Iterative Relaxation of Constraints: A Framework for Improving Automated Motion Planning*

O. Burchan Bayazit
Department of Computer Science and Engineering
Washington University in St. Louis
One Brookings Drive, Campus Box 1045, St. Louis, MO 63130-4899
bayazit@cse.wustl.edu

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

***Abstract*— This paper presents a technique for improving the efficiency of automated motion planners. Motion planning has application in many areas such as robotics, virtual reality systems, computer-aided design, and even computational biology. Although there have been steady advances in motion planning algorithms, especially in randomized approaches such as probabilistic roadmap methods (PRMs) or rapidly-exploring random trees (RRTs), there are still some classes of problems that cannot be solved efficiently using these state-of-the-art motion planners. In this paper, we suggest an iterative strategy addressing this problem where we first simplify the problem by relaxing some feasibility constraints, solve the easier version of the problem, and then use that solution to help us find a solution for the harder problem. We show how this strategy can be applied to rigid bodies and to linkages with high degrees of freedom, including both open and closed chain systems. Experimental results are presented for linkages composed of 9–98 links. Although we use PRMs as the automated planner, the framework is general and can be applied with other motion planning techniques as well.**

I. Introduction

Automatic motion planning has application in many areas such as robotics, computer animation, virtual reality systems, computer-aided design, computational biology and chemistry. Although many deterministic motion planning methods have been proposed, most are not used in practice because they are computationally infeasible except for some restricted cases, e.g., when the robot has very few degrees of freedom (dof) [19], [22]. Indeed, there is strong evidence that any complete planner (one that is guaranteed to find a solution or determine that none exists) requires time exponential in the number of dof of the robot [27].

For this reason, attention has focused on randomized approaches that are based on techniques for sampling points from the moveable object's configuration space (C-space). Representative examples include the *probabilistic roadmap methods* (PRMs) [1], [3], [18], [20], [26] and *rapidly-exploring random trees* (RRTs) [23]. In general, these methods can be quite effective for high dimensional problems when C-space is relatively uncluttered. Although these randomized approaches have solved many previously unsolved problems, they may fail in environments where the solution path requires the robot to pass through a narrow passage. This is because it is difficult for them to sample points in small volume corridors.

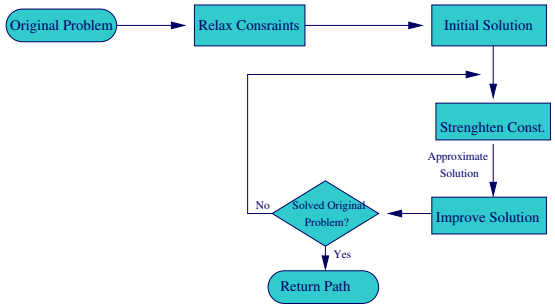


Fig. 1. Flow diagram of IRC. We first relax some constraints, solve the relaxed version of the problem and then use that solution to help solve the original problem.

In this work, we suggest a hierarchical strategy we call *Iterative Relaxation of Constraints (IRC)* for addressing the "narrow passage" problem. In IRC, we first relax some feasibility constraints, solve the relaxed version of the problem, and then use that solution as a guide to the solution of the original problem (see Figure 1). For example, we might first find a path that allows small collisions and then iteratively refine the path until it is collision-free. There are two benefits to this approach. First, in many cases relaxing the feasibility constraints increases the relative volume of the free C-space, making the relaxed version easier to solve than the original version. Second, the solution path for the relaxed problem can be used to identify a smaller region of C-space in which to focus the search for a solution to the original problem.

IRC is a general strategy that can be applied to any problem for which methods can be defined to (i) relax the feasibility constraints and for (ii) exploiting a solution to a relaxed version to focus the search for a solution to a less relaxed version. For example, in closed chain systems generating valid configurations using random sampling is a challenging problem whose difficulty increases with the dof [16], [31]. We overcome this difficulty with IRC by first reducing the number

*This research supported in part by NSF Grants ACI-9872126, EIA-9975018, EIA-0103742, EIA-9805823, ACR-0113971, CCR-0113974, EIA-9810937, EIA-0079874 and by the Texas Higher Education Coordinating Board grant ATP-000512-0261-2001. Some of this work was performed when Bayazit was with the Parasol Lab and his work was supported in part by the Turkish Ministry of Education.

of links (dof), finding a solution for the reduced-dof problem, and then using it to generate configurations for a system with more links. This is an important problem that arises when modeling proteins and other large molecules whose structures may contain (self) loops or when multiple molecules interact with each other, see, e.g., [29], [12], [24].

In the next section, we briefly cover some related work and in Section III we sketch the probabilistic roadmap (PRM) approach [21] which we use in our experiments. Section IV describes our IRC strategy. We discuss some applications of IRC to a traditional rigid body narrow passage problem and high dof open-chain and closed chain systems in Sections V. We present experimental results in Sections VI. Section VII concludes our paper.

