Factored Planning for Controlling a Robotic Arm

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

Abstract

Controlling robotic arms is important for real, physical world applications. Such control is hard because movement of one joint affects the position of many of the rest. In this paper we present an algorithm that finds plans of motion from one arm configuration to a goal arm configuration in 2D space. Our algorithm is unique in two ways: (a) It takes time that is only polynomial in the number of joints ($O(m^7 \cdot 2^n)(= O((2D\ Space)^3 \cdot m))$, with m joints and n island obstacles), thus allowing scaling up to complex arms; and (b) it decomposes the control problem to that of the separate joints, thus enabling future development of reactive modules. The algorithm leaves each joint somewhat independent of the rest by reformulating the domain description and re-partitioning it. Our algorithm is sound and complete given mild assumptions: it finds a plan, if there is one, and every returned plan leads to the goal. Also, it has bounded error with respect to the optimal path in the discretized environment. Our approach is promising because it leads naturally to a subsumption-architecture-like control of robotic arms.

1 Introduction

A robotic arm is a manipulator that has revoluting joints connecting oval-like pegs, much like a human's shoulder, elbow, wrist, palm, and fingers. The complexity of motion planning increases exponentially with the dimensionality of the space (the number of joints of the arm) (Canny 1987). Nonetheless, robotic arms are crucial for general purpose mobile robots, so much work has been invested in studying their controls (Kuffner. and LaValle 2000; Kavraki *et al.* 1996; Brock and Khatib 2000).

Unfortunately, current algorithms still depend exponentially on the dimensionality of the configuration space. This limits severely the number of joints that an arm can have in practice. In addition, the probabilistic approaches are not complete, and they provide a solution with probability that depends on the computation time spent. Most importantly, from the perspective of AI, current algorithms do not relate to a larger architecture for intelligent behavior.

In this paper we present a path planning algorithm that can scale to robotic arms with high degrees of freedom. We reduce the complexity of planning by applying independence assumptions to configurations of arm joints. Specifically the configurations of any joint is independent of other joints given the adjacent previous joint. We apply this to partition the planning problem into small subdomains associated with each joint. This enables scaling to arms of many joints.

Our approach is composed of two procedures: factoring and planning. We partition the problem manually into subdomains that correspond to each joint. Then, in each subdomain, we find a comprehensive set of plans of action and process a tree of subdomains in a dynamic programming fashion. This is done with a modified Factored Planning algorithm (Amir and Engelhardt 2003).

In the manually factored domain, the configuration space of our robotic arm is represented using distinctive groups of axioms in Propositional Logic (ground predicate calculus). On a joint, our algorithm finds valid actions with preconditions and effects in PDDL (McDermott 1998). The actions are grouped based on the locations of an end effector. If there are some obstacles that prevent the actions from being merged, the actions are split into several equivalence classes. Our algorithm finds all the possible PDDL actions by iterating from the inner-most joint to the outer-most one. Based on the achieved PDDL actions, a planner finds a path which is sound and complete [1].

We prove the complexity of our algorithm in various environments; without any obstacles, with 'n' convex islands, and with an infinite number of obstacles. The complexity of the algorithm without any obstacles is $O(m^7)$ (when m is the number of joints). Although the complexity of our algorithm increases exponentially with the number of convex islands, $O(m^7 \cdot 2^n)$, we also prove that the complexity of the algorithm is bounded by $O(m^5 \cdot c^m)$ (when n is the number of obstacles, and c is a constant). Thus, in many environments our algorithm is computationally superior to the previous approaches. Our algorithm is guaranteed to position the end effector in its target position. Furthermore, we give a condition that guarantees that the configuration error

[1] The returned paths of our algorithm are sound, because the paths always make the arm move to the goal location. The suggested algorithm is complete, because the algorithm returns a path if there is a path.

(distance from the requested overall configuration in some norm) in the planned path (with respect to the configuration in the optimal one) is bounded.

