Customizing PRM Roadmaps at Query Time*

Guang Song
Dept. of Computer Science
gsong@cs.tamu.edu

Shawna Miller
Dept. of Computer Science
slm5563@cs.tamu.edu

Nancy M. Amato
Dept. of Computer Science
amato@cs.tamu.edu

Technical Report TR00-029
Department of Computer Science
Texas A&M University
November 13, 2000

Abstract

In this paper, we propose a new approach for *building and querying* probabilistic roadmaps. In the roadmap construction stage, we build coarse roadmaps by performing only an approximate validation of the roadmap nodes and/or edges. In the query stage, the roadmap is validated and refined only in the area of interest for the query, and moreover, is *customized* in accordance with any specified query preferences. This approach, which postpones some of the validation checks (e.g., collision checks) to the query phase, yields more efficient solutions to many problems. An important benefit of our approach is that it gives one the ability to *customize* the same roadmap in accordance with multiple, variable, query preferences. For example, our approach enables one to find a path which maintains a particular clearance, or makes at most some specified number of sharp turns. Our preliminary results on problems drawn from diverse application domains show that this new approach dramatically improves performance, and shows remarkable flexibility when adapting to different query requirements.

*This research supported in part by NSF CAREER Award CCR-9624315 (with REU Supplement), NSF Grants IIS-9619850 (with REU Supplement), ACI-9872126, EIA-9975018, EIA-9805823, and EIA-9810937, and by the Texas Higher Education Coordinating Board under grant ARP-036327-017.

1 Introduction

Automatic motion planning deals with finding a feasible sequence of motions to take some moving object (the 'robot') from a given initial configuration to some specified goal configuration. While complete motion planning algorithms do exist, they are rarely used in practice since they are computationally infeasible in all but the simplest cases [16]. For this reason, attention has focused on probabilistic methods, which sacrifice completeness in favor of computational feasibility and applicability.

In particular, a class of algorithms known as as *probabilistic roadmap (PRM)* methods have been the subject of much recent work. The idea behind these methods is to create a graph of randomly generated collision-free configurations which are connected by a simple and fast local planning method. Actual global planning (queries) is carried out on this graph. The initial PRMs were shown to be very successful in solving difficult problems in high-dimensional configuration spaces (C-space) that had previously resisted efficient solution [15], and moreover, were simple and easy to implement, requiring only collision detection as a primitive operation. These successes sparked a flurry of activity in which PRM motion planning techniques were applied to a number of challenging problems arising in a variety of fields including robotics (e.g., closed-chain systems [10, 18]), CAD (e.g., maintainability [3, 7], deformable objects [2, 14]), and even computational Biology and Chemistry (e.g., ligand docking [22], protein folding [23]). Indeed, it can be argued that the PRM framework was instrumental in this broadening of the range of applicability of motion planning, as many of these problems had never before been considered candidates for automatic methods.

At the same time, these new applications served to point out some weaknesses in the PRM approach. In particular, it was observed that there were some classes of problems for which PRMs did not yield efficient solutions. For example, problems where the solution path must traverse *narrow passages* in C-space [12]. As a result, a number of PRM variants specifically targeted at this problem have been proposed, e.g., [1, 6, 12]. Also, some novel approaches for improving PRM efficiency have shown promising results [5, 20] (these methods do not address the narrow passage problem). Unfortunately, however, PRM solutions to many problems still take prohibitively long.

Another shortcoming of PRMs is that while they are very good at finding *a* path, they do not support applications which might impose particular, variable, requirements on the sought after path, e.g., maintaining a particular clearance from obstacles or minimizing the robot's rotation as it moves. This issue has not received as much attention as, e.g., the narrow passage problem, because simply finding any path was considered a necessary first step in many cases. However, this issue must be addressed before PRMs can be used in many practical instances.

In this work, we propose a new approach for *building and querying* probabilistic roadmaps that addresses both the above shortcomings of current PRMs.