II. Related Work

While IRC is first described in Bayazit's PhD dissertation [6], it grew out of and was inspired by our success in applying similar strategies to virtual prototyping [5], ligand binding [8] and motion planning for deformable objects [7].

We have recently learned about an approach called *Progressive Constraints (PC)* [13] that has a very similar philosophy to IRC. In particular, PC replaces the original problem with a series of progressively constrained problems whose freespaces decrease monotonically toward the freespace of the original problem and it uses a solution for one version as a heuristic for solving the next version. While PC and IRC have similar philosophies, the IRC framework is more general in the sense that it does not assume/require monotonic convergence of the freespaces of the different versions. The less rigid structure of IRC also provides for more flexibility in terms of feasibility constraints, relaxation methods, and techniques for improving the approximate solutions. Thus, in a sense, one can view IRC as a generalization of PC which exchanges completeness for the ability to apply it to a greater range of problems. Also, an approach similar to our rigid-body application was suggested in [4], where the robot size was changed to find a path. In [14], a sequential search framework that sequentially plan the motion of each link was proposed for manipulator arms, which can be viewed as a special case of relaxing the constraints for manipulator.

There have been some other PRM-based methods that bear some resemblance to the IRC strategy. Hsu et al. consider a "dilated" space [17]. They initially generate a roadmap in this space by allowing some penetration of the robot. Later, they push those nodes to the free space to generate a collision free roadmap. In contrast, we may find a path in the dilated space and then push the invalid configurations of the path to valid configurations. We believe our approach is more effective and efficient because it enables targeted exploration of C-space. In [10], a hierarchical sampling scheme that samples only as densely as the path clearance dictates was introduced. Our philosophy of initially finding an approximate solution also bears some similarity to Fuzzy PRM [25], Lazy PRM [9], or Customizable PRM [28] where the roadmap nodes and/or edges are not validated, or are only partially validated, during roadmap construction, and are validated completely only at query time.

Randomized approaches have also been applied to closed chain systems. In [31] gradient descent was used to obtain valid configurations from randomly generated, non-constrained configurations, which can be viewed as solutions with relaxed closure constraints. Han and Amato proposed a more efficient method, Kinematics-based PRM (KBPRM) [16], which uses both forward and inverse kinematics to guide the generation and connection of closed configurations. The KBPRM strategy has recently been extended to linkages with multiple loops [30]. Cortes et al. [11] propose 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. In this work, we handle high dof closed chain systems by applying IRC to KBPRM. Since closure constraints are intrinsic properties and the main difficulty for planning of closed chain systems, similar with [15], we focus our study on the first step of KBPRM which is to build a kinematic roadmap without considering environment obstacles. Also, an approach similar to our closed chain application was suggested in [29], where longer loop closure in proteins is solved by applying a shorter loop closure algorithm hierarchically.

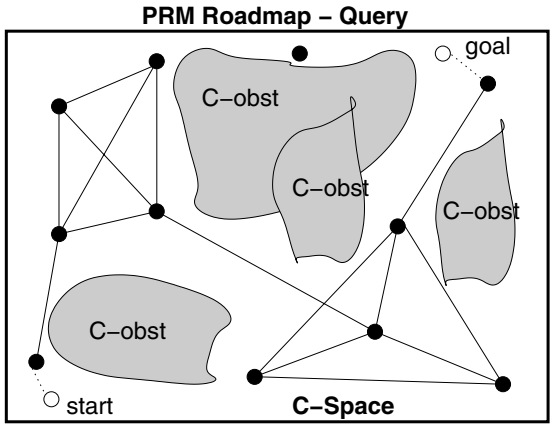


Fig. 2. Querying a PRM roadmap (C-space).

III. Probabilistic Roadmap Methods

While noting that our techniques could use any path planning algorithm, our current implementation is based on the probabilistic roadmap (PRM) approach to motion planning [21]. Briefly, PRMs work by sampling points 'randomly' from the robot's configuration space (C-space), and retaining those that satisfy certain feasibility requirements (e.g., they must correspond to collision-free configurations of the movable object). Then, these points are connected to form a graph, or roadmap, encoding representative feasible paths using some simple planning method to connect 'nearby' points. During query processing, the start and goal are connected to the

roadmap and a path is extracted from the roadmap using standard graph search techniques (see Figure 2).

In several of our experiments we use a PRM variant called Obstacle-Based PRM or OBPRM [2]. In OBPRM, nodes are generated on or near constraint surfaces instead of uniformly in C-space (e.g., contact configurations in traditional motion planning applications). As we will see, OBPRM is particularly well suited for the applications we study.

IV. Iterative Relaxation of Constraints

There are some important scenarios in which the critical feasible region of C-space is so small relative to the entire search space that global sampling approaches, such as PRMs cannot be effective. IRC (Iterative Relaxation of Constraints) is designed to address this situation. IRC initially solves a "relaxed" (easier) version of the problem and then iteratively refines the current solution by adding constraints until the solution meets the feasibility requirements of the original problem. Intuitively, the resulting search can be viewed as a hierarchical search of the C-space which focuses the search in the current iteration to a region of C-space localized around the previous solution.

