Factor-Guided Motion Planning for a Robot Arm

Jaesik Choi and Eyal Amir
Computer Science Department
University of Illinois at Urbana-Champaign
Urbana, IL 61801
{choi31,eyal}@cs.uiuc.edu

Abstract— **Motion planning for robotic arms is important for real, physical world applications. The planning for arms with high-degree-of-freedom (DOF) is hard because its search space is large (exponential in the number of joints), and the links may collide with static obstacles or other joints (self-collision). In this paper we present a motion planning algorithm that finds plans of motion from one arm configuration to a goal arm configuration in 2D space assuming no self-collision. Our algorithm is unique in two ways: (a) it utilizes the topology of the arm and obstacles to factor the search space and reduce the complexity of the planning problem using dynamic programming; and (b) it takes only polynomial time in the number of joints under some conditions. We provide a sufficient condition for polytime motion planning for 2D-space arms: if there is a path between two homotopic configurations, an embedded local planner finds a path within a polynomial time. The experimental results show that the proposed algorithm improves the performance of path planning for 2D arms.** [†]

I. Introduction

Robotic motion planning focuses on finding paths from one robot configuration to another. Robotic arms are particular robots that are made of connected links and joints (their degrees of freedom (DOF)). They are used for general-purpose manipulation of the work space, and planning with them is hard due to the high DOF [1], [2], [3], [4].

Current research on motion planning for robotic arms is limited in the number of joints that an arm can have in practice, especially with complex obstacles. Common approaches focus on mapping obstacles into the configuration space of an arm, and result in a search space of exponential size in the number of arm joints [8]. Recent progress in classical AI-planning suggests a different approach that focuses on factoring and dynamic programming [5], [6], [7]. This approach is promising because it can take advantage of the structure of the search space, and sometimes leads to polynomial-time planning.

In this paper we present a fast planning algorithm for a robotic arm in 2D. We focus on a manipulator that has revoluting joints connecting oval-like pegs, much like a human's shoulder, elbow, wrist, palm, and fingers. Joints of the arm are stacked sequentially, so that there is no self-collision. These simplifying assumptions keep the motion planning problem exponential still in the dimensionality of the space (the number of joints of the arm) [8], and they allow us to focus on the fundamentals of a factored planning approach for it.

Our path planning algorithm can scale to robotic arms with high DOF. It groups together configurations that share topological shape against obstacles, i.e., if they indicate the same end-point and smoothly deform into each other (we call those *homotopic configurations* and define them formally in Section II-D).

Our algorithm has three procedures: factoring and local planning. The planning problem (configuration space) is factored into subdomains whose elements share the topological shape. Then, our factoring algorithm finds a comprehensive set of plans between neighboring subdomains in a dynamic-programming fashion. An embedded planner (grid-based A* or sampling-based roadmap) finds local paths within each subdomain. We call the combined algorithm *Factor-Guided Motion Planning*.

In factoring, a dynamic program finds reachable locations from a subdomain. Each reachable location has an action which leads to the location with a specified reachable pivot point (acting joints) given a direction of movement (eg. left or right). The reachable configurations are found in order from the inner-most joint to the outer-most one. All the reachable configurations are found in polynomial time with dynamic programming, encoding homotopic one using a representative configuration (thus, we sidestep the possibly large (exponential) number of configurations).

In the second part of the algorithm, an embedded local planner is called a linear number of time to find a plan between two homotopic configurations. This procedure is done by any traditional motion planner (eg. grid-based A* planner or sampling-based roadmap planner). Thus, our local planner differs from local planners in motion planning literature (typically, straight-line planners in the configuration space) in that it looks for an arbitrary path between two homotopic configurations.

In practice, the combined algorithm can provide a path without self-collision, if we choose an appropriate local planner. There are many motion planners which find self-collision-free paths between two homotopic configurations. With the local planer, the combined algorithm finds a path with no self-collision, even though we assume self-collision in the factoring algorithm.

The bottleneck in our overall algorithm is the embedded

[†]Previous works *informally* appeared in workshops (5th International Workshop on Cognitive Robotics and AAAI 2006 Fall Symposium on Integrating Reasoning into Everyday Applications). However, the original work previously appeared in no official publication.