Finally, the architecture for this arm fits well with decomposition approaches to AI (McIlraith and Amir 2001), planning (Amir and Engelhardt 2003), and control (Amir and Maynard-Zhang 2004). This is promising for inclusion of such algorithms in a larger-scale cognitive architecture.

The previous works can be categorized into three groups; Potential Field (Khatib 1986), Cell Decomposition (Schwartz and Sharir 1983), and Roadmap (Canny 1987; Kavraki *et al.* 1996). They are based on the configuration space which is a set of all the possible configurations of all joints. Thus, the complexity of the methods for complete planning are exponentially proportional to the number of joints; the complexity of Potential Field is $O(c^m)$, the complexity of Cell Decomposition is $O(2^{2^m})$, and the complexity of Roadmap Method is $O(2^m)$ (when m is the number of joints).

The wavefront-expansion (Barraquand and Latombe 1991) is a complete algorithm, although the original Potential Field approach (Khatib 1986) is not complete. While Cell Decomposition approaches group the adjacent cells in the configuration space, we group the configurations of which the end effectors indicate the same location. Probabilistic Roadmap currently dominates the motion planning literature. Because it generates samples in the configuration space at random, the complexity is dramatically reduced. However, it is not complete without probabilistic assumption.

In the following sections, we define the states and actions, specify our planing algorithm, analyze the complexity and correctness of the algorithm, and provide the experimental results. In section 2, the PDDL actions and planning algorithm are presented. At the section 3, we suggested our path planning algorithm. In section 4, we analyze the complexity and correctness of our algorithm. In the following sections, we provide experimental results, discuss related works, and close with conclusions.

2 An Example and State Definition

In this section, we present the domain encoding used by our path planning algorithm. The input to our algorithm is a 2D chain of vertices (joints), edges (pegs) (e.g., see 1), their initial position, and a target location for the end effector. There, a state specifies the location of any joint of the arm. The only actions are rotations of joints.

A Motivating Example

Figure 1 shows an example of simple movements of an arm. The upper part of Figure 1 demonstrates the sequence of actions which move the arm from one position to another. The bottom part shows the corresponding actions that we represent here. For example, Figure 1(A) presents an action $RightJ3_{(W_0,W_1,P_3)}$(shown in Figure 1(a)) which means that the end effector of the arm moves from angular position W_0 to W_1 towards the $right$ direction by a 3rd joint($Joint_3$) at P_3.

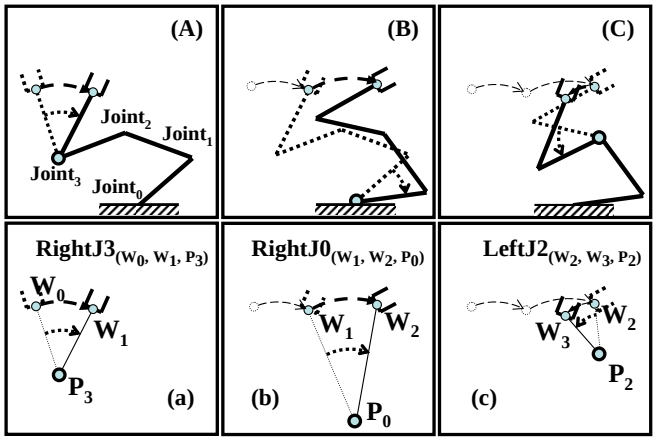

Figure 1: The upper part represents the sequence of movements of the arm. The three actions move the position of the arm to its target. The bottom part shows the corresponding actions that we represent here.

In this representation, each action has two preconditions and a postcondition. For example, the 1st action ($RightJ3_{(W_0,W_1,P_3)}$) has the following conditions. The two preconditions are 'The end effector of the arm is at W_0' and '3rd joint is at P_3'. The postcondition is 'The end effector of the arm is at W_1'. Intuitively, this action represents a right movement of 3rd joint which changes the location of end effector as a pivot point.

Our representations (Figure 1(a, b, c)) are simple and efficient compare to the configuration space representations (Figure 1(A, B, C)). Even though there are only 4 joints, the configurations space are complex to manage and memorize. However, our representation only care about the three locations (one pivot point and two locations of end effector) which are related to the action.