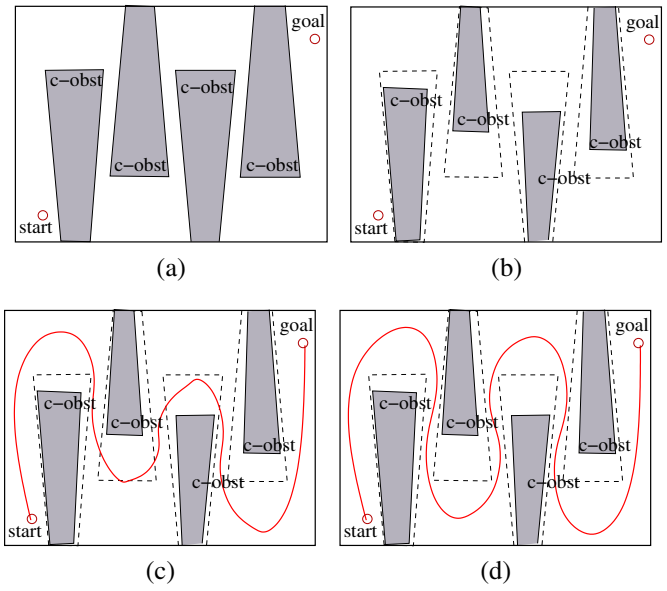


Fig. 3. Using an approximate (invalid) path to find a solution to the original problem. (a) Original C-space. (b) C-space after relaxing the constraints (reducing the size of the robot). (c) Approximate solution (invalid for the original problem). (d) Solution to the original problem after fixing the approximate path.

A. Algorithm

IRC is based on the principle that in many cases an approximate solution for a given problem can help us solve the original problem. For example, reducing the size of the robot (Figure 3(a)) would reduce the size of the C-space obstacles (Figure 3(b)). Although a solution in this new space may not be valid for the original problem (Figure 3(c)), we might be able to obtain a solution by fixing the invalid portions of the approximate solution. The potential for performance gains comes since we can restrict our exploration to the regions where the solution is difficult. That is, this technique assists us in identifying the easy and difficult parts of the problem and allows us to neglect some regions of the space in our search (Figure 3(d)).

By defining a hierarchical set of feasibility constraints, this approach yields an iterative process which starts with the least constrained problem and progressively tightens the constraints until the fully constrained problem is solved. That is, we first relax the feasibility constraints for a given problem and find a solution to this problem (*approximate solution*). Next, we check if this solution applies for the problem with strengthened constraints. If not, we improve the solution until it is valid for these new constraints (*improving solution*). This process of strengthening and improving the solution is repeated until a solution to the original problem is found (or the iteration limit is reached). A block diagram illustrating our approach is shown in Figure 1 and pseudo-code is provided in Algorithm IV.1.

Algorithm IV.1 Iterative Relaxation of Constraints

1: Identify and Relax feasibility constraints (set to minimal values)
2: Find a valid solution for the current feasibility constraints (an approximate solution)
3: **while** (problem $\neq$ original) and (#iters $<$ max iters) **do**
4: Strengthen feasibility constraints
5: Modify/Improve current solution until it is valid for current feasibility constraints
6: **end while**

Identifying and relaxing constraints (Step 1) and improving the solution for the strengthened constraints (Step 5) are the most important steps in Algorithm IV.1. These steps depend on the problem we are trying to solve. In Section V, different constraints and techniques for relaxing them are shown, as well as techniques to improve the solution for the strengthened constraints.

By relaxing a feasibility constraint, we hope to obtain an easier version of the problem whose solution, if invalid for the original problem, can be used to guide the automated planner in solving the original problem. For example, one common feasibility criterion in motion planning is that all configurations on the path should be collision-free. A relaxed version of this criterion might allow some penetration of the robot inside the obstacles [7]. This would have a similar effect as shrinking the robot or reducing the volume of the workspace obstacles, both of which would result in more free C-space and consequently a higher probability of generating nodes in narrow areas. Examples of other feasibility criterion that can be relaxed include the energy of a configuration [8], a heuristic collision detection routine [5], or the number of links in an articulated robot.

Once a feasible solution is found for the relaxed constraints, Step 4 of Algorithm IV.1 strengthens the feasibility constraints. In Step 5, we first check if the current path meets the feasibility criteria for the original problem; this is often done,

e.g., by verifying that each configuration on the path is valid. If the solution is not a valid solution for the original problem, then it will be refined. For example, one approach for this step might be to locally "push" the invalid path segments to the newly defined free C-space. We discuss techniques for this step in more detail in Section V.

Once an approximate solution is found, there is no guarantee that it can be improved to obtain a solution for a stronger version. An example of this case can be seen in the rigid body experiments in Section VI. Unfortunately as in the case with other probabilistic methods, there is no easy way to identify such cases, since to identify them we would need to know the exact shape of the C-space obstacles. However, careful examination of the problem can result in better selection of feasibility criteria, relaxation, and improvement methods. Furthermore, some heuristics can be employed to improve the performance of our algorithm.

