A Parallel Motion Planner for Systems with Many Degrees of Freedom

Pekka Isto

Laboratory of Information Processing Science
Helsinki University of Technology
P.O. Box 9700, FIN-02015 HUT
evp@cs.hut.fi

Abstract

During the several decades of research, a number of algorithms intended to solve practical motion planning problems have been presented. However, the intractability of the problem makes it difficult to design algorithms capable of solving hard problems, especially when the number of degrees-of-freedom is large. It is necessary to use all available means to extend the domain of practically solvable problem instances. This paper reports results for a parallel implementation of a motion planner based on two-level search algorithm. The planner can solve difficult problems with many degrees-of-freedom within practicable time limits. Furthermore, easier problems can be solved with unprecedented search resolution.

1. Introduction

The computation of collision-free motion for an object among obstacles is an important problem with applications in robotics and virtual reality [1]. However, the problem is intractable. The current understanding is that the planning of a collision-free motion for an object among static obstacles is PSPACE-hard problem with an upper bound that is exponential in the number of degrees-of-freedom (DOF) [2]. The analysis of the problem has produced several theoretical algorithms for motion planning, but none of the theoretical algorithms have been implemented for practical use [3]. Instead, a large number of approximate and heuristic algorithms have been developed to solve various instances of practical motion planning problems [4]. Due to the complexity of the problem even the best heuristic planners take impractical amount of time to compute a solution, if the number of DOF is sufficiently high and the task is geometrically difficult. In order to solve real-world motion planning problems as efficiently as possible and to extend the scope of practically solvable motion planning tasks, all the available techniques from computer science must be used, including parallel processing.

The purpose of this paper is to demonstrate that a two-level search algorithm [5] can be used to solve geometrically difficult motion planning problems for systems with many degrees-of-freedom (here up to 18). Furthermore, the algorithm can solve less difficult tasks with a search resolution that is an order of magnitude higher than other methods [6]. High-resolution planning is required in workspaces that contain tight clearances or thin obstacles like in assembly planning and spot welding.

The structure of the algorithm makes it easy to parallelize it efficiently. A parallel implementation of the algorithm running on a PC-cluster can solve the Hwang and Ahuja benchmark task [2] in average time comparable to the cycle time enabling near real-time operation. The Alpha Puzzle 1.2 benchmark task [7] can be solved in minutes demonstrating significant improvement over previous results.

The next section will present a brief overview of the previous related research in motion planning. Section 3 provides an overview of the implemented algorithm and the selected parallelization strategy. Section 4 presents the experimental results. Finally, section 5 presents conclusions.

2. Previous Related Research

A considerable amount of literature has been accumulated during the past three decades of research in motion planning. Several good reviews and books have been published on the subject [1,2,3,4,8]. Therefore, this section is limited to the context of the results presented here.

Like most contemporary algorithms for motion planning, the presented algorithm performs search in discretized representation of the robot's joint space or configuration space (*C-space*) [9]. In order to deal with the exponential cost of global planners and the susceptibility to local minima of local planners, Faverjon and Tournassoud proposed to combine both methods into a single two-level planner [10]. Glavina presented randomized planner using similar approach that builds a subgoal graph by connecting random subgoals or landmarks in *C-space* with a "sliding" local planner [11]. The SANDROS search strategy performs selective and non-uniform subdivision of the *C-space* and uses a "sliding" local planner to connect subgoals

from different portions of the *C-space* [12]. As their local planners are relatively weak, these motion planners are quite dependable on the quality of the generated subgoals. Recent developments along this line of research involve the heuristic placement of the subgoals [13,14,7].

Many of the motion planning approaches and algorithms originally introduced in serial form have been later formulated into parallel form. Only a limited number is covered here; for a more extensive discussion, see the review by Henrich [15]. Glavina conjectures that a significant reduction in time for the first solution can be attained by running his algorithm in parallel on multiple workstations [16]. Challou *et al.* [6] and later Caselli and Reggiani [17] use the same approach to construct a parallel implementation of the influential Randomized Path Planner (RPP) [18]. Instances of the RPP solving the original problem are run on multiple processors and the first solution is kept. Experimental results demonstrate good expected speed-up and reduction in variance of the planning time when the problems are sufficiently difficult.

