On The Movement of Robot Arms in 2-Dimensional Bounded Regions

J.E. Hopcroft*

D.A. Joseph*

S.H. Whitesides**

March 1982

TR 82-486

Department of Computer Science
Cornell University
Ithaca, New York 14853

---

* Cornell University

** On leave from Dartmouth College, Hanover, N.H.

This research was supported in part by ONR contract N00014-76-C-0018, NSF grant MCS81-01220, an NSF Postdoctoral Fellowship and a Dartmouth College Junior Faculty Fellowship.

On the Movement of Robot Arms in 2-Dimensional Bounded Regions

John Hopcroft
Deborah Joseph
Sue Whitesides[+]

Computer Science Department
Cornell University

ABSTRACT

The classical mover's problem is the following: can a rigid object in 3-dimensional space be moved from one given position to another while avoiding obstacles? It is known that a more general version of this problem involving objects with movable joints is PSPACE complete, even for a simple tree-like structure moving in a 3-dimensional region. In this paper, we investigate a 2-dimensional mover's problem in which the object is a robot arm with an arbitrary number of joints. In particular, we give a polynomial time algorithm for moving an arm confined within a circle from one given configuration to another. We also give a polynomial time algorithm for moving the arm from its initial position to a position in which the end of the arm reaches a given point within the circle.

Keywords: robotics, manipulators, mechanical arms, algorithms, polynomial time.

This work was supported in part by ONR contract N00014-76-C-0018, NSF grant MCS81-01220, an NSF Postdoctoral Fellowship and a Dartmouth College Junior Faculty Fellowship.

[+] On leave from the Mathematics Department, Dartmouth College.

1. Introduction

With current interests in industrial automation and robotics, the problem of designing efficient algorithms for moving 2- and 3-dimensional objects subject to certain geometric constraints is becoming increasingly important. The mover's problem (see Schwartz and Sharir [4,5], Reif [3]), is to determine, given an object X, an initial position P_i, a final position P_f and a constraining region R, whether X can be moved from position P_i to position P_f while keeping X within the region R.

In the classical problem, X is a rigid 2- or 3-dimensional polyhedral object, and R is a region described by linear constraints. Recently, several authors (Schwartz and Sharir [4,5], Reif [3], Lozano-Perez [2]) have presented polynomial time algorithms for solving this type of problem.

A more difficult problem, which is related to problems in robotics, assumes that the object X has joints and is hence nonrigid. Again, one desires a fast (polynomial time) algorithm for moving X from position P_i to P_f within a region R. Unfortunately, such an algorithm is unlikely, as Reif [3] has shown that the problem of deciding whether an arbitrary hinged object can be moved from one position to another in a 3-dimensional region is PSPACE complete.

Our paper investigates variants of the mover's problem which we believe are of practical interest. We begin in Sections 2 and 3 by considering the problem of folding a carpenter's ruler -- that is, a sequence of line segments hinged together consecutively. This problem arises because a natural strategy for moving an arm in a confining region is to fold it up as compactly as possible at the beginning of the motion. Unfortunately, deciding whether an

arbitrary carpenter's ruler (whose link lengths are not necessarily equal) can be folded into a given length is NP-complete. Because of this, it turns out to be at least NP-hard to decide whether or not the end of an arbitrary arm (i.e., a carpenter's ruler with one end fixed) can be moved from one position to another while staying within a given 2-dimensional region.

In Sections 4 and 5 we consider the problem of moving an arm inside a circular region, and we are able to give polynomial time algorithms for changing configurations and reaching points.

2. Folding a Ruler

In this section, we ask how hard it is to fold a carpenter's ruler consisting of a sequence of n links $L_1, \ldots, L_n$ that are hinged together at their endpoints. These links, which are line segments of integral lengths, may rotate freely about their joints and are allowed to cross over one another. We assume that the endpoints of the links are consecutively labeled $A_0, \ldots, A_n$ and for $1 \leq i \leq n$, we let $\mathbf{l}_i$ denote the length of link L_i. We define the RULER FOLDING problem to be the following:

Given: Positive integers n, $\mathbf{l}_1, \ldots, \mathbf{l}_n$, and k.

Question: Can a carpenter's ruler with lengths $\mathbf{l}_1, \ldots, \mathbf{l}_n$ be folded (each pair of consecutive links forming either a 0° or 180° angle at the joint between them) so that its folded length is at most k?

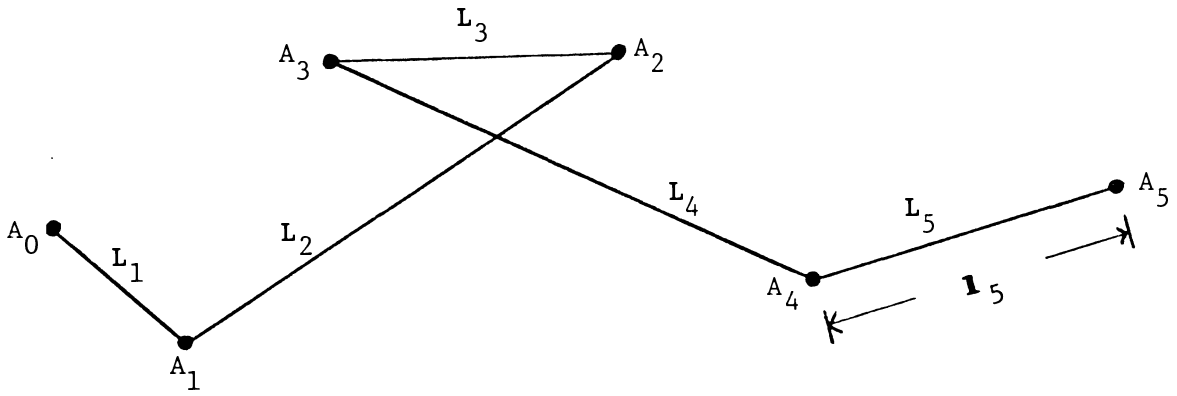


Fig. 2.1: A typical ruler with five links.

By a reduction from the NP-complete PARTITION problem (see Garey and Johnson [1]) we can easily show that the RULER FOLDING problem is also NP-complete. The PARTITION problem asks whether, given a set S of n positive integers $\mathbf{l}_1, \ldots, \mathbf{l}_n$, there is a subset $S' \subseteq S$ such that

$$\sum_{\mathbf{l}_i \in S'} \mathbf{l}_i = \sum_{\mathbf{l}_j \in S-S'} \mathbf{l}_j .$$

Theorem 2.1: The RULER FOLDING problem is NP-complete.

