configuration in the optimal one) is bounded.

The previous works can be categorized into three groups; Potential Field (Khatib 1986), Cell Decomposition (Schwartz & 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). Although the original Potential Field approach (Khatib 1986) is not complete, the wavefront-expansion (Barraquand & Latombe 1991) is a complete algorithm. 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 specify our algorithm, analyze the complexity, and prove an error bound. In section 2, the state definitions in propositional logic, the PDDL actions and planning algorithm are suggested. At the section 3, we analyze the complexity of our algorithm. In section 4, we prove the error bound of the suggested algorithm comparison to the previous approaches. We respectively state the related works and conclusion in the last 2 sections

2 Factored Planning for a Robotic Arm

In this section, we present the state definitions and a path planning algorithm for a robotic arm. Each subdomain of the robotic arm corresponds to a subset of the fluents and actions that are only related to a joint. A fluent represents the location of any joint of an arm. An action represents the movement of a joint. The fluents and actions are only shared by the adjacent joints.

An Illustrative Example

We want to start this section with an illustrative example. Figure 1 shows a simple movement of the joint. The left side of Figure 1 represents the Workspace of 2nd joint. Suppose that the location W_1 is $(x = 0, y = 0, \theta = \pi/2)$. That is, x, y, and θ are respectively 0, 0, and $\pi/2$. The location of W_2 is $(3, 3, \pi/4)$. When we move the 1st joint in W_1 toward the right, the location of 2nd joint changes from $W_2(3, 3, \pi/4)$ to $W_2'(4.1, 1.1, \pi/12)$. We call the movement an action of W_2 ($act_{(W_2,W_1,moveright)}$). Although there are many actions which are related to the 2nd joint and the location (W_2), we only consider the unit movement of any previous joints.

Our task one is relocating joint 3 from one location to another. Formally, our task is finding a plan for relocating the location of joint 3 from the start location to a goal one.

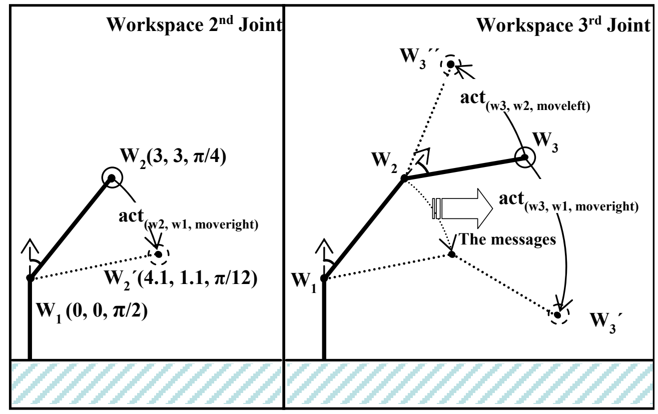

Figure 1: The left side of figure represents the Workspace of the 2nd joint. W_2 and W_2' are the locations of the 2nd joint. For example, x, y, and θ of W_2 are respectively 3, 4, and $\pi/4$. The $act_{(W_2,W_1,moveright)}$ is an action which moves the location of the 2nd joint from the $W_2(3, 4, \pi/4)$ to the $W_2(4.1, 2.1, \pi/12)$. The right side of figure represents the Workspace of the 3rd joint. W_3, W_3', and W_3'' are the locations of 3rd joint. The 2nd joint sends the message to the 3rd joint. An action ($act_{(W_3,W_1,moveright)}$) is constructed from the received action, $act_{(W_2,W_1,moveright)}$. Moreover, the 3rd joint makes a new action, $act_{(W_3,W_2,moveleft)}$.

Suppose that all the possible positions of joint 3 are W_3, W_3', and W_3''. A possible task is relocating joint 3 from W_3 to W_3', from W_3 to W_3'', from W_3' to W_3'', or the same in the opposite direction.

