Randomized Query Processing in Robot Motion Planning

L. Kavraki* J-C. Latombe† R. Motwani‡ P. Raghavan§

November 30, 1994

Abstract

The subject of this paper is the analysis of a randomized preprocessing scheme that has been used for query processing in robot motion planning. The attractiveness of the scheme stems from its general applicability to virtually any motion-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 a preprocessing step following which queries can be answered quickly. En route, we pose and give solutions to related problems on graph connectivity in the *evasiveness* model, and art-gallery theorems.

*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.

†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 IBM Faculty Development Award, an OTL grant, and NSF Young Investigator Award CCR-9357849, with matching funds from IBM, Schlumberger Foundation, Shell Foundation, and Xerox Corporation.

§IBM T.J. Watson Research Center, Yorktown Heights, NY 10598.

1 Introduction

Planning obstacle-avoiding motion for a robot or a robot arm from a given initial configuration to a goal configuration is an important problem in robotics [Can88, Lat91]. Typically, the environment is static and the goal configuration is dynamic as the robot performs a series of complicated maneuvers. A number of recent papers in the robotics literature [KL94a, KL94b, KSLO94, OŠ94, Ove92, HST94] have described the striking success of a class of randomized preprocessing heuristics for query processing in robot motion 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 result is a general approach that can easily be tailored to any particular motion planning problem. The method 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 exact algorithms for every robot in existence. Figure 1 (page ii) depicts several positions of a robot with 7 degrees of freedom to which the method has been successfully applied. This paper initiates a theoretical basis for explaining the success of the method.

The configuration of a robot at any instant is characterized 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 space as a set of forbidden regions that may 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 joints. We refer to the subset of the configuration space that is *not* forbidden as the *free space*; in general, it may consist of more than one connected component.

This notion of a configuration space is central to the randomized preprocessing method we are about to describe. We assume 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.

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 [KSLO94, Ove92] 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 [BL91]. 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 a 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 sufficient resampling. 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* [KSLO94].

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 [Lat91, KL94a, KSLO94]: 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 [KL94a, KL94b, KSLO94] 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. (For readability in this abstract, we defer the issues of Lebesgue measurability and other topological considerations to the final version.)

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.

1.1 Contributions and Organization

The first contribution of this paper is a model of computation appropriate for the analysis of such schemes, 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 in this abstract 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 [LY91], 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 (page ii); 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 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 do this primarily for brevity, and our usage is inspired by a commonly used simple planner [KSLO94, Ove92] 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 (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 $s = (c/\epsilon)(\ln 1/\epsilon)$. We first describe the algorithm for preprocessing.

The Preprocessing Algorithm:

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.
4. Invoke the **Randomized Permeation** algorithm (page 8) on these representatives.

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 above algorithm 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.

Next, we describe the processing of a query. Given the query points q_1 and q_2, we connect them to milestones m_1 and m_2 using B_S as follows.

The Query Processing 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.

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.

Call a set of milestones M *good* if the volume of the subset of $\mathcal{F}$ not visible from any milestones in M is at most $(\epsilon/2)\mu(F)$. 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.

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

Note that Theorem 2.1 only says that most of $\mathcal{F}$ is likely to be visible from some milestone in M. In fact, 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 B_S simply checks visibility between milestones). In such unusual configuration spaces, the use of the complex planner B_C in Step 4 is inevitable to ensure a good overall success probability.

Theorem 2.2 *Given a set S of s milestones lying in k connected components denoted $S_1, \ldots, S_k$, with high probability the preprocessing stage will determine the partition correctly. The expected number of invocations of B_C is*

$$O(|S_1| + 2|S_2| + \cdots + k|S_k|).$$

With high probability the number of invocations of B_C is within $O(\log n)$ of its expectation.

Theorem 2.3 *Suppose that the set of milestones chosen during preprocessing is good. 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.*

The next two sections are devoted to proving Theorems 2.1, 2.2, and 2.3.

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

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

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

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

By the Markov inequality, we have

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

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 Theorems 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 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 page 5, 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 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* [LY91]. Let $N(n, K)$ denote the non-deterministic complexity of this problem.