Proof: Given an instance of the PARTITION problem with $S = \{\mathbf{l}_1, \ldots, \mathbf{l}_n\}$, let $d = \sum_{i=1}^{n} \mathbf{l}_i$. Then the desired subset S' of S exists if and only if a ruler with links of length 2d, d, $\mathbf{l}_1, \ldots, \mathbf{l}_n$, d, 2d (in consecutive order) can be folded into an interval of length at most 2d. To see that this is the case, imagine that the ruler is being folded into the real line interval [0,2d], and notice that both the initial endpoint A_0 of link L_1 (the third link in our ruler) and the terminal endpoint A_n of link L_n (the third from last link) must be placed at integer d. The set S' in the PARTITION problem then corresponds to the set of links L_i whose initial endpoints A_{i-1} appear to the left of their terminal endpoints A_i in a successful folding of the ruler. □

The RULER FOLDING problem and the PARTITION problem share not only the property of being NP-complete, but also the property of being solvable in pseudo-polynomial time. The time complexity of the RULER FOLDING problem is bounded by a polynomial in the number of links, n, and the maximum link length, m. In fact, it is possible to find the minimum folding length in time proportional to n*m by a dynamic programming scheme. However, in order to carry out this scheme we need to know that a ruler with maximum link length m can always be folded to have length at most 2m.

Lemma 2.1: A ruler with lengths $\mathbf{l}_1, \ldots, \mathbf{l}_n$ can always be folded into length at most 2m, where $m = \max \{\mathbf{l}_i \mid 1 \leq i \leq n\}$.

Proof: Place link L_1 into the interval [0,2m] with A_0 at 0. Having placed links $L_1, L_2, \ldots, L_{i-1}$ into the interval, position L_i as follows: Place L_i with A_i to the left of A_{i-1}, if possible. Otherwise, place L_i with A_i to the right of A_{i-1}. To see that this is possible, suppose that p is the position of A_{i-1} and note that if A_i cannot be placed to the left of A_{i-1}, then $p \leq l_i \leq m$. Hence A_i can surely be placed to the right of A_{i-1}. □

Using this result, we can now give a dynamic O(m*n) programming algorithm for determining the minimum folding length of a ruler, where n is the number of links in the ruler and m is the maximum length of any given link.

Algorithm 2.1: Ruler Folding in Minimum Length

Given a ruler with links $L_1, \ldots, L_n$, compute the maximum link length m. Then, for each k, $1 \leq k \leq 2m$, construct a table with rows numbered 0 to n and columns numbered 0 to k. Row i corresponds to endpoint A_i, and column j corresponds to the position j in the interval [0,k]. Fill in row 0 by writing a T in each column j for which L_0 fits in [0,k] with A_1 at integer j, and F's

in the other columns. Once row i-1 has been filled in, fill in row i by writing a T in each column j for which the linkage L_1, ..., L_i fits in [0,k] with endpoint A_i at integer j. To do this, examine row i-1 to obtain the possible locations for A_{i-1}. The last row of the completed table contains a T if and only if the ruler can be folded into [0,k]. Find the smallest k for which the table contains a T in the last row, and read the table from bottom to top to reconstruct the desired folds. □

The next example shows that 2m is, in fact, the best upper bound for the minimum folding length.

Example 2.1: A ruler with minimum folding length $2m-\epsilon$.

Consider a ruler which has $n = 2k-1$ links L_1, ..., L_n. Suppose that links with odd subscripts have length m and that links with even subscripts have length $m-\epsilon$, where $\epsilon = m/k$. It is easy to check that this ruler cannot be folded into length less than $2m-\epsilon$. □

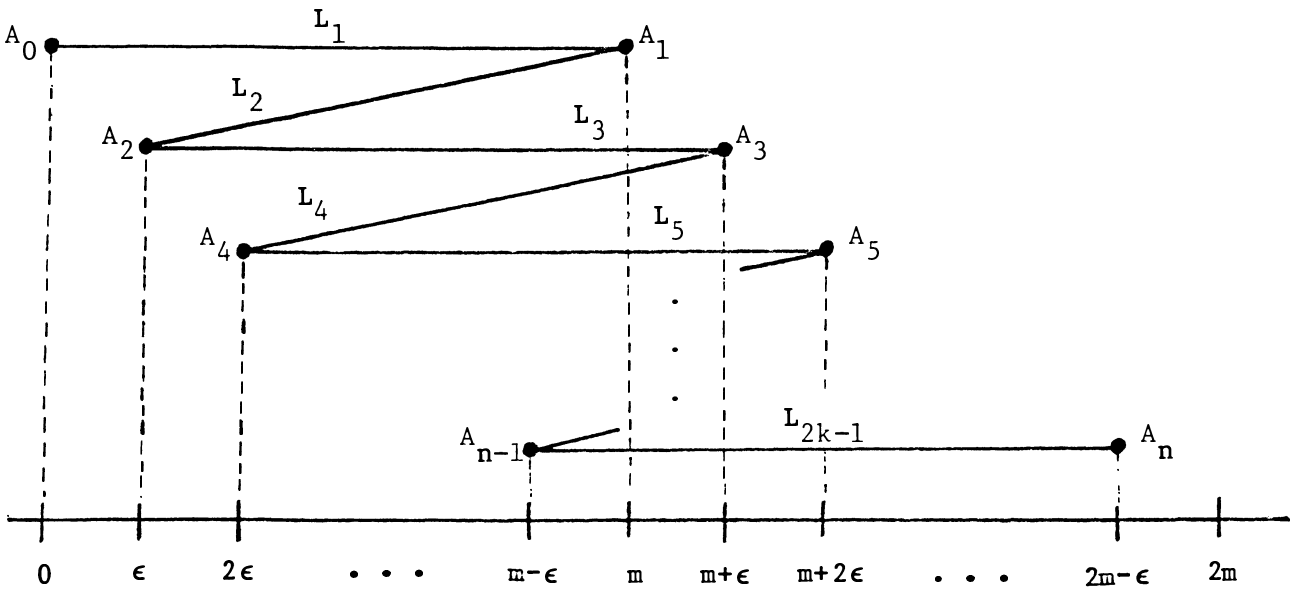


Fig. 2.2: The ruler of Example 2.1.

Having established some basic results about folding rulers, we now return to the original problem of moving such objects.

3. Moving an Arm in Two Dimensions

The remainder of this paper is concerned with moving a ruler that has one endpoint, A_0, pinned down. We will refer to such a ruler as an arm.

Unrestricted Movement

It is easy to find out what points can be reached by the free end of an arm placed in the plane. The answer is given in the next lemma, whose simple proof we omit. (The lemma extends readily to three dimensions.)

Lemma 3.1: Let $L_1, \ldots, L_n$ be an arm positioned in 2-dimensional space, and let $r = \sum_{i=1}^{n} \mathbf{l}_i$, the sum of the lengths of the links. Then the set of points that A_n can reach is a disc of radius r centered at A_0 -- unless some $\mathbf{l}_i$ is greater than the sum of the other lengths. In that case, the set of points A_n can reach is an annulus with center A_0, outer radius r, and inner radius $\mathbf{l}_i - \sum_{j \neq i} \mathbf{l}_j$.

