A Kinematics-Based Probabilistic Roadmap Method for High DOF Closed Chain Systems*

(To appear in the 2004 IEEE International Conference on Robotics and Automation (ICRA'04))

Dawen Xie and Nancy M. Amato
Parasol Lab, Department of Computer Science
Texas A&M University, College Station, TX 77843-3112
Email: {dawenx,amato}@cs.tamu.edu

***Abstract*— In this paper we consider the motion planning problem for arbitrary articulated structures with one or more closed kinematic chains in a workspace with obstacles. This is an important class of problems and there are applications in many areas such as robotics, closed molecular chains, graphical animation, reconfigurable robots. We use the kinematics-based probabilistic roadmap (KBPRM) strategy proposed in [1] that conceptually partitions the linkage into a set of open chains and applies random generation methods to some of the chains and traditional inverse kinematics methods to the others. The efficiency of the method depends critically on how the linkage is partitioned into open chains. The original method assumed the partition was provided as input to the problem. In this paper, we propose a fully automated method for partitioning an arbitrary linkage into open chains and for determining which should be positioned using the inverse kinematic solver. Even so, the size (number of links) of the closed loops that can be handled by this method is limited because the inverse solver can only be applied to small chains. To handle high dof closed loops, we show how we can use the Iterative Relaxation of Constraints (IRC) strategy proposed by Bayazit to efficiently handle large loops while still only using inverse kinematics for small chains.**

Our results in 3-dimensional workspaces both for planar and spatial linkages show that our framework performs well for general linkages. We also use our planner to simulate an adjustable lamp called Luxo. Using IRC, our planner can handle a single loop of up to 98 links.

I. INTRODUCTION

Closed chain mechanisms arise in many applications both in and beyond robotics, such as the Stewart Platform [2], parallel robots [3], closed molecular chains [4], graphical animation [5], reconfigurable robots [6], [7], and the closed chain system formed by multiple robots grasping an object [8] (Fig. 1).

In this paper, we consider the motion planning problem for arbitrary linkage structures with one or more closed kinematic chains in a workspace with obstacles. Since there is strong evidence that any deterministic planner requires time exponential in the number of degrees of freedom (dof) of the movable object [9], attention has focused on randomized approaches. In particular, *Probabilistic Roadmap Methods (*PRM*s)* [10] have been used successfully for many high dof problems.

While closed chains can offer advantages over open chains in terms of the rigidity of the mechanism, motion planning and control of closed chains is complicated by the need to maintain the closed chain structure, the so-called *closure constraints*. It is very difficult to find configurations satisfying the closure constraints using purely randomized techniques since the probability that a randomly generated configuration lies on the constraint surface is zero [11].

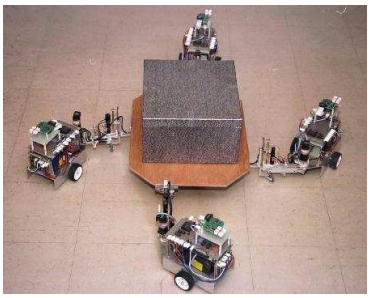
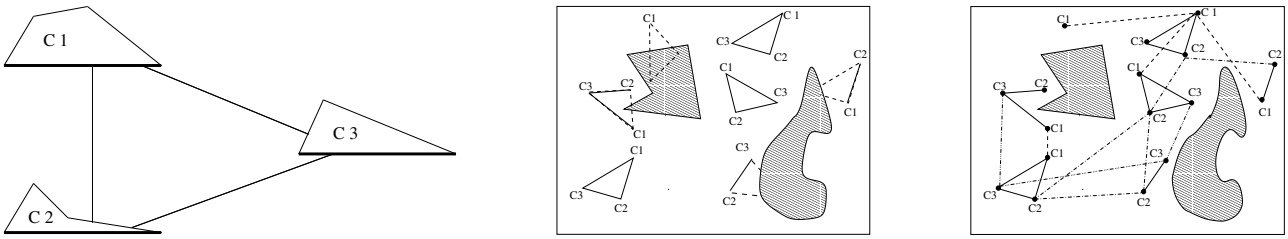