Theorem 4.1 *For $1 \leq k \leq n$, $N(n,k) = \Theta(n + k^2)$.*

We now characterize the worst-case *deterministic* complexity of this problem, denoted $T(n,k)$. Consider the following algorithm: by probing all edge slots incident on an arbitrary vertex i determine the neighborhood of i, say $\Gamma(i)$; let $C_1 = \{i\} \cup \Gamma(i)$, and output C_1. Now, recur on the vertex-induced subgraph $G[V \setminus C_1]$. This algorithm probes $O(nk)$ edge slots in the worst case.

Theorem 4.2 *For $1 \leq k \leq n$, $T(n,k) = O(nk)$.*

The following lower bound establishes that the above algorithm is optimal. The proof uses a non-trivial adversary argument and can be found in the Appendix.

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

We now give a randomized algorithm that beats the lower bound of Theorem 4.3, especially when the sizes of the k cliques differ significantly. This is crucial in our application to motion planning because in practice the free space $\mathcal{F}$ usually consists of one large component and a few small components.

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. We will establish the following theorem.

Theorem 4.4 *There is a Las Vegas algorithm which correctly determines the clique structure whose expected cost is at most*

$$2(w_1 + 2w_2 + \cdots + kw_k).$$

Furthermore, with high probability, the number of probes will be

$$O((w_1 + 2w_2 + \cdots + kw_k)\log n).$$

Observe that the worst case is when all w_i are equal to n/k, in which case the expected cost is $\Theta(nk)$. 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.

The randomized algorithm is derived from the deterministic algorithm described earlier.

The Randomized Permeation Algorithm:

1. Mark all vertices in V as being LIVE.
2. Permute the vertices randomly so each is labeled by an integer in $\{1,\ldots,n\}$.
3. Initialize $x \leftarrow 1$.
4. **While** $x < n$ **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 if there are no LIVE vertices left.

We omit the proof of correctness of this algorithm, and give a brief sketch of the analysis of the expected running time. Also, the high probability bound is deferred to the final version of the paper. For each edge slot, we will determine the probability that it is probed during an execution of this randomized algorithm, and the sum of these probabilities over all edge slots will give the desired bound on the expected running time.

We first consider the edge slots whose end-points lie in distinct cliques of G. Suppose that an edge slot (x, y) has one end-point in a clique C_i and another in a clique C_j such that $i < j$. We claim that the edge slot (x, y) is probed if and only if either x or y is assigned the smallest (random) label from among all the vertices in $C_i \cup C_j$. The probability of this event is exactly $2/(w_i + w_j)$. Summing over all $w_i w_j$ edge slots between these two cliques and over all choices of two distinct cliques, the expected number of probes to slots not containing edges of G is at most

$$\sum_{j=1}^{k}\sum_{i<j}\frac{2w_i w_j}{w_i + w_j} \leq 2\sum_{j=1}^{k}\sum_{i<j} w_j \leq 2\sum_{j=1}^{k}(j-1)w_j.$$

Consider now those edge slots (x, y) with both end-points in the same clique C_j. We claim that the edge slot (x, y) is probed if and only if either x or y is assigned the smallest (random) label from among all the vertices in the clique C_j. The probability of this event is exactly $2/w_j$. Summing over all $w_j(w_j - 1)/2$ edge slots in the clique, and over all cliques C_j, the expected number of probes to slots containing edges of G is at most

$$\sum_{j=1}^{k}\frac{w_j(w_j-1)}{2} \times \frac{2}{w_j} \leq \sum_{j=1}^{k} w_j.$$

Adding the two expressions together, we obtain the desired bound on the expected number of edge slots probed by this algorithm.

5 Experiments and the Extended Definition

The robot arm of Figure 1 (page ii) was tested for ϵ-goodness using 9000 random samples; 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). Perhaps as a consequence, in practice the *resampling* step (Step 3) from our high-level outline of Section 1 (page 2) helps in situations such as Figure 1. To address these observations, we introduce a generalization of the notion of ϵ-goodness and use it to explain the success of the resampling step in situations such as the robot arm. 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 $\mu(\{q \in S(p) \mid q \text{ is } (\epsilon, t-1)\text{-good}\}) \geq \mu(S(p))/2$. We say that a configuration space is (ϵ, t)-good if every point in it is (ϵ, i)-good for $i \leq t$. If a configuration space 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 algorithm 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 a good 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. The precise algorithm and the analysis that result will be given in the final version. 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 [O'Rou87] 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? 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" or cover the entire free space with high probability. In practice it may be reasonable to assume that number of obstacle components 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 obstacle components.*

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 regions in the plane. The proof, which is deferred to the final version, is based on a Helly-type theorem from topology due to Molnár [Mol57]: for any collection of compact, simply-connected sets in the Euclidean plane, if the pairwise intersection of the sets is compact and simply-connected, and the triplewise intersection of the sets is non-empty, then the common intersection of the sets in the collection is non-empty.

