ON THE MOVEMENT OF ROBOT ARMS IN 2-DIMENSIONAL BOUNDED REGIONS*

John Hopcroft, Deborah Joseph and 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. In this paper, we investigate a 2-dimensional mover's problem in which the object being moved is a robot arm with an arbitrary number of joints. We reduce the mover's problem for arms constrained to move within bounded regions whose boundaries are made up of straight lines to the mover's problem for a more complex linkage that is not constrained. We prove that the latter problem is PSPACE-hard even in 2-dimensional space and then turn to special cases of the mover's problem for arms. 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.

ON THE MOVEMENT OF ROBOT ARMS IN 2-DIMENSIONAL BOUNDED REGIONS*

John Hopcroft, Deborah Joseph and Sue Whitesides**

Computer Science Department, Cornell University

1. Introduction

With current interests in industrial automation and robotics, it is becoming increasingly important to design efficient algorithms for moving 2- and 3-dimensional objects subject to certain geometric constraints. In this paper, we will be concerned with the mover's problem, which is to determine, given an object X, an initial position P_1, a final position P_I and a constraining region R, whether X can be moved from position P_1 to position P_I while keeping X within the region R. Here, X could represent an object being moved by some sort of manipulator, or it could represent the manipulator itself; the constraints determining region R could arise either from the presence of other objects in the work space or from the boundary walls of the space. Recently, several authors (Schwartz and Sharir [10,11], Reif [9], Lozano-Perez [7]) have studied the mover's problem for the situation in which X is a rigid 2- or 3-dimensional polyhedral object and R is a region described by linear constraints.

A more difficult problem is to assume that X has joints and hence is nonrigid. This problem is directly related to the design and control of robot arms as well as to the planning of obstacle-avoiding paths for objects being moved. Again, one desires a fast (polynomial time) algorithm for moving X from position P_1 to P_I within a region R. Unfortunately, such an algorithm is probably not possible, as Reif [9] has shown that the problem of deciding whether or not an arbitrary hinged object can be moved from one position to another in a 3-dimensional, nonconvex region is PSPACE-complete.

The motion of hinged objects called linkages, which are collections of rigid rods arbitrarily fastened together by pivots at their endpoints, was extensively studied in the 1800's. Of special interest was the synthesis problem, which is to design a mechanism such that the locus of points reachable by a designated point on the linkage is some specified curve. Many solutions were given for the case in which the curve is a straight line, and we will make use of one of them. Also, we will build on work of Kempe [6] that showed that certain simple computations such as multiplication can be performed by linkages.

Our first main result is that, given a linkage within a region with straight line boundaries, we can design a new linkage that has the boundaries incorporated into its design in the sense that the new linkage has a designated joint for each joint of the original linkage, and the locus for a designated joint of the new linkage without boundaries is exactly the locus of the corresponding joint in the old linkage in the presense of boundaries. In terms of robotics, the original linkage might represent a mechanical arm constrained to move in an enclosed space.

Unfortunately, as we will show, 2-dimensional linkages can simulate arbitrary space-bounded Turing machines. From this result it follows that the problem of deciding whether or not a joint in a linkage can reach a designated point is PSPACE-complete even in a 2-dimensional space with no constraining boundaries.

Because of the above results, it becomes

important to discover natural restrictions of the mover's problem for which fast, general algorithms can be found. In unrestricted 2- or 3-dimensional space, it is easy to solve the mover's problem for an arm, by which we mean a sequence of links joined consecutively at their endpoints by fully rotating joints, the first of which is fastened down. However, the problem of folding such an arm into a given length turns out to be NP-complete. Because of this, we can show that it is at least NP-hard to decide whether or not the end of an arbitrary arm can be moved from one position to another while staying within a given 2-dimensional region. The folding problem is still NP-complete for a sequence of joined links with both ends free, which we call a carpenter's ruler. (See Figure 1.1). The problem of folding a carpenter's ruler arises because a natural strategy for moving such an object in a confining region is to fold it up as compactly as possible at the beginning of the motion.

