is to say that edges are present, as far as possible.

The adversary maintains a graph H in which the edges are those edges of $\overline{G}$ which have been probed already *and* for which the response was that the edge is present. When the adversary is forced to concede that an edge (i,j) is absent in $\overline{G}$, it then *collapses* the two vertices i and j into a single meta-vertex whose neighborhood is the union of the neighbors of i and j. Collapsing two vertices is equivalent to conceding that they are in the same independent set of the k-partition; meta-vertices can also be collapsed into each other. The missing edges in H correspond to edge slots in G that have not been probed so far. The adversary maintains the following invariants at all times.

1. The chromatic number of H is k; in particular, it maintains a partition of the (meta)-vertices into k non-empty color classes $C_1, \ldots, C_k$ such that each color class is an independent set.

2. For each meta-vertex, every vertex therein has had at least $k-1$ incident edges already probed that were deemed to be present in $\overline{G}$.

Initially, the adversary arbitrarily partitions the vertices into k non-empty color classes; since H is empty then, this ensures the first invariant. The second invariant holds trivially since there are no meta-vertices at the beginning.

```
1. Mark all vertices in $V$ as being LIVE.
2. Initialize $x \leftarrow 1$.
3. **While** $x \leq n$ **do:**
   (a) $\Gamma(x) \leftarrow \emptyset$.
   (b) **For** $y = x + 1$ to $n$ **do:**
       i. **If** vertex $y$ is marked LIVE
          **then** probe the edge $(x, y)$ in $G$.
       ii. **If** edge $(x, y)$ is probed and found present
           **then** mark $y$ as DEAD and add $y$ to $\Gamma(x)$.
   (c) Output $\{x\} \cup \Gamma(x)$ as being a clique.
   (d) Mark $x$ as being DEAD.
   (e) Set $x$ to the smallest numbered LIVE vertex, or $n + 1$ if there are no LIVE vertices left.
```

Figure 4: *The Deterministic Permeation Algorithm*

Thereafter, the adversary responds as follows to a probe (i, j) by the algorithm. Note that a probe involving an edge (i, j), where i is contained in a meta-vertex i^*, will be treated as referring to the edge (i^*, j).

- If i and j belong to distinct color classes, it will say that the edge is present and will add this edge to the graph H.

- If i and j belong to the same class C_r, then it will check to see if there exists a color class C_t with $t \neq r$ such that at least one of i and j does not have neighbors in C_t. Suppose that i does not have any neighbors in C_t, then the adversary will transfer i from C_r to C_t and will then respond as in the previous case (i.e., say that the (i, j) edge is present).

- Finally, there is the case where both i and j belong to the same component C_r and each has at least one neighbor in every other color class. In this case, the adversary will concede that the edge (i, j) is indeed absent and will then collapse i and j together.

The first invariant holds since edges are only introduced between vertices in distinct color classes. The color classes remain non-empty since a vertex is transferred from a color class only when it has at least two vertices. To verify the second invariant, observe that when two vertices (i, j) are collapsed, both have at least one neighbor in the remaining $k - 1$ color classes.

The algorithm can terminate only when the number of (meta)-vertices in each color class is down to one, and there is an edge between each pair of color classes, since otherwise the algorithm cannot be certain of the k-partition of $\overline{G}$, or even whether there is a k-partition in the first place.

We claim that, upon termination, every one of the n vertices must have at least $k - 1$ edges incident on it which were probed and deemed to be present in $\overline{G}$. The second invariant implies that this is true for any vertex which participated in a collapse and is a part of some meta-vertex when the algorithm terminates. A vertex which did not participate in any collapse must also have at least $k - 1$ edges incident on it since it is the only vertex in its color class, and there is an edge from its color class to every other class. Thus, the total number of edges probed and deemed present in $\overline{G}$ is at least $n(k - 1)/2$. Also, there must be at least $n - k$ edges which were probed and deemed absent in $\overline{G}$, since in going from n vertices to k vertices at least $n - k$ collapses need to be performed and each collapse requires a distinct absent edge. Thus, the total number of probes must be $\Omega(nk)$. $\square$

We now give a randomized algorithm that beats the lower bound of Theorem 4.3 when the sizes of the k cliques differ significantly. This is crucial in our application to motion planning because in practice the free space $\mathcal{F}$ often consists of components of very different sizes. The Randomized Permeation Algorithm (see

Figure 5) labels the vertices in a random order and then invokes the Deterministic Permeation Algorithm.