State Definition in Propositional Logic

Here, we formally define the Workspace (W) and the set of actions (Act). In this section, we assume that there is no obstacle (we lift this assumption in section 2.3). 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 workspace (W) 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). That is, the set of all the locations is $W_{robot} = \{(x, y, \theta) | (x \in X) \wedge (y \in T) \wedge (\theta \in \Theta)\}$ [2].

A_i is the set of actions of the i_{th} joint. $A_i(w_i)$ is the set of actions that are grouped by a location $w_i \in W$. An action ($Left_{(i,w_i,w_j)}$) of $A_i(w_i)$ is defined PDDL form as following,

[2]The assumptions of this research can be also adapted to the 3-dimensional workspace environment, $w = (x, y, z, \alpha, \beta, \gamma)$

$$Left_{(i,w_i,w_j)} = \begin{cases} pre: & end_j(w_j) \wedge end_i(w_i) \\ del: & end_i(w_i) \\ add: & end_i(w_i') \end{cases}$$

When $end_j()$ is the predicate for the location of j_{th} joint and w_i, w_i' and w_j are locations in W_{robot}.

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

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

An action of $i + 1_{th}$ joint can be described with the pre-condition (w_{i+1}) of the joint and an action ($Left_{(i,w_i,w_j)}$) of the previous (i_{th}) joint. That is, the movement of the j_{th} joint at w_j not only changes the location of i_{th} joint from w_i to w_i' but also changes the location of $i + 1_{th}$ joint from w_{i+1} to w_{i+1}'. However, the new action $Left_{(i+1,w_{i+1},w_j)} \in A_{i+1}(w_{i+1})$ can be defined without the propositions of i_{th} joint ($end_i()$), because an action require only 3 conditions (one pivot point and two locations).

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

End-effector Space

In the most general case (with many obstacles), the simple Workspace is not enough to represent the actions, because some configurations, which result in w_i, may not execute some of actions in $A_i(w_i)$. Thus, we need to group the set of configurations that can share their actions. We called the the set of configurations as a point es of ES (End-effector Space).

The End-effector Space of the i_{th} joint (ES_i) is the set of pairs of a location and its representative configuration; $es = (w, \langle c_j \rangle_{j \le i})$ when c_j is an angular configuration of the j_{th} joint (the direction of 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 End-effector Space, we represent an action ($Right_{(i,es(loc),w_j)} \in A_i(es)$) that changes the location of i_{th} joint from $w_i (= es(loc))$ to w_i' by the movement of j_{th} joint at w_j as following.

$$\begin{aligned} pre :& end_j(w_j) \wedge end_i(es(loc)) \\ del :& end_i(es(loc)) \\ add :& end_i(w_i') \end{aligned}$$

3 Factoring and Planning for a Robotic Arm

In this section, we explain our planing algorithm and the way that we make factoring possible.

A Modified Factored Planning Algorithm

Algorithm: RobotArmPlan
Input: the start configuration(es_{start}); the goal configuration(es_{goal}); number of steps in planning($depth$); the lengths of pegs($\{len_i\}_{i \le m}$)
Output: Planned actions
$ES_0 \leftarrow \{(x_{base}, y_{base}, \theta_{base})\}$ /* The mounted base */
for $j \leftarrow 1$ **to** *m(m is the last joint)* **do**
  for $es \in ES_{j-1}$ **do**
    foreach *angle(ang) of* j_{th} *joint* **do**
      $\langle es', act \rangle \leftarrow$ *SingleJointPlan*$(es, ang, len_j, A_{j-1})$ **if** $act \neq nil$ **then** *StorePartPlan*(es', act, ES_j, A_j)
$\langle path \rangle \leftarrow$ *PathPlan*$(es_{start}, es_{goal}, A_m, ES_m, depth)$
return $\langle path \rangle$

Algorithm 1: *RobotArmPlan*

Procedure *RobotArmPlan* is presented in Algorithm 1 and its subroutines are presented in Algorithms 2, 3, 4 and 5. Our overall algorithm partitions the problem domain at each joint, and each partition precompiles all possible macro actions for the joint and sends those macro action description in PDDL to the adjacent outer joint. Algorithm *RobotArmPlan* is given a start and a goal configuration, the search depth for planning, the lengths of pegs and obstacle information. It returns the sequence of actions that moves the arm to the goal location.