We find a plan of movement for the joints using the following dynamic programming algorithm. We process each joint separately as follows, starting from joint 1, and proceeding to joint 2 and finally 3. Processing each joint involves running a planner on all possible input starting states and all possible goal conditions. It records those input and goal conditions that have a valid plan, and transfers them all as new macro actions to the subsequent joint. For joint 1, we compute all the possible positions of joint 2 (W_2, W_2') that joint 1 movements can entail. Then we record all those positions as new macro actions that joint 2 can request from joint 1. The planning problem for joint 2 becomes finding all the positions that it can entail for joint 3, given its original actions and its (new) macro actions.

We describe this algorithm more precisely now. Each joint has an associated subdomain and messages that it computes and sends to its parent joint subdomain. A message from joint i includes actions that are constructed by subdomain i and i's child subdomains. All messages are of the form "if X_0 holds, then actions of the 2nd joint can make X_1 hold". In this example, the 2nd joint has a message "if the first joint is on $W_1(0, 0, \pi/2)$ and the 2nd joint is on $W_2(3, 3, \pi/4)$, then an action($act_{(W_2,W_1,moveright)}$) can make the 2nd joint be on $W_2'(4.1, 1.1, \pi/12)$". This message is sent to the 3rd joint. The 3rd joint can convert the message into an action, because the 3rd joint has information for manipulating the shared fluents. From the message, the 3rd joint constructs a new action "if the first

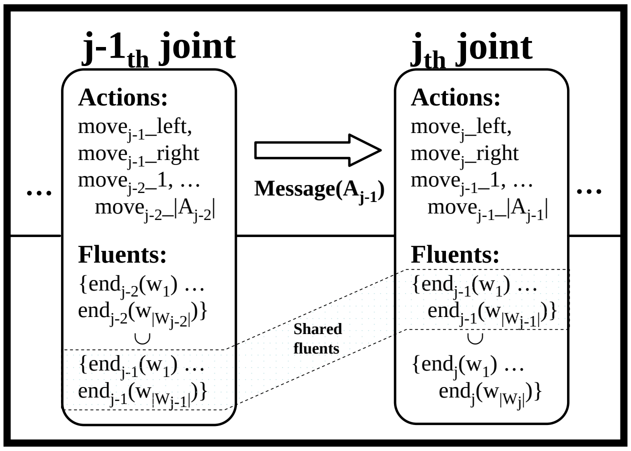

Figure 2: Subdomains of controlling a robotic arm; Each joint have actions that includes its movement ($move_{j-1}\_left$ and $move_{j-1}\_right$) and the movement of previous joints ($move_{j-2}{}_1$ ... $move_{j-2}{}_{|A_{j-2}|}$); After the actions are converted to the A_{j-1}, the set of actions are sent to j_{th} joint as a message.

joint is on $W_1(0, 0, \pi/2)$ and the 3rd joint is on W_3, an action($act_{(W_3,W_1,moveright)}$) can make the 3rd joint be on W_3'".

The right side of Figure 1 represents the Workspace of the 3rd joint. Because the 2nd joint sends the messages (the set of actions which are related to the 3rd joint) to the 3rd joint, the 3rd joint can construct some actions from the received actions. For example, $act_{(W_3,W_1,moveright)}$ is constructed from a received action ($act_{(W_2,W_1,moveright)}$). Moreover, the 3rd joint makes new actions which are caused by the movement of the 2nd joint. For example, $act_{(W_3,W_2,moveleft)}$ is a new action of the 3rd joint.

In the general case, we decompose the problem of controlling a robotic arm into the subdomains in Figure 2. Each joint (eg. j_{th}) receives the messages from the previous joint(eg. $j-1_{th}$) and sends the messages to the next joint (eg. $j+1_{th}$).

State Definition in Propositional Logic

Here, we formally define the Workspace (W), End-effector Space (ES) and the set of actions (Act). In the simple case (without any obstacles), we can represent the state only with the Workspace and the set of actions without the End-effector Space. However, in the most general case (with many obstacles), the End-effector Space is also required for representing the complex environment, because a location of Workspace is divided into multiple points of End-effector Space by the obstacles.

The Workspace of the robot (W_{robot}) is the set of discretized locations that can be occupied by any joint of the arm. To simplify, we assume a 2-dimensional workspace. 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 is the discretized x axis, Y is the discretized y axis, and Θ is the discretized angular orientations). 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].