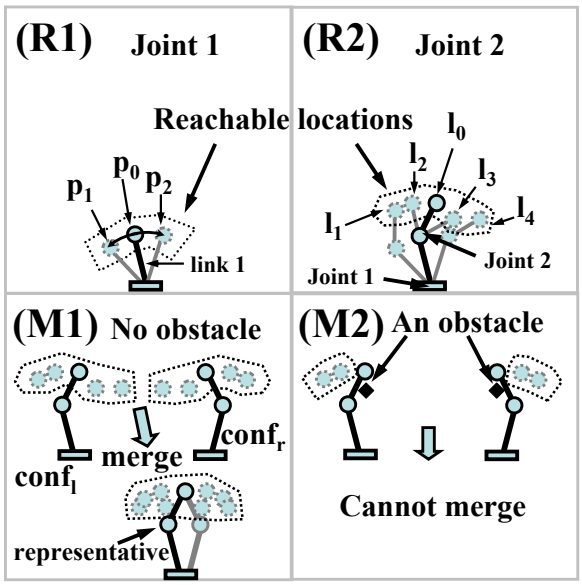

Fig. 1: (R1)(R2) shows the reachable locations with actions from an end-effector location. (M1)(M2) shows the condition of merging two sets of actions. (R1) represents a location which is reachable by a link and a joint. The two neighboring locations are reached by moving the joint 1 toward left or right. (R2) represents a position which is reachable by two links. Four neighboring positions (l_1, l_2, l_3, and l_4) are respectively lead by 4 actions: moving joint 1 toward the left (l_1); moving joint 2 toward the left (l_2); moving joint 2 toward the right (l_3); and moving joint 1 toward the right (l_4). (M1) shows the two set of configurations and their actions which can be merged into a group. (M2) shows the two separated set of configurations and their actions which cannot be merged because they have different topology due to an obstacle.

planner which may take exponential time in the worst case. In some cases, an embedded planner can find a path between homotopic configurations in polynomial time ($O(c^h)$) where c is the discretized angles of a joint and h is a constant. In those cases, the complexity of the algorithm is $O(m \cdot l^4 \cdot 2^n + l \cdot c^h)$ (m is the number of joints, n is the number of island (floating) obstacles, and l is the length of a longer side of workspace). Otherwise, the overall complexity still depends on the configuration space $O(m \cdot l^4 \cdot 2^n + l \cdot c^m)$.

Our experiments show that our method performs better than one of the most successful grid-decomposition methods [1]. It is also at least as good or compatible with grid-decomposition based planners and probabilistic roadmap based planners because we may use them as an embedded planner.

1) Related Work: Most previous motion-planning work is based on planning in configuration spaces (e.g., Potential Field [9], Cell Decomposition [10], and Roadmap [8], [3]). The configuration space of a robot arm is the set of all the possible configurations of all joints, thus this space is exponentially large in the number of joints of the arm. This state-space size makes planning infeasible for moderately complex arms (e.g., ≥ 10 joints).

Recent approaches try to overcome the state-space size problem. [11] groups close points in configuration space into a set of balls. The method is computed in configuration space, resulting in large computational burden. [12] solves

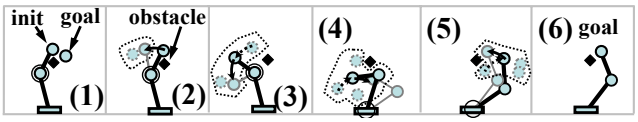

Fig. 2: Figures from (1) to (6) show the sequence of planning. The figures focus only on each step of the plan, although our algorithm finds all the possible pathes with breath-first search. (1) is the planning task from an initial configuration to a goal one. There is no direct path for the task because the obstacle splits two types of topologies of the arm (left sides and right sides). In each step (2)(3)(4)(5), our algorithm provides neighboring positions from the current position. The goal position is found after searching the 2D workspace which limits search space.

this problem by decomposing the workspace instead of the configuration space. They find a set of balls that can contain the entire robot, later find homotopic paths for the mobile robot in the workspace. Unfortunately, it is impossible to implement this method for a robotic arm that needs to manipulate elements in the complex environment. A grid-based planning algorithm [1] efficiently finds a solution for a robotic arm with a distance-based heuristic function. However, it may not find paths even when relatively simple ones exist.