V. Applications of IRC

We evaluate the IRC framework using three different types of applications: (i) a rigid body system, (ii) an open chain system, and (iii) a closed chain system. These applications were selected because each defines a difficult problem, requires different feasibility criteria, and may have different techniques to improve the approximate solution.

A. Rigid-body motion planning

The first scenario we consider is a free-flying polyhedral object in three-dimensional space which should pass through a narrow passage in the environment. The size of the robot affects the probability of finding a valid solution. Within the IRC framework, we could reduce the size of the robot (effectively making the problem easier) and find a solution to this version of the problem. A solution path for the original sized robot is found by repeatedly increasing the robot size and fixing any resulting problems in the path for the new robot size.

Feasibility criteria: The feasibility criteria is the depth the original robot is allowed to penetrate into an obstacle. The smaller the current robot, the larger the penetration permitted for the original robot.

Approximate Solution: The approximate solution is the path (i.e., sequence of configurations) for a smaller robot. This path can be found by any algorithm; we use OBPRM.

Improving Approximate Solution: Clearly, a valid path for a smaller version (relaxed constraints) of the robot may not be valid for a larger version (strengthened constraints) because some configurations in the path may be in collision with the environment. The technique used in the results reported here for improving the path is as follows: (i) identify invalid path configurations (in collision with the environment), (ii) replace each invalid path segment (one or more consecutive invalid configurations) with nearby valid configurations found by some local search method, e.g., sample around the invalid configurations, (iii) try to make connections that repair the damaged path segments, and if not possible, then try to generate additional configurations that can be used to do so. We achieve this by building a roadmap using only the configurations in the path or with additional configurations near the path.

The key advantage of this strategy is that in several representative problems, such as those studied in this paper, this concentrates the planner on promising regions for finding a solution.

B. Open chain systems

The second scenario we consider is an articulated robot without closed chains. In particular, we consider serial linkages of identical sized 3D polyhedral links, and we will decrease the difficulty of problem by removing links. Hence, our approach will be to find a solution for a robot with fewer links, and then gradually add links, fixing newly invalid configurations in the path, until a solution to the original problem is found.

Feasibility criteria: The number of links is the feasibility criteria. Removing links makes it easier to find a path.

Approximate Solution: The approximate solution is the path found using a robot with fewer links. Similar to the rigid body problem, we find our initial path (with the smallest number of links) using OBPRM. We found the initial solution by treating the first two links as a two-link (7 dof) robot.

Improving Approximate Solution: The approximate solution is for a robot with smaller dof. To improve it, we first need to transform it into a set of configurations compatible with the current robot's dof. We did this by maintaining the parameters for the existing dofs and finding values for the new (extra) dofs. There are many ways one could set these values, such as initializing them to zero, or assigning them the average of the other parameters. We found that assigning random values to new links worked well for the problem instances we studied, and avoided biasing the robot configurations. After the path is applicable to the current robot (using all its dof), we can apply the same strategy as in the rigid body case where the invalid path segments are replaced by nearby valid configurations.

C. Closed chain systems

The third scenario we consider is an articulated robot with closed chains. We use KBPRM [16], [30] to plan for the closed chain system. For each closed chain, we select some *pivot* joints in the original chain (See Figure4(d)), and we will decrease the difficulty of problem by fixing joint angles between two consecutive pivots. The pivot positions are arbitrarily selected over the actual joints. Intuitively, this is similar to connecting consecutive pivots with *virtual links* to create a *virtual closed chain*. The number of links in the virtual closed chain is smaller than the original one if the span (number of links between two consecutive pivots) is greater than 1. Our strategy to generate closed configurations is to first generate a number of random *seed* configurations and to then try to generate configurations similar to the seed configurations by rotating pivot joint angles. To generate a seed configuration, we first randomly generate joint angles

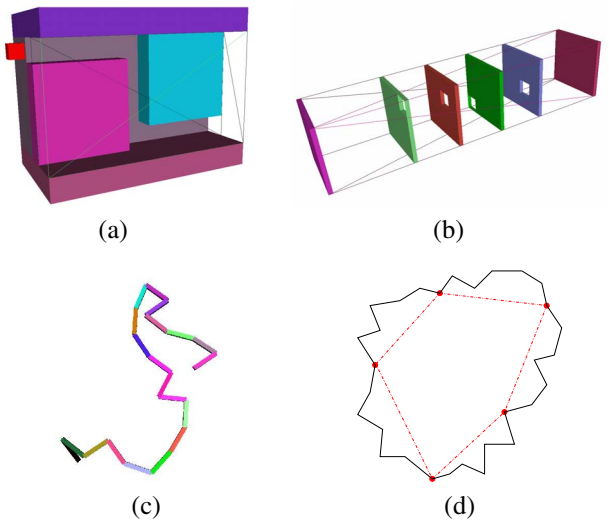


Fig. 4. Experimental environments. (a) S-shaped tunnel with the red cube as the robot. (b) Walls environment for open chain experiments. (c) Open chain robot with 26 dof. (d) An example of closed chain robot. The circle stands for the "pivot" and the dashed line represents the virtual link. The IRC simplification is shown as the polygon inside the chain.

between consecutive pivots to set the virtual links. A virtual link is accepted if its length is within a certain acceptable range. Then we randomly generate joint angles for pivots and try to use an inverse kinematics solver to obtain a closed configuration which will be used as a seed configuration.