Fig. 1. Four manipulators coordinate to handle an object. (http://www.robotics.is.tohoku.ac.jp/lab/robot/)

We are aware of two efforts to extend the PRM technique to systems with closed kinematic chains. LaValle et al. [11] proposed the first method which demonstrated PRM could be applied to closed chains; in subsequent work [12], they obtained somewhat improved performance applying the RRT approach [13]. Next, Han and Amato [1] proposed a more efficient method called KBPRM that conceptually partitioned the linkage into a collection of open chains, and then used PRM-based random sampling to position some of the chains and applied inverse kinematics to the remaining chains to enforce the closure constraints. The efficiency of their approach depends critically on how the linkage is partitioned into open chains. In [1], it was assumed that the partition was provided as input to the problem.

In this paper, we propose a fully automated method to perform the partitioning needed by KBPRM. In particular, we show how a technique from graph theory called *ear decomposition* can be adapted to partition a complex linkage containing an arbitrary number of closed loops into a set of open chains suitable for KBPRM. While the KBPRM strategy is quite efficient, in practice the size (number of links) of the closed loops that can be handled by this method is limited because the inverse solver can only be applied to relatively

*This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0081510, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874.



(a) (b) (c)

Fig. 2. KBPRM first builds a kinematic roadmap without considering the environment and then places copies of it in the environment. (a) A three-node kinematic roadmap for a 4-link closed chain (C-Space). (b) Copying the kinematic roadmap to the environment (C-space); only the feasible (solid) edges are retained. (c) Connecting configurations of the same closure type (C-space); solid edges are from the kinematic roadmap and dashed edges represent connections between configurations with the same closure type from different copies of the kinematic roadmap.

small chains. To handle high dof closed loops, we show how we can use the Iterative Relaxation of Constraints (IRC) strategy [14] to deal with loops with large numbers of links, while still only computing inverse kinematics solutions for very small systems.

II. Related Work

The general methodology of PRMs [10] is to first construct a graph (roadmap) that represents the connectivity of the robot's free configuration space (C-space), and then query the roadmap to find a valid path for a given motion planning task. Initially, PRMs were mainly limited to rigid bodies and articulated objects without closed chains.

In the approach used in [11], roadmap nodes are found by first sampling random nodes in C-space and then applying an iterative random gradient descent method. The method minimizes error functions that express the closure constraints. Roadmap edges are computed by executing a randomized traversal of the constraint surface between two nodes. Planar test problems with up to 8 links and two loops were solved in several hours. In [12], which is an extension of [11], the same authors proposed a method based on PRM and rapidly-exploring random trees (RRTs) [13]. Problems with similar complexity were tested for both planners. The PRM-based planner achieved similar results as [11], and the RRT based planner achieved significantly better results, solving problems in 8–20 minutes that previously took hours.

As previously mentioned, we use the kinematics-based PRM (KBPRM) approach proposed by Han and Amato [1]. Planar and spatial single loop linkages with 7–9 links were solved under a minute. (See Section III or the paper [1] for more details.) Cortes et al. [15] proposed an extension of KBPRM called Random Loop Generator (RLG) to improve the generation of random configurations. It samples parameters one at a time to guide the end-frame of a subchain toward a region which is in the reachable workspace of the subchain which is computed using inverse kinematics. Here, we handle high dof closed chain systems by applying IRC [14] to KBPRM.