Section 2 presents the intuition of our planning algorithm. Section 3 describes the details of our path algorithm. Section 4 analyzes the complexity and correctness of our algorithm. Section 5 and 6 provide experimental results and close with conclusions.

II. An Example and Problem Definition

This section shows an example of path planning for an arm. Figure 1 shows the steps to find all the possible actions for the arm given the environment. Figure 2 shows the steps to find a path from an initial configuration to a goal one. We also present the domain encoding used by our path planning algorithm. The input of our algorithm is a 2D chain of vertices (joints), edges (links), their initial positions, and a target location of the end effector. The only actions are rotations of joints.

A. A Motivating Example

We are interested in finding sets of plans that move the position of end-effector (eg. an action from p_0 to p_1 in Figure 1(R1)). The algorithm first finds the set of accessible positions (eg. p_0) for joint 1 and their neighboring positions (eg. p_1 and p_2). For each position, it marks the possible *entrance configurations*, any angle with which link 1 can approach from joint 1 to joint 2 at that position (in this case, the location of joint 1 is fixed in the base, so there is at most one such angle for every position).

The algorithm proceeds in a dynamic-programming fashion as follows. For each position for joint 2, we find the set of accessible positions. Each position of joint 2 (eg. l_0 in Figure 1(R2)) has reachable positions (eg. l_1, l_2, l_3, and l_4). Some (eg. l_2 and l_3) are reachable by moving the current joint. The others (eg. l_1 and l_4) are found by using the plans of the previous joint. For example, the l_0 moves to l_1, when

the p_0 in joint 1 moves to p_1. That is, a plan (from l_0 to l_1) is built by the plan of the joint 1 (from p_0 to p_1).

Our algorithm considers the topology of the arm. In Figure 1 (M1), a position has two *entrance configurations*: $conf_l$; and $conf_r$. If the two configurations have the same topology, it merges the two sets of plans and selects one representative (eg. $conf_l$). However, in Figure 1 (M2), it maintains the separated sets of plans, because their topologies are difference due to an obstacle in the middle.

Finally, for each possible position of joint 2 we find the set of accessible positions of joint 3. Every positions for joint 3 may have many entrance configurations as well, so we mark those configurations with a set of segments. Similarly, we proceed for joint 4 and 5.

Thus, we can factor the whole configuration space into the segments of configurations. Moreover, we have all the valid actions between the neighboring segments. Our algorithm finds a trajectory of endpoint by recursion (eg. a wavefront expansion method [13]) in Figure 2.

An embedded local planner finds local paths within each segments. We maintain a representative configuration for a segment and the set of actions. In theory, the size of possible configurations in a segment (*Closed Kinematic Chain*) can be large ($O(c^{n-2})$) (when c is a constant, and n is the number of joints). [1] Thus, the embedded local planner moves the joint to the pivot position, when a joint of the current configuration differs from the pivot point of the selected action.

B. State Definition in Propositional Logic

Here, we formally define a unit action, the Workspace (W) and the set of actions (Act). Our dynamic-programming uses these definitions to pass the set of actions from an inner joint to the next joint. In this section, we assume that there is no obstacle (we lift this assumption in Section II-C).

We present a unit action with only 3 parameters: a previous location of end-effector, a result location of end-effector, and a location of the pivot joint. For example, Figure 1(R1) demonstrates a unit action of joint 1 toward the right direction. (Here, we call the action $Move^2_{(P_0,P_1,Base)}$.) $Move^2_{(P_0,P_1,Base)}$ has two preconditions and a postcondition. The two preconditions are 'The end effector of the arm is at P_0' and 'The 1st joint is at *Base*'. The postcondition is 'The end effector of the arm is at P_1'.

The Workspace of the robot (W_{robot}) is the set of discretized locations that can be occupied by any joint of the arm in the 2D space. A *location* in the W_{robot} is represented by the $w = (x, y, \theta)$ ($x \in X$, $y \in Y$, and $\theta \in \Theta$, when X, Y, and Θ are the discretized x axis, y axis, and angular orientations respectively).