- In the roadmap construction stage, we build coarse roadmaps by performing only approximate validation of roadmap nodes and edges.
- In the query stage, the roadmap is validated and refined only in the area of interest for the query, and moreover, is *customized* in accordance with any specified query preferences.

This approach, which postpones some of the validation checks (e.g., collision checks) to the query phase, yields more efficient solutions to many problems. For example, consider a roadmap with n nodes, each of which is connected to its k nearest neighbors, who are at an average distance of d from it. Constructing this roadmap in its entirety would require $O(nkd)$ validity checks – the majority of which are not needed for any particular solution path. Our strategy of postponing validation leads to faster roadmap construction, while incurring only slightly higher query times. As discussed below, similar strategies have been used by other researchers [5, 20].

An important benefit of our approach is that it gives one the ability to *customize* the roadmap in accordance with any specified, variable, query requirements. For example, it might be specified that the solution path should maintain a certain clearance from the obstacles, that the robot's rotation should be minimized, or that the path should contain at most some fixed number of bends. And moreover, these requirements might change for different queries in the same roadmap, or might actually be preferences (as opposed to requirements) with different priorities. For example, one might want a path with a particular clearance and also desire, if possible, a minimal amount of rotation. As another example, suppose that in one query a path is desired with a certain clearance from the obstacles, but in another query the path needs to respect a different clearance threshold. Current PRMs do not support such customizable use of the *same* roadmap.

2 Related Work

In this section we briefly mention other approaches targeted at either improving the efficiency of PRMs or in supporting requirements on the query path other than the usual collision-free requirement. First, there has some recent work aimed at improving the efficiency of PRMs by postponing most of the validation to the query stage [5, 20].

Like PRM, Lazy PRM [5] randomly generates nodes in the configuration space, and edges in the roadmap represent paths between roadmap nodes. Unlike PRM, all nodes and edges are initially assumed to be collision-free. This dramatically reduces the amount of time spent in the preprocessing phase. When presented with a query, a shortest path is found between the start and goal configurations. Then, the nodes (first) and edges (second) along the path are checked for collision. If a node or edge is found to be in collision, it is removed from the roadmap, and a new shortest path is extracted from the roadmap. This process repeats until a collision-free path is found or a path no longer exists between the start and goal.

A similar approach called Fuzzy PRM was independently proposed in [20]. Like Lazy PRM, this method does not check the validity of the roadmap edges in the preprocessing phase, but it does ensure that the roadmap nodes are collision-free. Instead of initially assuming the edges to be collision-free, as Lazy PRM does, Fuzzy PRM assigns each edge a weight that corresponds to the probability that the edge represents a feasible path (1 for collision-free). During the query phase, the algorithm selects the path with the highest probability. Each edge along the path is inserted into a priority queue. Then, the edge with the lowest probability is checked at a slightly higher resolution, assigned a new probability, and reinserted in the priority queue. This process repeats until all edges on the path have probabilities of 1 or a collision is found.

We are aware of no PRM work which supports variable query preferences. However, there are a few probabilistic methods that enable one to specify, but not subsequently alter, query requirements [17, 4]. These methods are a bit different from PRMs in that they grow components rooted at the start (and possibly the goal) configuration. Since they incrementally construct paths, query requirements can be implemented as the roadmap is built. However, they do not provide support for later querying the same roadmap with different requirements.

3 Building and Querying Customizable PRM Roadmaps

In this section, we describe our new algorithm for the fast construction of PRM roadmaps that support multiple, and variable, query preferences. The general structure of our Customizable PRM (or C-PRM) follows the traditional PRM paradigm. In the standard PRM approach [15], a graph (the roadmap) is constructed, usually during preprocessing, where vertices (nodes) correspond to randomly generated valid configurations (typically verified as collision-free) and where edges correspond to feasible paths between nearby roadmap nodes found by some simple local planning method (often, the straightline in C-space). Queries are processed by connecting the start and goal configurations to the same connected component of the roadmap, and then searching for a path (usually the shortest) connecting them.