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 $es(conf) = \langle c_j \rangle_{j \le i}$.

A_i is the set of actions of i_{th} subdomain. $A_i(es)$ is the set of actions that are grouped by an element $es \in ES_i$. An action ($act_{(i,es,w_j,moveleft)}$) of $A_i(es)$ is defined with PDDL form as following,

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

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

A complex action ($act_{(i,es,w_j,moveleft)} \in A_i(es)$) means that a unit left movement of the j_{th} joint at w_j changes the location of i_{th} joint from $es(loc)$ 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.

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

The location of $i+1_{th}$ joint can be described with the previous location of the joint ($es'(loc)$) and the $act_{(i,es,w_j,moveleft)}$ of 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 $es(loc)$ to w_i' but also changes the location of $i+1_{th}$ joint from $es'(loc)$ to w_{i+1}'. Now, the new action $act_{(i+1,es',w_j,moveleft)} \in A_{i+1}(es')$ can be defined without $end_i()$.

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

Figure 3 conceptually explains the reason why two joints are independent. Suppose that we make an action 'moveleft' of j_{th} joint located in w_j. The moveleft of j_{th} joint at w_j with configuration 'a' moves the location of i_{th} joint from w_i to w_i'. The movement of j_{th} joint at w_j with configuration 'b' also moves the location of i_{th} joint from w_i to w_i'. Here, we want to note that the specific configuration of the arm is irrelevant given the location of j_{th} joint (w_j) and the location of i_{th} joint (w_i) of an action, $act_{(i,es,w_j,moveleft)}$ and $es(loc) = w_i$.

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

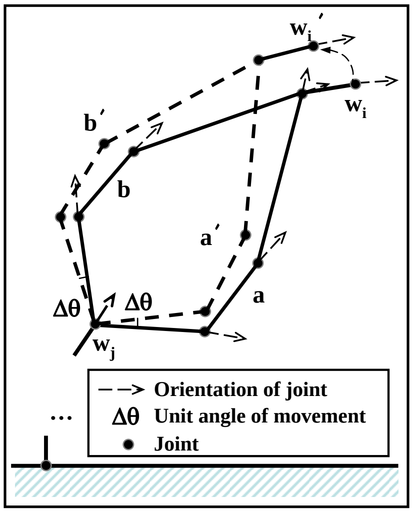

Figure 3: The 'moveleft' of a joint; w_j. w_i, w_i' are the locations of the i_{th} joint; w_j is the location of the j_{th} joint; $\Delta\Theta$ is the unit angle of the left movement; the $act(i, es, w_j, moveleft)$ with $es(loc) = w_i$ respectively changes the configurations from 'a' to 'a'' and from 'b' to 'b''

Modified Factored Planning for the Robotic Arm

Procedure *RobotArmPlan* is presented in Algorithm 1 and its subroutines are presented in Algorithms 2 [3], 3, 4 and 5. This path planning algorithm is based on the Factored Planning (Amir & Engelhardt 2003) which sends messages between the partitioned domains and has no back-tracking in searching for a plan. However, a characteristic of robotics (a large amounts of shared fluents between consecutive joints) prevents us using the Factored Planning algorithm, because the complexity of the algorithm is exponentially proportional to the number of shared fluents. Thus, we modify the algorithm to reduce the size of messages. For the same reason, we make a few assumptions, setting both the interaction (k) and the depth (d) are equal to 1 [4].

We reduce the number of messages with the domain knowledge. In the general-purpose factored planner, sub domains have to find the actions for all the possible combinations of preconditions and effects of shared fluents. Thus, the number of messages is exponentially proportional to the number of shared fluents. However, in this special domain, it is enough to search the actions that are the subset of the combinations of shared fluents, because the actions of the domain is limited. Here, we need only a following type of actions, although all the possible locations of j_{th} joint ($end_j(w)$) are shared by the j_{th} joint and $j+1_{th}$ joint.

