Motions of a Short-Linked Robot Arm in a Square

Vitit Kantabutra

Department of Mathematics and Computer Science, Drexel University,
Philadelphia, PA 19104, USA

Abstract. The main result of this paper is an $O(n)$ -time algorithm to determine whether a given configuration of an n -link short-linked robot arm confined in a square can be reached from another given configuration. The arm is constructed from n links, which are straight line segments of real lengths, each no greater than half the side length of the confining square. This result represents a continuation of previous work published by Hopcroft *et al.* and by Kantabutra and Kosaraju.

1. Introduction

A problem related to the movements of multilink robot arms in two dimensions is considered. A robot arm is constructed from n links, which are straight line segments of arbitrary real lengths. These links are joined together end-to-end by freely rotating joints so that the links are allowed to cross each other. One end of the arm, called the shoulder, is pinned down positionwise. Free rotation about the pin is allowed. The arm is confined within a square each of whose sides is at least twice as long as the longest link of the arm.

The main result of this paper is an $O(n)$ -time algorithm to determine whether a given configuration of the arm¹ can be reached from another given configuration. A key result that makes the algorithm work is that a configuration is reachable from another if, in these configurations, the same links are “stuck” to a corner of the constraining square.

This paper represents a step toward understanding the computational complexity of motion-planning problems. One of the first groups of researchers to study the computational complexity of motion planning was Hopcroft *et al.*, who had a considerable amount of success on the study of arms of the type described above

¹ The *configuration* of an arm (at any instant) is specified by the positions of all the joints.

[1], [2]. In [2] they showed that reachability decision problems are NP-hard if the arm is constrained by arbitrary polygonal walls. However, when the boundary is a circle, then they were able to give an algorithm that determines the reachability of any arm configuration from any other configuration in $O(n)$ time. They also gave an algorithm of $O(n^3)$ -time complexity that moves the arm to any specific reachable configuration in $O(n^3)$ arm moves.² In [3] Kantabutra and Kosaraju presented a more efficient solution, involving only $O(n)$ time and $O(n)$ arm moves. In [1] Hopcroft *et al.* developed a polynomial-time algorithm for computing the regions reachable by the joints of a robot arm in a circle. This polynomial is at least quadratic in the input size. Kantabutra and Kosaraju presented in [3] a linear-time solution to this problem. It was also established there that linear-time computation of a reachable configuration of an arm in a circle is possible, when the final positions of the members of an arbitrary subset of the set of joints are specified.

2. Mathematical Preliminaries

A robot arm (Fig. 1) is a sequence of n straight line segments or *links* of arbitrary finite lengths moving in the Euclidean plane. These links are joined together end-to-end by freely rotating *joints*. One end of the arm, the *shoulder*, is fixed positionwise, but can rotate freely. The other end is called the *hand*. The joints (including the shoulder and the hand) are denoted by $A_0, A_1, \dots, A_n$. The link between A_{i-1} and A_i , $i > 0$, is called L_i , and L_i 's length is called l_i . The *joint angle* at A_i , $i > 0$, is the angle between the links L_i and L_{i+1} , and the joint angle at A_0 is the angle that L_1 makes with horizontal. A *configuration* of an arm is specified by the positions of all the joints.³ The *tail (of an arm) beginning from A_i* is the part of the arm from A_i to A_n . So an entire robot arm is the same as the tail beginning from A_0 .

Fig. 1. A robot arm.

² They defined an *arm move* to be any arm motion in which at most four joint angles (a joint angle is an angle between two links) move simultaneously, and each joint angle moves only in one direction.

³ Formally, the configuration is the $2(n+1)$ -tuple $(x_0, y_0, x_1, y_1, x_2, y_2, \dots, x_n, y_n)$, where x_i, y_i are, respectively, the X and the Y coordinates of A_i .

Fig. 2. Defining link orientation.

In the movements of the arm we allow all joint angles to change simultaneously, such that the rate of change of each angle may be an arbitrary function of time.

The constraining square, denoted by S , has sides of length s parallel to the X and Y coordinate axes. The square constrains the arm in the sense that, at any instant, each joint's position must be within the closed region delimited by S .

A link L_i is said to be *oriented toward* a corner (vertex) A of S (Fig. 2) if in case we erect orthogonal axes through A_{i-1} and parallel to the sides of S , A_i and A are in the same orthant. Note that, in Fig. 2, L_i is oriented toward A , and L_j is oriented toward *both* A and B . Any link L_i is *free* if it is oriented toward a corner of S that is *farthest* from A_{i-1} (farthest when compared with the other corners of S). An arm or a tail is *free* if all its links are free. A link is *locked* if the arm cannot be moved into a configuration in which that particular link is free.