The main differences are that in C-PRM, only a coarse, approximate roadmap is built during preprocessing, and detailed validation of the path, including customization for any particular query preferences, is performed during the query itself. That is, we postpone much of the time consuming validation until the query phase, and then only perform detailed validation on the portions of the roadmap deemed necessary for solving the particular query. In addition to yielding big performance gains, this strategy enables us to support additional, and variable, requirements or preferences on the properties of the solution path obtained. We note that some query preferences cannot be encoded in a roadmap, i.e., they must be enforced at query time. For example, if the same roadmap will be used to answer queries with different clearance thresholds, or if validity can only be tested after the start and goal are known (e.g., paths with at most k sharp turns). Our strategy supports such customizable use of the *same* coarse roadmap to meet different query requirements or preferences.

We now describe in more detail how we construct coarse roadmaps during preprocessing (Section 3.1) and how these roadmaps are used to provide solution paths adhering to requirements or preferences specified during the query phase (Section 3.2).

3.1 Roadmap construction using fast, approximate evaluation

The fundamental primitive operation in the construction (and querying) of PRM roadmaps is the evaluation of a configuration's feasibility (frequently, this evaluation is collision detection). This is true for both the node generation phase, where invalid nodes are rejected, and in the connection phase, where connections between nearby nodes are validated by verifying a sequence of intermediate nodes between them.

One of the strengths of the PRM method is that the global sampling (and validation) yields roadmaps that encode useful information about the connectivity of the free configuration space. Unfortunately, however, as previously noted, complete, exact evaluation of configurations in the roadmap construction phase leads to infeasible construction times for many important applications. Our strategy is to replace the exact validation, of nodes and/or edges, with faster, approximate methods. The goal is to perform 'good enough' evaluations so that the resulting roadmap provides some global view of the connectivity of the planning space, while postponing as much of the time consuming detailed validation until it is really needed.

While our strategy is similar in motivation to the recently proposed Lazy PRM [5] and Fuzzy PRM [20] methods, it is different in that we advocate approximate validation methods for the roadmap nodes *and* edges in the roadmap construction phase. In contrast, Lazy PRM performs no validation of either nodes or edges during roadmap construction, and Fuzzy PRM performs exact evaluation of nodes but no validation of edges (initial edge probabilities are set based on inter-node distances). We believe our approximate edge evaluation will produce roadmaps which better reflect the planning space, which is very important for difficult problems, such as those containing *narrow passages* in C-space [12]. In addition, as our approximate evaluation methods can be tuned to be more or less accurate (and consequently more or less time consuming), our methodology allows the user to determine the degree of exactness appropriate for the given application.

Below, we describe some methods for approximate node and approximate edge evaluation.

3.1.1 Approximate node evaluation

Using faster, coarser, collision detection. Suppose that paths with variable clearances will be desired when planning in a particular environment. Note that it may not be known *a priori* what the maximum possible clearance is, or what clearance requirements will be imposed in the query.

One approach is to determine the clearance of all nodes and edges during the roadmap construction phase, and then to use a standard graph search to query the roadmap for the solution path with maximal clearance. This requires a collision detection method which computes clearance information, such as the C-Space Toolkit (CSTk) [24].

Another strategy is to build a roadmap which does not contain clearance information. Such a roadmap would be faster to construct and could employ any collision detection method suitable for the environment (thus allowing the use of perhaps faster packages such as RAPID [9]). Then, at the query stage, the roadmap could be refined using more expensive clearance computations. This can be done by (iteratively) extracting a shortest path, and then checking it for compliance with the given clearance requirement – first the nodes, and then the edges. Performance is improved because clearance computations for many edges will not be performed, and in particular, we will only check edges that connect nodes with acceptable clearances. An intermediate approach might compute clearance information for the nodes, but not the edges, during roadmap construction.