Algorithm: SingleJointPlan
Input: an end-effector point(es); the orientation to the next joint(ang); the length of the j_{th} peg(len_j); the actions of j_{th} joint(A_{j-1})
$es'(loc) \leftarrow es(loc) + trans(es(loc), ang, len_j)$ [3]
$es'(conf) = es(conf) + \langle ang \rangle$
$act_j \leftarrow \emptyset$
foreach $act_{j-1} \in A_{j-1}(es)$ **do**
  Make act_{new} ($end_i(w_i)$: a pre-condition of act_{j-1})
    pre: $end_i(w_i) \wedge end_j(es'(loc))$
    eff: $\neg end_j(es'(loc)) \wedge end_j(w_j')$
  if act_{new} *do not collide with any obstacle* **then**
    $act_j \leftarrow act_j \bigcup \{act_{new}\}$
foreach *left and right move of* j_{th} *joint* **do**
  Make act_{new} as following
    pre: $end_{j-1}(es(loc)) \wedge end_i(es'(loc))$
    eff: $\neg end_j(es'(loc)) \wedge end_j(w_j'')$
  if act_{new} *do not collide with any obstacle* **then**
    $act_j \leftarrow act_j \bigcup \{act_{new}\}$
return $\langle es', act \rangle$

Algorithm 2: *SingleJointPlan*

We discuss three of the subroutines of *RobotArmPlan* in detail. The first, *SingleJointPlan*[3], is shown at Algorithm

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

2. It finds all the macro actions (local plans) 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. Given the actions of a location of the current joint, *StorePartPlan* merges them to an existing group (if there is an identical location) or inserts them into a new group (if there is no identical location). The *PathPlan* 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 $depth$ actions.

This method is similar to Factored Planning (Amir and Engelhardt 2003) 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 size 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 $|W|^2$.

How do we make Factoring Possible?

We reduce the complexity of the planning problem by grouping partial plans based on the location of a given joint. Thus, we do not consider a specific configurations of joints of an arm, if the location of end effector is identical to another one that we already considered. This avoids storing the exponentially many specific configurations.

Algorithm: StorePartPlan
Input: an end-effector point(es'); the actions of the $es'(act)$; the j_{th} end-effector space of (ES_j); the actions of j_{th} joint(A_j)
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
$A_j(es') \leftarrow act$

Algorithm 3: *StorePartPlan*

We define the *Homotopic Configurations* for grouping sets of actions. Two configurations indicating the same endpoints are homotopic if one can be continuously deformed into the other. The concept of Homotopic Configurations is based on *Homotopic Paths* (Brock and Khatib 2000) [4]. For example, in Figure 2, the two configurations 'a' and 'c' are Homotopic and 'a' and 'b' are not Homotopic.

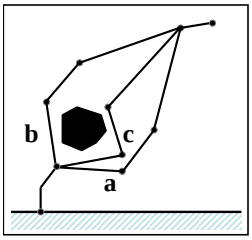

Figure 2: The Homotopic relationship among the configurations; 'a', 'b' and 'c' are the configurations of the arm; 'a' and 'b' are not Homotopic configurations; 'a' and 'c' are Homotopic configurations

A *local planner* is used to find a path between the two Homotopic Configurations. However, in the real environment, two Homotopic configurations are not always continuously deformable each other, due to the rigid body of link[5]. In Algorithm *PathPlan*, actions are validated by a *local planner*. This can be achieved by any inverse-kinematics algorithm which avoids obstacles. There are many local path planning algorithms for such a calculation (Zlajpah and Nemec 2002).