[3] $trans(w_{j-1}, ang, length_j)$ returns the position of next (j_{th}) joint given the location of $j-1_{th}$ joint (w_{j-1}) and the angle (ang) and the length of j_{th} link

[4] The interaction (k) is the maximum interactions between two subdomains. The depth (d) is the searching depth of planning

Algorithm 1 RobotArmPlan
es_{start}: the initial location and its configuration
es_{goal}: the goal location and its configuration
$depth$: the maximum depth for global path planning
$\{length_i\}_{i \leq m}$: the length of each link. *RobotArmPlan* iterates from the innermost joint to the outermost joint.

PROCEDURE *RobotArmPlan* (es_{start}, es_{goal}, $depth$, $\{length_i\}_{i \leq m}$)
1. Insert $\{(0,0,0), \{\}\}$ into ES_0, $j \leftarrow 1$
2. Do until j = m (m is the last joint)
 (a) For each $es \in ES_{j-1}$
  i. For each angle, ang, of j_{th} joint
   $\langle es', act\rangle \leftarrow$ *SingleJointPlan*$(es, ang, length_j, A_{j-1})$
   If $act \neq nil$,
    StorePartPlan(es', act, ES_j, A_j)
 (b) j ← j+1
3. $\langle path\rangle \leftarrow$ *PathPlan*$(es_{start}, es_{goal}, A_m, ES_m, depth)$
4. Return path

$$pre: end_j(w)$$
$$eff: \neg end_j(w) \wedge end_j(w')$$

In the general case, all the possible pairs of preconditions and effects are $2^{2|W|}$, because the any location (w) in W can be TRUE of FALSE. However, the complexity is reduced, if we focus on the fact that the action requires one proposition ($end_j(w)$) in precondition and another proposition ($end_j(w')$) in effect. When the w and w' are locations of j_{th} joint in the working space (W), all the possible combination of fluents is only $|W|^2$.

The Complexity of this Algorithm

The Factored Planning algorithm (Amir & Engelhardt 2003) is sound and complete, given the subdomains and certain parameters, k (interactions) and d (depth). The algorithm terminates at time $O\left(m \cdot 2^{2k+l} \cdot min((a+k)^d, k \cdot 2^v)\right)$ with parameters m, a, and v (when m represents the number of subdomains; a is the largest number of action symbols; v is the largest number of fluent symbols in any subdomains; and l is the largest number of shared symbols between the two subdomains.).

The complexity of our modified Factored Planning algorithm is reduced as following, because we need only the part of the truth assignments of shared fluents. Moreover, we assign both k and d a value of 1.

$$O(m \cdot l \cdot min((a+1)^1, 1 \cdot 2^v) = O(m \cdot l \cdot a)$$

Here, m is the number joints, l is the size of discretized Workspace($|W_{robot}|$), and a is the number of actions at the last (m_{th}) joint.

We can simply bound the size of Workspace $W \equiv (X, Y, \Theta)$. Suppose a robotic arm which has m joints and each joint has $c\left(= \frac{2\pi}{\Delta\theta}\right)$ discrete angles. The space of the end effector can be bounded by $[-N..N] \times [-N..N] \times [0..2\pi]$, if the N is defined by the average length of the joints ($N = mL = \sum_{i=1}^{m} length_i$). Thus, the complexity of Workspace is following

$$O(l) = O(|W|) = O(c \cdot N^2) = O(m^2)$$

The largest number of actions ($a = |A_m|$) is also represented by the Workspace($|W|$) and End-Effector Space($|ES_m|$). All the actions are grouped based on the each location in $|ES_m|$. In each location, there are at most $2|W|$ actions, because each action is unit movement (left or right) of any previous joint in the Workspace (W).

$$O(a) = O(|A_m|) = O(|W| \cdot |ES_m|)$$

The complexity of the modified Factored Planning is following (when $c \left(= \frac{2\pi}{\Delta\theta}\right)$ is the number of discrete orientations, and $\Delta\theta$ is a unit angular displace.)

$$O(m \cdot |W| \cdot |A_m|) = O(m \cdot |W|^2 \cdot |ES_m|) = O\left(\frac{m^5}{(\Delta\theta)^2} \cdot |ES_m|\right)$$