III. Overview of KBPRM

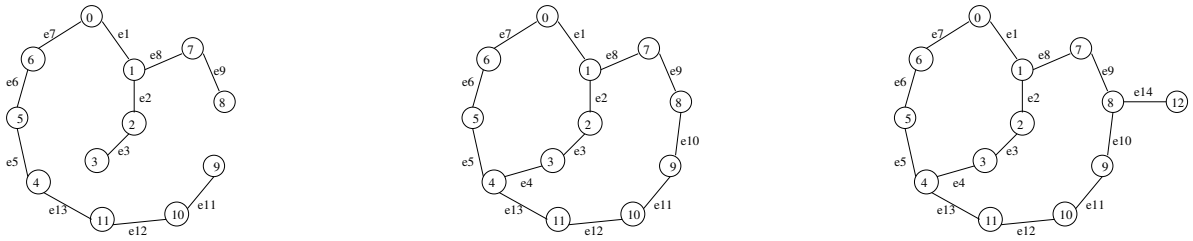
We now give a brief overview of kinematics-based PRM (KBPRM) [1]. Briefly, KBPRM (conceptually) 'breaks' all closed loops into a set of open subchains. Standard PRM random sampling techniques and forward kinematics are applied to one subset of the subchains, and then traditional inverse kinematics solutions are applied to the remaining subchains to enforce the closure constraints. This strategy preserves the PRM sampling philosophy, while addressing the fact that the probability that a random configuration will satisfy the closure constraints is zero, which proved problematical in previous attempts to apply the PRM methodology to closed chain systems.

In many cases the efficiency of KBPRM can also be improved by applying it in a two-stage strategy, both of which employ the PRM framework. First, we disregard the environment, fix the position and orientation of one link (the "virtual" base) of the system, and construct a *kinematic roadmap* (Fig. 2(a)) which contains different self-collision-free closure configurations. Next, we place copies of (portions of) the kinematic roadmap (nodes and edges) in the environment (Fig. 2(b)), and then use rigid body planners to connect configurations of the same closure type (Fig. 2(c)). This two-stage approach enables us to amortize the cost of computing and connecting closure configurations.

The efficiency and effectiveness of KBPRM depends critically on how the linkage structure is partitioned into a set of open chains. For a closed chain linkage involving more than one loop, a link or joint may be involved in multiple closed chains, and in this case the effectiveness of the method will also depend on the relative order in which one tries to enforce the closure constraints. In the original KBPRM method [1], it was assumed that the partition into open chains and the decision on how each chain would be processed were provided as input to the method. A major contribution of this paper is to provide an automated method for performing this processing and assignment, and to demonstrate that it achieves good results for arbitrary linkage structures.

IV. Breaking Closed Loops

In this section, we describe a method to break an arbitrary linkage structure into a set of open chains and to determine how each chain should be positioned, i.e., by random sampling (forward kinematics) or using inverse kinematics, and in what order. This is preprocessing that is done before applying KBPRM.



(a) (b) (c)

Fig. 3. (a) An open chain linkage in which the underlying graph G is a tree; vertices 3, 8 and 9 are artificial vertices we added for unary links e3, e9 and e11. (b) A closed chain linkage. (c) A compound chain linkage.

A. General Structure of Linkage

The parts of a mechanism, regardless of their shape or the type of connections between them, are called *links*. The *joints*, which are connections between links, are formed by at least two links coming into contact. A set of links connected by joints is called a *kinematic chain* [16].

We consider a two-dimensional or three-dimensional world, $W \subset R^N$, with $N = 2$ or $N = 3$. Let $L = \{L_1, L_2, ..., L_m\}$ represent a collection of m links. Let $J = \{J_1, J_2, ..., J_n\}$ be a collection of n joints, each of which connects a subset of links in L and attaches links at their endpoints. We define $KC = (L, J)$ to be a kinematic chain. A *configuration* of the kinematic chain is a vector of real-valued parameters which determines the position and orientation of all links.

It will be convenient to view a kinematic chain as a graph in which the vertices represent joints and the edges represent links. An artificial vertex needs to be created in the graph for each unary link (a link connected to a single joint) and it will be connected only to the edge of the corresponding to the unary link. In particular, let G denote the graph that corresponds to a kinematic chain. If G is a tree, then we define it as an *open chain linkage* (Fig. 3(a)). If G is cyclic and there exists no edge whose removal disconnects G, the chain is called a *closed chain linkage* (Fig. 3(b)). Another class is the *compound chain linkage* in which G is cyclic with at least one edge whose removal disconnects G (Fig. 3(c)).