Algorithm: PathPlan
Input: the start(es_{start}); the goal(es_{goal}); the actions(A); the end-effector space(ES); the depth of planning($depth$)
$es'_{start} \leftarrow$ *FindES*(es_{start}, A, ES)
$es'_{goal} \leftarrow$ *FindES*(es_{goal}, A, ES)
$j \leftarrow 1$, $R_0 \leftarrow \{es'_{start}\}$, $R_{total} = \emptyset$, $A' = \emptyset$
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 **then**
      with a *local planner* $es_0 \leftarrow$ moved es by the $act_{(m,es,w_i,...)}$
      $es' \leftarrow$ *FindES*(es_0, A, ES)
      if $es' \notin R_{total}$ **then**
        Make new Action($move_{(es,es')}$)
          pre: $es \wedge done_{j-1} \wedge \neg done_j$
          eff: $es' \wedge done_j$
        $Act_{global} \leftarrow \{move_{(es,es')}\} \bigcup Act_{global}$
        $R_{total} \leftarrow R_{total} \bigcup \{es'\}$
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 4: *PathPlan* with a *local planner*

[4]Two paths with the same endpoints are homotopic if one can be continuously deformed into the other

[5]We check the Homotopic relationship with *local planner*. We allow the end point can be moved, if the movement can be managed by the *local planner*.

4 The Analysis of The Algorithm

In this section, we analyze the complexity and correctness of the suggested algorithm. In the proofs, we assume that the *local planner* could find a path from one configuration to another, if the two configurations are Homotopic and there is a path between the two.

Complexity Analysis

Here we analyze the complexity of the suggested algorithm. We prove the complexity of this algorithm in various environments; without obstacles; with a convex island obstacle; with 'n' convex island obstacles; and with infinite number of obstacles. If we find the path with an exact path planning algorithm (Khatib 1986) (Brock and Khatib 2000) in the dimensional space, the complexity is

$$O\left(\min\left(c^m, m^{depth}\right)\right)$$

Theorem 1.1: The complexity of our algorithm is bounded by (When $|ES_m|$ is the search space of the end effector.)

$$O(m \cdot |W|^2 \cdot |ES_m|) \ = O(m^5 \cdot |ES_m|)$$

We have $|W|(= cN^2 = \frac{2\pi}{\Delta\theta}(mL)^2)$ axioms representing the positions of the end effector of the robotic arm. Moreover, we have all the possible "move" actions between the axioms (positions). In Algorithm 4, we can find all the reachable position with t actions at the t_{th} step. If we execute further $depth$ steps, we can find all the possible paths whose lengths are shorter than $depth$. However the $depth$ is also bounded by $|W|$

Without Obstacles *Lemma 1.1:* Without any obstacles, the complexity of $RobotArmPlan$ is following given m links [6].

$$O(m \cdot |W|^2 \cdot |ES_m|) \ = O(m \cdot |W|^3) \ = O(m^7)$$

With Convex Islands *Lemma 1.2:* If there exists a convex island, the size of the end effector, $|ES_m|$, is bounded by $O(|ES_m|) = O(2 \cdot |W|)$.

That is, there are at most 2 distinct elements (es_0 and es_1) for a location w. (when $w \in W$, $es_0(loc) = es_1(loc) = w$, and $es_0(conf) \neq es_1(conf)$)

Lemma 1.3: The search space the end effector with n convex islands is bounded by $O(2^n \cdot |W|)$.

With an Infinite Number of Obstacles *Lemma 1.4:* If there is an infinite number of obstacles, the search space of the end effector, $|ES_m|$, is bounded by

$$O\left(\left(\frac{2\pi}{\Delta\theta}\right)^m\right) = O(c^m).$$

The Size of $|ES_m|$ *Theorem 1.2:* If there are n convex islands (obstacles), the search space of the end effector, $|ES_m|$, is bounded by

$$O\left(\min\left(2^n m^2, c^m\right)\right).$$

When n is the number of obstacles, c is $\frac{2\pi}{\Delta\theta}$, and m is the number of joints.