Feasibility criteria: The number of links is the feasibility criteria. It is harder to generate a configuration as the number of links increases.

Approximate Solution: The approximate solution is the path found for the closed chain with fixed joint angles between consecutive pivots and it is generated using KBPRM.

Improving Approximate Solution: To expand the approximate solution, we need to generate joint angles between two consecutive pivots. Since the portion of the chain between two consecutive pivots and corresponding virtual link is itself a closed chain, another application of KBPRM can be used. We do this by sampling some joint angles between consecutive pivots and computing the remaining joint angles. If there are too many joints for KBPRM to be used effectively, another level of pivots can be introduced.

VI. Experiments

In our experiments, we compare the performance of the IRC framework with the performance of other PRM based techniques. For each of our experiments, we selected several versions of a specific environment, and evaluate the performance of PRM based techniques and IRC based techniques. For the rigid body environment, we have selected an obstacle that contains an s-shaped tunnel. The robot is a cube (see Figure 4(a)). To find a solution to the problem, the robot has to pass through the tunnel. In the open chain experiments, our environment is made up of 5 rooms, and each room is connected to adjacent regions by small passages (Figure 4(b)). The robot has a base that has 6 dof. Each extra link increases the dof of the system (Figure 4 (c)). In the closed chain experiments, we focus our study on the first step of KBPRM which is to build a kinematic roadmap without considering environment obstacles. We have used a similar robot representation, but in this case the robot has to satisfy the closure constraints as well (Figure 4(d)).

The rigid body and open chain experiments were run on a Pentium-4, 3GHz machine with 2GB memory. The closed chain experiments were run on a Pentium-4, 2.8GHz machine with 512MB memory. The movies of the experiments can be found at *http://www.cse.wustl.edu/ bayazit/index.php?page=irc*.

A. Rigid-body motion planning:

In the rigid body experiments, we keep the same obstacle in each environment but change the size of the robot. We have selected the smallest robot size as our base model (1.0x) and scaled it to 1.1x, 1.2x, 1.3x and 1.4x. IRC started with the path found by OBPRM for the base model. As the size of the robot increases, sampling in the tunnel becomes harder for a PRM based algorithm. As a result, OBPRM could not find solutions in a reasonable time if the robot's size was greater then 1.1x (Figure 5). Even with a small change in the robot's size, OBPRM's speed dramatically changes (Figure 5(a)). However, the running time for IRC stays nearly linear with the scale of the robot and is far faster than OBPRM. Since the IRC solution to a given robot requires the solution of a problem with the smaller versions of the robot first, the values for number nodes in the final roadmap and the running time to build that roadmap are cumulative in Figure 5. The efficiency of IRC can also be seen in the number of connected components in the roadmap. After the 1.2x version, IRC would generate fully connected roadmap (Figure 5(c)).

We have also unsuccessfully tried a 1.5x version. In this version, the robot is large enough to prevent any rotation inside the tunnel. Since the smaller versions of the robot can rotate inside the tunnel, the paths we found up to 1.4x contains rotations. That means there are some consecutive configurations in the path with different orientations. Since our current improving technique only replaces invalid path segment with nearby valid configurations or adds additional configurations, we were not able to improve a path for the 1.4x version to one for the 1.5x version. Similar problems may surface when the approximate solution passes through infeasible regions of the original problem, that is, when it may not be possible to transform the approximate solution into the original version. We can try to avoid such deadends by carefully selecting the approximate solution and the improving techniques. For example, in our case, instead of fixing the path, we could have found a new path using a PRM that is biased to generate configurations close to approximate path configurations. In fact a similar technique performed well in [5] to find a solution passing through a very tight area.

B. Open chain systems

In this environment, we have used OBPRM to find a path for a two-link (7 dof) robot as an approximate path. IRC started from this path and increased the dof until we found a path for a 26 dof robot. The results of this experiment can be seen in Figure 6. Once again the values for the running times and number of nodes are cumulative. We have stopped experimenting with OBPRM after 7 links since the planner becomes inefficient after that. This behavior can be explained with Figures 6(b) and (c). The number of nodes required by OBPRM to solve the problem is relatively small yet the time to find a solution increased dramatically. Figure 6(c) shows that the number of connected components increases in OBPRM as the number of links increases. So OBPRM spends most of its time connecting the separate components. A similar behavior for IRC is not seen. Since IRC concentrates on the broken parts of the path, it is able to quickly fix them. Note that, the time difference between two consecutive versions represents the time to improve the previous solution.