We first note that we can 'break' a compound chain linkage into components where each component is either a closed chain linkage or an open chain linkage. For example, the linkage in Fig. 3(c) can be broken into a closed chain linkage component (Fig. 3(b)) and an open chain linkage component which only has an edge $e14$. Secondly, a closed chain linkage can alternatively be viewed as a linkage system consisting of a collection of open chain linkages, where we 'break' each closed chain, and then satisfy the closure constraints, if any, by forcing the break points to coincide. Thus, we can 'break' any system into a set of open chain linkages.

B. Ear Decomposition

We use the *ear decomposition* technique [17] from graph theory to partition the kinematic chain into an *ordered* set of open chains. The ordering will be used to determine the order in which the closure constraints are enforced. Without loss of generality, in the discussion below we assume that we have a closed chain linkage (the open chains can easily be identified and positioned using random sampling).

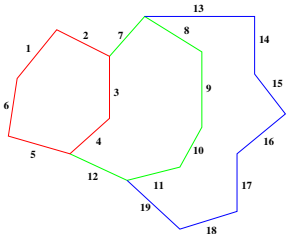


Fig. 4. An ear decomposition of G is an ordered partition of the set of edges $E = \{P_0, P_1, ..., P_k\}$. In this example, an ear decomposition starting from P_0 is $E = \{P_0, P_1, P_2\}$ where $P_0 = \{1, 2, 3, 4, 5, 6\}$, $P_1 = \{7, 8, 9, 10, 11, 12\}$, and $P_2 = \{13, 14, 15, 16, 17, 18, 19\}$.

Let $G = (V, E)$ be an undirected graph, with $|V| = n$ and $|E| = m$. Let P_0 be an arbitrary simple cycle of G. A simple cycle is formed by a non-tree edge and a spanning tree of G. An ear decomposition of G starting with P_0 is an ordered partition of the set of edges $E = \{P_0, P_1, ..., P_k\}$, such that each endpoint of P_i, for $i \geq 1$, is contained in some P_j, for $j < i$, and none of the internal vertices of P_i are contained in any P_j, for $j < i$ [17] (Fig. 4). Each simple path P_i is called an *ear*. An ear is *open* if it is noncyclic and is *closed* otherwise.

An undirected graph $G = (V, E)$ has an ear decomposition if and only if it is bridgeless (i.e., there exists no edge whose removal disconnects the graph). The underlying graph G we get for a closed chain linkage is a bridgeless graph, and we can apply ear decomposition on G directly. Thus, the ear decomposition will decompose the closed chain linkage into an ordered set of *simple chains* (which has at most one loop).

C. Partition Algorithm

After the kinematic chain has been decomposed into a set of simple chains, we need to partition them into two subsets, one which will be positioned using PRM random sampling techniques and one which will be positioned using inverse kinematics to enforce the closure constraints. Hence, the task here is to partition the joint variables into two groups, the *sampled variables* θ_s and the *computed variables* θ_c, where the θ_c will be determined after the θ_s using inverse kinematics. Algorithm IV.1 outlines our approach for partitioning an arbitrary linkage using the ear decomposition technique.