Let $w_1 \geq w_2 \geq \cdots \geq w_k$ be the sizes of the cliques in an instance G arranged in a non-increasing order, where $n = \sum_{i=1}^{k} w_i$. Denote by C_i the ith largest clique in G.

Theorem 4.4 *The Randomized Permeation Algorithm correctly determines the clique structure and incurs an expected cost that is at most*

$$2g(w_1, w_2, \ldots, w_k) - n - k.$$

Furthermore, with high probability, the cost is at most

$$O(g(w_1, w_2, \ldots, w_k) \log n).$$

Remark: Observe that the worst case is when all w_i are equal to n/k, in which case the expected cost is $O(nk)$. On the other hand when there is one giant clique and $k-1$ cliques of size $O(1)$ the expected cost is $\Theta(n+k^2)$, which is essentially the non-deterministic lower bound.

Proof Sketch: The proof of correctness follows from that for the Deterministic Permeation algorithm. We first sketch the analysis of the expected cost.

We say that a clique C_i *beats* another clique C_j if *some* vertex of C_i occurs before *all* vertices of C_j in the random permutation chosen by the Random Permeation Algorithm. The probability that C_i beats C_j is the same as the probability that a uniformly random choice from $C_i \cup C_j$ yields a vertex of C_i, and, clearly, the latter is $w_i/(w_i + w_j)$.

We divide the edge slots of the graph into intra-clique and inter-clique edge slots. For each i, the number of intra-clique edge slots in C_i that are probed is precisely $w_i - 1$, since the only such probes are between the earliest vertex (according to the random permutation) of C_i and the remaining vertices of C_i. The total number of such probes is

$$\sum_{i=1}^{k}(w_i - 1) = n - k.$$

Fix some i and j, and suppose that C_i beats C_j. The inter-clique edge slots between these two cliques are between the earliest vertex of C_i and all vertices of C_j. This gives a total of w_j probes that are "charged" to C_j (the beaten clique). The expected total charge to clique C_j is given by

$$\sum_{i \neq j} \frac{w_i}{w_i + w_j} \times w_j.$$

To bound the expected total number of inter-clique edge slots that are probed, we sum the charges to the various cliques and obtain

$$\begin{aligned}
\sum_{j=1}^{k}\sum_{i \neq j} \frac{w_i w_j}{w_i + w_j} &= \sum_{j=1}^{k}\sum_{i<j} \frac{2w_i w_j}{w_i + w_j} \\
&\leq \sum_{j=1}^{k}\sum_{i<j} 2w_j \\
&= 2\sum_{j=1}^{k}(j-1)w_j \\
&= 2g(w_1, \ldots, w_k) - 2n.
\end{aligned}$$

Adding together the bounds on the expected number of intra- and inter-clique edge-slots that are probed, we obtain the desired bound.