Grid-based approximation. Another method is to compute an approximate value for each cell in a grid-based decomposition of C-space. During roadmap construction, the value for each configuration is determined by a simple, fast table lookup. Of course, due to the high-dimensionality of most interesting C-spaces, this method is infeasible in many cases. Nevertheless, there exist some applications where useful approximations can be computed on a coarse grid. For example, as discussed in Section 4, we have employed this technique for ligand binding, which arises in drug design [22].

3.1.2 Approximate edge evaluation

As with approximate node evaluation, our goal here is to perform some fast, approximate validation of the edges. The motivation is that many invalid edges can be easily discovered, resulting in roadmaps that better reflect the connectivity of the free C-space.

Binary resolution approximation. This technique is very general and can be applied in any scenario. The strategy is to validate the intermediate configurations on an edge (a straight-line in C-space) according to a binary partitioning strategy. That is, we first validate the midpoint, then the midpoints of the two resulting subsegments, etc. Our experience shows this strategy discovers many invalid edges quickly. The resolution at which to terminate the process during roadmap construction can be some computed value related to the environment characteristics, or can be set by the user. A similar idea of increasing resolution checks is used in Fuzzy PRM [20] when attempting to increase edge probabilities during the query phase.

Overlapping spheres approximation. Another way to estimate whether or not an edge is collision-free is use a test based on the C-space clearance of its two endpoints. The C-space clearance c, if known, of a node defines a sphere centered at that node of radius c which is entirely contained in the free C-space. If the clearance spheres of the two endpoints of an edge intersect, then there exists a collision-free path connecting them. (A path could still exist even if the spheres do not intersect.) Since we cannot compute clearances in C-space, we compute an approximation as follows. We select k directions at random, and move away from the node in those directions until a collision occurs; the minimum distance until collision is used as the C-space clearance approximation. Clearly, the accuracy of the approximation is very sensitive to the number k. In practice, the overlapping sphere approximation seemed to work well for values as small as $k = 3$.

Running time for C-PRM vs. PRM						
	# Rdmap Edges		Rdmp Const. Time (s)		Query Time (s)	
Application	C-PRM	PRM	C-PRM	PRM	C-PRM	PRM
Moving-Piano	2133	2019	**24**	**104**	1.73	1.38
CAD: alpha1.5	508	441	**57**	**478**	3.77	3.07
Ligand Binding	705	958	**206**	**9523**	41.16	38.54
Protein Folding	127296	96919	**22280**	**48490**	1626	684

Table 1: Running times of C-PRM and a traditional PRM for various applications.

3.2 Query Phase

In C-PRM queries are processed by adding the start and goal to the roadmap, and connecting them to their k closest neighbors (as was done with the other roadmap nodes, but with a larger k). If the start and goal are in the same connected component of the roadmap, the shortest path connecting them is retrieved using Dijkstra's algorithm. Since the path consists of edges and nodes that have only been approximately validated, it must be validated now. For example, finer resolution collision checking must be performed on all path segments. In addition, any desired query specifications will be enforced at this stage. If any node or edge on the path fails to meet the query requirements, it is removed from the roadmap, a new shortest path is found in the refined roadmap, and the process iterates with this new path. The process terminates when either a valid path is found, or when the start and goal are no longer in the same connected component.

3.2.1 Additional, multiple query preferences

Usually, there are other desirable properties for a path in addition to the basic requirement that it be collision-free. Some common preferences include large clearance, small rotation, low curvature (smoothness), few sharp corners, avoiding singularities for manipulators, or low potential energies for ligand binding and protein folding (see Table 2).

Common Query Perferences	
Application	**Preference**
CAD	clearance
Mobile robots	clearance, smoothness
Manipulators	singularity, clearance
Ligand binding	low potential
Protein folding	low potential

Table 2: Some common query preferences for different applications.