One interesting observation is that an approximate solution may require dramatic adjustment for harder versions. In our experiment, we have seen this behavior when trying to find a solution to the 14-link version. A significant adjustment of the path can be observed in Figure 6(a). Before the 14-link version, the improving time is almost linear. IRC spends more time improving the path of the 13-link version when trying to find a solution to the 14-link version. After the 14-link version, the improving time again becomes linear again. This suggests that the solution to the 13-link version is not easily applicable for the 14-link version. Note that another execution would likely generate different adjustment characteristics, i.e., there is nothing special about 13 or 14 links.

Another interesting observation is that, the number of connected components increases every two or three links (Figure 6(c)). We believe this could be explained as follows. Once an approximate path is corrected, i.e., a roadmap is built using that path and a new path is found on the roadmap, the new path would satisfy the feasibility constraints of the next few iterations, i.e., building a roadmap would be easier, resulting in fewer connected components. But after a few iterations, another adjustment of the path is required since the number of connected components increases (i.e., the connectivity of the roadmap reduces).

C. Closed chain systems

In the results presented here from our preliminary implementation, we have only applied IRC to the node generation phase of KBPRM[16], [30]. That is, we did not actually find a solution path for the simplified linkage, but we generated closed configurations for it and used them to assist in the generation of configurations for the larger linkage. Note that in closed chain systems, generating valid configurations itself is a challenging problem.

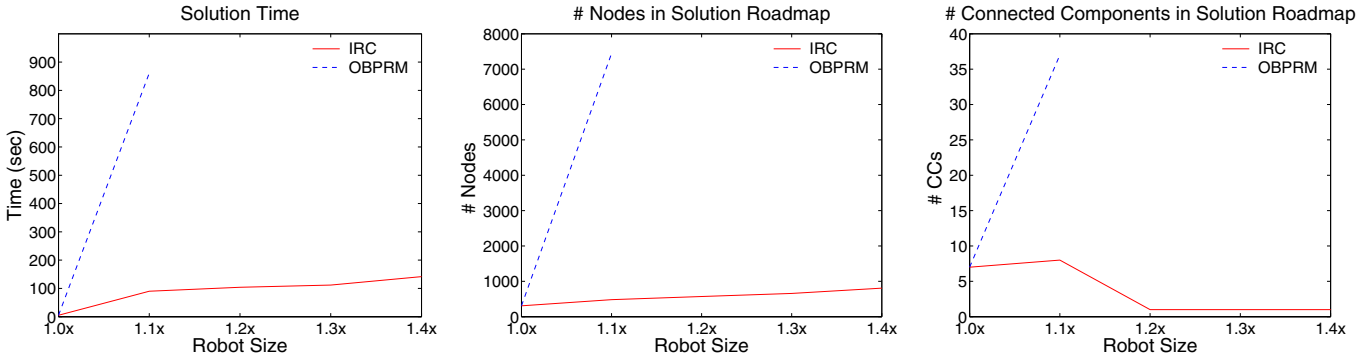
In all experiments, we tested our planner for a single loop planar closed chain which has m identical sized 3D polyhedral links, all joints are revolute. We implemented two sets of experiments for different purposes.

In the first set of experiments, we used KBPRM without IRC as the comparison method for IRC. We consider closed chains with 9 to 26 links, which is near the upper bound on the size that KBPRM can solve efficiently. We use the node generation time and the number of connected components of the resulting roadmap as criteria for comparison. The span (the number of links between two consecutive pivots) varies from 4 to 6 and we attempted to generate 100 nodes. As we can see from Figure 7, KBPRM's node generation is faster for chains with less than 14 links. This is because IRC does have some overhead. KBPRM and IRC have similar times for 14-link chains, and IRC is faster for chains with more than 14 links. In all cases, the IRC roadmaps are better, i.e., fewer connected components.

In the second set of experiments, we tested IRC on some higher-dimensional problems (20 to 98 links) where KBPRM is inefficient. When the number of links ranges from 20 to 44, there is only one level of pivots. When the number of links ranges from 74 to 98, a second level of pivots was introduced. In our experiments, we choose first level pivots every 6 to 7 joints and second level pivots every 4 first level pivots. In all experiments, we first try to generate 5 seed configurations, then we try to generate 10 similar C_1 configurations to each seed configuration by only adjusting angles for pivots. We then transform each C_1 configuration by sampling some joint angles between consecutive pivots and computing the rest joint angles. For each transformation, we set a maximum attempt number to 2000 and repeat this transformation 10 times to get different configurations from a C_1 configuration. In Table I, we show results for single loop k-link closed chains, Cfg and CC denote the average number of nodes and connected components in the roadmap, respectively. The results are averaged over 10 runs. Table I shows that it takes more time to generate a node as the number of links increases, there are two main reasons: (i) the dof itself increases as the number of links increases, (ii) we need to transform all joint angles between consecutive pivots and the total number of pivots increases as well. This also explains why we get fewer nodes when the number of links increases: as the number of links increase, it is more difficult to transform a configuration within 2000 attempts. An interesting observation is that, when the number of links is 74 and 98, the number of connected components is smaller than the results for 44-link even though the dof is much higher. This is because we use two levels of relaxation and we only adjust joint angles for second level pivots when we try to get C_1 nodes from a seed configuration.