Algorithm IV.1 Partition Algorithm

1: Get an underlying graph G from the input to represent the closed chain linkage
2: **if** G does not have loop **then**
3: **return** NO-LOOP
4: **else**
5: Compute ear decomposition $E = \{P_0, P_1, \ldots, P_k\}$ of G
6: Choose the last three consecutive joint variables in P_0 as θ_c
7: **for** each P_i, $i > 0$ **do**
8: Choose three consecutive joint variables from one of its end points as θ_c
9: **end for**
10: Choose the rest joint variables as θ_s
11: **end if**

For a system that has any closed chains, we first use the ear decomposition technique (Step 5) to partition it into a sequence of simple chains. We then divide the joint angles in each simple chain into two groups: θ_s and θ_c. From an algorithmic point of view, θ_s needs to be chosen such that for a given value of θ_s, the corresponding value of θ_c satisfying the closure constraints, if any, can be computed efficiently. While solutions are not known for inverse kinematics problems for general linkages, closed-form solutions do exist for simple chains (such as 4-link chains) and most industrial robots. In our current implementation, we have a solution for a 3-joint/4-link chain [18], which is why we select three consecutive joint variables as θ_c in steps 6 and 8.

V. Roadmap Construction

To construct the roadmap, we first build a kinematic roadmap which encodes the connectivity of the closure configurations and does not depend on the environment, and then place copies of the kinematic roadmap in the environment and make additional connections between them.

Node Generation. The node generation algorithm is sketched in Algorithm V.1. Here θ_s denotes the sampled variables and θ_c denotes the computed variables as determined by Algorithm IV.1. For a system involving closed chains, we first randomly sample θ_s. Steps 5 – 11 process the computed chains θ_c in the order provided by the ear decomposition using the inverse kinematics solver. After checking for self-collision, the closed configurations are added as nodes to the kinematic roadmap. In step 7, if multiple solutions exist for the inverse kinematics problem, we can either keep all solutions or randomly choose one. Fig. 5 shows some closed configurations generated by our planner.

Node Connection. An edge between two closure configurations in the roadmap consists of a sequence of intermediate closure configurations. The general strategy of node connection here is as same as [1]. In particular, we use the local planner to connect the sampled variables θ_s of the two 'nearby' closure configurations, and then compute the corresponding computed variables θ_c along the local path.

Two-stage Approach. The general idea here is as same as [1]. In addition, we apply some heuristics to improve the connectivity of the roadmap.

Algorithm V.1 Node Generation Algorithm

1: **if** NO-LOOP is returned in partition algorithm **then**
2: Randomly generate all the joint variables θ
3: **else**
4: Randomly sample θ_s
5: **for** $i = 0, i \leq k$ (k is the number of loops) **do**
6: Use forward kinematics to compute the transformation for computed variables in P_i
7: Use inverse kinematics to compute the θ_c in P_i
8: **if** no solution is found **then**
9: **return** failure
10: **end if**
11: **end for**
12: **if** $\theta = (\theta_s, \theta_c)$ is self-collision free **then**
13: **return** θ
14: **end if**
15: **end if**

VI. Using IRC with KBPRM

Recall that the effectiveness of KBPRM decreases as the number of links increases. In this section, we discuss how to apply the *Iterative Relaxation of Constraints (IRC)* [14] strategy to enable us to handle high dof closed chain systems.

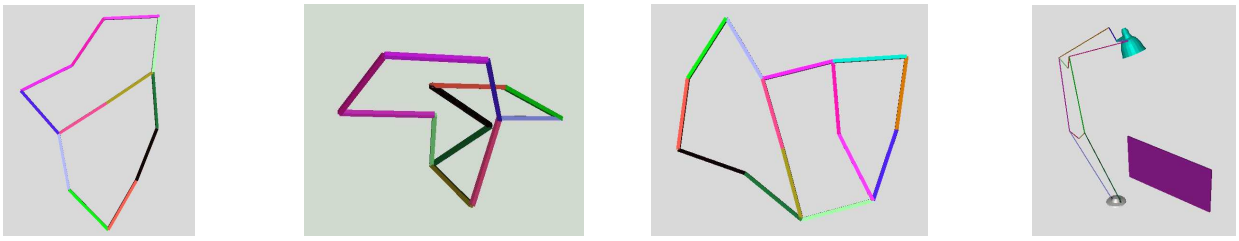
The general philosophy of IRC is first to relax the feasibility constraints so that the planner can solve an easier problem and then use the solution to the easier problem to help find a solution for the more constrained problem. In this case, the original closed chain is approximated by a *virtual closed chain* with fewer links, and the approximate solution is the path found for the smaller virtual closed chain.