Note that it is not practical to store all information needed to verify such properties in the roadmap. Moreover, in some cases it is difficult, or even impossible, to associate such information with roadmap nodes or edges because it relates to global properties of the solution path. For example, one might ask for a path for a mobile robot that makes at most k sharp turns. Since this is not a local property, and since the start and goal are not known in advance, one cannot prune the roadmap so that it contains only valid solution paths. In this example, and indeed in many interesting queries, the only feasible way to enforce such preferences is during the query process.

In many cases, one would actually like to request a path meeting several query preferences but with different priorities. For example, one might ask for a path that: (1) is collision free, (2) restricts certain dofs to a particular range (e.g., to control tilt), (3) has some particular clearance, and (4) requires at most some specified amount of rotation. The numbers indicate priorities.

C-PRM can support such queries by iteratively refining, or customizing, the roadmap. First, a collision-free path is found. Next, the path is checked to see if all dofs are within the specified limits. Then, clearances on the path are checked, and finally the rotation along the path is checked. At each stage, invalid nodes and edges are removed as they are discovered, and the process iterates. Finally, a path that meets the preferences to the degree possible is found, and in the process, the roadmap is customized with respect to these requirements.

4 Examples and Results for Various Applications

In this section, we consider examples drawn from a wide variety of application domains, including traditional motion planning applications, intelligent CAD, and computational Biology and Chemistry. We demonstrate how C-PRM can be applied to support different query preferences. We also show that in all cases C-PRM shows significant performance gains over a traditional PRM.

A summary of our results on a variety of applications is shown in Table 1. From these results it can be seen that C-PRM is always faster than the PRM method, but that the magnitude of the improvement varies for different types of applications. All experiments were performed on a Pentium III 550 MHz PC using our C++ OBPRM library, which includes implementations of many PRM variants.

In the following, we describe in more detail how C-PRM has been applied to each problem.

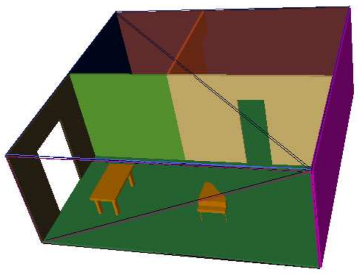

Figure 1: The piano mover's problem.

4.1 Robotics Applications

In this section we describe how C-PRM can be applied to motion planning problems arising in robotics.

The Piano Mover's Problem. Figure 1 shows a house containing a piano and a table. The goal is to move the piano out of the house. For this problem, all roadmap nodes were checked for collision, but the edges were only approximately verified using the binary resolution approximation scheme, where we checked the edges down to a resolution of 10 times the resolution required for exact validation. The results in Table 1 show that the C-PRM roadmap was constructed approximately five times faster than the PRM roadmap, while the cost paid in terms of a more expensive query was negligible (a fraction of a second). One can also see by comparing the number of edges in the C-PRM and PRM roadmaps that the binary approximation does fairly well in pruning invalid edges.

We also use this problem to illustrate the ability of C-PRM to support multiple, prioritized query preferences (see Table 3). The most basic query requirement is certainly finding a collision-free path. Secondary preferences might be small tilt (keep it upright), little rotation, and large clearances (to give more 'maneuvering' room). As previously discussed, it is either impractical or impossible to encode all information needed for these preferences in the roadmap itself. So instead, we compute a coarse roadmap, containing collision-free nodes (without clearances) and edges that are only partially validated by the binary approximation scheme. The preferences are enforced during the query, which considers the preferences one by one and iteratively refines the roadmap by removing nodes and edges invalid for the specified preferences. The roadmap construction used the fast RAPID [9] collision detector, while the query stage used CSTk [24] if clearance information was needed.

