From Days to Seconds: A Scalable Parallel Algorithm for Motion Planning[*]

Sam Ade Jacobs and Nancy M. Amato
Parasol Lab., Dept. of Computer Science and Engineering, Texas A&M Univ
College Station, Texas, 77843-3112, USA
(sjacobs,amato)@cse.tamu.edu

ABSTRACT

This work describes a scalable method for parallelizing sampling-based motion planning algorithms. It subdivides configuration space (C-space) into (possibly overlapping) regions and independently, in parallel, uses standard (sequential) sampling-based planners to construct roadmaps in each region. Next, in parallel, regional roadmaps in adjacent regions are connected to form a global roadmap. By subdividing the space and restricting the locality of connection attempts, we reduce the work and inter-processor communication associated with nearest neighbor calculation, a critical bottleneck for scalability in existing parallel motion planning methods.

We show that our method is general enough to handle a variety of planning schemes, including the widely used Probabilistic Roadmap (PRM) and Rapidly-exploring Random Trees (RRT) algorithms. We compare our approach to two other existing parallel algorithms and demonstrate that our approach achieves better and more scalable performance. Our approach achieves almost linear scalability on a 2400 core LINUX cluster and on a 153,216 core Cray XE6 petascale machine.

Categories and Subject Descriptors

I.6.8 [**Computing Methodologies**]: SIMULATION AND MODELING—*Types of Simulation, Parallel*

General Terms

Algorithms, Performance

Keywords

Robotics, Motion Planning, High Performance Computing

[*]This research supported in part by NSF Grants EIA-0103742, ACR-0081510, ACR-0113971, CCR-0113974, ACI-0326350, CRI-0551685, CCF-0833199, CCF-0830753, by the DOE, Chevron, IBM, Intel, HP, and by King Abdullah University of Science and Technology (KAUST) Award KUS-C1-016-04.

1. PROBLEM AND MOTIVATION

The need for solving large problems within an acceptable time frame has been at the center of demand for parallel computing. Numerous areas of computing and various applications now require such solutions. One such area is motion or path planning. Motion planning, which is traditionally a robotics problem, now finds applications in other areas of scientific computing; from protein folding and drug design[21, 5, 22] to virtual prototyping and computer-aided design [16, 3, 9, 4]. These new application areas test the limit and capability of existing sequential motion planners[20].

Sampling-based motion planners are one such existing sequential planner and are considered state-of-the-art for solving motion planning problems. Sampling-based motion planning algorithms have been highly successful at solving previously unsolved problems [10], and much research has focused on developing more sophisticated variants of them. Sampling-based approaches are efficient and can be applied to high dimensional problems. While not guaranteed to find a solution, they are probabilistically complete, meaning that the probability of finding a solution given one exists increases with the number of samples generated [15].

Despite their advantages, the efficiency of sampling-based motion planning algorithms degrades as the ratio of obstacle space to free space increases – common in high dimensional problems. Thus, substantial resources in time and hardware are still required to solve computationally intensive applications. For example, the authors in [24] reported that it took several hours on a desktop PC to compute a roadmap modeling the folding motion of a small protein using coarse approximations for energy calculations. This time increases to several weeks if more accurate energy calculations are used or if a larger protein were studied.

One solution to this resource problem may lie in parallel computing. For many application areas, parallel processing offers the advantage of not only reducing computation time, but also improving the solution quality and enabling larger problems to be solved than were feasible before.

As parallelism become ubiquitous and research advances in motion planning, one untapped potential is in making effective use of parallel processing. It remains a potential area of research to fully explore the effective use of parallelism in in solving high-dimensional motion planning problems and its application in other areas of computational sciences.

In this work, we present a strategy for parallelizing sampling-based motion planning algorithms. Our strategy uses C-space subdivision to achieve scalability. First, the planning space is separated into (possibly overlapping) regions, at least one per processor. Then, each processor independently applies a sampling-based planner in its region and produces a regional roadmap. Finally, adjacent regional roadmaps are connected to form a global roadmap. By subdividing the space, we reduce the amount of work and inter-processor communication required for nearest neighbor calculations, which has been seen to be a critical bottleneck for scalability in previous methods.