Restricted Movement

If an arm is constrained to avoid certain specified objects during its motions, then determining whether A_n can reach some given point p is difficult. In the following example, we use a reduction of RULER FOLDING to show that even for "walls" consisting of a few straight line segments, this problem can be NP-hard.

Example 3.1: A hard decision problem.

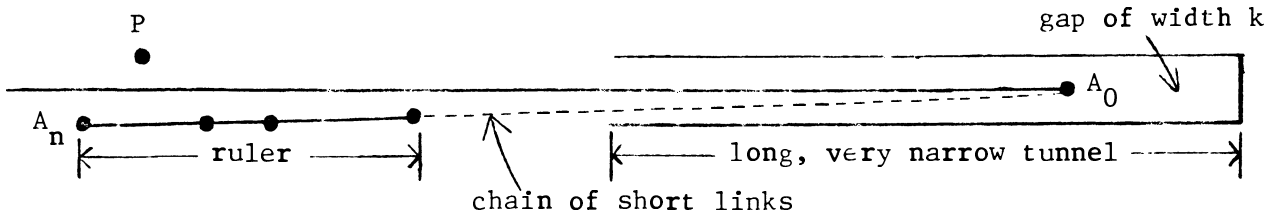


Fig. 3.1: A point that is hard to reach.

We want to know whether the arm shown in Fig. 3.1 can be moved so that A_n reaches the given point p. The arm consists of a ruler with links of integral lengths attached to a chain of very short links. The chain links are short enough to turn freely inside the tunnel, which is sufficiently narrow that links of the ruler can rotate very little once they are inside. Since the ruler cannot change its shape very much while moving through the tunnel, it must be foldable into length at most k in order to move through the gap of width k. Thus, point p can be reached if and only if the ruler can be folded into length at most k. □

We would like to find natural classes of regions for which questions concerning the movement of arms are decidable in polynomial time. Certainly the simplest such region is the inside of a circle, since there are no corners in which an "elbow" might be caught. We believe that studying motions inside a circle sheds light on the underlying movements of the arm without the complexities that arise in situations where a link can jam in a corner. For the remainder of this paper, we will discuss polynomial algorithms for moving an arm within a circle. In a subsequent paper, we hope to treat more general situations.

4. Changing Configurations Inside a Circle

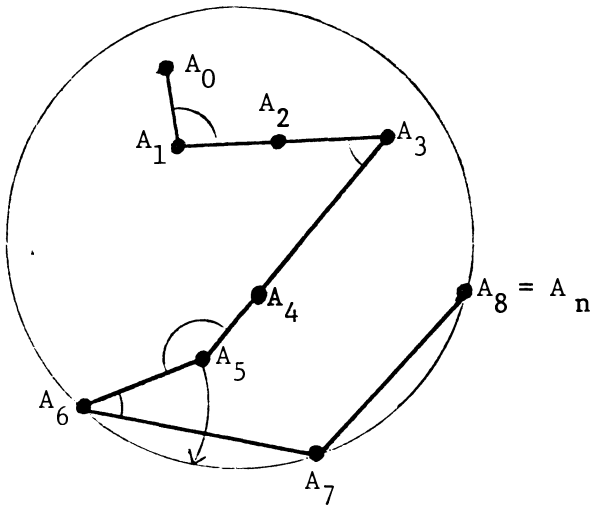
In this section, we solve the problem of moving an arm from one given configuration to another inside a circular region. Simply determining whether this can be done turns out to be a matter of checking that links whose "orientations" differ in the two configurations can be reoriented. This checking can be done in time proportional to the number of links. Assuming that it is feasible to change configurations, we show how to move the arm to its desired final position by first moving it to a certain "normal form" and then putting each link into place, correcting its orientation if necessary. Correcting orientation involves destroying and then restoring the positions of previous links. Our algorithm consists of a sequence of "simple motions" (which we are about to define), and the length of this sequence is on the order of the cube of the number of links.

Simple Motions

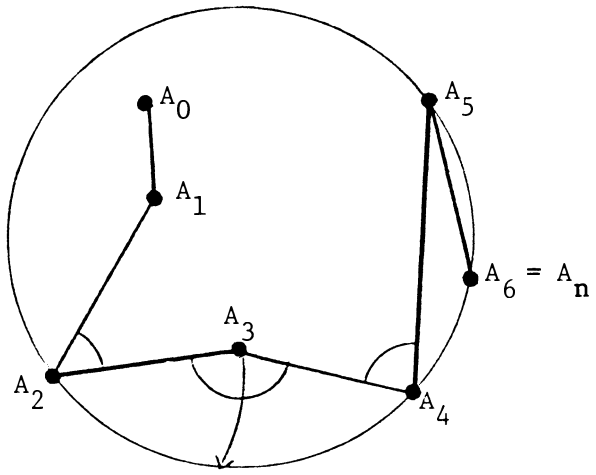
A definition of a "simple motion" is needed in order to make clear the sense in which our algorithms for moving an arm are polynomial. This definition should not limit the positions the arm can reach nor should it complicate the algorithms and proofs. With these considerations in mind, we define a "simple motion" of an arm as follows. (There are many other definitions which would give similar results.)

Definition 4.1: A simple motion of an arm is a continuous motion during which at most four joint angles change. (The angle between the first link and some reference line through the fixed point A_0 may be one of these.) Moreover, a changing angle is not allowed both to increase and to decrease during one simple motion.

Fig. 4.1 illustrates some simple motions of the type we use. Note that in the motions shown, the joints where angles are changing are connected together by straight sections of the arm. This is true of all the simple motions we will use.



A_5 is moving to the circle by a simple motion. The locations of A_0, A_1, A_6, A_7, and A_8 remain fixed. The angles at A_1, A_3, A_5, and A_6 are changing.



A_3 is moving to the circle by a simple motion. The locations of A_0, A_1, and A_2 remain fixed. A_4, A_5, and A_6 move first counter-clockwise, then clockwise around the circle. Only the angles at A_2, A_3, and A_4 are changing.

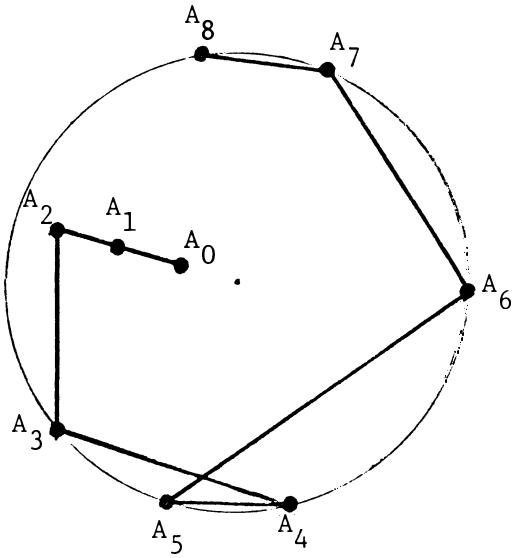
Fig. 4.1: Examples of simple motions.