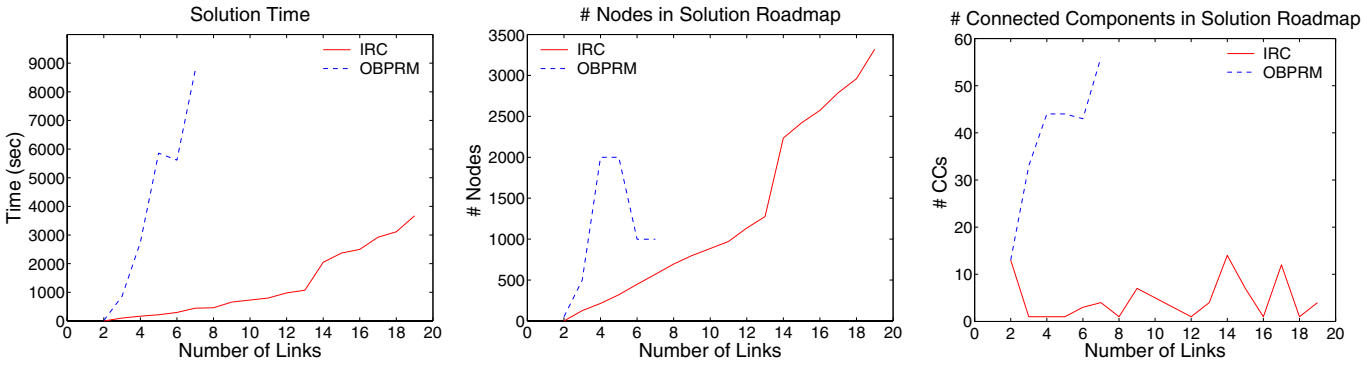
VII. Conclusion

In this paper, we presented a framework, IRC, to solve motion planning problems. This framework first solves a simplified version of the problem and uses the solution to guide a motion planner to solve the original problem. There may be multiple levels of simplification. In this case, the algorithm would solve the easiest problem and iteratively



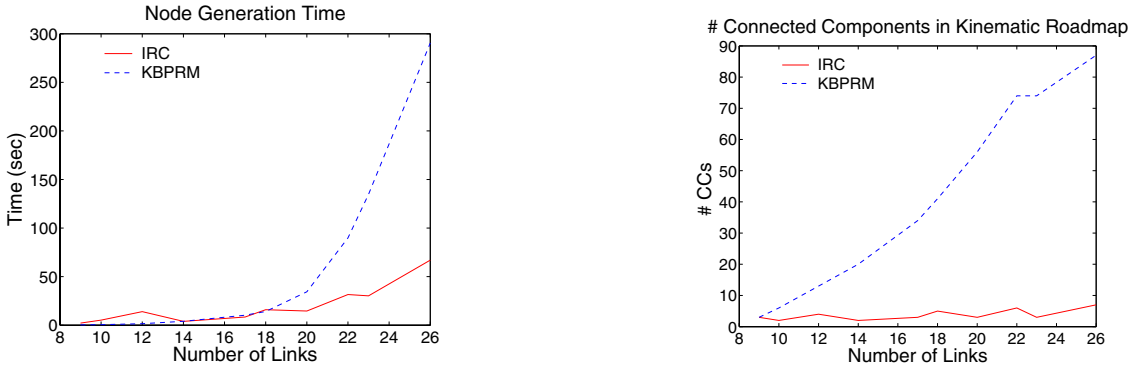
(a) (b) (c)

Fig. 5. Comparison of IRC vs. OBPRM in rigid body experiments. (a) Running time. (b) Minimum number of configurations required to solve the query. (c) The number of connected components in solution roadmap. Note that IRC values are cumulative for (a) and (b).



(a) (b) (c)

Fig. 6. Comparison of IRC vs. OBPRM in open chain experiments. (a) Running time. (b) Minimum number of configurations required to solve the query. (c) The number of connected components in solution roadmap. Note that IRC values are cumulative for (a) and (b).



(a) (b)

Fig. 7. Comparison of IRC vs. KBPRM in closed chain experiments. (a) Node generation time. (b) The number of connected components in kinematic roadmap.

TABLE I

KINEMATIC ROADMAP CONSTRUCTION TIMES (SECONDS) AND STATISTICS. RESULTS ARE AVERAGED OVER 10 RUNS.

Links	Cfg	time	CC
20	457.3	46.8	3.9
32	451.8	161.4	8.2
44	410.9	442.5	15.3
74	350.0	1148.6	6.5
98	258.0	3455.2	9.2

increase the hardness until it finds a solution to the original problem. The framework is applicable to any motion planning algorithm as long as methods are provided to simplify the problem and to transform a solution of the simplified version into a solution to the previous version. In our experiments, we have shown that IRC performs better than a PRM based algorithms on several types of problems.