In particular, the virtual chain is constructed by selecting some *pivot* joints of the original chain (Fig. 6(a)), and then connecting them with *virtual links* to create a *virtual closed chain*. A configuration of the original closed chain is determined by first computing a closed configuration of the virtual closed chain, and then computing closed configurations for the portions of the original chain between two consecutive pivots (whose positions are now fixed). If all these subproblems are small enough, then KBPRM can be used on them directly. Otherwise, another level of virtual links can be introduced. Fig. 6(b) shows a configuration for a 44-link single loop linkage generated by our planner.

VII. Experimental Results

In this section, we show how our planner performs in practice. All experimental results reported in this section were performed on a Pentium-4, 2.4GHz machine with 640MB memory.

We tested different examples in 3D environments for both planar and spatial linkages to show that our framework performs well for general linkages (Section VII-A). We also use our planner to simulate an adjustable lamp called Luxo (http://www.pixar.com/shorts/ljr/) (Section VII-B). In Section VII-C we show our results for high-dimensional problems using IRC.



(a) (b) (c) (d)

Fig. 5. Some closed configurations generated by our planner. (a) A 2loop-12link planar linkage. (b) A 2loop-12link spatial linkage. (c) A 3loop-14link planar linkage. (d) An adjustable lamp called "Luxo", which is made up by a 3-loop planar linkage and an adjustable head, the board is the obstacle in the environment.

TABLE I

TIMES (SECONDS) AND STATISTICS FOR ROADMAPS CONSTRUCTED WITH AND WITHOUT THE KINEMATIC ROADMAP

Roadmap Construction							
Chain	Kinematic Map			Generation		Connection	
Links	sec	cfg	CC	sec	cfg	sec	CC
P2-12	3.3	30	2	6.0	580	6.3	1
P2-12	–	–	–	53.6	580	34.8	12
S2-12	447.0	15	10	447.3	140	1.2	3
S2-12	–	–	–	6.3K	140	19.1	13
P3-14	8.59	30	2	15.57	1195	49.57	6
Luxo	–	–	–	170.3	1000	46.9	3

A. General linkages

We tested our planner for different examples in the "Walls" environment (Fig. 6(c)) for both planar and spatial linkages. The chains we consider in this section have k identical links and all joints are revolute. To study the benefit of using the kinematic roadmap, we compare roadmaps constructed with and without the two-stage approach.

In Table I, planar and spatial chains are labeled respectively with P and S followed by the number of loops and links, respectively, i.e., P2-12 stands for a 2loop-12link planar linkage, and cfg and CC denote the number of nodes and connected components in the roadmap (there are two), respectively. In planar and spatial 2loop-12link examples (Fig. 5(a) and (b)), when roadmaps were generated without kinematic preprocessing, the nodes were generated using Algorithm V.1; the base configuration was randomly generated and collision was checked with the obstacles in the environment. When roadmaps were generated with the two-stage approach, we first generated a small size kinematic roadmap, i.e., 30 nodes for the planar linkage and 15 nodes for the spatial linkage. Then, 20 different base configurations were generated for the kinematic roadmap and it was copied to the environment. Configurations with the same closure type were treated as rigid bodies and connected by a straight line planner. As can clearly be seen from the table, the roadmaps constructed using kinematic preprocessing are superior in all aspects: faster computation and improved roadmap quality (fewer connected components). Table I also shows the results for a 3loop-14link planar linkage (Fig. 5(c)) using the two-stage approach.