Normal Form

It is convenient to begin by showing that any arm positioned within a circle can be moved by a short sequence of simple motions into a normal form

that has as many joints as possible positioned on the circle. We immediately dispense with the case in which the distance from A_0 to the circle is greater than the length of the entire arm, since in this case the circle is irrelevant.

Definition 4.2: Suppose A_0 is fixed at some point distance d_0 from the circle, and suppose that j is the smallest integer such that $\sum_{i=1}^{j} l_i \geq d_0$. Then the arm is in normal form if and only if $L_1, \ldots, L_j$ contains at most one bent joint, and for each k, $j \leq k \leq n$, A_k is on the circle. Moreover, if $L_1, \ldots, L_j$ is bent, the bend is at joint A_{j-1}. (See Fig. 4.2.) In any event, $L_1, \ldots, L_{j-1}$ lie on a radius.



A_0, A_1, and A_2 lie on a radius. A_3 is the first joint that can reach the circle. The successors of A_3 lie on the circle.

Fig. 4.2: An arm in normal form.

Lemma 4.1 (Normal Form): For any given configuration of an arm within a circle there is a sequence of O(n) simple motions that moves the arm to normal form. Moreover, this sequence can be computed in O(n) time.

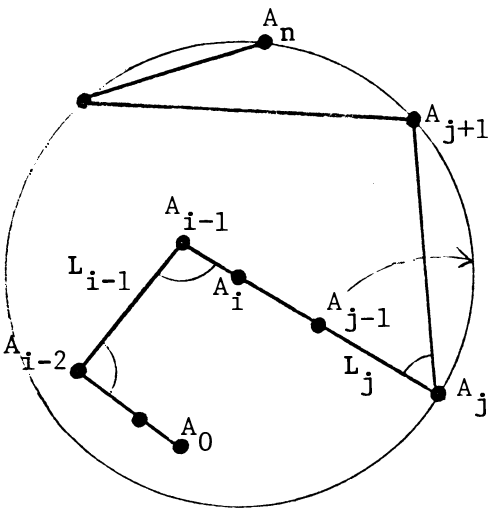
Proof: The process consists of two stages. First, the tail will be straightened until A_n reaches the circle. Then, starting with A_{n-1}, the other

joints will be moved one by one onto the circle.

Suppose L_j, L_{j+1}, ..., L_n form a straight line segment. Move A_n toward the circle by rotating this segment about A_{j-1} until A_n reaches the circle or L_{j-1} is added to the straight segment. In this latter case, rotate the extended straight segment about A_{j-2}. Eventually, A_n reaches the circle or the entire arm becomes a straight segment that can be rotated about A_0 to place A_n on the circle. (Recall that we are assuming that the arm is long enough to reach the circle.) This process requires at most O(n) simple motions and can be computed in O(n) time.

Now assume that A_n, A_{n-1}, ..., A_j are on the circle, and let L_i, L_{i+1}, ..., L_j be the maximal straight segment leading back from A_j. Keeping L_i, L_{i+1}, ..., L_{j-1} straight and the positions of A_j and A_{i-2} fixed, rotate L_j about A_j moving A_{j-1} away from A_{i-2}. (See Fig. 4.3.) L_j is rotated until A_{j-1} hits the circle (in which case we have a new joint on the circle), or L_{i-1} is added to the straight segment L_i, ..., L_{j-1}, or A_{i-1} hits the circle. If L_{i-1} is added to the straight segment, then the process of rotating L_j is continued with the straight segment replaced by a new one containing at least L_i, ..., L_{j-1} and L_{i-1}. If A_{i-1} hits the circle, then A_{i-1} is held fixed while the angles at joints A_{i-1}, A_{j-1} and A_j are adjusted so as to push A_{j-1} to the circle while keeping A_j and its successors on the circle. In this way, one can force onto the circle as many joints as possible (i.e., A_j can be placed on the circle, where j is minimum such that the sum of the lengths of the first j links exceeds the distance from A_0 to the circle). Once these joints are on the circle, it is easy to position the links at the beginning of the arm as desired. This process requires O(n) simple motions and once again, these motions can be computed in O(n) time. Thus, a total of O(n) simple

motions is needed to put an arm into normal form, and O(n) time is needed to compute the motions. □

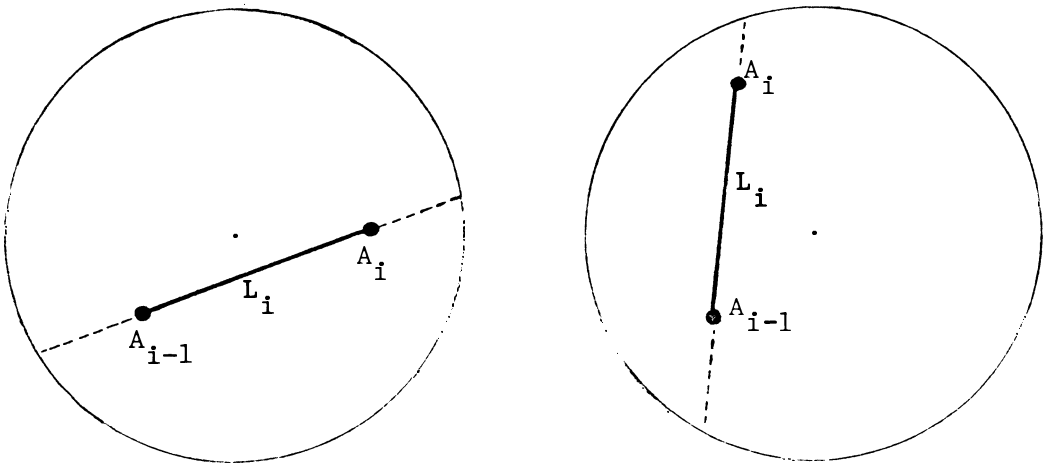


A_{j-1} moves toward the circle away from A_{i-2}. The locations of A_{i-2} and its predecessors and the locations of A_j and its successors remain fixed. Only the angles at A_{i-2}, A_{i-1}, A_{j-1}, and A_j are changing.

Fig. 4.3: Moving an arm to normal form.

Reorientation of Links

For any given position of an arm inside a circle, we define each link to have either "left" or "right" orientation. This is done by first observing that the straight line extension of a link L_i cuts the circle into two arcs. L_i is said to have left orientation if the arc on the left of the extension, viewed from A_{i-1} to A_i, is no longer than the arc on the right. Right orientation is defined in a similar manner. (See Fig.4.4.) Note that a link that is on a diagonal of the circle can be regarded as having either orientation and that a link must move to a diagonal in order to change orientation.



left orientation right orientation

Fig. 4.4: Link orientations.