A_i is the set of all actions of the i_{th} joint. $A_i(w)$ is the set of actions having the same end-effector location $w \in W$. w_i and w'_i are locations of the i_{th} joint. w_j is a location of the j_{th} joint. An action ($Move^i_{(w_i,w'_i,w_j)}$) in $A_i(w_i)$ is defined in PDDL form as follows,

[1] In some cases, the planning within the segment is much easier than the planning in the whole configuration space.

$$Move^i_{(w_i,w'_i,w_j)} = \begin{cases} pre: & end_j(w_j) \wedge end_i(w_i) \\ del: & end_i(w_i) \\ add: & end_i(w'_i) \end{cases}$$

$end_x()$ is a predicate for the location of the x_{th} joint.

A complex action $Move^i_{(w_i,w'_i,w_j)} \in A_i(w_i)$ means that a unit left movement of the j_{th} joint at w_j changes the location of the i_{th} joint from w_i to w'_i. Based on the action of the i_{th} joint, the movement of the next $(i+1_{th})$ joint can be described as follows.

$$\begin{aligned} pre &: end_j(w_j) \wedge end_i(w_i) \wedge end_{i+1}(w_{i+1}) \\ del &: end_i(w_i), end_{i+1}(w_{i+1}) \\ add &: end_i(w'_i), end_{i+1}(w'_{i+1}) \end{aligned}$$

An action of the $i+1_{th}$ joint is built from a precondition ($end_{i+1}(w_{i+1})$) and an action ($Move^i_{(w_i,w'_i,w_j)}$). That is, $Move^i_{(w_i,w'_i,w_j)}$ not only changes the location of the i_{th} joint but also changes the location of the $i+1_{th}$ joint from w_{i+1} to w'_{i+1}. Fortunately, the new action $Move^{i+1}_{(w_{i+1},w'_{i+1},w_j)} \in A_{i+1}(w_{i+1})$ can be defined without the propositions of the i_{th} joint ($end_i()$), because an action requires only 3 conditions (one pivot point and two locations).

$$Move^{i+1}_{(w_{i+1},w'_{i+1},w_j)} = \begin{cases} pre: & end_j(w_j) \wedge end_{i+1}(w_{i+1}) \\ del: & end_{i+1}(w_{i+1}) \\ add: & end_{i+1}(w'_{i+1}) \end{cases}$$

C. End-effector Space

In general (with many obstacles), the simple workspace-based representation is not enough to build a correct set of actions. Figure 3 (Init 2) describes such a situation. The arm needs to detour an island (floating) obstacle to reach the goal position. That is the reason why most roadmap based approaches use the configuration space [14]. To solve the problem, we group the set of configurations that share the topology. A *representative configuration* represents the set of configurations. We call the set of *representative configurations* as *End-Effector Space* (ES).

The *End-effector Space* of the i_{th} joint (ES_i) is the set of pairs of a location (w) and its representative configuration ($\langle c_j \rangle_{j \le i}$); $es = (w, \langle c_j \rangle_{j \le i})$ when c_j is an angular configuration of the j_{th} joint. Each element is indexed by loc and $conf$, that is $es(loc) = w$ and $es(conf) = \langle c_j \rangle_{j \le i}$.

With the definition of ES, we represent an action ($Move^i_{(es(loc),w'_i,w_j)} \in A_i(es)$) that changes the location of the i_{th} joint from $es(loc)$ to w'_i by the movement of the j_{th} joint at w_j.

D. Homotopic Configurations

Two configurations ($Conf_1$ and $Conf_2$) are *Homotopic*, if they are smoothly deformed each other without moving endpoints. Formally, a homotopy between two configurations $Conf_1$ and $Conf_2$ is defined to be a continuous function $H : [0,1] \to Y$ from the unit interval $[0,1]$ to a topological space Y that includes all the possible configurations such that $H(0) = Conf_1$ and $H(1) = Conf_2$. We say that two configurations are homotopic if there is a homotopy between them and their endpoints are identical.