Iteratively Refined Roadmap				
	time	**Roadmap Statistics**		
Query	**(sec)**	**#N**	**#E**	**#CCs**
makemap	468.00	2618	33914	3(2087:531:1)
C-free	14.64	2620	34081	3(2089:531:1)
tilt $< 90°$	67.36	2072	21562	4(1665:206:200:1)
rot $< 15°/s$	51.47	2072	21546	4(1665:206:200:1)
cl > 0.1	25.00	1059	7231	3(885:97:77)
cl > 0.2	23.00	767	4391	4(647:67:52:1)

Table 3: The evolution of the roadmap for the piano mover's problem based on prioritized query preferences. The clearance requirement units are environment units. The rotation requirement units are degrees per step.

The results of this process are summarized in Table 3. The first query, which enforces the collision-free requirement, results in the addition of the start and the goal to the roadmap, along with some edges. The tilting and clearance requirements are properties that can be tested locally for each node and edge, and as such, they result in the loss of both nodes and edges from the roadmap. For the tilting and clearance preferences, we first go through the roadmap and remove all nodes that do not satisfy the current requirement. This turned out to be useful since many of our nodes did not meet the requirements, and so pruning them initially greatly reduced the number of iterations required to find a feasible path. We note that the pass checking all nodes would not be as efficient if the check was more expensive (such as in the computational biology applications) or if the roadmap were larger. The rotation requirement does not result in the loss of any nodes, but does remove some edges from the roadmap – this is because it is a property related to the edges, not nodes.

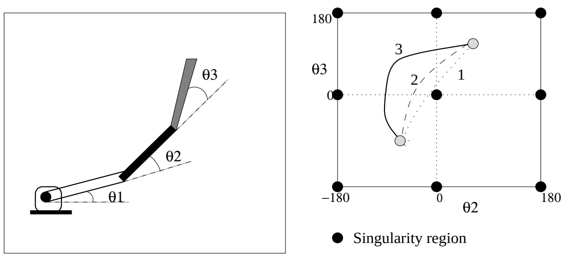

Figure 2: A 3-R manipulator

Manipulators. Another practical application of our method involves multi-linked manipulators. For manipulator motion planning, it it important to find collision-free paths which avoid singularities. Consider the example in Figure 2, which shows an RRR planar manipulator, where the singularity regions are shown in the plot on the right. It is important to avoid singular configurations when planning because in those configurations the manipulator loses one or more degrees of freedom [8]. That is, at such a configuration, there exists a direction along which it is impossible to move the end effector of the manipulator, regardless of the joint velocities. Even at configurations close to a singularity, joint velocities required to maintain a certain end effector velocity can be very large and impractical. In the figure, path 1 is the shortest path from the start to the goal; it is undesirable because it passes very close to a singularity region. If the user were to add an additional query requirement specifying some minimum value on the absolute value of the determinant of the Jacobian, then path 2 or path 3 might be returned, depending on the particular minimum value specified.

4.2 Intelligent CAD

An important design criteria for complex mechanical systems is that certain parts should be easy to service and/or replace. Motion planning algorithms for testing such conditions automatically on CAD models would be a great aid to engineers [3, 7].

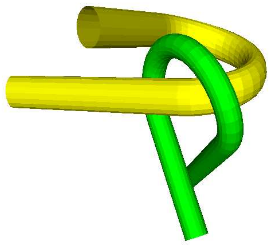

Figure 3: Alpha Puzzle (1.5 scaled version)

The CAD model we consider is the alpha puzzle, which consists of two identical twisted, intertwined tubes (one the obstacle and one the robot, 1008 triangles each); the objective of the puzzle is to separate the tubes (see Figure 3). An easier version of problem was obtained by scaling the obstacle tube by a factor of 1.5 in one dimension, which widened the gap between the prongs of the alpha shape.

For this problem, collision-free configurations were generated and edges were partially validated by the binary approximation scheme (again by checking at ten times the resolution required for exact validation). The results in Table 1 show that C-PRM solves this problem nearly 10 times faster than the PRM. The savings are due to the fact that C-PRM never completely validates many of the edges.