2. BACKGROUND AND RELATED WORK

2.1 Background

Motion planning is the problem of finding a valid path (e.g., collision-free) for a movable object from a specified start configuration to a goal configuration in an environment with obstacles[10]. As simple as the motion planning problem may sound, its solution has been shown to be computationally intensive. A complete solution to the motion planning problem is known to be PSPACE-hard with an upper bound that is exponential in the number of degrees of freedom of the movable object [10]. In other words, for any complete planner to guarantee that a solution to a motion planning problem exists or not, exponential time in the number of degree of freedom is required. However, there are efficient heuristics and approximate algorithms that trade completeness for efficiency. One such algorithm is the sampling-based motion planning approach.

One well established sampling-based motion planning approach is the Probabilistic Roadmap (PRM) [15]. PRMs first construct a graph $G = (V, E)$, called a roadmap, to capture the connectivity of C-space. A node in the graph represents a valid (e.g., collision-free) placement of the robot (movable object), and an edge is added between two nodes if a simple path can be defined and validated by a local planner.

In the original method, nodes are generated using uniform random sampling and connections are attempted between a node and its k-nearest neighbors as computed using some distance metric (e.g., Euclidean). Once the roadmap is constructed, query processing is done by connecting the start and goal configurations to the roadmap and extracting a path through the roadmap between them. Many variants of PRMs have been proposed that bias node generation or connection in various ways [10].

Tree-based methods, such as the Rapidly-exploring Random Tree (RRT) [18] and Hsu's expansive planner [14], are other popular approaches to sampling-based motion planning. RRTs attempt to grow a tree from the start configuration by expanding outward into the unexplored areas of C-space. They first generate a uniform random sample and identify the nearest node in the tree to that sample. They then attempt to make a step from the nearest node toward the generated sample and add the step to the tree if successful. When solving a query, RRTs continue until the goal configuration can be connected to the tree in a similar fashion. RRT-connect [17] is a variant of the original RRT algorithm that grows two trees, one from the start configuration and one from the goal configuration, until the two trees can be connected.

2.2 Related Work

For years, researchers have proposed and studied different types of parallel algorithms for motion planning. For a detailed survey of early work in general parallel motion planning, please see [12]. Recently, research efforts have focused on parallel sampling-based motion planning due to the success of sequential sampling-based motion planning in solving high dimensional problems [13]. We present a brief review of most recent related work in parallel sampling-based motion planning.

The sequential PRM algorithm was first parallelized in [2] and then specifically for protein folding applications in [24]. Both papers present a similar parallel approach. Each processor generates an "equal" number of nodes in the entire C-space in parallel and adds them to the roadmap. Then, each processor attempts to connect its nodes with their k-nearest neighbors in the entire roadmap. The major drawback of this approach was the all-to-all computation and communication involved in the $O(n^2)$ method used to find nearest neighbors which did not scale to large systems or problem sizes.

A basic parallelization of RRTs is given in [8] where the problem is replicated on each processor. Each processor then constructs its own RRT and concurrently explores the entire C-space along with all the other processors. The first processor to find a solution sends a termination message to other processors. More recent research on parallelization of RRTs is presented in [11, 6]. While the work in [11] extends the initial idea of RRT parallelization, the work in [6] explores GPU implementation of parallel RRT and RRT* algorithms with a primary focus on parallelizing the collision detection phase.

Parallel Sampling-based Roadmap of Trees (pSRT) [1, 19, 20] combines the multiple query sampling characteristics of PRMs with the efficient local planning capabilities of single query of RRTs. Unlike previous approaches, the nodes of a pSRT roadmap are trees instead of individual configurations. The collections of these trees form the roadmap. Connections between trees are attempted between closest pairs of configurations between the two trees. The authors adopted the scheduler (master) – processor (slave) architecture. Each slave processor computes a predefined number of trees in the entire C-space. The master is responsible for arbitration of tree ownership, nearest neighbor computations, and determination of which pairs of trees to attempt for connection. Edge validation is distributed to the slave processes.

In most existing work surveyed, we identify inter-processor communication, redundant computation, and load imbalance (in master-slave architecture) as the key bottlenecks to scalable performance.

To address some of these drawbacks, we propose a C-space subdivision approach.