We now turn to the task of proving the high probability bound. Fix a clique C_j and note that the total charge to C_j is the size of C_j multiplied by the number of other cliques that beat it. Since there are $j-1$ cliques that are larger than C_j, at most $j-1$ of the cliques that beat C_j are larger than C_j. Let X_j be the random variable denoting the number of cliques *smaller* than C_j that beat C_j; let Y_j be the random variable denoting the total number of vertices from cliques smaller than C_j that are earlier than *all* vertices in C_j; and, finally, let Z_j be a random variable having the geometric distribution with parameter $p_j = w_j / \sum_{i=j}^{k} w_i$ and expectation $1/p_j$. Clearly, $X_j \leq Y_j$, and Y_j is stochastically dominated by Z_j. The probability that Z_j is larger than $2p_j^{-1} \ln n$ is bounded by

$$(1 - p_j)^{2p_j^{-1} \ln n} \leq e^{-2 \ln n} = \frac{1}{n^2}.$$

Thus with probability at least $1 - 1/n$, we have, for each j, $X_j \leq 2p_j^{-1} \ln n$. This implies that, with high probability, the total number of inter-clique edges probed is given by

$$\sum_{j=1}^{k}(j - 1 + X_j)w_j \quad \leq \quad \sum_{j=1}^{k}(j - 1 + 2p_j^{-1} \ln n)w_j$$

1. Permute the vertices randomly so that each is labeled by an integer in $\{1, \ldots, n\}$.
2. Invoke the Deterministic Permeation Algorithm.

Figure 5: *The Randomized Permeation Algorithm*

$$
\begin{aligned}
&\leq \sum_{j=1}^{k}((j-1)w_j + 2\ln n \sum_{i=j}^{k} w_i) \\
&= \sum_{j=1}^{k} jw_j - n + 2\ln n \sum_{j=1}^{k} jw_j \\
&= (1 + 2\ln n)g(w_1, \ldots, w_k) - n.
\end{aligned}
$$

Adding in the number of intra-clique edge slots that are probed, we obtain the desired result. $\square$

5 Experiments and Extensions

The robot arm of Figure 1 was tested for ϵ-goodness using 9000 random samples; on a DEC Alpha workstation, it took 9.24 seconds to create the random configurations, and 1399 seconds[*] to try connecting all pairs using B_S. (These figures underscore that random sampling is not a significant component of the cost.) The samples with the "most" visibility could see about 0.06 (i.e., 6%) of the remaining samples, suggesting that they are 0.06-good. As many as 3.3% of the random samples could see no other random samples, and fully 22% could see 0.001 (i.e., 0.1%) or less; in other words, only about 78% of the configuration space is 0.001-good or better. (For $\epsilon = 0.001$, we have $(1/\epsilon)\ln 1/\epsilon = 6908$, which is of the same order as our number of samples.) We conjecture that the resampling step (Step 3 from our high-level outline of Section 1) leads to better coverage of the space in situations such as Figure 1. We have observed that it helps eliminate the need for the Complex Permeation step of the outline of Section 1 in some examples. To address this we introduce a generalization of the notion of ϵ-goodness.

[*]In implementations used in practice, several additional techniques offer substantial savings over the timings reported here. For instance, we dynamically update a representation of the connected components after testing each pair of configurations. Thus we would not test a new pair if they are known to belong to the same connected component.

5.1 The Extended Definition

Let us say that a point p in free space is $(\epsilon, 1)$-good if $\mu(S(p)) \geq \epsilon\mu(\mathcal{F})$, corresponding to our original definition of ϵ-goodness for a point. Next, we say a point p in free space is (ϵ, t)-good if $\mu(\{q \in S(p) \mid q \text{ is } (\epsilon, t-1)\text{-good}\}) \geq \mu(S(p))/2$. For $t > 1$, we say that $\mathcal{F}$ is (ϵ, t)-good if $\mu(\{p \in \mathcal{F} | p \text{ is } (\epsilon, 1)\text{-good}\}) \geq \mu(\mathcal{F})/2$ and every point of $\mathcal{F}$ is (ϵ, i)-good for $i \leq t$. If $\mathcal{F}$ is (ϵ, t)-good for a small value of t, we can give a theoretical basis for the resampling step (Step 3 in the outline of Section 1). The main idea is that single links discovered by B_S in the algorithms of Section 2 are now simulated using t-link paths found by resampling and connecting using B_S. This leads to a generalized definition of an adequate set of milestones, and eventually to a version of Theorem 2.3 in which the number of invocations of B_S is larger by a factor of 2^t. This extension requires that we can still sample the visibility region of a query point. In practice, this is often accomplished by defining an appropriate "neighborhood" for any point p, from which a sample likely to be in $S(p)$ can be chosen. We are currently designing experiments to check the (ϵ, t)-goodness of practical examples; the experiment design is non-trivial since the parameter 2 in the above definition (while sufficient for theorems) is somewhat arbitrary, and affects the value of t observed.