4.3 Computational Biology/Chemistry

We have also applied C-PRM to several problems arising in computational Biology and Chemistry such as ligand binding and protein folding. For these applications, different criteria are used to judge the validity of nodes and edges. In particular, instead of geometric collision detection tests, we need to determine if the *potential energy* of the configurations (called conformations) is low enough. Similarly, the validity of an edge is determined by computing a weight which is dependent on the potential values along the edge [22, 23]:

$$Weight = \sum_{i=0}^{N-1} -log(P_i),$$

where,

$$P_i = \begin{cases} e^{\frac{-\Delta E}{kT}} & \text{if } \Delta E > 0 \\ 1 & \text{if } \Delta E \leq 0 \end{cases}$$

Note that some intermediate nodes along the edge need to be checked to estimate the path weight. Thus, approximation schemes, such as the overlapping spheres, which depend only on the endpoints cannot be utilized for such applications.

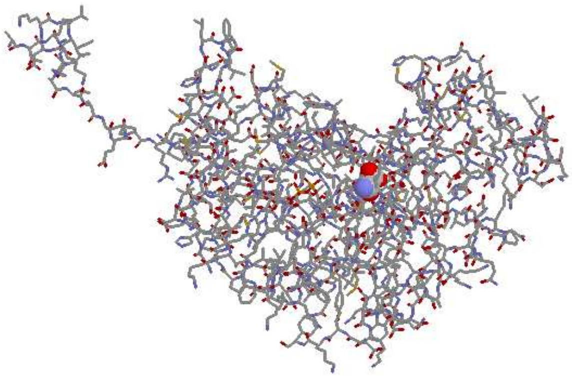

Figure 4: Ligand Binding

Ligand binding, also referred to as molecule docking, is an important problem in the design of new drugs [22]. We applied C-PRM to the example shown in Figure 4. The protein (M4 Lactate Dehydrogenase Ternary Complex, pdb file: 1LDM) is shown in wire-frame and the binding ligand, shown in space-fill, is an oxamate molecule. The roadmap was constructed using a grid-based potential approximation for the nodes (as in [22]) and 'exact' validation for the edges, but again using the grid-based potential approximation for the internal nodes on the edge. During the query, exact potential computations were used. The results in Table 1 show C-PRM is nearly 50 times faster than the traditional PRM method. Moreover, even though C-PRM has postponed all exact potential computations to the query phase, the query times for the two approaches are still quite comparable. This example shows the grid-based approximation for node validation is a very powerful technique for this application.

We have also applied C-PRM to compute folding pathways for several small proteins (see [23] for more details). One of the most important outstanding problems in computational biology is protein folding, i.e., folding a one-dimensional amino acid chain into a three-dimensional protein structure [13, 19]. Our focus is on computing folding pathways to the known native fold from some initial state. Many researchers have remarked that knowledge of the folding pathways might provide insights into and a deeper understanding of the nature of protein folding [11, 21].

The roadmap for protein A (Figure 5) was constructed using exact potential computations for the nodes and the binary approximation scheme for the edges which used a resolution six times that required for exact evaluation. Here, C-PRM is roughly twice as fast as the PRM approach. Less improvement is obtained here because C-PRM mainly optimizes the roadmap connection phase, and this application spends more of its time in the node generation phase than the other applications studied do.

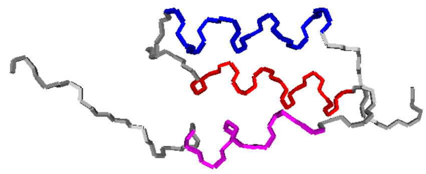

Figure 5: The native fold of Protein A.