3. STRATEGY FOR PARALLELIZING SAMPLING BASED MOTION PLANNING

3.1 Overview of Approach

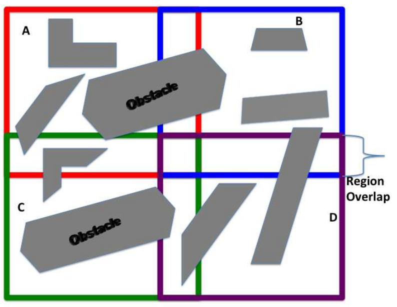
An overview of our approach is given in Algorithm 1. Initially, the environment describing the obstacles and robot is subdivided into regions. A simple illustration of a 2D environment subdivided into four regions is shown in Figure 1(a). The subdivision is represented by a *region graph*, whose vertices represent regions and whose edges encode the adjacency information between regions. Figure 1(b) shows the region graph corresponding to the subdivision shown in Figure 1(a). The circles and straight lines connecting adjacent circles represent vertices and edges of the region graph, respectively.

In Step 2, roadmaps are constructed independently, in parallel, in each region. Hence, this approach does not depend on the underlying sampling-based motion planning algorithm or strategy and can handle a variety of planning schemes.

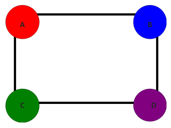
In Step 3, we connect nearby regional roadmaps to form a roadmap representing the entire C-space. The region graph

Algorithm 1 Parallel Sampling-based Motion Planning

```
Input: An environment $E$, A set of motion planners $S$, number of regions $N_R$
Output: A roadmap graph $G$
1: Decompose $E$ into $N$ regions
2: Make a region graph $R = (V_R, E_R)$ with $V_R$ and $E_R$ representing each region and adjacency information between regions, respectively
3: Independently and in parallel, construct roadmaps in each region using any desired planner $s \epsilon S$
4: Connect regional roadmaps in adjacent regions to form a roadmap $G$ for the entire problem
```



(a) A 2D environment subdivided into 4 regions with user-defined overlap between regions



(b) Region Graph

Figure 1: Space subdivision

is the enabling infrastructure facilitating the process of connecting the region roadmaps. The region graph infrastructure aids identification of adjacent regions between which connections are attempted. In this way, communication is only limited to adjacent regions.

3.2 Novelty and Uniqueness of Approach

The uniqueness of our approach can be summarized as follow:

- The first reported work in parallel sampling-based motion planning based on C-space subdivision.
- Experiments that show we achieve better and more scalable performance on thousands of processors than previous parallel sampling-based planners.
- An approach that is compatible with any sampling-based algorithm, including the Probabilistic Roadmap (PRM) and Rapidly-exploring Random Trees (RRT) algorithms.

In the following, we describe our method in more detail.

3.3 Space Subdivision and Region Graph Construction

We subdivide a given environment by breaking up the planning space into a set of regions. While the approach is general, in this work, we consider the x, y, and z dimensions of the original workspace only. Each processor is then assigned at least one region and the task of building a regional roadmap in its assigned region(s). In this initial work, we maintain some user-defined overlap between regions to allow sampling in the portions of the space that are at the boundaries that may facilitate connection between regional roadmaps.

We made use of two different graphs as underlying data structures: the roadmap graph and the region graph.

The roadmap graph stores the nodes and edges representing the configurations sampled and the connections between the samples, respectively.

The region graph, shown in Figure 1(b)represents the regions and the adjacencies between regions with its vertices and edges, respectively. This information is used to limit communication to adjacent regions. In addition, the region graph also maintains additional information that keeps track of the connected components in each region which is used when connecting adjacent regions.

3.4 Constructing Regional Roadmaps

Step 2 of Algorithm 1 involves construction of regional roadmaps. At this step, any of the existing sampling-based motion planning algorithms, such as PRM (and its variants) or RRT (and its variants) can be used. This step is independent of the sampling strategy employed. At this step, each processor independently generates and connects samples in its assigned region with no or minimal communication with other regions. The roadmaps built at this step are added to the roadmap graph.

To facilitate and streamline the connection at the next step, we keep track of the size and a vertex representative for each connected component in the regional roadmap.