Algorithm 2 SingleJointPlan

es: a point in the end effector space
ang: the direction to the next link
$length_j$: the length of j_{th} link
$A_{j-1}(es)$: the actions of es

SUBROUTINE *SingleJointPlan*(es, ang, $length_j$, $A_{j-1}(es)$)
1. Let es' new position and configuration of j_{th} joint, and $act \leftarrow \emptyset$
   $es'(loc) \leftarrow es(loc) + trans(es(loc), ang, length_j)$ [3]
   $es'(conf) = es(conf) \bigcup ang$
2. For each $act_{j-1} \in A_{j-1}(es)$
   (a) Make act_{new} from act_{j-1} (The $end_i(w_i)$ is in the precondition of act_{j-1})
   pre: $end_i(w_i) \wedge end_j(es'(loc))$
   eff: $\neg end_j(es'(loc)) \wedge end_j(w'_j)$
   (b) If j_{th} joint of act_{new} do not collide with any obstacle
   Insert act_{new} into act
3. For each leftmove and rightmove of j_{th} joint by $\Delta\theta$
   (a) Make act_{new} as following
   pre: $end_{j-1}(es(loc)) \wedge end_i(es'(loc))$
   eff: $\neg end_j(es'(loc)) \wedge end_j(w''_j)$
   (b) If j_{th} joint of act_{new} do not collide with any obstacle
   Insert act_{new} into act
4. Return $\langle es'$, act$\rangle$

Grouping the Set of Actions

We reduce the complexity of the planning problem by grouping the actions based on the location of joint. That is, we don't consider a specific configurations of joints of an arm, if the location of end effector are same. If we separately store the set of actions for the specific configuration, the complexity is same with the configuration space which is exponentially proportional to the number of joints.

We group the set of actions, because storing the sets of actions for every configurations is not only expensive but also redundant. Thus, we merge some sets of actions into a large group, when the locations of end-effector are on the same location. Our grouping method may loss some data, because a specific configuration only contributes to the part of actions. This prevent our algorithm finding an optimal path in some case. However, this method is beneficial for

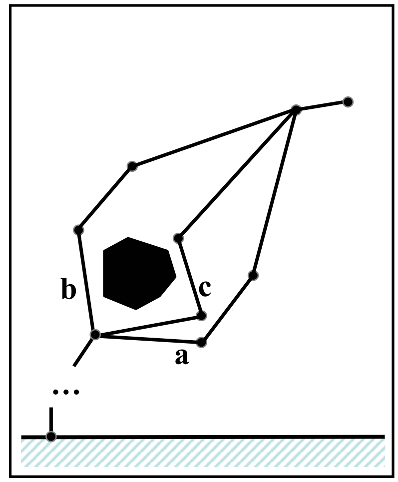

Figure 4: The Homotopic relationship among the configurations; 'a', 'b' and 'c' are the configurations of the arm; 'a' and 'b' are not Homotopic configurations; 'a' and 'c' are Homotopic configurations

reducing complexity.

For grouping the sets of actions, we define the Ho-

Algorithm 3 StorePartPlan using any *local planner*

es': a point in the end effector space
act: the set of actions related to the es'
ES_j: the current end effector space of j_{th} joint
A_j: the current set of actions of j_{th} joint

SUBROUTINE *StorePartPlan*(es', act, ES_j, A_j)
1. For each $es \in ES_j$ when $es(loc) = es'(loc)$
   (a) If there is a path from $es'(conf)$ to $es(conf)$ with a *local planner* then
   Add act to $A_j(es)$
   Exit
2. $A_j(es') = act$ {Assign new action set}

motopic Configurations. Two configurations with the same endpoints are homotopic if one can be continuously deformed into the other. The concept of the Homotopic Configurations is based on the Homotopic Paths (Brock & Khatib 2000) [5]. Although it is similar to the concept with Homotopic Paths, we adapt the notion of Homotopic to the configuration. For example, in Figure 4, the two configurations 'a' and 'c' are Homotopic and 'a' and 'b' are not Homotopic.