Theorem 6.1 *Let R be a compact, simply-connected region in the plane that is $2/3$-good. Then there is a point $p \in R$ such that $S(p) = R$.*

We can extend this to higher dimensions using Helly's theorem [Hel30]: for any collection of compact, convex sets in Euclidean d-space, if the intersection of every $d+1$ sets is non-empty, then the common intersection of the sets in the collection is non-empty.

Theorem 6.2 *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 $S(p) = R$.*

Various interesting (but hard) open questions remain. For instance, even the existential version of Conjecture 6.1 would be useful: given an ϵ-good space $\mathcal{F}$ with ω obstacle components, there must exist set of $poly(\omega) + 1/poly(\epsilon)$ points which covers $\mathcal{F}$.

Acknowledgements

We thank Don Knuth for helpful discussions.

References

[BL91] J. Barraquand and J.-C. Latombe. Robot motion planning: A distributed representation approach. *Int. J. Robotics Research*, 10:628–649, 1991.

[Bol85] B. Bollobás. *Random Graphs.* Academic Press, 1985.

[Can88] J.F. Canny. *The Complexity of Robot Motion Planning.* MIT Press, 1988.

[ER59] P. Erdös and A. Rényi. On random graphs. *Publ. Math. Debrecen*, 6:290–297, 1959.

[Hel30] E. Helly. Über Syteme abgeschlossenen Mengen mit gemeinschaftlichen Punkten. *Monatsh. Math.*, 37:281–302, 1930.

[HST94] 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.

[KL94a] 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.

[KL94b] L. Kavraki and J.-C. Latombe. Randomized preprocessing of configuration space for path planning: Articulated robots. In *Proc. IEEE/RSJ/GI Int. Conf. Intelligent Robots and Systems*, München, Germany, 1994.

[KSLO94] 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.

[Lat91] J.-C. Latombe. *Robot Motion Planning.* Kluwer Academic Publishers, 1991.

[LY91] L. Lovász and N. Young. Lecture notes on evasiveness of graph properties. Tech. Rep. CS-TR-317-91, Computer Science Dept., Princeton University, 1991.

[Mol57] J. Molnár. Über den zweidimensionalen topologischen Satz von Helly. *Mat. Lapok*, 8:108–114, 1957. [Hungarian with German and Russian summaries]

[O'Rou87] J. O'Rourke. *Art Gallery Theorems and Algorithms.* Oxford University Press, 1987.

[OŠ94] M. Overmars and P. Švestka. A probabilistic learning approach to motion planning. In *Proc. of Workshop on Algorithmic Foundations of Robotics*, 1994.

[Ove92] M. Overmars. *A Random Approach to Motion Planning.* Tech. Rep. RUU-CS-92-32, Dept. Comput. Sci., Utrecht Univ., 1992.

[ŠO94] P. Švestka and M. Overmars. *Motion Planning for Car-Like Robots, Using a Probabilistic Learning Approach.* Tech. Rep. RUU-CS-94-33, Dept. Comput. Sci., Utrecht Univ., 1994.

[Šve93] P. Švestka. *A Probabilistic Approach to Motion Planning for Car-Like Robots.* Tech. Rep. RUU-CS-93-18, Dept. Comput. Sci., Utrecht Univ., 1993.