In our current implementation, if we fail to improve an approximate solution, we discard all previous solutions and then re-start solving the problem with a different initial solution. A better approach would be implement some type of backtracking mechanism, i.e., re-relax the feasibility constraints and find another approximate solution. We plan to implement this approach in the future.

Acknowledgments

We would like to thank the robotics group in the Parasol lab at Texas A&M, especially Jyh-Ming Lien, Marco Morales, Guang Song and Shawna Thomas, for valuable discussions. We would also like to thank to Jean-Claude Latombe for bringing Progressive Constraints [13] to our attention.

REFERENCES

[1] J. M. Ahuactzin and K. Gupta. A motion planning based approach for inverse kinematics of redundant robots: The kinematic roadmap. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, volume 2, pages 3609–3614, 1997.

[2] N. M. Amato, O. B. Bayazit, L. K. Dale, C. V. Jones, and D. Vallejo. OBPRM: An obstacle-based PRM for 3D workspaces. In *Robotics: The Algorithmic Perspective*, pages 155–168, Natick, MA, 1998. A.K. Peters. Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics (WAFR), Houston, TX, 1998.

[3] N. M. Amato and Y. Wu. A randomized roadmap method for path and manipulation planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 113–120, 1996.

[4] B. Baginski. Local motion planning for manipulators based on shrinking and growing geometry models. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 3303–3308, 1996.

[5] O. B. Bayazit, G. Song, and N. M. Amato. Enhancing randomized motion planners: Exploring with haptic hints. *Autonomous Robots, Special Issue on Personal Robotics*, 10(2):163–174, 2001. Preliminary version appeared in *ICRA 2000*, pp. 529–536.

[6] O. Burchan Bayazit. *Solving Motion Planning Problems By Iterative Relaxation Of Constraints*. Ph.D. dissertation, Dept. of Computer Science, Texas A&M University, May 2003.

[7] O. Burchan Bayazit, Jyh-Ming Lien, and Nancy M. Amato. Probabilistic roadmap motion planning for deformable objects. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2126–2133, May 2002.

[8] O. Burchan Bayazit, Guang Song, and Nancy 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*.

[9] R. Bohlin and L. E. Kavraki. A randomized algorithm for robot path planning based on lazy evaluation. In P. Pardalos, S. Rajasekaran, and J. Rolim, editors, *Handbook on Randomized Computing*, pages 221–249. Kluwer Academic Publishers, 2001.

[10] A. D. Collins, P. K. Agarwal, and J. K. Harer. HPRM: A hierarchical PRM. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 4433–4438, 2003.

[11] 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)*, pages 2141–2146, 2002.

[12] J. Cortes, T. Simeon, M. Remaud-Simeon, and V. Tran. Geometric algorithms for the conformational analysis of long protein loops. *J. Computat. Chem.*, 25, 2004.

[13] P. Ferbach and J. Barraquand. A method for progressive constraints for manipulation planning. *IEEE Trans. Robot. Automat.*, 13(4), 1997.

[14] K. K. Gupta and Z. Guo. Motion planning for many degrees of freedom: Sequential search with backtracking. *IEEE Trans. Robot. Automat.*, 11(6):897–906, 1995.

[15] L. Han. Hybrid probabilistic roadmap - monte carlo motion planning for closed chain systems with spherical joints. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 920–926, 2004.

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

[17] D. Hsu, L. E. 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)*, pages 141–153, 1998.

[18] 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.

[19] Y. K. Hwang and N. Ahuja. Gross motion planning – a survey. *ACM Computing Surveys*, 24(3):219–291, 1992.

[20] L. Kavraki and J. C. Latombe. Randomized preprocessing of configuration space for fast path planning. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 2138–2145, 1994.

[21] L. E. Kavraki, P. Svestka, 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.

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

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

[24] Ming Lei, Maria I. Zavodszky, Lesllie A. Kuhn, and M. F. Thorpe. Sampling protein conformations and pathways. *J. Comput. Chem.*, 25:1133–1148, 2004.

[25] 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.

[26] M. Overmars. A random approach to path planning. Technical Report RUU-CS-92-32, Computer Science, Utrecht University, The Netherlands, 1992.

[27] J. H. Reif. Complexity of the mover's problem and generalizations. In *Proc. IEEE Symp. Foundations of Computer Science (FOCS)*, pages 421–427, San Juan, Puerto Rico, October 1979.

[28] G. Song, S. L. Miller, and N. M. Amato. Customizing PRM roadmaps at query time. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 1500–1505, 2001.

[29] W. J. Wedemeyer and H. A. Scheraga. Exact analytical loop closure in proteins using polynomial equations. *J. Comp. Chem.*, 20:819–844, 1999.

[30] D. Xie and N. M. Amato. A kinematics-based probabilistic roadmap method for high dof closed chain systems. In *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, pages 473–478, 2004.

[31] Jeffery H. Yakey, Steven M. LaValle, and Lydia E. Kavraki. Randomized path planning for linkages with closed kinematic chains. *IEEE Transactions on Robotics and Automation*, 17(6):951–958, 2001.