In our algorithm, the grouping is decided by a *local planner*. The *local planner* could find a path from one configuration to another, if the two configurations are Homotopic. This can be achieved by any inverse-kinematics algorithm which avoids obstacles. There are many local

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

path planning algorithms for such a calculation (Zlajpah & Nemec 2002). The *local planner* is used for finding the Homotopic Configurations which are deformable each other. However, in the real environment, two Homotopic configurations are not always continuously deformable each other, due to the rigid body of link[6].

Soundness and Completeness

Algorithm 4 PathPlan
es_{start}: the initial location and its configuration
es_{goal}: the goal location and its configuration
A_m: the set of actions of m_{th} joint
ES_m: the end effector space of m_{th} joint
$depth$: the depth for searching path

SUBROUTINE *PathPlan*(es_{start}, es_{goal}, A_m, ES_m, $depth$)
1. $es'_{start} \leftarrow$ *FindES*(es_{start}, A_m, ES_m)
2. $es'_{goal} \leftarrow$ *FindES*(es_{goal}, A_m, ES_m)
3. $j \leftarrow 1$, $R_0 \leftarrow \{es'_{start}\}$, $R_{total} = \emptyset$, $A'_m = \emptyset$
4. Do until j=$depth$
   (a) For each $es \in R_{j-1}$
      i. For each $act_{(m,es,w_i,...)} \in A_m(es)$
       If the $act_{(m,es,w_i,...)}$ is valid for es
         $es_0 \leftarrow$ moved es by the $act_{(m,es,w_i,...)}$
         $es' \leftarrow$ *FindES*(es_0, A_m, ES_m)
         If $es' \notin R_{total}$ then
           Make new Action($move_{(es,es')}$) as following
             pre: $es \wedge done_{j-1} \wedge \neg done_j$
             eff: $es' \wedge done_j$
           Add $move_{(es,es')}$ to Act_{global}
           Add es' to R_{total}
   (c) j ← j + 1
5. Init(es'_{start}, $done_0$, $\neg done_1$, ..., $\neg done_{depth}$)
6. Goal(es'_{goal}, $done_0$, $done_1$, ..., $done_{depth}$)
7. Search for plans (Φ) in $Init$, $Goal$, Act_{global}
8. return Φ

We prove this path planning algorithm is sound and complete. Our planning algorithm is sound, because all the returned paths are valid. That is, we can control the robotic arm along the returned path. 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 position.

Lemma 1.1: With the robotic arm, we can find movements for every $move(es, es') \in Act_{global}$ in *PathPlan*

Proof. Suppose that there is a $move(es, es') \in Act_{global}$ and an $act_{(m,es,w_i,...)}$ contributes to the movement. We can assign a location es_0 which is moved from es by the $act_{(m,es,w_i,...)}$ in the *PathPlan*. There are movements from es_0 to es', because the *FindES*(es_0, ...) returns an es' only if the *local planner* finds a path from es_0 to es'. □

[6]We check the Homotopic relationship with *local planner*. To reduce the complexity of this algorithm we allow the end point can be moved, if the movement can be managed by the *local planner*. This makes the algorithm require a verification function in the *PathPlan*, because the transition between the configurations may cause collision at the outer link, if the location of end point is not fixed.

Theorem 1.1:(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})

Proof. By the *Lemma 1.1*, 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 1.2:(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, if the path is a movement of robotic arm, we can simply find it. It is because we have all the actions for each point ($es \in ES_m$). That is, if there exists a unit movement from a location es to a location es_1, we have a complex action for the movement in $A(es)$.
For the $n-1_{th}$ step, we assume that if there exist $n-1$ movements from a location es to a location es_{n-1}, we can find a path.
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 assumption of 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 $\langle es_0, es_1, ..., es_{n-1} \rangle$, $act_{(m,es_{n-1},...)}$, es_n. □

Algorithm 5 FindES
es: a configuration
Act: the set of actions
ES: the end effector space

SUBROUTINE *FindES*(es, Act, ES)
1. For each $es' \in ES$ for $es'(loc) = es(loc)$
   (a) If $es'(conf)$ and $es(conf)$ are Homotopic Configurations
      i. If *local planner* find a path from es' to es
        return es
2. Make a new $es''(\equiv es)$
3. Add es'' to ES
4. $Act(es'') \leftarrow Act(es)$
5. return es''

3 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

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

of obstacles.