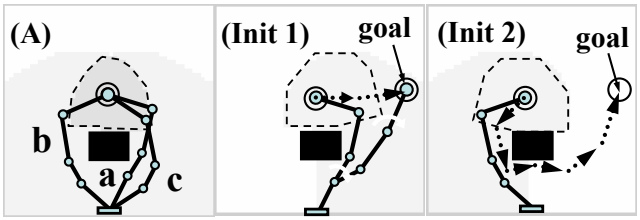

Fig. 3: (A) shows the Homotopic relationship among the configurations; '*a*', '*b*' and '*c*' are the configurations of the arm; '*a*' and '*b*' are not Homotopic configurations; '*a*' and '*c*' are Homotopic configurations. (Init 1) presents a path (a roadmap) to a goal position while (Init 2) presents another path to the same goal position. Our planner provides a correct path based on the current *entrance configuration*.

For example, in figure 3(A), two configurations 'a' and 'c' are *Homotopic*, and 'a' and 'b' are not *Homotopic*. The concept of *Homotopic Paths* is used in another approach [4]. [2] However, we use the *Homotopic* to group configurations while they use the *Homotopic* to group paths along which the center of a mobile robot moves.

III. Factoring and Planning for a Robotic Arm

In this section, we explain our planning algorithm and the way that we enable factoring and dynamic programming.

A. A Factor-Guided Motion Planning Algorithm

Algorithm: Factor-GuidedMotionPlanning

Input: the start configuration(es_{start}); the goal configuration(es_{goal}); number of steps in planning($depth$); the lengths of pegs($\{len_i\}_{i\leq m}$)
Output: Planned actions
$ES_0 \leftarrow \{(x_{base}, y_{base}, \theta_{base})\}$ /* A mounted base */
for $j \leftarrow 1$ **to** *m(m is the last joint)* **do**
  for $es \in ES_{j-1}$ **do**
    foreach *angle of the* j_{th} *joint* **do**
      $\langle es', act\rangle \leftarrow SinglePlan(es, ang, len_j, A_{j-1})$
      if $act \neq nil$ **then**
      foreach $es \in ES_j$ *when* $es(loc) = es'(loc)$ **do**
        if $es'(conf)$ *and* $es(conf)$ *are Homotopic* **then**
          $A_j(es) \leftarrow A_j(es) \bigcup act$
return $PathPlan(es_{start}, es_{goal}, A_m, ES_m, depth)$

Algorithm 1: *Factor-GuidedMotionPlanning*

Procedure *Factor-GuidedMotionPlanning* is presented in Algorithm 1 and its subroutines are presented in Algorithms 2 and 3. Our algorithm partitions the problem domain at each joint, and each partition precompiles all possible macro actions [3] for the joint and sends those macro action descriptions in PDDL to the adjacent outer joint. Algorithm *Factor-GuidedMotionPlanning* is given a start and a goal configuration, the search depth for planning, the lengths of links and obstacle information. It returns the sequence of actions that moves the arm to the goal location.

[2]Two paths having the same initial and goal configurations are homotopic if one can be continuously deformed into the other.

[3]We call these actions as macro actions because the actions may include a sequence of actions.

Algorithm: SinglePlan - Planning for a single joint

Input: an end-effector point(es); the orientation to the next joint(ang); the length of the $j-1_{th}$ peg(len_j); the actions of the j_{th} joint(A_{j-1})
$es'(loc) \leftarrow es(loc)$ + $trans(es(loc), ang, len_j)$ [4]
$es'(conf) = es(conf)$ + $\langle ang\rangle$, $act_j \leftarrow \emptyset$
foreach $act_{j-1} \in A_{j-1}(es)$ **do**
  Make a new act' from act_{j-1} (pivoting: joint k)
    pre: $end_k(w_k) \wedge end_j(es'(loc))$
    eff: $\neg end_j(es'(loc)) \wedge end_j(w'_j)$
  if act' *does not collide with any obstacle* **then**
    $act_j \leftarrow act_j \bigcup \{act'\}$
foreach *left and right move of the* j_{th} *joint* **do**
  Make a new act'' as follows
    pre: $end_{j-1}(es(loc)) \wedge end_i(es'(loc))$
    eff: $\neg end_j(es'(loc)) \wedge end_j(w''_j)$
  if act'' *does not collide with any obstacle* **then**
    $act_j \leftarrow act_j \bigcup \{act''\}$
return $\langle es', act\rangle$