6 Related Combinatorial Results

A number of combinatorial problems concerning art-gallery theorems [11] are related to our work. For instance, given a simple polygon that is ϵ-good we ask: how many guards are necessary and sufficient to cover the entire polygon? (Another way of thinking of this is to imagine point sources of light being placed in the polygon with the objective of illuminating the entire interior.) The following would be an ideal result: given an ϵ-good configuration space S, a random sample of $poly(1/\epsilon)$ points from the free space $\mathcal{F}$ will "illuminate" the entire free space with high probability. In practice it may be reasonable to assume that the number of "holes" in the free space ω is "small" (for instance, bounded by a slowly growing function of the input size).

Conjecture 6.1 *A random sample of $poly(\omega + 1/\epsilon)$ points is likely to cover an ϵ-good free space with ω holes.*

At present we only have the most rudimentary results of this type; for instance, we give an upper bound on ϵ so that one guard suffices to cover an ϵ-good simply-connected region. In fact, a Helly-type theorem due to Krasnosselsky [9] immediately yields:

Theorem 6.1 *Let R be a compact, simply-connected ϵ-good region in Euclidean d-space for $\epsilon > d/(d+1)$. Then there is a point p in R such that $S(p) = R$.*

Broder, Dyer, Frieze, Raghavan and Upfal [2] have initiated progress in extending the above result: they show that if $\epsilon = 1/3 + \delta$ for a simply-connected region in the plane, the number of guards is polynomial in $1/\delta$. Various interesting questions remain. For instance, even the existential version of Conjecture 6.1 would be useful: given an ϵ-good space $\mathcal{F}$ with ω holes, there exists a set of $poly(\omega + 1/\epsilon)$ points which covers $\mathcal{F}$.

Acknowledgement

We thank Don Knuth for helpful discussions.

References

[1] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotics Research*, 10:628–649, 1991.

[2] A.Z. Broder, M.E. Dyer, A.M. Frieze, P. Raghavan and E. Upfal. Private communication, 1995.

[3] J.F. Canny. *The Complexity of Robot Motion Planning*. MIT Press, 1988.

[4] Th. Horsch, F. Schwarz, and H. Tolle. Motion planning for many degrees of freedom — random reflections at c-space obstacles. In *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2138–2145, San Diego, CA, 1994.

[5] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 2138–2145, San Diego, CA, 1994.

[6] L. Kavraki. Random Networks in Configuration Space for Fast Path Planning. PhD Thesis, Stanford University, 1995. Technical Report STAN-CS-TR-95-1535.

[7] L. Kavraki, P. Švestka, J.-C. Latombe, M. Overmars. *Probabilistic roadmaps for path planning in high dimensional configuration spaces*. STAN-CS-TR-94-1519, Stanford University, Stanford, CA, 1994.

[8] J.-C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[9] S.R. Lay. *Convex Sets and their Applications*. John Wiley, 1982.

[10] L. Lovász and N. Young. *Lecture notes on evasiveness of graph properties.* Tech. Rep. CS-TR-317-91, Computer Science Dept., Princeton University, 1991.

[11] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[12] M. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Proc. of Workshop on Algorithmic Foundations of Robotics*, 1994.

[13] M. Overmars. *A Random Approach to Motion Planning*. Tech. Rep. RUU-CS-92-32, Dept. Comput. Sci., Utrecht Univ., 1992.