If we find the path with an exact path planning algorithm (Khatib 1986) (Brock & Khatib 2000) in the dimensional space, the complexity is

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

Without Obstacles

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). At the t_{th} step, we can find the reachable position with, at most, t actions through dynamic programming in Algorithm 4. If we execute further $depth$ steps, we can find all the possible paths whose lengths are shorter than $depth$. Even if there are certain non-valid paths, which are caused by the incomplete previous step, we can locate a path from the returned paths.

Lemma 2.1: Without any obstacles, the complexity of $RobotArmPlan$ is following given m links.

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

Proof. For the one positions in ES_m, we have maximum ($|W| = cN^2$) neighbor positions which are reachable with a move action. Because we have a total of cN^2 positions and visit each position once, the total step is not more than cN^2. Moreover, there are no two es_0 and es_1, which are in ES_m and $es_0(loc) = es_1(loc)$. In the $StorePartPlan$, es_1 merges into the es_0, if the es_0 is already added to the ES_m. Without obstacles, the *local planner* [8] finds a path between two configurations if they are Homotopic Configurations. □

With a Convex Island

Lemma 2.2: If there exists a convex island, the size of end effector, $|ES_m|$, is bounded by

$$O(|ES_m|) = O(2 \cdot |W|).$$

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

Proof. Suppose that there are 3 (or more) distinctive groups (a, b, and c) whose end effector locate on the same location, and there is an convex island obstacle are blocked by each other. That is, 'a' and 'b' are blocked by the island; 'b' and 'c' are blocked by the island; and 'c' and 'a' are blocked by the island.

Given the two distinct groups, we can label that one as 'left-side' of the island the other as 'right-side' of the island. Without loss of generality, suppose that 'a' is the left-side and 'b' is the right-side. In that case, 'c' can be assigned to neither left-side nor right-side. There is no 'c' in the given space. □

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

With 'n' Convex Islands

Lemma 2.3: The search space for n convex islands is bounded by $O(2^n \cdot |W|)$.

Proof. Suppose the search space with $n-1$ convex islands is bounded by $O(2^{n-1} \cdot |W|)$ for mathematical induction. Assume that each location (w) in the $n-1$ convex islands environment can have up to 2^{n-1} distinct groups of configurations. That is, $|ES_{m,n-1}(w)| \leq 2^{n-1}$, when $ES_{m,n-1}(w) = \{es | es(loc) = w \wedge es \in ES_m \textit{ with n-1 convex islands}\}$. With an additional island, O_n, each distinct group can be divided into two (and no more than two) groups, which would be either left-side or right-side with respect to O_n. Thus, the distinctive groups for each position, $w \in W$, can be bounded by $O(2^n)$ □

With an Infinite Number of Obstacles

Lemma 2.4: If there is an infinite number of obstacles, the search space of the end effector, $|ES_m|$, is bounded by

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

Proof. The size of $|ES_m|$ cannot be greater than all possible configurations of m joints ($O((\frac{2\pi}{\Delta\theta})^m)$) when $\Delta\theta$ is the unit angles and c is $\frac{2\pi}{\Delta\theta}$. □

The Size of $|ES_m|$

Theorem 2: If there are n convex islands (obstacles), the search space of the end effector, $|ES_m|$, is bounded by

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

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

Proof. by *Lemma 1.1, 1.2 and 1.3* □

4 Error Bound

Here, we examine the error of the suggested algorithm with respect to the discretized configuration space algorithm. [9]

Our algorithm approximately discretized the position of each joint, at (x, y, θ). That is, if the position of the n_{th} joint of an arm is close enough to a position (x', y', θ'), we assumed that the n_{th} joint of arm is located on the (x', y', θ'). The position error of inner joints cumulatively influences the position of the end effector, due to the dynamic programming fashion of our algorithm.