Algorithm 2: *SinglePlan*

We discuss subroutines of *Factor-GuidedMotionPlanning* in detail. The *SinglePlan*[4] is described in Algorithm 2. It finds all the macro actions of a specific location of the current joint given all the accessible possible locations and actions of the previous joint. The returned macro actions include all the necessary rotations of internal joints as subroutines. The actions of the current joint are merged into an existing group (if there is a homotopic set) or into a new group (otherwise). The *PathPlan* in Algorithm 3 receives all the planned actions as input and returns the sequence of actions as output, if the goal configuration can be reached by less than $depth$ macro actions.

This method is similar to Factored Planning [5] in that the domain is split into sub domains and the subdomains send messages as a form of actions. However, our algorithm has important differences. The characteristic of robotics (a large amounts of shared fluents between consecutive joints) prevents us from using the general Factored Planning algorithm because the complexity of the algorithm (which depends on the number of messages of macro actions) is exponential in the number of shared fluents. This number increases with the number of joints in the naive use of Factored Planning. Thus, we modify the algorithm to reduce the sizes of messages.

Our contribution in this regard is that we reduce the number of messages using the characteristics of our robotic arm planning problem. With $|W|$ propositions, the number of potential messages is proportional to the number of possible pairs of preconditions and effects, i.e., $2^{2|W|}$. However, the complexity is reduced if we focus on the fact that the action requires one proposition ($end_j(w)$) in the precondition and another proposition ($end_j(w')$) in the effect. When the w and w' are locations of the j_{th} joint in the working space (W), the number of possible combinations of fluents is only $|W|^2$.

[4]$trans(...)$ returns the position of next joint given the location of the current joint, the angular orientation, and the length of peg

B. How Do We Make Factoring Possible?

We reduce the complexity of the planning problem by grouping partial plans based on the topology of the arm. Thus, we do not consider a specific configuration of the arm, if the topology of a configuration is identical to another one that already exists in our segments. This prevent us from storing the exponentially many specific configurations.

A *local planner* is used to find a path between the two *Homotopic Configurations*. In Algorithm *PathPlan*, actions are validated by a *local planner*. This can be achieved by any complete planner (eg. ARA*[1]). Thus, we can find a path between two Homotopic configurations, if there is a path.

Algorithm: PathPlan
Input: the start(es_{start}); the goal(es_{goal}); the actions(A); the end-effector space(ES); the depth of planning($depth$)
$j \leftarrow 1$, $R_0 \leftarrow \{es'_{start}\}$, $R_{total} = \emptyset$, $A' = \emptyset$
for $j \leftarrow 1$ **to** $depth$ **do**
  foreach $es \in R_{j-1}$, $act_{(m,es,w_i,...)} \in A(es)$ **do**
    if $act_{(m,es,w_i,...)}$ *is valid for es with a* local planner **then**
      $es' \leftarrow$ results from $act_{(m,es,w_i,...)}$
      $Act_{global} \leftarrow \{move_{(es,es')}\} \bigcup Act_{global}$
      $R_{total} \leftarrow R_{total} \bigcup \{es'\}$
Init(es_{start}, $done_0$, $\neg done_1$, ..., $\neg done_{depth}$)
Goal(es_{goal}, $done_0$, $done_1$, ..., $done_{depth}$)
Search for plans (Φ) in *Init*, *Goal*, Act_{global}
return Φ

Algorithm 3: *PathPlan* with a *local planner*

IV. The Analysis of The Algorithm

In this section, we analyze the complexity and correctness of the suggested algorithm. In the proofs, we assume two conditions: (1)the *local planner* can find a path from one configuration to another, if the two configurations are Homotopic and there is a path between the two; (2) the *local planner* terminates within a polynomial time in the number of joints ($O(m^h)$). These are the sufficient conditions for the polytime path-planning.

A. Complexity Analysis

Here, we analyze the complexity of our algorithm. We prove the complexity in the environment with n convex island obstacles. If we find the path for a m joint robotic arm with an exact path planning algorithm [9] [4], the complexity is $O(c^m)$ when c is a constant.