Baginski presents a parallel version of Glavina's algorithm for time-varying workspaces [19]. Rather than running multiple instances of the algorithm on the processors, the implementation distributes the local planners into the available slave processors and builds the subgoal graph in the master processor. Little experimental data is provided, but a network of 30 workstations provides only a speed-up of four. A similar approach is used by Qui and Henrich but with a better speed-up [20]. Their local planner is rule-based.

Mazer *et al.* present a motion planner based on parallel genetic algorithms [21]. In addition to running the components of the planner in parallel, they implement parallel collision detection. The planner should be quite scalable due to the perfectly parallel nature of genetic algorithms.

3. A Parallel Motion Planner

The motion planner presented in this paper is a parallel and improved version of the adaptive two-level heuristic search algorithm presented earlier [5]. The upper level is a subgoal graph built with random subgoals. A unique feature of the algorithm is that rather than using a single local planner or a combination of local planners to attempt the path segments between the subgoals, it uses a local planner with continuously adaptable capability. As more subgoals are needed and generated for solving the problem, the global planner increases the capability of the local planner. The subgoal graph is essentially a task decomposition method and the global planner attempts to construct a solution from solutions to subproblems with a cost measure below a certain increasing limit.

The local planner originates from the free-space enumeration algorithm presented by Kondo [22]. It uses a bi-directional A^* algorithm guided by four different heuristics to search a grid representation of the *C-space*. The heuristics are executed in round-robin fashion and the efficiency of each heuristics $t = 1,...,4$ at configuration node C is estimated by the formula:

$$p_t(C) = \frac{g(C)^{DOF}}{F_t(C)}$$

where $g(C)$ is the distance from the start configuration to the current configuration C in grid steps (Manhattan distance) and $F_t(C)$ is the total number of configurations examined by the heuristics t until the examination of C. The overall efficiency $P_t(j)$ of heuristics t at round j is estimated by taking an average of $p_t(C)$ over the last 20 configurations.

At the first round of execution each heuristics is allocated 25 node expansions in the A^* algorithm. Subsequently, the number of examinations is determined by the relative efficiency of the heuristics with the following formula:

$$E_t(j+1) = \max\left\{25 \times \frac{P_t(j)}{\max\{P_1(j),..., P_T(j)\}}, 1\right\}$$

where E_t stands for the number of examinations allocated for heuristics t. A second evaluation value is calculated for each examined configuration:

$$O_t(C) = \frac{\sum_t F_t(C)}{g(C)}$$

If the evaluation value $O_t(C)$ for the heuristics t increases above a threshold value O_{th}, the execution of the heuristics is discontinued for that round. If all heuristics are discontinued, the local planner fails.

The heuristic evaluation of a configuration uses the standard form of A^* guiding function $f(C) = g(C) + h(C)$, where $g(C)$ is as above and $h(C)$ is an estimate for the cost from the evaluated configuration to the goal configuration. The following four weight sets are used in the estimate:

Manipulator heuristics:

$$a_i = \left\lceil 9\frac{DOF + 1 - i}{DOF} \right\rceil, \quad i = 1,..., DOF.$$

Position heuristics:

$$a_i = \begin{cases} 9, & i = 1,\ldots,d \\ 1, & i = d+1,\ldots, DOF \end{cases}$$

Rotation heuristics:

$$a_i = \begin{cases} 1, & i = 1,\ldots,d \\ 9, & i = d+1,\ldots, DOF \end{cases}$$

Even heuristics:

$$a_i = 5, \quad i = 1,..., DOF.$$

$d = \lfloor (DOF + 0.5)/2 \rfloor$. An additional "greediness" multiplier $A=3$ and a tie-breaking term ρ are added to the full expression of $h(C)$:

$$h(C) = A\left(\sum_{i=1}^{DOF} a_i D_i(C,G) - \rho a_j \right),$$

$D_i(C,G)$ is the distance between the current configuration and the goal configuration G in grid steps along axis i. The tie-breaking term has a value of 0.5 if the evaluated node was expanded in the same direction as the parent node along the axis j or 0 otherwise.

All of the heuristics share the search space representation, and therefore progress made by any of the heuristics will immediately and continuously benefit all of them. The evaluation values control the search so that relatively more efficient heuristics are used more and the individual heuristics and eventually the local planner are discontinued if sufficient progress is not made.