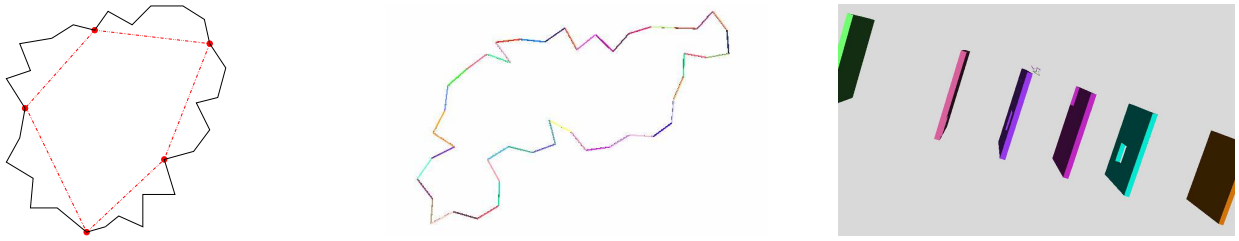
B. Luxo lamp

In this experiment, we use our planner to simulate a lamp called Luxo, which consists of a 3-loop linkage and an adjustable head (Fig. 5(d)). In this case, the base of the lamp is fixed and we don't need to use the two-stage approach. Furthermore, the underlying graph G of Luxo indicates the linkage is a compound chain linkage. We first 'break' it into a closed chain component which has three loops and an open chain linkage which only has one link. We applied KBPRM to the closed chain and randomly generated joint angles for the open chain. As shown in Table I, it takes 170.3 seconds to generate 1000 roadmap nodes and 46.9 second to connect them. The roadmap has 3 connected components and the largest one has 997 nodes. The query takes at most 1 second to find a path from a given start posture to a given end posture.

C. Using IRC for high dof chains

In the results presented here from our preliminary implementation, we have only applied IRC to the node generation phase of KBPRM[1]. That is, we did not actually find a solution path for the smaller linkage, but we generated closed configurations for it and used them to assist in the generation of configurations for the larger linkage.

We tested our planner for a single loop closed chain which has k identical links, $k = 20, 30, 44, 56, 74$ and 98, all joints are revolute. One level of IRC relaxation was used for $k = 20, 30$, and 44 and two levels of relaxation were used for $k = 56, 74$, and 98. In Table II, cfg and CC denote the number of nodes and connected components in the roadmap (there are two), respectively. In all these experiments: we first generated small number (20, 30 or 50) kinematic roadmap nodes, then 30 different base configurations were generated for the kinematic roadmap and it was copied to the environment. When the number of links ranges from 20 to 44, we chose pivots every other 6 or 7 joints. As we can see from the table, it is harder to generate a configuration as the number of links increases. When there are 44 links, it is very hard to connect nodes in the kinematic roadmap. When the number of links ranges from 56 to 98, a second level of relaxation was applied. In our experiments, we chose first level pivots every other 6 or 7 joints and second level pivots every other 3 or 4 first level pivots. Note that when the number of links is 56, the performance is better than the results for 44-link linkage even



(a) (b) (c)

Fig. 6. Applying IRC to closed chain. (a) The circle stands for the "pivots" and the dashed line represents the virtual links. All the virtual links form a 5-link closed chain while the original closed chain has 30 links. (b) A 44-link single loop configuration. (c) The "Walls" Environment.

TABLE II

TIMES (SECONDS) AND STATISTICS FOR SINGLE LOOP CLOSED CHAINS USING IRC

Roadmap Construction							
Chain Links	Kinematic Map			Generation		Connection	
	sec	cfg	CC	sec	cfg	sec	CC
20	6.38	50	4	12.59	957	165.35	7
30	19.67	50	3	29.73	966	289.09	11
44	65.02	30	13	70.41	671	344.67	8
56	21.01	20	4	25.12	564	170.99	8
74	165.8	30	4	175.11	882	574.83	13
98	238.08	30	6	251.42	821	947.04	15

though the dof is much higher. This is because we use two levels of relaxation for 56-link linkage and only one-level for the 44-link structure. We believe this is a good example of how iterative relaxation can improve performance.