Theorem 1: The complexity of our algorithm is bounded by (When $|ES_m|$ is the size of the ES_m; $|W|$ is the size of workspace; l is the length of a larger side of workspace.)

$$O(m \cdot |W| \cdot |ES_m| + l \cdot m^h) = O(m \cdot l^2 \cdot |ES_m| + l \cdot m^h).$$

Environments	ARA* (sec)	Factored Motion Planning	
		Preprocess (Sec)	Planning (Sec)
(A)	> 580	19	83
(B)	78	12	5
(C)	17	12	2
(D)	4	27	3
(E)	>580	27	9

Fig. 5: Planning times for the different environment setting of Figure 4. In (A) and (E), ARA* finds no path within 580 secs and cannot continue due to lack of memory.

Proof: We have all the possible "move" actions between the axioms (positions). For each joint ($O(m)$), our algorithm finds all the reachable positions given a pivot point $O(|W|)$ and a point in ES $O(|ES_m|)$. Given the conditions, the possible movement is bounded by $O(2)$ because the pivot point only rotates toward the left direction or the right direction. The local planner (m^h) called at most ($O(l)$) times. ■

1) With 'n' Island Obstacles: The $|ES_m|$ with n islands is bounded by $O(2^n \cdot |W|)$.

Proof: Given an island obstacle, there are at most two distinct elements (es_{left} and es_{right}) for a location w. (when $es_{left}(loc) = es_{right}(loc)$ and $es_{left}(conf) \neq es_1(conf)$) For every island obstacles, the arm can be either left or right side (eg. $es_{left,right,...,right}$). Thus, the size of $|ES_m|$ is bounded by $|2^n|$.[5] ■

V. Experimental Result

We verify our algorithm by using the environment in Figure 4 (A)(B)(C)(D)(E). In each figure, the circle indicates the goal position. (A) has larger complexity (30 DOFs) than others (20DOFs). The difficulties of (B) and (C) are the narrow holes that the joints have to pass through. (D) and (E) have the same experimental settings (joints and obstacles) except goal positions. However, (E) is much more difficult than (D) because one needs to find the path that detours the island obstacle. In (E), many algorithms often fail to find solution if they use a simple distance-based heuristic function, or they use a naive reachability method without considering homotopic configurations.

We compare the performance of our algorithm with ARA* [1] which is one of the fastest grid-based motion planning algorithms. ARA* is fast algorithm for easy problems. However, it suffers from the curse of dimensionality for

[5]We assume that the arm rings around an island obstacle without a circular loop.

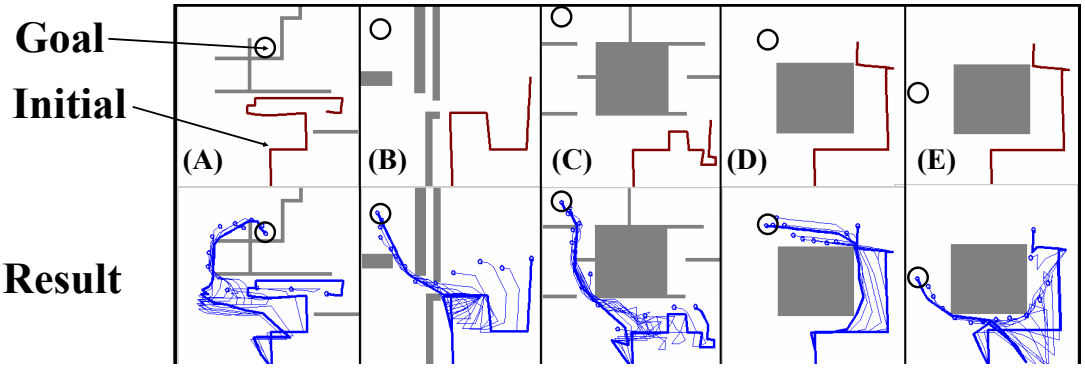

Fig. 4: (A),(B),(C),(D) and (E) are path planning problems. The arm in (A) has 30 joints (over 10^{36} states). The arms in (B),(C),(D) and (E) have 20 joints (over 10^{25} states).

global problems. [7] Thus, we use it as a local planner in our experiment. That is, our motion planner finds a global path (a roadmap) in the workspace. Then, the local steps in the global path is found by the local planner. We use an algorithm [15] to check the homotopy between two configurations. In Figure 5, our methods outperform the original ARA* method in all case, because it uses simple distance-based heuristic function. Our algorithm is significantly faster than ARA* and can plan effectively even when it fails to plan a path within reasonable time. However, our factor-guided motion planning needs to preprocess once for new environment. It respectively takes 19 secs, 12 secs, 12 secs, 27 secs and 27 secs to preprocess settings from (A) to (E).