The memory consumption of the local planner grows exponentially as the number of degrees-of-freedom is increased. The threshold parameter O_{th} can be used to control the memory consumption, but an additional limit on the maximum number of configuration nodes for each subtask is needed. This is a machine specific parameter that depends on the amount of main memory in the machine.

The global planner generates random subgoals and builds two trees of subgoals and successful subpaths, one rooted at the start configuration and the other at the goal configuration. Pohl's cardinality principle [23] is used to select the tree for expansion. As more subgoals are generated, the threshold parameter is updated according to the formula $O_{th} = O'_{th} R^S$, where S is the current number of subgoals, O'_{th} and R are constants with values of *3* and *1.05*. This means that the local planner becomes gradually more powerful and the balance of computation shifts increasingly from global planning to local planning.

The main motivation for the parallel implementation of the motion planner is to reduce the wall-clock time spent in solving the planning tasks. Although no established limit exists to denote what is "practicable" planning time, Gupta considers run times in few minutes and few tens of minutes "reasonable" [8] and times in hours "impracticable" [24]. A parallel implementation should be able to reduce the running time from hours to minutes or tens of minutes with reasonable computing resources.

Several alternative parallelization strategies could be used. The whole planner can be run on multiple machines for the first solution like parallel RPP. Or the local planners can be distributed to the slave processors while the global planner is run at the master processor. A hybrid strategy would combine these approaches by running multiple copies of the whole planner on subtasks at the slave processors and running an additional global planner at the master. The results presented here are for the second strategy as it has

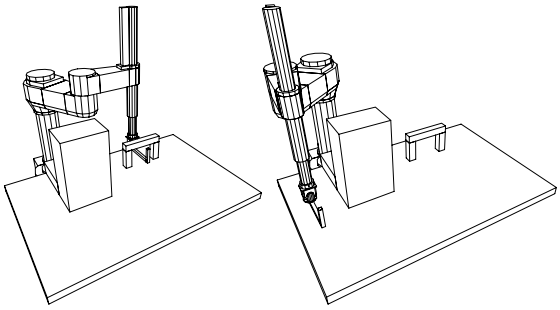

Figure 1: A version of the benchmark problem proposed by Hwang and Ahuja.

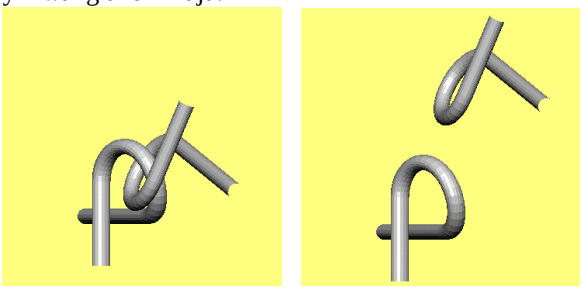

Figure 2: Alpha Puzzle 1.2 benchmark task. Two intertwined ∝ shaped tubes must be separated.

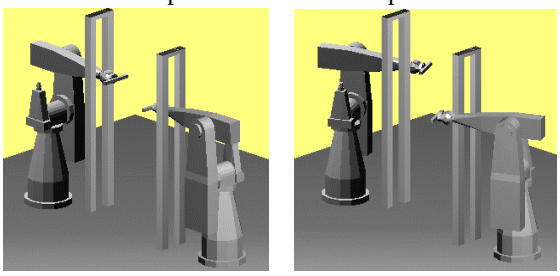

Figure 3: The start and goal configurations for the 12DOF task "dogs with bones".

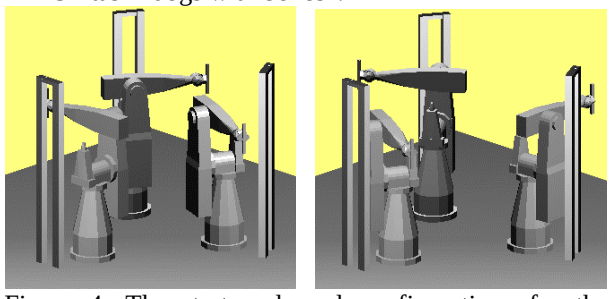

Figure 4: The start and goal configurations for the 18DOF task. The robots must avoid the gates and each others while performing a right hand rotation.