VIII. CONCLUSION

KBPRM is one of the most efficient methods for motion planning for linkages containing closed kinematic linkages. The efficiency of KBPRM depends critically on how the linkage is partitioned into open chains, and the original method assumed the partition was provided as input to the problem. In this paper, we described a fully automated method for partitioning an arbitrary linkage into open chains, and for determining which chains should be positioned by random sampling and which should be positioned using inverse kinematics.

Even so, the size (number of links) of the closed loops that can be handled by KBPRM is limited because the inverse solver can only be applied to small chains. We showed that the Iterative Relaxation of Constraints (IRC) strategy [14] can be used to handle loops that are much larger than can be handled using KBPRM alone. Indeed, our preliminary implementation of IRC can generate closed configurations for loops with up to 98 links.

Acknowledgments

We would like to thank the Parasol Algorithms and Applications group, especially Burchan Bayazit for his help with IRC and Jyh-Ming Lien, Marco Morales, Guang Song and Shawna Thomas, for their help regarding the OBPRM software library and valuable discussions.

REFERENCES

[1] L. Han and N. M. Amato, "A kinematics-based probabilistic roadmap method for closed chain systems," in *Robotics:New Directions*. Natick, MA: A K Peters, 2000, pp. 233–246, book containts the proceedings of the International Workshop on the Algorithmic Foundations of Robotics (WAFR), Dartmouth, March 2000.

[2] D. Stewart, "A platform with six degrees of freedom," *Proc. of the Institute of Mechanical Engineering*, vol. 180, no. part I(5), pp. 171–186, 1954.

[3] J. P. Merlet, "Still a long way to go on the road for parallel mechanisms," in *Proc. ASME Int. Mech. Eng. Congress and Exhibition*, 2002.

[4] A. Singh, J. Latombe, and D. Brutlag, "A motion planning approach to flexible ligand binding," in *7th Int. Conf. on Intelligent Systems for Molecular Biology (ISMB)*, 1999, pp. 252–261.

[5] M. Kallmann, A. Aubel, T. Abaci, and D. Thalmann, "Planning collision-free reaching motion for interactive object manipulation and grasping," in *Eurographics*, 2003.

[6] K. Kotay, D. Rus, M. Vona, and C. McGray, "The self-reconfiguring robotic molecule: Design and control algorithms," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 1998, pp. 375–386.

[7] A. Nguyen, L. J. Guibas, and M. Yim, "Controlled module density helps reconfiguration planning," in *Proc. Int. Workshop on Algorithmic Foundations of Robotics (WAFR)*, 2000.

[8] O. Khatib, K. Yokoi, K. Chang, D. Ruspini, R. Holmberg, and A. Casal, "Vehicle/arm coordination and multiple mobile manipulator decentralized cooperation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1996, pp. 546–553.

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

[10] L. Kavraki, P. Svestka, J. C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Automat.*, vol. 12, no. 4, pp. 566–580, August 1996.

[11] S. LaValle, J. Yakey, and L. Kavraki, "A probabilistic roadmap approach for systems with closed kinematic chains," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 1671–1676.

[12] J. H. Yakey, S. M. LaValle, and L. E. Kavraki, "Randomized path planning for linkages with closed kinematic chains," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 951–958, 2001.

[13] S. M. LaValle and J. J. Kuffner, "Randomized kinodynamic planning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 1999, pp. 473–479.

[14] O. B. Bayazit, "Solving motion planning problems by iterative relaxation of constraints," Ph.D. Dissertation, Texas A&M University, May 2003.

[15] J. Cortes, T. Simeon, and J. P. Laumond, "A random loop generator for planning the motions of closed kinematic chains using PRM methods," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2002, pp. 2141–2146.

[16] S. Molian, *Mechanism Design: The Practical Kinematics and Dynamics of Machinery*, 2nd ed. Elsevier Science, Inc., 1997.

[17] J. Jaja, *In Introduction to Parallel Algorithms*, 2nd ed. Addison-Wesley Pub. Co., 1992.

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