An obvious necessary condition for being able to move the arm from one configuration to another is that it be possible to reorient each link whose orientation differs in the two configurations. (It turns out that this condition is also sufficient.) We are about to show that determining whether a link can be reoriented is simply a matter of determining how far its endpoints can be moved from the circle.

For an arm with A_0 fixed within a circle C, let c_i and d_i denote the minimum and maximum distance that A_i can be moved from C by arbitrary motions of the arm within C. Of course, distance is measured along a radius of C, so $0 \le c_i \le d_i \le d/2$, where d is the diameter of C.

Since A_0 is fixed, c_0 and d_0 are determined by the position of A_0. The Normal Form Lemma (4.1) shows that each successive A_i can get closer to the circle by the amount $\mathbf{l}_i$ until the circle is reached. Thus,

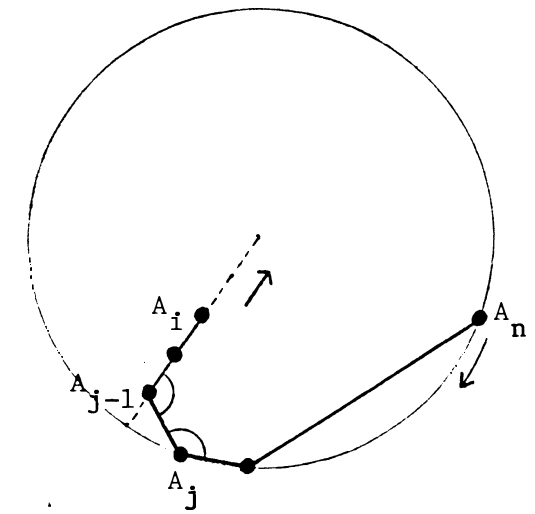
$$c_i = \max \{c_{i-1} - \mathbf{l}_i, 0\}.$$

Computing the d_i's is slightly more complicated. We begin by computing for each i, $0 \le i \le n$, the maximum distance t_i that A_i could move from the circle if it were constrained only by the tail of the arm (i.e., if $L_{i+1}, \ldots, L_n$ were freed from $L_1, \ldots, L_i$ and $L_1, \ldots, L_i$ were discarded). Then we compute d_i from t_i and d_{i-1}.

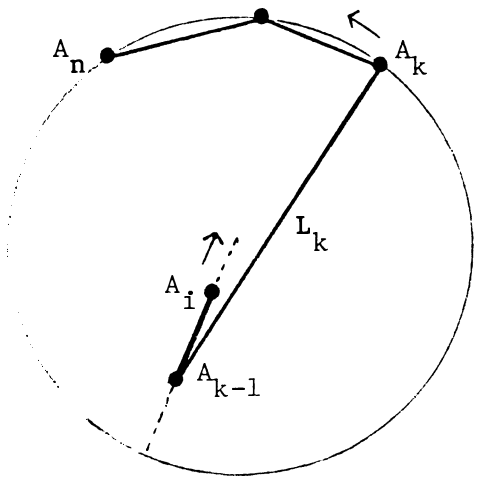
Lemma 4.2: For any arm $L_1, \ldots, L_i, \ldots, L_n$ inside a circle of diameter d,

$$t_i = \begin{cases} d/2 & \text{if no link beyond } A_i \text{ is longer than } d/2; \\ \min\{d/2,\ d - \mathbf{l}_k + \sum_{i<j<k} \mathbf{l}_j, \text{ where } \mathbf{l}_k \text{ is the length of the} \\ \quad \text{first link beyond } A_i \text{ longer than } d/2\} & \text{otherwise.} \end{cases}$$

Proof: Think of the links beyond A_i as an arm with A_i fixed. Move this arm to normal form. Let A_j be the first joint on the circle. If $j \ge i+2$, the straight section of arm between A_i and A_{j-1} lies on a radius of the circle. (If $j = i$ or $i+1$, this section is just the point A_i.) While changing only the angles at joints A_{j-1} and A_j, one can push this straight section along the radius toward the circle's center while A_j and its successors move around the circle. (See Fig. 4.5.) New links are added to the moving straight section until A_i reaches the center or the first long link L_k prevents further travel because it has folded against the straight section (or reached the diagonal in the case $L_k = L_{i+1}$). □



$A_0, \ldots, A_{i-1}$ have been removed. $A_i, \ldots, A_{j-1}$ move along the radius while $A_j, \ldots, A_n$ move around the circle. Only the angles at A_{j-1} and A_j are changing.



Joint A_{k-1} is about to fold completely, preventing further travel of A_i along the radius.

Fig. 4.5: Moving A_i distance t_i from the circle.

Now that we have calculated the t_i's, it is easy to calculate the d_i's. For $i > 0$:

$$d_i = \begin{cases} \min\{t_i, d_{i-1} + \mathbf{l}_i\} & \text{if } \mathbf{l}_i < d/2 - d_{i-1}; \\ \min\{t_i, d/2\} & \text{if } d/2 - d_{i-1} \le \mathbf{l}_i \le d/2 - c_{i-1}; \\ \min\{t_i, d - \mathbf{l}_i - c_{i-1}\} & \text{if } \mathbf{l}_i > d/2 - c_{i-1}. \end{cases}$$

For any given distance x between c_i and d_i, there is obviously some way

to move A_i to a position that is distance x from the circle. The point of the next remarks and lemma, which we need before we can give an algorithm for reorienting the links of an arm, is that this can be done using a short sequence of simple motions.

Remark 4.1: Suppose that the tail $L_{j+1}, \ldots, L_n$ has been detached from the arm $L_1, \ldots, L_n$. Then note that this tail can be moved from its initial position so that the distance between A_j and the circle monotonicly increases or decreases. To see this, put the tail (regarded as an arm with initial point A_j fixed) into normal form. Then move the straight segment of links containing A_j along the radius on which it lies, adding or deleting links from the segment as A_i gets closer to or farther from the center of the circle. □

Remark 4.2: Consider the arm as a whole, and suppose the tail beginning at A_j is in normal form. Then L_j can be rotated about A_{j-1} to push A_j closer to or farther from the circle while the angles at A_j and two other joints in the tail are adjusted to keep the tail constantly in normal form. In fact, Remark 4.1 shows that any rotation of L_j for which the distance between A_j and the circle is either an increasing or a decreasing function can be carried out in at most n-j simple motions. □

Lemma 4.3: Let A_j be a joint of an n-link arm positioned within a circle. For any x between c_j and d_j, there is a sequence of $O(n^2)$ simple motions that moves the arm from its original position to a position in which A_j is distance x from the circle.