	CPUs	Min.	25%	50%	75%	Max.	Ave.	Std.
5DOF	1	2.7	14.7	23.1	34.6	158.0	28.5	22.4
	11	0.6	1.6	2.3	3.1	11.7	2.6	1.6
HR5DOF	1	75.7	216.0	390.5	623.0	3609	500.0	470.0
	11	21.8	41.3	58.0	82.5	312	68.6	42.1
6DOF	1	14.1	209.5	450.0	909.5	5636	764.4	956.9
	11	0.8	15.2	32.2	66.6	417	54.7	69.6
HR6DOF	1	23.5	1302	4171	7407	21377	5045	4666
	11	12.1	120	331	655	1855	444.4	409.1
10DOF	1	45.9	498.5	1117.5	2579	10648	1981	2272
	11	9.3	42.0	96.6	183.5	987	149.1	181.0
12DOF	1	10.9	286.0	940.5	5553	34974	4112	6300
	11	4.5	17.5	58.1	305.0	1945	225.6	345.6
18DOF	1	31.2	476.5	1297	3143	25584	3330	5026
	11	10.8	38.0	82.0	186.5	1661	214.8	325.3

Table 1: Run times in wall clock seconds for the various tasks on a single 500 MHz CPU and on the whole cluster of 11 CPUs. Sample size is 100 runs. Percentiles are rounded up.

the smallest granularity of computing, and thus is the worst case for scalability.

4. Experimental Results

The parallel motion planner was tested on a Linux PC cluster comprised of 11 processors with clock speeds between 450 MHz and 550 MHz and memory sizes of 128 MB or 512 MB. The computing nodes are connected with 100 Mbit Ethernet. For the scalability experiments the computing nodes are grouped so that each group has an average clock speed of 500 MHz. The implementation uses RAPID collision detection library [25] and MPICH message passing library [26].

As seen in table 1, the planner can solve the Hwang and Ahuja benchmark task (figure 1) in seconds with a 296×171×42×191×105 grid representation of the *C-space* (label: 5DOF). The search resolution is the same as that of the similar "AdeptOne" task [12]. A 2960×1710×420×1910×1050 high-resolution version of the task (HR5DOF) can be solved in few minutes using the whole cluster. The benchmark task is designed to force a large backtracking motion to the solution and many other planners would need considerable effort for producing the backtracking motion at such a high search resolution.

The Alpha Puzzle 1.2 task (figure 2) is intended to represent disassembly problems with a "narrow passage". The parallel planner can solve it in minutes with a resolution of 128 positions for each DOF (6DOF). A high-resolution version with 1280 positions can be solved in tens of minutes on the whole cluster (HR6DOF). These are significant improvements, since the earlier results are in hours on a single processor [7]. The planner has also demonstrated capability to plan motions through narrow space in the previous experiments [5].

The main motivation of this paper is to complement earlier experiments with many degrees-of-freedom problems and demonstrate that the planner can be used for such problems despite the potentially excessive memory consumption of the local planner [5].

Various notions of what is considered a large number of degrees-of-freedom exist. Gupta states that from the practical point of view, systems with more that four DOF must be included in the category [24,8]. Faverjon and Tournassoud would only include systems with 8 or more DOF [10]. Kavraki and Latombe introduced the PRM approach especially for systems with many degrees-of-freedom [27]. Their test cases have 8 and 12 DOF. The minimum number of degrees-of-freedom required to place an object into an arbitrary position and orientation in 3D space is six. Therefore, motion planners intended for general use should be able to generate motions for 6 degrees-of-freedom systems. Industrial robot cells usually have auxiliary degrees-of-freedom for placement of the work piece. Multirobot cells can easily have two dozens degrees-of-freedom. Recently introduced new applications, such as planning for flexible objects, call for several dozens or hundreds of degrees-of-freedom [28,29,30]. In this paper, systems with 10 or more degrees-of-freedom are included in the category of problems with many degrees-of-freedom.