VI. Conclusion

Three contributions of our work are (1) it reduces the complexity of planning problem by factoring the large configuration space into small segments with smaller sizes; (2) it takes only polynomial time in the number of joints under some conditions; and (3) it efficiently know the feasibility of the planning problem, when the goal is not reachable. Therefore, it is promising for inclusion of such algorithms in a larger-scale cognitive architecture. In the future work, the architecture for this arm may fit well with our decomposition approaches to AI [16], planning [5], and control [17].

VII. Acknowledgements

This material is based upon work supported by the National Science Foundation under Grant No. 05-46663

References

[1] M. Likhachev, G. Gordon, and S. Thrun, "ARA*: Anytime A* search with provable bounds on sub-optimality," in *NIPS'03*, S. Thrun, L. Saul, and B. Schölkopf, Eds. MIT Press, 2003.

[2] J. J. Kuffner and S. M. LaValle, "RRT-connect: An efficient approach to single-query path planning." in *ICRA'00*, 2000, pp. 995–1001.

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

[4] O. Brock and O. Khatib, "Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths." in *ICRA'00*, 2000, pp. 550–555.

[5] E. Amir and B. Engelhardt, "Factored planning," in *Proc. Eighteenth International Joint Conference on Artificial Intelligence (IJCAI '03)*. Morgan Kaufmann, 2003, pp. 929–935.

[6] R. Brafman and C. Domshlak, "Factored planning: How, when, and when not," in *Proc. National Conference on Artificial Intelligence (AAAI '06)*. MIT Press, 2006.

[7] E. Kelareva, O. Buffet, J. Huang, and S. Thiebaux, "Factored planning using decomposition trees," in *Proc. Twentieth International Joint Conference on Artificial Intelligence (IJCAI '07)*. Morgan Kaufmann, 2007.

[8] J. Canny, *The Complexity of Robot Motion Planning*. Cambridge, MA: MIT Press, 1987.

[9] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile manipulators," *Int. J. of Rob. Res.*, vol. 5, no. 1, pp. 90–98, 1986.

[10] J. T. Schwartz and M. Sharir, "On the Piano Movers' Problem: I." *Commu. on Pure and Applied Math.*, vol. 36, pp. 345–398, 1983.

[11] L. Yang and S. M. LaValle, "A framework for planning feedback motion strategies based on a random neighborhood graph." in *ICRA*. IEEE, 2000, pp. 544–549.

[12] O. Brock and L. Kavraki, "Decomposition-based motion planning: A framework for real-time motion planning in high-dimensional con guration spaces," in *ICRA'01*, 2001, pp. 1469–1474.

[13] J. Barraquand and J.-C. Latombe, "Robot motion planning: a distributed representation approach," *Int. J. Rob. Res.*, vol. 10, no. 6, pp. 628–649, 1991.

[14] R. Geraerts and M. H. Overmars, "Reachability analysis of sampling based planners," in *ICRA'05*, 2005, pp. 406–412.

[15] S. Cabello, Y. Liu, A. Mantler, and J. Snoeyink, "Testing homotopy for paths in the plane," in *SCG'02*. New York, NY, USA: ACM Press, 2002, pp. 160–169.

[16] S. A. McIlraith and E. Amir, "Theorem proving with structured theories." in *IJCAI'01*, 2001, pp. 624–634.

[17] E. Amir and P. Maynard-Zhang, "Logic-based subsumption architecture." *Artif. Intell.*, vol. 153, no. 1-2, pp. 167–237, 2004.

[7] Here, global problems are the complex problems that need to analyze the global environment setting such as locations and sizes of obstacles.