Proof: Compute the c_i and d_i for each predecessor A_i of A_j. Then, given x, compute the sequence of numbers defined by the following recursive formula:

$$x_i = x \quad \text{for} \quad i = j;$$

$$x_{i-1} = \max\{c_{i-1}, x_i - \mathbf{l}_i\} \quad \text{for} \quad 2 \le i \le j.$$

(Note that $c_i \le x_i \le d_i$.) To position A_j distance x_j from the circle, first put the entire arm into normal form (O(n) steps). Then, beginning with A_1, move each A_i in turn to a position distance x_i from the circle. This is done by rotating L_i about A_{i-1} while keeping the tail in normal form. All together, at most $(n-1)+(n-2)+\cdots+(n-j)$ additional simple motions are needed, so the entire repositioning sequence contains $O(n^2)$ motions. Note that this sequence can be computed in $O(n^2)$ time. □

We are now ready to give the conditions under which links can be reoriented.

Lemma 4.4: A link L_i can be reoriented if and only if at least one of the following inequalities holds:

i) $d - \mathbf{l}_i \le d_{i-1} + d_i$;

ii) $d_i \ge \mathbf{l}_i + c_{i-1}$;

iii) $d_{i-1} \ge \mathbf{l}_i$.

Furthermore, if L_i can be reoriented, then this can be done with $O(n^2)$ simple motions that can be quickly computed.

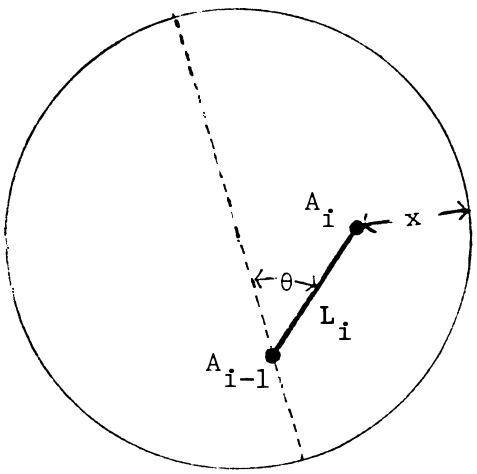
Proof: As we noted at the beginning of this subsection, L_i must lie on a diagonal in order to be reoriented. Hence, the above conditions are obviously necessary because i) holds when L_i is on a diagonal and the center of the circle is between A_{i-1} and A_i, ii) holds when L_i lies on a radius with A_i closer to the center than A_{i-1}, and iii) holds when L_i lies on a radius with A_{i-1} closer to the center than A_i.

To prove that the conditions are also sufficient, first suppose that inequality i) holds. Using the method in the proof of Lemma 4.3, move A_{i-1} to a position distance d_{i-1} from the circle in $O(n^2)$ simple motions. If inequality iii) holds, move A_{i-1} to a position distance d_{i-1} from the circle, again using $O(n^2)$ simple motions. After this has been done, hold A_{i-1} fixed, and rotate L_i about A_{i-1} to bring L_i to the radius through A_{i-1}. By Remark 4.2 this takes at most n-i simple motions, and these can be quickly computed.

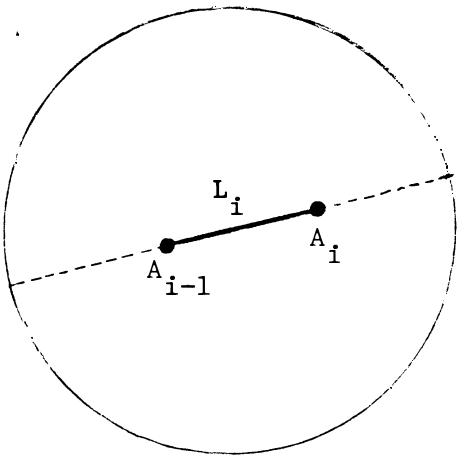
If inequality ii) holds, then $c_{i-1} \leq d/2 - \mathbf{1}_i \leq d_{i-1}$. Move A_{i-1} distance $d/2 - \mathbf{1}_i$ from the circle, and then rotate L_i to the diagonal. □

We need to make one more observation before we can show how to change configurations.

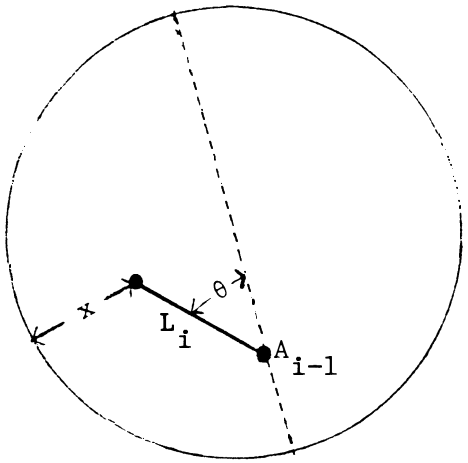
Remark 4.3: Suppose L_i is a link that can be reoriented. Then starting from any initial configuration of the arm, we can reorient L_i and with $O(n^2)$ additional motions, return $A_1, \ldots, A_{i-1}$ to their starting positions without changing the new orientation of L_i. To see this, bring L_i to a diagonal with $O(n^2)$ simple motions, and then "undo" these motions but with the orientation of L_i reversed. That is, keep the angle at A_{i-1} adjusted so that at corresponding moments before and after L_i reaches the diagonal through A_i, L_i forms the same angle with this diagonal but lies on the opposite side of it. This keeps A_i the same distance from the circle at corresponding times. (See Fig. 4.6.) To check that the tail can be moved in a compatible fashion, note that reversing the changes in the size of the angles in the tail indeed keeps A_i the same distance from the circle at corresponding times. Although the tail does not return to its original position, it does return to its original shape. □



At time $t_0 - t$, L_i forms an angle θ with the diagonal through A_{i-1}, and A_i is distance x from the circle.



At time t_0, L_i reaches a diagonal.



At time $t_0 + t$, A_{i-1} has returned to the position it occupied at time $t_0 - t$. L_i again forms angle θ with the diagonal through A_{i-1}, but has changed orientation. The distance between A_i and the circle is again x.

Fig. 4.6: Reorientation of a link L_i with restoration of $A_1, \ldots, A_{i-1}$.

An Algorithm for Changing Configurations

Suppose we are given an initial configuration and a desired final configuration of an arm within a circle. Using the formulas of the preceding subsection, we can quickly compute the c_i's, d_i's, and t_i's. Using Lemma 4.4, we can then quickly check whether each link with differing initial and final configuration can be brought to the diagonal. If this necessary and sufficient condition holds, then the following motion algorithm shows that the arm can be moved to the desired final configuration with $O(n^3)$ simple motions.