Algorithm 2 Region Roadmap Connection

```
Input: A region graph $R$, connection method, $k$ number of candidates, local planner $lp$
Output: A roadmap graph $G$.
for all edges $E \epsilon R$ par do
  if (connection method == closest) then
    sourceCC = select $k$ center of mass based closest CC to target region from $E.source$
    targetCC = select $k$ center of mass based closest CC to source region from $E.target$
  end if
  if (connection method == largest) then
    sourceCC = select $k$ largest CCs from $E.source$
    targetCC = select $k$ largest CCs from $E.target$
  end if
  for all pairs($sourceCC, targetCC$) do
    if lp.IsConnectable($sourceCC, targetCC$) then
      Add the edge($sourceCC, targetCC$) to $G$.
    end if
  end for
end for
Return $G$.
```

3.5 Connecting Regional Roadmaps

The final step in constructing the full roadmap is to connect the regional roadmaps. Prior to this step, we track the sizes and number of connected components in each region. The regional graph stores this information which is input to the region connection algorithm shown in Algorithm 2. Other inputs to the algorithm include: k, the number of connections to be attempted between adjacent regions, the type of connection method, and a local planner used to verify connections.

For every edge identifying neighboring regions in the region graph, we attempt a connection between candidate node(s) of connected components in the source region to candidate node(s) of connected components in the target region.

Even though our implementation is independent of which region connection method is used, in this work, we attempt to connect regions based on the size of connected components in each region and the distance between connected components across regions. For the size-based connection, we attempt connections between a user-defined k largest connected components from the source region and k largest connected components from the target region. For the distance-based connection, we attempt to connect the k-closest connected components between the regions based on the distance between them. This distance is computed between the centers of mass (a measure of average of all configurations in the connected component) of the two connected components.

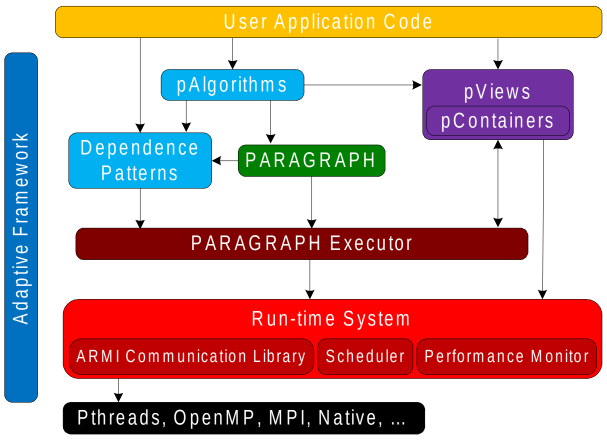


Figure 2: STAPL software architecture.

3.6 Implementation using STAPL

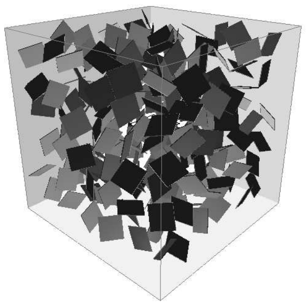
Our code was written in C++ using the Standard Template Library (STL) and the Standard Template Adaptive Parallel Library (STAPL) as supporting libraries. STAPL [7] is a platform independent superset of STL being developed in our lab. It provides a collection of building blocks for writing parallel programs. These building blocks (as shown in Figure 2) include a collection of parallel algorithms (*pAlgorithms*), parallel and distributed containers (*pContainers*), a general mechanism to access the data of the pContainer, similar to STL *iterators* called *pView*, an abstraction of task graph of computation called *PARAGRAPH* and an Adaptive Runtime System that includes a communication library, scheduler and perfomance monitor. For detailed information on the STAPL project, please see [7, 23].

In this work, we made use of the STAPL pGraph, one of the STAPL pContainers, as the parallel data structure for representing both the region graph and the roadmap graph. Our proposed method was implemented as a STAPL pAlgorithm.

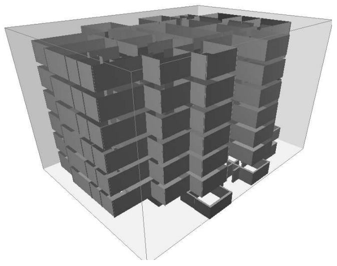
4. RESULTS AND CONTRIBUTIONS

4.1 Experimental Setup

4.1.1 Algorithms



(a) Clutter



(b) Building

Figure 3: Environments studied

We implemented four different algorithms. The first two were based on our proposed approach but with two different strategies as the underlying sequential planner. These two implementations are referred to as pSBMP-RRT, a parallel sampling-based motion planning method with RRT as the underlying sequential planner, and pSBMP-PRM, a parallel sampling-based motion planning method with PRM as the underlying sequential planner. For evaluation and comparison, we implemented two parallel algorithms mentioned in related work section: the parallel PRM (pPRM) [2] and parallel sampling-based roadmap of trees (pSRT)[1, 19].