The error can be divided into two categories: (1) the displacement of location; and (2) the difference of angles. When we assume that the size of cell in the discretized Workspace is small, relative to the length of joints, the error

[9] Here, we use the discretized configuration space in contrast to the continuous configuration space. That is, all the algorithms, which uniformly split the configuration space into the cells of a configuration, are included in the definition

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

To analyze the angular error, suppose that $\Delta\theta$ is a unit angular displacement ($\frac{2\pi}{c}$) and c is the number of discretized angles. The maximum difference of angles with respect to the original configuration can be described as $m\frac{\Delta\theta}{2}$. When the maximum error of angle at each joint is bounded by $\frac{\Delta\theta}{2}$, the angular error of the last joint is bounded by $m\frac{\Delta\theta}{2}$. [10] This is because the angular error is also additively accumulated. If the length of m joints is N, the maximum displacement of x (Δx_{ang}) is $N \sin{(m \cdot \frac{\Delta\theta}{2})}$.

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

Proof. The maximum error of Δx is sum of Δx_{pos} and Δx_{ang} because Δx_{pos} and Δx_{ang} are respectively bounded by $\frac{1}{2} size(cell)$ and $L \sin{(m \cdot \frac{\Delta\theta}{2})}$. To bound Δx to r, we simply bound both the Δx_{pos} and Δx_{ang} to $\frac{1}{2}r$. □

5 Related Work

The problem of finding an optimal path from the current position to a goal position at the configuration space of a robotic arm received the attention of many works in the robotics and planning literature. Potential Field (Khatib 1986), Probabilistic Roadmap (Kavraki *et al.* 1996) and Rapidly-Exploring Random Trees (Kuffner. & LaValle 2000) are currently used for solving this problem (Ghallab, Nau, & Traverso 2004). Although wavefront-expansion Potential Field algorithm (Barraquand & Latombe 1991) provides completeness, the algorithm have to search huge configuration space. Though the efficiency of these approaches is dramatically improved when Probabilistic Roadmap is used, the solutions still depend on the large dimensionality of the configuration space.

In contrast, this paper modified a Factored Planning algorithm (Amir & Engelhardt 2003) to solve this problem. Our algorithm takes advantage of the fact that each angle of a robotic arm is independent of the rest, given some parameters. That is, the configuration space of a joint is independent of other joints with exception of previous joint. The configuration space of an joint depends on the location in the workspace. This allow us to partition the planning problem into small subdomains, and the resulting algorithm has a running time that depends on the dimensionality of the space only polynomially.

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

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. Although the complexity is exponential in the number of obstacles, it is only polynomial to the number of joints. Our theoretical results show that the planning algorithm scales well in planning the path of a robotic arm.

References

Amir, E., and Engelhardt, B. 2003. Factored planning. In Gottlob, G., and Walsh, T., eds., *IJCAI*, 929–935. Morgan Kaufmann.

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

Brock, O., and Khatib, O. 2000. Real-time replanning in high-dimensional configuration spaces using sets of homotopic paths. In *IEEE International Conference on Robotics and Automation (ICRA)*, 550–555.

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

Ghallab, M.; Nau, D.; and Traverso, P. 2004. *Automated Planning*. Morgan Kauffman.

Kavraki, L. E.; Svestka, P.; Latombe, J.-C.; and Overmars, M. 1996. Probabilistic roadmaps for path planning in high dimensional configuration spaces. *IEEE Transactions on Robotics and Automation* 12(4):566–580.

Khatib, O. 1986. Real-time obstacle avoidance for manipulators and mobile manipulators. *International Journal of Robotics Research* 5(1):90–98.

Kuffner., J. J., and LaValle, S. M. 2000. Rrt-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, 995–1001.

Schwartz, J. T., and Sharir, M. 1983. On the Piano Movers' Problem: I. The case of a two-dimensional rigid polygonal body moving amidst polygonal barriers. *Communications on Pure and Applied Mathematics* 36:345–398.

Zlajpah, L., and Nemec, B. 2002. Kinematic control algorithms for on-line obstacle avoidance for redundant manipulators. In *IEEE/RSJ/ International Conference on Intelligent Robots and Systems*, 1898–1903. Lausanne, Switzerland: IEEE/RSJ.