The Hwang and Ahuja benchmark task can be extended into a 10 DOF version by planning the motion

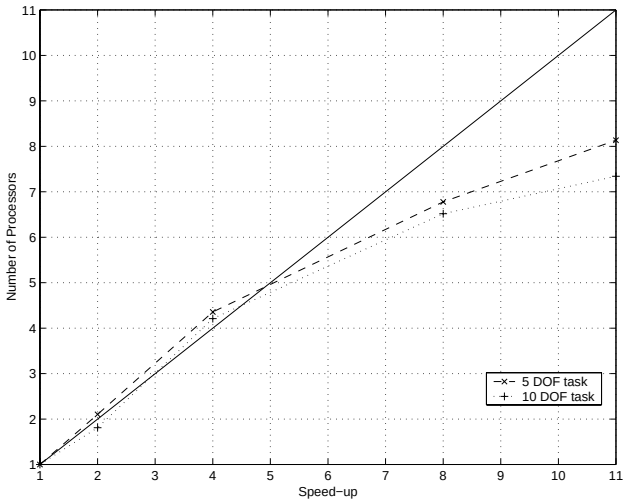

Figure 5: Speed-up for tasks 5DOF and 10DOF. The speed-up is calculated from the average run time for 5 runs of the planner.

simultaneously for two independent SCARA robots (10DOF). Similarly, 12 DOF and 18 DOF tasks are constructed from two and three Puma type robots as shown in figures 3 and 4. The 12DOF and 18DOF tasks are planned with 100 discrete positions for each degree-of-freedom. The data in table 1 shows that also these problems can be solved in minutes with the cluster.

Finally, the scalability of the planner is shown in figure 5. Even the worst-case parallelization strategy provides an acceptable speed-up. Additional reduction in planning time could be attained by adding more processors to the cluster. However, the graph shows diminishing returns and eventually a larger granularity strategy must be used. The superlinear speed-up for small number of processors is probably caused by cache effects.

5. Conclusions

This paper presented experiments with a parallel motion planner based on two-level heuristic search in the configuration space. The experiments demonstrated a moderately good scalability of the algorithm on an inexpensive parallel computer built form commodity hardware and free software. While the algorithm shows diminishing speed-up as the number of processors is increased, alternative parallelization strategies may be used to provide improvement in the behavior of the planner. Even in the current form, the parallel implementation can be used to solve very difficult motion planning problems within near real-time and practicable off-line time limits. The test problems included cases with many degrees-of-freedom demonstrating that A^* search-based approach can be used to solve such cases. Additionally, easier but non-trivial problems can be solved with exceptionally high-resolution representation of the *C-space*. The results on Alpha Puzzle 1.2 presented here are a significant improvement over previous results.

Acknowledgments

This research is funded by the Helsinki Graduate School in Computer Science and Engineering. The parallel implementation is based on software developed by Johannes Lehtinen and the IMPACT group of Tik-76.115 Software Project course. Mr. Lehtinen also provided the geometric model for the 5DOF test problem. Alpha Puzzle 1.2 was designed by Boris Yamrom, GE Corporate Research & Development Center. The model was provided by the DSMFT research group at Texas A&M University. Janne Ravantti provided access to the cluster computer at the Bamford Laboratory, University of Helsinki. The author thanks Dr. Juha Tuominen for helpful comments and suggestions.

References

[1] J. C. Latombe, Robot Motion Planning, Kluwer Academic Publishers, Norwell, Mass. 1991.

[2] Y. K. Hwang, N. Ahuja, Gross Motion Planning - A Survey, ACM Computing Surveys, vol. 24, no. 3, Sep. 1992, 219-291.

[3] J. C. Latombe, Motion Planning: A Journey of Robots, Molecules, Digital Actors, and Other Artifacts, International Journal of Robotics Research, vol. 18, no. 11, November 1999, 1119-1128.

[4] K. Gupta, A. P. del Pobil (eds.), Practical Motion Planning in Robotics: Current Approaches and Future Directions, John Wiley & Sons, West Sussex, 1998.

[5] P. Isto, A Two-level Search Algorithm for Motion Planning, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, IEEE Press, 2025-2031.

[6] D. J. Challou, D. Boley, M. Gini, V. Kumar, C. Olson, Parallel Search Algorithms for Robot Motion Planning, In [4], 115-131.

[7] D. Vallejo, C. Jones, N. M. Amato, An Adaptive Framework for `Single Shot' Motion Planning, Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2000.

[8] K. Gupta, Motion Planning for “Flexible'” Shapes (Systems with Many Degrees of Freedom): A Survey, The Visual Computer, vol. 14, no. 5/6, 288-302.