4.1.2 Environments and Robots

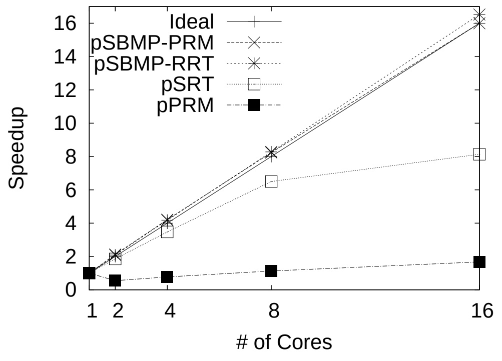
We used two different kinds of environments. The first is a homogeneous cluttered environment with dimensions of 512x512x512 units. The cluttered elements span the x-axis. The cluttered environment has a total of 216 obstacles, each of size 2x64x64 units, as shown in Figure 3(a).

The second environment shown in Figure 3(b) is a non-homogeneous cluttered environment. This particular environment models the floor plan of the H.R. Bright building (HRBB), the building that houses the Departments of Computer Science and Engineering and Aerospace Engineering at Texas A&M University.

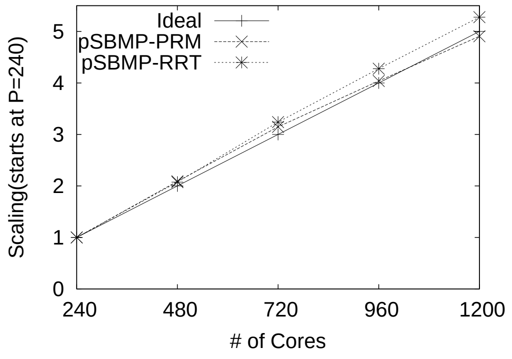
In both environments, we use two different kinds of robots: a 4x4x4 unit cube-like rigid body robot and a three-link articulated linkage robot, with each link having dimensions of 7x1x1 units.

4.1.3 Machine Architectures

Our experiment was carried out on two massively parallel computers. The first is a Cray XE6 petascale machine at Lawrence Berkely National Laboratory. It has 6384 nodes and a total of 153,216 cores with 217 TB of memory and peak performance of 1.288 peta-flops. The second machine is a major computing cluster at Texas A&M University. It has a total of 300 nodes, 172 of which are made of two quad core Intel Xeon and AMD Opteron processors running at 2.5GHz



(a) Comparison with previous approaches



(b) Scaling on Cray XE6 machine

Figure 4: Experimental Results

with 16 to 32GB per node. The 300 nodes have 2400 cores in all with over 8TB of memory and a peak performance of 24 Tflops. Our code was written in C++ using the STAPL library [7, 23] and compiled with gcc 4.4.4 on the LINUX cluster and gcc 4.6.1 on the Cray XE6 machine.

4.2 Experimental Results

We tested the four algorithms (pSBMP-PRM, pSBMP-RRT, pPRM and pSRT) on the LINUX cluster varying the processor count from 1 to 16. The input sample size was fixed at 9600 for each of the four algorithms. Each experiment was run five times and the average maximum time for the 5 runs was computed. Figure 4(a) shows the speedup for the four algorithms. From Figure 4a, one will observe that our proposed method (pSBMP-PRM and pSBMP-RRT) achieves good scalability compared to the existing methods.

To study scalability and test the limit of our method, we explore further experiments on a Cray XE6 petascale machine. In this experiment, we tested processor counts of 240, 480, 720, 960 and 1200. The results are shown in Figure 4b. We observe that scalability is still possible on a massively parallel machine such as the Cray XE6. The results also suggest that, to the extent possible, our proposed method is independent of machine architecture. Thus, though there may be variance in results, we still expect to see similar performance and scalability across different platforms. .

5. SUMMARY

In this work, we describe a scalable approach for parallelizing sampling-based motion planning algorithms. Our framework uses the subdivision of C-space to achieve scalability. We compare implementation of our method to two existing parallel algorithms for sampling-based motion planning and demonstrate that our approach achieves better and more scalable performance. We presented experimental results using both LINUX cluster and Cray XE6 petascale machines. We demonstrated that our framework is flexible enough to support different planning schemes.