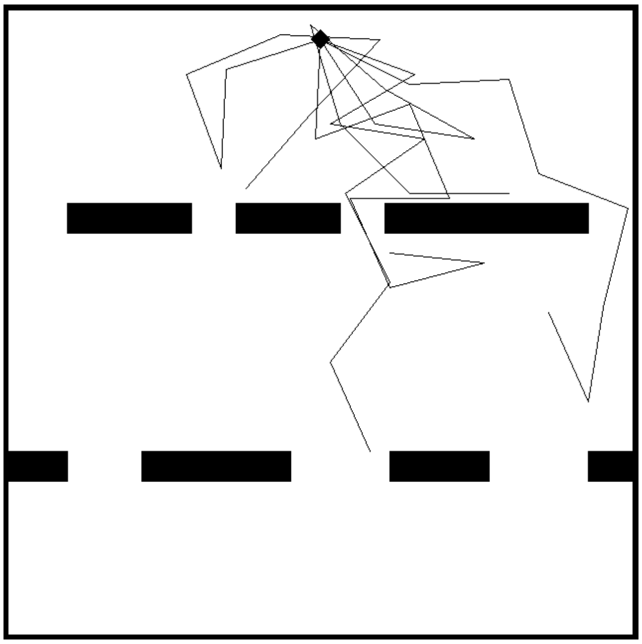

Figure 1: *Several configurations of a robot with 7 degrees of freedom. This robot arm has 7 articulated joints, and must maneuver through gaps in two walls.*

Appendix

Proof of Theorem 4.3

It will be convenient to present this lower bound 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. This problem is exactly equivalent to the problem of determining a partition into k cliques of the complement graph G.

We use an adversary argument to derive this lower bound. 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 chooses a value k initially, and ensures that the graph it constructs (adaptively) is a complete K-partite graph for some $K \in \{k-1, k, k+1\}$. The algorithm can be provided this information without affecting the following argument.

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 the two nodes together is equivalent to conceding that i and j are in the same independent set of the k-partition. In general, meta-vertices can be repeatedly collapsed into each other to obtain meta-vertices containing a large number of "real" vertices. Finally, note that the missing edges in H correspond to edge slots in G which have not been probed so far.

Any probe involving an edge (i, j), where i is contained in a meta-vertex i^* obtained by some earlier collapses, will be treated as referring to the edge (i^*, j) since all vertices in i^* have exactly the same set of neighbors. The adversary can reveal this graph H together with the meta-vertex structure to the algorithm without affecting the lower bound argument, and so we can assume that the algorithm never makes redundant queries such as probing for an edge between two vertices which belong to the same meta-vertex.

Initially, the graph H has n vertices but no edges or meta-vertices. At all times, the adversary ensures the following invariants.

1. The graph H is k-colorable; 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. Note that by the definition of H, none of the edges between the (meta-)vertices in a color class have been probed yet, and all edge that were probed and deemed present are between two distinct color classes.
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 $\overline{G}$.

It is clear that the state of knowledge of the algorithm is exactly captured by the structure of the graph H. Initially, the adversary arbitrarily partitions the vertices into k non-empty color classes and thereby ensures that the invariants hold at the beginning.

The adversary strategy for responding to the probes made by an algorithm must preserve the invariants. At each stage, given a probe for an edge (i, j) by the algorithm, the adversary will respond as follows.

- 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 color 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.

It is easy to verify that first invariant holds since edges are always introduced between vertices in two distinct color classes, and the color classes are always non-empty since a vertex is collapsed or transferred from a color class only when it has at least two vertices in it. To verify the second invariant, first observe that when two vertices (i, j) are collapsed, both have at least one neighbor in the remaining $k-1$ color classes. Thus, any time a non-meta-vertex is collapsed (for the first time), it must have at least $k-1$ incident edges in H. Of course, either one or both of i and j may be meta-vertices, but that does not affect the invariant.

We now claim that as long as there are at least $k+1$ (meta-)vertices in H, the algorithm cannot be certain of the k-partition of $\overline{G}$, or even whether there is a k-partition in the first place. This is because with $k+1$ vertices, some color class C_r must have at least two distinct (meta-)vertices, say i and j. The adversary can choose to make the edge (i, j) absent from $\overline{G}$, and thereby ensure that the current k-partition into color classes corresponds to the correct k-partition of $\overline{G}$. On the other hand, it could choose to decide that the vertex j is adjacent to all other vertices (in particular, to i), and thereby obtain a complete $(k+1)$-partite graph. Thus, the algorithm cannot terminate at any stage where the number of vertices in H exceeds k.

Furthermore, we claim that upon termination at least one edge is present in H between each pair of color classes. Otherwise, the adversary could collapse a pair of color classes and obtain a $(k-1)$-partite graph.

We can now determine a lower bound on the total number of probes. When the algorithm terminates, there are k (meta-)vertices in k non-empty color classes, i.e., one in each color class. We claim that 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 $\Omega(nk)$.