[9] T. Lozano-Pérez, M. Wesley, An Algorithm for Planning Collision-free Paths among Polyhedral

Obstacles, Communications of the ACM, vol. 22, no. 10, October 1979, 560-570.
[10] B. Faverjon and P. Tournassoud, A local approach for path planning of manipulators with a high number of degrees of freedom, Proceedings of the 1987 IEEE International Conference on Robotics and Automation, Raleigh, NC, 1987, 1152-1159.
[11] B. Glavina, Solving Findpath by Combination of Goal-Directed and Randomized Search, Proceedings of the 1990 IEEE International Conference on Robotics and Automation, IEEE, 1990, 1718-1723.
[12] P. C. Chen, Y. K. Hwang, SANDROS: A Dynamic Graph Search Algorithm for Motion Planning, IEEE Transactions on Robotics and Automation, vol. 14, no. 3, June 1998, 390-403
[13] D. Hsu, J. C. Latombe and R. Motwani, Path Planning in Expansive Configuration Spaces, Proceedings of the 1997 IEEE International Conference on Robotics and Automation, IEEE, 1997, 2719-2726.
[14] R. Bohlin and L. E. Kavraki, Path Planning Using Lazy PRM, Proceedings of the 2000 IEEE International Conference on Robotics and Automation, IEEE, 2000.
[15] D. Henrich, Fast Motion Planning by Parallel Processing – a Review, Journal of Intelligent and Robotic Systems, vol. 20, no. 1, September 1997. 45-69.
[16] B. Glavina, A Fast Motion Planner for 6-DOF Manipulators in 3-D Environments. Proceedings of the Fifth International Conference on Advanced Robotics, IEEE Press, 1991, 1176-1181.
[17] S. Caselli and M. Reggiani, Randomized Motion Planning on Parallel and Distributed Architectures, 7th Euromicro Workshop on Parallel and Distributed Processing, PDP'99, IEEE, 1999.
[18] J. Barraquand, J.C. Latombe, Robot Motion Planning: A Distributed Representation Approach, The International Journal of Robotics Research, vol. 10, no. 6, Dec. 1991, 628-649.
[19] B. Baginski, Fast Motion Planning in Dynamic Environments with the Parallelized Z^3-Method, Proceedings of the Sixth International Symposium on Robotics and Manufacturing ISRAM, ASME Press, 1996, 81-86.
[20] C. Qin, D. Henrich, Randomized Parallel Motion Planning for Robot Manipulators, Technical Report 5/96, Computer Science Department, University of Karlsruhe, 1996.
[21] E. Mazer, J. Ahuactzin, G. Talbi and P. Bessiere, The Ariadne's Clew Algorithm, Journal of Artificial Intelligence Research vol. 9, 1998, 295-316.
[22] K. Kondo, Motion Planning with Six Degrees of Freedom by Multistrategic Bidirectional Heuristic Free-Space Enumeration. IEEE Transactions on Robotics and Automation, Vol. 7, No. 3, June 1991, 267-277.
[23] I. Pohl, Bi-directional Search, Machine Intelligence 6, Edinburgh University Press, Edinburgh, 1971, 127-140.
[24] K. Gupta, Overview and State of the Art, In [4], 3-8.
[25] S. Gottschalk, M. C. Lin, D. Manocha, OBB-Tree: A Hierarchical Structure for Rapid Interference Detection, Technical report TR96-013, Department of Computer Science, University of N. Carolina, Chapel Hill.
[26] W. Gropp, E. Lusk, N. Doss, A. Skjellum, A High-Performance, Portable Implementation of the MPI Message Passing Interface Standard, Parallel Computing, vol. 22, no. 6, September 1996, 789-828.
[27] L. Kavraki, J.C. Latombe, Randomized Preprocessing of Configuration Space for Fast Path Planning, Proceeding of the 1994 IEEE International Conference on Robotics and Automation, IEEE Press, Los Alamitos, 1994, 2138-2145.
[28] L. E. Kavraki, F. Lamiraux, and R. Holleman, Towards planning for elastic objects. In P. K. Agarwal, L. Kavraki, and M. Mason (eds.), Robotics: The Algorithmic Perspective. AK Peters, 1998.
[29] A.P. Singh, J.C. Latombe, and D.J. Brutlag, A Motion Planning Approach to Flexible Ligand Binding, Proceedings 7th International Conference on Intelligent Systems for Molecular Biology (ISMB), AAAI Press, Menlo Park, CA, 1999, 252-261.
[30] G. Song and N. M. Amato, A Motion Planning Approach to Folding: From Paper Craft to Protein Structure Prediction, Technical Report 00-001, Department of Computer Science, Texas A&M University, January 17, 2000.