6. REFERENCES

[1] M. Akinc, K. E. Bekris, B. Y. Chen, A. M. Ladd, E. Plaku, and L. E. Kavraki. Probabilistic roadmaps of trees for parallel computation of multiple query roadmaps. In *The International Symposium on Robotics Research (ISRR)*, Sienna, Italy, October 2003.

[2] N. M. Amato and L. K. Dale. Probabilistic roadmap methods are embarrassingly parallel. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 688–694, 1999.

[3] O. B. Bayazit, J.-M. Lien, and N. M. Amato. Better flocking behaviors using rule-based roadmaps. In *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 95–111, Dec 2002.

[4] 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.

[5] O. B. Bayazit, G. Song, and N. M. Amato. Ligand binding with OBPRM and haptic user input: Enhancing automatic motion planning with virtual touch. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 954–959, 2001. This work was also presented as a poster at *RECOMB 2001.*

[6] J. Bialkowski, S. Karaman, and E. Frazzoli. Massively parallelizing the rrt and the rrt*. In *Proc. IEEE Int. Conf. Intel. Rob. Syst. (IROS)*, 2011.

[7] A. Buss, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, G. Tanase, N. Thomas, X. Xu, M. Bianco, N. M. Amato, and L. Rauchwerger. STAPL: Standard template adaptive parallel library. In *Proc. Annual Haifa Experimental Systems Conference (SYSTOR)*, pages 1–10, New York, NY, USA, 2010. ACM.

[8] S. Carpin and E. Pagello. On parallel rrts for multi-robot systems. In *Proc. Italian Assoc. AI*, pages 834–841, 2002.

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

[10] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementations.* MIT Press, Cambridge, MA, June 2005.

[11] T. S. D. Devaurs and J. Cortes. Parallellizing rrt on distributed-memory architectures. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2011.

[12] D. Henrich. Fast motion planning by parallel processing - a review. *Journal of Intelligent and Robotic Systems*, 20(1):45–69, 1997.

[13] D. Hsu, L. Kavraki, J.-C. Latombe, and R. Motwani. Capturing the connectivity of high-dimensional geometric spaces by parallelizable random sampling techniques. In *Proc. IEEE Workshop Randomized Parallel Computing(WRPC)*, 1998.

[14] D. Hsu, J.-C. Latombe, and R. Motwani. Path planning in expansive configuration spaces. In *Proc.*

IEEE Int. Conf. Robot. Autom. (ICRA), pages 2719–2726, 1997.

[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. Automat.*, 12(4):566–580, August 1996.

[16] Y. Koga, K. Kondo, J. Kuffner, and J. Latombe. Planning motions with intentions. In *Proc. ACM SIGGRAPH*, pages 395–408, 1995.

[17] J. J. Kuffner and S. M. LaValle. RRT-Connect: An Efficient Approach to Single-Query Path Planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 995–1001, 2000.

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

[19] E. Plaku, K. E. Bekris, B. Y. Chen, A. M. Ladd, and L. E. Kavraki. Sampling-based roadmap of trees for parallel motion planning. *IEEE Trans. Robot. Automat.*, 2005.

[20] E. Plaku and L. E. Kavraki. Distributed sampling-based roadmap of trees for large-scale motion planning. *IEEE Transactions on Robotics and Automation*, 38:793–884, 2005.

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

[22] G. Song and N. M. Amato. Using motion planning to study protein folding pathways. In *Proc. Int. Conf. Comput. Molecular Biology (RECOMB)*, pages 287–296, 2001.

[23] G. Tanase, A. Buss, A. Fidel, Harshvardhan, I. Papadopoulos, O. Pearce, T. Smith, N. Thomas, X. Xu, N. Mourad, J. Vu, M. Bianco, N. M. Amato, and L. Rauchwerger. The STAPL Parallel Container Framework. In *Proc. ACM SIGPLAN Symp. Prin. Prac. Par. Prog. (PPoPP)*, pages 235–246, San Antonio, Texas, USA, 2011.

[24] S. Thomas, G. Tanase, L. K. Dale, J. M. Moreira, L. Rauchwerger, and N. M. Amato. Parallel protein folding with STAPL. *Concurrency and Computation: Practice and Experience*, 17(14):1643–1656, 2005.