Algorithm 4.1: Algorithm for Changing Configuration

Step i) Move the arm to normal form (O(n) simple motions);

Step ii) Once the predecessors of A_i are in their final positions, reorient L_i if necessary, restoring the predecessors of A_i to their final positions ($O(n^2)$ motions, by Remark 4.3). Then rotate L_i about A_{i-1} to put A_i in final position (n-i simple motions, by Remark 4.2). Increment i, and repeat Step ii) until $i > n$. □

Notice that since the c_i's and d_i's depend only on the l_i's, the very existence of the desired final configuration assures us that the distance from A_i to the circle will stay between c_i and d_i while L_i is being rotated about A_{i-1}. This is because the distance between A_i and the circle changes monotonicly during this rotation.

Notice also that the question of whether the desired final configuration can be attained can be answered in linear time on a machine that does real arithmetic (+, -, *, /2, min(,)) since it is necessary only to compute the c_i's, d_i's, and t_i's, determine the links which must be reoriented, and check

that the conditions of Lemma 4.4 hold for these links.

In the next section, we show how to reduce the problem of reaching a given point with A_n to a problem of changing configurations.

5. Reaching a Point with an Arm Inside a Circle

In this section, we will solve the problem of deciding whether an arm inside a circle can be moved from a given initial position to one which places A_n at some given point p. We will do this by showing that this problem can be reduced to the problem of changing configurations, which we solved in the last section.

Points on the Circle Reached by the A's

We want to compute a feasible configuration (i.e., one to which the arm can be moved from its initial configuration) that places A_n at a given point p (inside or on the circle). In order to find such a configuration, we first construct the set R_j of points on the circle that can be reached by A_j from the given initial position of the arm.

Lemma 5.1: Each R_j consists of at most two arcs of the circle.

Proof: (Induction on j) Clearly, $R_0 = \{A_0\}$ if A_0 is on the circle. Otherwise, the Normal Form Lemma 4.1 shows that the first non-empty R_j is the one for which

$$\mathbf{1}_1 + \cdots + \mathbf{1}_{j-1} < c_0 = d_0 \leq \mathbf{1}_1 + \cdots + \mathbf{1}_j ,$$

and that all subsequent R_j's are non-empty. It is easy to see that the first non-empty R_j consists of at most two arcs.

Now consider a j for which R_{j-1} is nonempty but consists of at most two arcs. If A_j is at some point in R_j, we can move A_{j-1} to the circle while moving A_j around the circle. (This can be done in the same way that an arm is put into normal form.) Of course, A_j stays in R_j during this process. Thus, each point in R_j belongs to an arc of R_j that contains a point reached by A_j with A_{j-1} in R_{j-1}. Hence, counting the number of arcs in R_j is equivalent to counting how many of its arcs contain a point that A_j can reach with A_{j-1} in R_{j-1}.

Suppose that A_{j-1} and A_j are on the circle and that $d_{j-1} \geq \mathbf{1}_j$. Then we can reorient L_j while moving A_j around the circle, keeping A_j in R_j. Our observation about counting arcs shows that each arc of R_{j-1} gives rise to only one arc in R_j. Thus in this case, R_j consists of at most two arcs.

Now suppose that A_{j-1} and A_j are on the circle and that $d_{j-1} \leq \mathbf{1}_j$. Then we can move A_{j-1} from any point in R_{j-1} to any other point in R_{j-1} without ever taking A_j off the circle or changing the orientation of L_j. Hence, all the points of R_j that are reached from R_{j-1} by L_j with left orientation are in the same arc of R_j. The same is true for L_j with right orientation, so again R_j consists of at most two arcs. □

In our algorithm for reaching a point p, we will need to find for any given point in R_j a feasible configuration of the arm that positions A_j at that point. In the next section, we show how to compute this information quickly.

Determining the R's

First we will show that each set R_j is a union of certain contributions from its predecessors, and then we will describe an algorithm for calculating

the R_j's and determining how to reach them.

The following lemma, whose proof we omit, can easily be established using the ideas in the proof of the Normal Form Lemma 4.1.

Lemma 5.2: Suppose an arm is positioned inside a circle so that A_j is located at a point p_j on the circle. Then A_j can be kept fixed at p_j while the arm is moved to a position where one of the following conditions holds:

i) links $L_1, \ldots, L_j$ form either a straight line (with no folds) or an "elbow" whose only bend is as A_{j-1};

ii) for some $i < j$, A_i is on the circle, and links $L_{i+1}, \ldots, L_j$ form either a straight line or an elbow whose only bend is at A_{j-1}.

Given a value for j, we need to find out for each R_i, $i < j$, which points of R_j can be reached from R_i by the straight lines and elbows of Lemma 5.2.

Suppose that p_i is a point in R_i and that $\mathbf{l}_{i+1} + \cdots + \mathbf{l}_j \leq d$. If all the links between A_i and A_j can be given the same orientation, then p_i contributes a point to R_j by means of a straight line. (If both orientations are possible, then p_i contributes two points to R_j.) Contributions of this type from points in R_i form at most four arcs, two for each arc of R_i. These arcs amount to shifts of R_i around the circle.

Now consider the possibilities for joining a point p_i in R_i to a point p_j in R_j by an elbow whose last joint is the one which is bent. Certainly $\mathbf{l}_{i+1} + \cdots + \mathbf{l}_{j-1}$ must be at most d. Since L_j and the straight line from A_i to A_{j-1} might have either orientation, there are four types of elbows to consider. Consider a particular feasible elbow, and note that it must place A_{j-1} somewhere on an arc of a circle of radius $\mathbf{l}_{i+1} + \cdots + \mathbf{l}_{j-1}$ centered

at p_i. Since the orientations of the links in the elbow are specified, this arc is bounded by the circle at one end and by the diagonal through A_i at the other. The set of points that can then be reached by L_j in its specified orientation, with A_{j-1} on the arc, forms an arc on the circle. Hence, each feasible elbow type allows R_i to contribute a widened shift of itself to R_j.

The contributions of A_0 to R_j can be determined in a similar fashion.

It is now easy to give an $O(n^2)$ algorithm to do the following: compute the endpoints of the R_j's, and build a table that allows one, given a p_j in R_j, to find in O(n) time (where n is the number of links in the arm) a feasible configuration having A_j at p_j.

Algorithm 5.1: Finding R's

First, determine how the links can be oriented (O(n) time). Next, compute the contributions from A_0 of straight lines and elbows whose last joint is the one that is bent. Record these contributions by listing the endpoints of the arcs together with the description of the lines or elbows that generated them (O(n) time). At this stage, the first non-empty R_i has been completely determined, and so its endpoints (of which there are at most four) can be computed (O(n) time). Finally, for each R_i in turn, compute the contribution of R_i to its successors, and then compute the endpoints of R_{i+1} (O(n) time per iteration). □

In the next subsection, we use the information about the R_j's to solve the problem of moving A_n to an arbitrary point inside the circle.