[6] We assume that the *local planner* terminates within a constant time.

Algorithm: FindES

Input: an end-effector point(es); the set of actions(Act); the end-effector space(ES)

foreach $es' \in ES$ *for* $es'(loc) = es(loc)$ **do**
  if $es'(conf)$ *and* $es(conf)$ *are Homotopic Configurations* **then**
    if local planner *find a path from* es' *to* es **then**
      return es
    $es'' \leftarrow es$ (es'': new point)
    $ES \leftarrow ES \bigcup \{es''\}$
    $Act(es'') \leftarrow Act(es)$
    return es''

Algorithm 5: *FindES*

Proof. by *Lemma 1.1, 1.2, 1.3, and 1.4* □

Soundness and Completeness

We prove this path planning algorithm is sound and complete. Our planning algorithm is sound, because all the returned paths are valid. Moreover, the planning algorithm is complete. If there is a path from the start to the goal, the algorithm finds a path that can reach to the goal.

Theorem 2.1:(Soundness of *RobotArmPlan*) When a path $\langle es_0, es_1, es_2, ..., es_n\rangle$ is returned by the algorithm, there is a path that passes through the configurations $\langle es_0(conf), es_1(conf), es_2(conf), ..., es_n(conf)\rangle$ with the robotic arm (when es_0 is es_{start} and es_n is es_{goal})

Lemma 2.1: With the robotic arm, we can find movements for every $move(es, es') \in Act_{global}$ in *PathPlan*

Proof. By the *Lemma 2.1*, we can find movements for any consecutive es points. For any es_{j-1} and es_j pair, we can find movements that control the robotic arm from an $es_{j-1}(conf)$ to an $es_j(conf)$. Thus, we can use mathematical induction for the whole path. □

Theorem 2.2:(Completeness of *RobotArmPlan*) If there exist a unique path whose number of movements is less than $depth$, our algorithm finds a path.

Proof. We use the mathematical induction to prove the completeness. For the 1_{st} step, we can simply find it because we have all the actions for each point ($es \in ES_m$). We have a complex action in $A(es)$, if there exists a unit movement from a location (es) to another (es_1). For the $n-1_{th}$ step, we assume that we can find a path if there exist $n-1$ movements from a location (es) to another (es_{n-1}).

For the n_{th} step, suppose that a path $\langle es_0, es_1, ..., es_{n-1}, es_n\rangle$ is unique from es_0 to es_n. Based on the mathematical induction, we already have a path from es_0 to es_{n-1}. Moreover, we have an action ($act_{(m,es_{n-1},...)}$) from es_{n-1} to es_n [7], because a unit movement of the arm really exists from es_{n-1} to es_n. Thus, the algorithm finds a path $\langle es_0, es_1, ..., es_{n-1}\rangle, act_{(m,es_{n-1},...)}, \langle es_n\rangle$. □

[7] Here, the action does not always guarantee the optimal path, because we don't store the actions of a specific configuration.

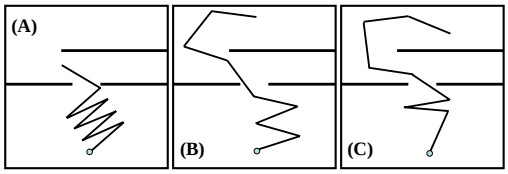

Figure 3: A path planning problem to measure the performance of the algorithm. (A) is the initial condition and (B) is the goal.

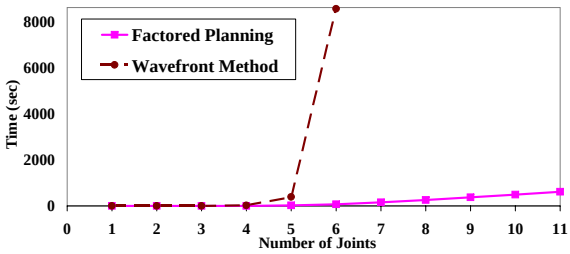

Figure 4: The planning time for the different number of joints.