3. Main Result

Now we address the reachability problem. Let the initial and the final configurations be denoted by Γ_{init} and Γ_f , respectively. We will establish that Γ_f is reachable from Γ_{init} if and only if the highest-indexed locked link is the same link in both these configurations.⁴ It turns out, however, that an arm configuration can have at most one locked link. Our reachability problem thus becomes a problem of deciding whether the two input configurations have the same locked link or not. (Hence we can consider the set of all possible arm configurations to be classified into equivalence classes of mutually reachable configurations based on which link (if any) the locked link is.) Much of the work in this paper involves showing that if the two input configurations have the same locked link, or if neither configuration has a locked link, then each configuration is reachable from the other. This reaching is by way of normal form configurations, defined as follows.

⁴ Note that reachability is symmetric.

Fig. 3. An arm in normal form.

An arm is in normal form (see Fig. 3) if all the following conditions hold for some j :

1. The initial joint angles at $A_1, \dots, A_{j-2}$ are straight, and the joint angle at A_{j-1} is folded (zero); the joint angle at A_0 is allowed to be anything.
2. All the links indexed higher than j are free.
3. L_j is oriented toward the corner of S closest to A_{j-1} .
4. $l_j - \sum_{k < j} l_k > \max\{s/2 - |x_0|, s/2 - |y_0|\}$, where x_0 and y_0 are the X and Y coordinates of A_0 . Note that x_0 and y_0 are the perpendicular distances from A_0 to sides α and β , as shown in Fig. 3.

Observe that conditions 3 and 4 hold vacuously when $j = 0$.

Lemma 1. *Consider an isolated tail from A_i in any free configuration. The joint A_i can be guided through any path within S (and its boundary) in such a manner that the tail is kept free all the time.*

Proof. The proof is by induction on the joint indices.

Erect axes, as in Fig. 4, parallel to the sides of S , such that the origin lies at the center of the region enclosed by S .

We now show that the lemma holds for index $n - 1$. Consider the tail from A_{n-1} (which is L_n) as a free, isolated link. As A_{n-1} traces any path without crossing an axis, we do not allow L_n to rotate. The link L_n , being no longer than $s/2$, is always free. Now assume that A_{n-1} lies on an axis and is about to enter a new quadrant. As an illustration assume that A_{n-1} now lies in the first quadrant, and is just about to cross the positive X -axis into the fourth quadrant. Observe that L_n is free whenever it (the A_{n-1} to A_n line) makes an angle between 90° and 270° with the X -axis.⁵ Consequently L_n can be swung within that range and be kept free at all times. Thus we let A_{n-1} pause on the positive X -axis and swing L_n about A_{n-1} until L_n makes a

⁵ If A_{n-1} were strictly inside the first quadrant, then L_n would be free when this angle was between 180° and 270° . On the other hand, if A_{n-1} were strictly inside the fourth quadrant, then L_n would be free when this angle was between 90° and 180° .

Fig. 4. An axis system and free link L_n for the proof of Lemma 1.

180° angle with the X -axis. Now A_{n-1} can continue on its path so that L_n remains free upon A_{n-1} 's entry into the fourth quadrant.

Now assume that the lemma holds for index $i + 1$. When A_i follows any path, move L_{i+1} through the same positions we would have moved it if it had been the only link, guaranteeing that L_{i+1} is always free. By the induction hypothesis, we can move the tail from A_{i+1} in such a way that it is kept free all the time. Hence A_i can follow any path in such a manner that the tail from A_i is always free. We have thus proven the lemma by induction. $\square$

Algorithm 1. (Moving an arm to normal form.) Erect axes dividing the region enclosed by S into quadrants as before.

Suppose L_j is the highest-indexed locked link. (Let $j = 0$ if there is no locked link.) A subalgorithm will be described that frees all the links indexed higher than j , one by one in descending order of their indices. After applying this procedure to such links, we apply it to L_j . The effect of this last application is not to free L_j (which is locked), but to bring the arm into normal form, as desired. No previous knowledge of j is required for the usage of this subalgorithm.

Subalgorithm. Assume that, at the present stage, all the links indexed greater than m , for some $m > j$, are free. Suppose m is the minimum index for which this assumption holds, and thus L_m needs to be freed, if possible. Let α and β respectively be the vertical and the horizontal sides of S adjacent to a quadrant in which A_{m-1} lies, and let Q denote the corner of S at which α and β intersect. See Fig. 5.

Note that the tail from A_m , being free, can be ignored while freeing L_m .

We may assume that L_m is oriented toward Q . (If it is not, then L_m can be freed simply by swinging A_m about A_{m-1} , away from Q .) Swing A_m about A_{m-1} toward Q until L_m becomes collinear with the line joining A_{m-1} and Q . From now on keep L_m collinear with the line from A_{m-1} to Q . (This line moves as A_{m-1} moves.) Let us now assume that A_{m-1} never hits S , since if A_{m-1} hits S , then (supposing that L_m is not already free) L_m can be freed simply by swinging A_m about A_{m-1} .

Fig. 5. Showing a locked link L_j referred to in Algorithm 1.