How to Reach a Point

If we want to place A_n at a point p on the circle, we merely compute R_n

and test p for membership. If p is in R_n, we use the table generated by Algorithm 5.1 to determine a feasible arm configuration that has A_n at p. Then we can use Algorithm 4.1 to move the arm to this configuration.

Now suppose p is inside the circle. If the arm can be moved to a configuration in which A_n is at p and some other joint is on the circle, then p can be reached by a feasible configuration in which some A_i is on the circle and links L_{i+1}, ..., L_n form either a straight line or an elbow with the bend at A_{i+1}. To see whether this happens, we compute the R_j's and then look for an appropriate straight line or elbow reaching from p back to a non-empty R_j. If no such line or elbow can be found, we check to see whether p can be reached by a configuration that does not touch the circle.

Lemma 5.3: Suppose that an arm L_1, ..., L_n can be moved to a configuration in which A_n is at a given point p inside the circle, but that no such feasible configuration can have any joint on the circle. Then the arm can be moved to a configuration in which A_n is at p and at most two joints are bent.

[redacted]: Consider a feasible configuration with A_n at p. If it has more than two bends, proceed as follows. Let A_i, A_j, and A_k, where $0 < i < j < k < n$, denote the first three bent joints. Let A_m denote the fourth bent joint if one exists; otherwise, set $A_m = A_n$. Keeping A_k and its successors pinned down, rotate the line of links between A_0 and A_i about A_0 so that A_i moves <u>away</u> from A_m. (See Fig. 5.1.) Eventually, one of three events must occur:

i) some joint straightens (in which case we can start over with a smaller number of bends);

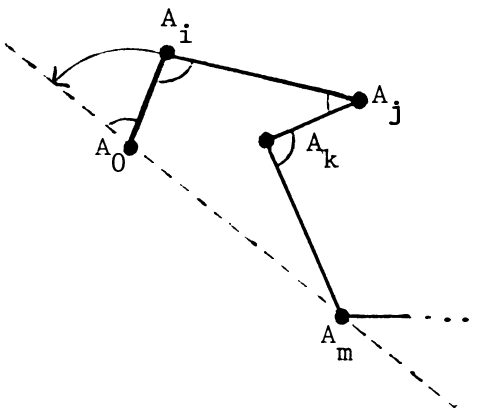
ii) A_i moves close enough to A_k to fold the joint A_j completely;

iii) A_i reaches the line through A_0 and A_m.

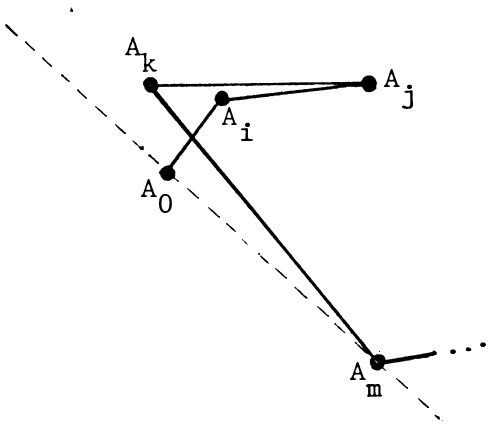
Note that by hypothesis, no joint can hit the circle.

If ii) occurs, keep joint A_j folded, unpin A_k, and continue the rotation. Since A_i is moving away from A_m, the rotation can continue until joint A_k straightens or A_i reaches the line through A_0 and A_m.

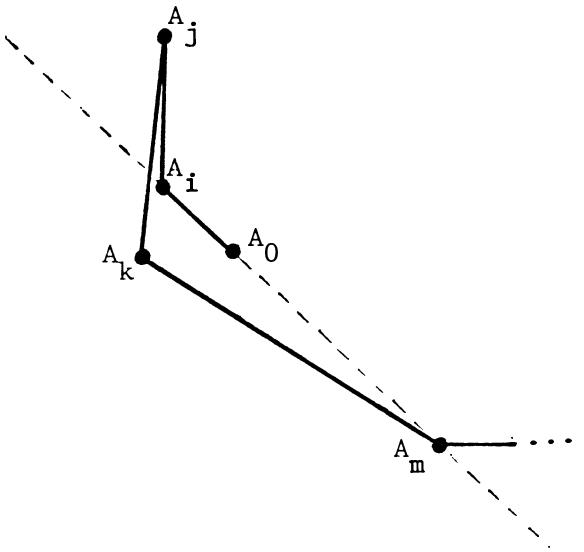
Assume that A_i, A_0, and A_m are collinear. Pin down A_0, ..., A_i and A_m, ..., A_n, and rotate the line of links between A_i and A_j about A_i so that A_j moves <u>away</u> from A_m. One of the joints A_i and A_k <u>must</u> straighten during this rotation. □



The locations of A_k and its successors are held fixed while A_i is rotated about A_0 away from A_m. Joint A_i or A_j may straighten, A_i may reach the line through A_0 and A_m, or . . .



joint A_j may fold, preventing continued rotation of A_i about A_0.



Then A_k is unpinned, joint A_j is kept folded, and the rotation is continued until A_i reaches the line through A_0 and A_m.

Fig. 5.1: Reaching p with at most two bent joints.

There are $O(n^2)$ configurations of the type described in Lemma 5.3, and

each one can be tested for feasibility in constant time. All together, then, we need $O(n^2)$ time to compute the R_j's, $O(n)$ additional time to check for a feasible configuration with some joint on the circle, and if no such configuration exists, $O(n^2)$ time to check for feasible configurations with no joint on the circle. If a feasible configuration is found, we can then use Algorithm 4.1 to move A_n to p with $O(n^3)$ simple motions. Note that our method can be used to solve the problem of moving any arbitrary joint A_j to a specified point.

References

[1] Garey, Michael R., and David S. Johnson. Computers and Interactability, A Guide to the Theory of NP-Completeness. W.H. Freeman and Company, San Francisco, California, 1979.

[2] Lozano-Perez, Thomas. Automatic Planning of Manipulation Transfer Movements. M.I.T. Artificial Intelligence Laboratory, A.I. Memo 606, December 1980.

[3] Reif, J. Complexity of the Mover's Problem and Generalization. Proceedings 20th IEEE Symposium on the Foundations of Computer Science, 1979, pp. 421-427.

[4] Schwartz, Jacob T., and Micha Sharir. On the 'Piano Movers' Problem I. The Case of a Two-dimensional Rigid Polygonal Body Moving Amidst Polygonal Barriers. Department of Computer Science, New York University, TR 39, October 1981.

[5] Schwartz, Jacob T., and Micha Sharir. On the 'Piano Movers' Problem II. General Techniques for Computing Topological Properties of Real Algebraic Manifolds. Department of Computer Science, New York University, TR 41, February 1982.