Error Bound

Here, we examine the error of the suggested algorithm with respect to the discretized configuration space algorithm. [8] The error can be divided into two categories: (1) the displacement of location; and (2) the difference of angles.

The error of x and y is simply additive to the position of the last link, when cell size is small enough compared to the length of pegs. At each step, the maximum error of x axis (Δx_{pos}) increases $\frac{1}{2}|cell|$ at each joint when $|cell|$ is the size of a cell. Thus, the maximum error is $\frac{m}{2}|cell|$. Similarly, the maximum error of y axis (Δy_{pos}) is also $\frac{m}{2}|cell|$.

We analyze the angular error by supposing that $\Delta\theta$ is a unit angular displacement ($\frac{2\pi}{c}$) and c is the number of discretized angles. The angular error of the last joint with respect to the original configuration is bounded by $m\frac{\Delta\theta}{2}$ [9] because the maximum error of angle at each joint is bounded by $\frac{\Delta\theta}{2}$. Thus the maximum displacement of x (Δx_{ang}) is $N \sin(m \cdot \frac{\Delta\theta}{2})$ when N is the total length of joints.

Theorem 3: Given an optimal path of discretized configuration space algorithm, the path of *RobotArmPlan* has at most $\sqrt{2}r$ distance error, when we configuration that the $|cell|$ is less than $\frac{r}{m}$ and $\Delta\theta$ is less than $\frac{2}{m} \cdot \sin^{-1}(\frac{r}{N})$.

5 Experimental Result

We verify our algorithm by using the environment in Figure 3 (borrowed from (Latombe 1991)). The initial condition is Figure 3 (A). The goal condition is Figure 3 (C). The difficulty of this experiments is the narrow hole that the joints have to pass through.

We compare the performance of our algorithm with the Wavefront Method (Barraquand and Latombe 1991) which is one of the fastest complete algorithms. In the experiments, we change the number of joints that a robotic arm controls from 1 to 9. When the number of joints increases, the time for both planning algorithms increases shown in Figure 4. Our algorithm is significantly faster than the Wavefront Method and can plan effectively even when the Wavefront method fails.

6 Conclusion

Two contributions of our work are (1) an algorithm whose complexity is polynomial to the number of joints, and (2) the decomposition of the control problem into sub problems. When the optimal path approaches the goal location within finite steps ($depth$), the planning algorithm is sound and complete. It is only polynomial in the number of joints, although the complexity is exponential in the number of obstacles. Our theoretical results show that the planning algorithm scales well in planning the path of a robotic arm.

[8] We use the discretized configuration space in contrast to the continuous configuration space. Here we consider the uniformly split configuration space

[9] The worst case is occurred when the outermost link is very larger than the inner joints.

References

E. Amir and B. Engelhardt. Factored planning. In *IJCAI*, pages 929–935, 2003.

E. Amir and P. Maynard-Zhang. Logic-based subsumption architecture. *Artif. Intell.*, 153(1-2):167–237, 2004.

J. Barraquand and J.-C. Latombe. Robot motion planning: a distributed representation approach. *Int. J. Rob. Res.*, 10(6):628–649, 1991.

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

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

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.*, 12(4):566–580, 1996.

O. Khatib. Real-time obstacle avoidance for manipulators and mobile manipulators. *Int. J. of Rob. Res.*, 5(1):90–98, 1986.

J. J. Kuffner. and S. M. LaValle. RRT-connect: An efficient approach to single-query path planning. In *ICRA'00*, pages 995–1001, 2000.

J.-C. Latombe. *Robot Motion Planning*. Kluwer, Boston, MA, 1991.

D. McDermott. The planning domain definition language manual., 1998.

S. A. McIlraith and E. Amir. Theorem proving with structured theories. In *IJCAI'01*, pages 624–634, 2001.

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

L. Zlajpah and B. Nemec. Kinematic control algorithms for online obstacle avoidance for redundant manipulators. In *IROS'02*, pages 1898–1903, 2002.