5 Conclusion

In this paper, we describe a new approach for *building and querying* probabilistic roadmaps. This approach, which postpones some of the validation checks (e.g., collision checks) to the query phase, was seen to yield more efficient solutions to a wide variety of problems, ranging from traditional robotics applications to ligand docking and protein folding problems in computational Biology and Chemistry. An important benefit of our approach is that it gives one the ability to *customize* the roadmap in accordance with multiple, variable, query preferences. This feature addresses an aspect that had been absent in previous PRMs, and potentially makes the PRM framework more attractive for use in many practical instances.

Acknowledgements

We would like to thank the robotics group at Texas A&M. The alpha puzzle was designed by Boris Yamrom of the Computer Graphics & Systems Group at GE's Corporate Research & Development Center.

References

[1] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 155–168, 1998.

[2] E. Anshelevich, S. Owens, F. Lamiraux, and L. Kavraki. Deformable volumes in path planning applications. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2000.

[3] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 529–536, 2000.

[4] P. Bessiere, J. M. Ahuactzin, E.-G. Talbi, and E. Mazer. The ariadne's clew algorithm: Global planning with local methods. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, volume 2, pages 1373–1380, 1993.

[5] R. Bohlin and L. E. Kavraki. Path planning using lazy prm. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 521–528, 2000.

[6] V. Boor, M. H. Overmars, and A. F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1018–1023, 1999.

[7] H. Chang and T. Y. Li. Assembly maintainability study with motion planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1012–1019, 1995.

[8] John J. Craig. *Introduction to Robotics: Mechanics and Control, 2nd Edition.* Addison-Wesley Publishing Company, Reading, MA, 1989.

[9] S. Gottschalk, M.C. Lin, and D. Manocha. Obb-tree: A hierarchical structure for rapid interference detection. Technical Report TR96-013, University of N. Carolina, Chapel Hill, CA, 1996.

[10] L. Han and N. M. Amato. A kinematics-based probabilistic roadmap method for closed chain systems. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[11] B. Honig. Protein folding: From the levinthal paradox to structure prediction. *J. Mol. Bio.*, 293:283–293, 1999.

[12] D. Hsu, L. Kavraki, J-C. Latombe, R. Motwani, and S. Sorkin. On finding narrow passages with probabilistic roadmap planners. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[13] G. N. Reeke Jr. Protein folding: Computational approaches to an exponential-time problem. *Ann. Rev. Comput. Sci.*, 3:59–84, 1988.

[14] L. Kavraki, F. Lamiraux, and C. Holleman. Towards planning for elastic objects. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998.

[15] L. E. Kavraki, P. Švestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Trans. Robot. Autom.*, 12:566–580, 1996.

[16] J. C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA, 1991.

[17] S. M. LaValle and J. J. Kuffner. Randomized kinodynamic planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–479, 1999.

[18] S.M. LaValle, J.H. Yakey, and L.E. Kavraki. A probabilistic roadmap approach for systems with closed kinematic chains. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999.

[19] M. Levitt, M. Gerstein, E. Huang, S. Subbiah, and J. Tsai. Protein folding: the endgame. *Annu. Rev. Biochem.*, 66:549–579, 1997.

[20] C. L. Nielsen and L. E. Kavraki. A two level fuzzy prm for manipulation planning. Technical Report TR2000-365, Computer Science, Rice University, Houston, TX, 2000.

[21] E.I. Shakhnovich. Theoretical studies of protein-folding thermodynamics and kinetics. *Curr. Op. Str. Bio.*, 7:29–40, 1997.

[22] A.P. Singh, J.C. Latombe, and D.L. Brutlag. A motion planning approach to flexible ligand binding. In *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, pages 252–261, 1999.

[23] G. Song and N. M. Amato. A motion planning approach to folding: From paper craft to protein folding. Technical Report TR00-017, Department of Computer Science, Texas A&M University, July 2000.

[24] P. G. Xavier and R. A. LaFarge. A configuration space toolkit for automated spatial reasoning: Technical results and ldrd project final report. Technical Report SAND97-0366, Sandia National Laboratories, 1997.