At this stage we straighten out the joints indexed lower than $m - 1$. To this end, let us introduce a variable k , whose value is $m - 1$ initially. Swing A_k about A_{k-1} away from Q , always keeping $L_{k+1}, L_{k+2}, \dots, L_m$ collinear with the line through Q and A_{m-1} . (We may assume that A_k never hits S , because otherwise A_{m-1} would also hit S , contradicting our assumption of the contrary.) Now decrement k by 1 and repeat the swing. Stop after swinging A_1 about A_0 .

The situation now is that $L_1, L_2, \dots, L_{m-1}$ and L_m are all collinear with the line joining Q and A_{m-1} , and all of these links except L_m point away from Q . The link L_m points toward Q . There are two possibilities: either $l_m - \sum_{i < m} l_i \leq \max\{s/2 - |x_0|, s/2 - |y_0|\}$, or not. In the former case, it is easy to free L_m by pivoting A_0 , keeping $A_1, \dots, A_{m-1}$ fixed and keeping the tail of the arm from A_m free. In the latter case, the algorithm stops altogether, because the arm has come to normal form. The link L_m cannot be freed, i.e., $m = j$. To see this, simply notice that A_{m-1} cannot be moved so far as l_m from α or β . Observe that the quantity $l_m - \sum_{i < m} l_i$ can be computed in constant time if we have stored the corresponding quantity from the previous iteration.

Figure 6 illustrates a possible snapshot from the execution of our subprogram.

Lemma 2. *If a locked link exists, then it is unique.*

Proof. If L_j is the highest-indexed locked link, then Algorithm 1, just given, brings the arm into a normal-form configuration, in which all links except L_j are free. $\square$

Now we show that if two configurations have the same locked link, then each of the configurations is reachable from the other.

Algorithm 2. (This algorithm moves the arm from Γ_{init} to Γ_f if these two configurations both have the same locked link, L_j , or both have no locked link, in which case we let $j = 0$.)

Fig. 6. A possible snapshot from the execution of the subprogram of Algorithm 1.

Move the arm from Γ_f to normal form, recording all the moves made. Call this normal-form configuration Γ_f^{norm} . Now move the arm from Γ_{init} to normal form, calling the resulting configuration $\Gamma_{\text{init}}^{\text{norm}}$. It only remains to be seen how to move the arm from $\Gamma_{\text{init}}^{\text{norm}}$ to Γ_f^{norm} .

Since in both normal-form configurations every one of the joints indexed up to and including j is placed at the same position, it suffices to bring the joints indexed $j + 1$ and higher, one at a time in the order of their indices, to the positions they will assume in Γ_f^{norm} .

Assume that all the joints indexed no greater than k have been brought to their destinations, leaving the tail from A_k free. Since the tail from A_{k+1} is also free, we can ignore that tail and swing A_{k+1} about A_k until A_{k+1} reaches its destination, keeping the tail free at all times. The joint A_{k+1} never becomes obstructed by S because L_{k+1} only moves through free positions. Thus we have demonstrated how successively higher-indexed joints can be brought to their destinations.

Finally the main result of the paper can be presented, as follows:

Theorem 1. *Given two configurations Γ_{init} and Γ_f , it can be determined in linear time whether Γ_f can be reached from Γ_{init} .*

Proof. We will show that determining whether Γ_f can be reached from Γ_{init} is equivalent to determining whether the two configurations have the same locked link. Then the proof will be finished when we show that the latter activity can be performed in linear time.

If the two configurations have the same locked link, then Algorithm 2, just given above, can be used to move the arm from Γ_{init} to Γ_f . However, if the two configurations have different locked links (or if one of the two configurations has a locked link and the other one does not), then it is easy to see that the arm cannot be moved from Γ_{init} to Γ_f , as follows: If the locked links are different, then let the higher-indexed one be L_j . Assume without loss of generality that Γ_{init} is the

configuration with L_j locked. There must be a configuration Γ_{jfree} , in which L_j is free, such that Γ_{jfree} is reachable from Γ_f . Because the reachability relation among configurations is transitive, it follows that if Γ_f were to be reachable from Γ_{init} , then Γ_{jfree} would have also been reachable from Γ_{init} . However, this is impossible, since L_j is locked in Γ_{init} .

Acknowledgments

The author would like to thank the referee, who gave very useful and explicit advice on how to reorganize this paper.

References

1. J. E. Hopcroft, D. A. Joseph, and S. H. Whitesides, Determining points of a circular region reachable by joints of a robot arm, Technical Report TR82-516, Computer Science Department, Cornell University, 1982.
2. J. E. Hopcroft, D. A. Joseph, and S. H. Whitesides, On the movement of robot arms in two-dimensional regions, *SIAM J. Comput.* Vol. 14, No. 2, 1985, pp. 315-333.
3. V. Kantabutra and S. R. Kosaraju, New algorithms for multilink robot arms, *J. Comput. System Sci.*, Vol. 32, No. 1, February 1986, pp. 136-153.

Received May 10, 1989, and in revised form October 10, 1989.