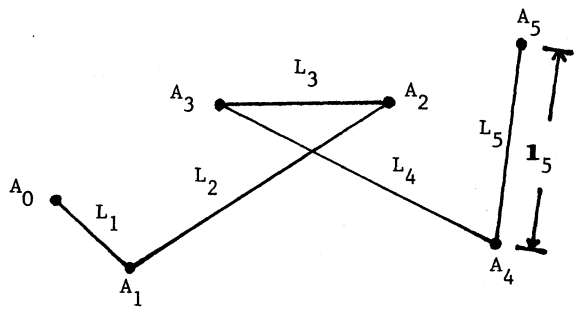


Figure 1.1: A Carpenter's Ruler. If the location of A_o is fixed, the ruler is called an arm. (L_1 still rotates about A_o).

Since the mover's problem is NP-hard for an arm moving in an arbitrary 2-dimensional region, it is natural to restrict the problem to convex regions. In the case that the region is the inside of a circle, we will give polynomial time algorithms for changing configurations and reaching points.

2. The Reachability Problem for 2-Dimensional Linkages

The arms that we discussed in Section 1 are a very simple form of linkage -- their links are joined together consecutively to form a chain. Historically, a great deal of importance has been placed on more complex types of linkages whose links are joined in an arbitrary fashion. Examples of 2-dimensional linkages of this type are straight-line motion devices and pantographs, which are mechanisms used to scale drawings. In this section of the paper, we discuss the reachability problem for arbitrary 2-dimensional linkages. The reachability problem is a simpler version of the mover's problem in which only the desired final position of one joint is specified.

We begin by showing how to polynomially reduce the reachability problem for any linkage constrained by a polygonal region to a reachability problem for a more complex linkage constrained only to keep certain joints at fixed locations. Unfortunately, we are also able to show that the reachability problem for linkages in 2-dimensional space without boundaries is PSPACE-hard. The proofs of these results are quite intricate, and we will only be able to sketch them here. The interested reader is referred to [4] for the full proofs.

Replacing Boundaries by Adding Links

We first consider the special case in which the region R that restrains linkage L is a convex polygon. In this case, R is the intersection of half-planes $H_1,\ldots,H_k$ corresponding to the sides of the polygon. Since R is convex, we know that all links of L lie in R whenever all joints lie in R, i.e., whenever all joints lie in each H_i. We would like to ensure that a joint J of L does not leave H_i by identifying J with a joint of some appropriate linkage L_i. Of course L_i should not restrict the motion of J inside R in any way. (It does not matter whether L_i itself lies in R since we are looking for a reduction of the original problem to a new problem without a bounding region.) If a linkage can be constructed for each joint and half-plane pair, then we can ignore region R and study the motions of the original joints of L with the new linkages attached.

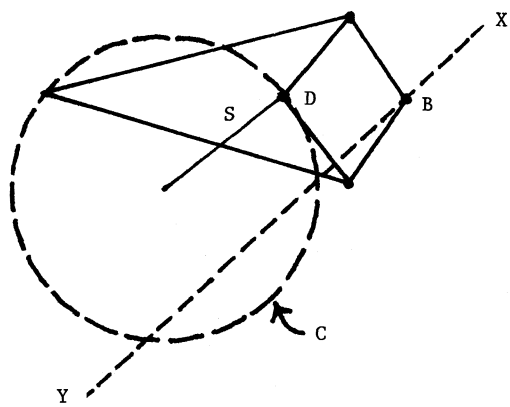


Figure 2.1: Peaucellier's Straight Line Motion Linkage.

The linkages we need can in fact be constructed by using a device, invented by Peaucelier [8] in 1864 that converts circular motion to linear motion. (See Figure 2.1.) As point D moves around the circle C, point B traces out the line segment XY. This linkage has the additional property that if the point D is allowed to move within the circle C by the creation of a joint at the midpoint of link S, then the region R of points that B can reach is the intersection of a circle and a half-plane. As the lengths of the links in the linkage are increased, but ratios of their lengths kept constant, the size of region R increases but its shape remains the same. The description of the new linkages can be computed in time on the order of a polynomial in the description of the original linkage L. (See [4] for details.) The formal statement of the result whose proof we have just outlined is given by the next theorem.

Theorem 2.1: Let L be a linkage constrained to lie within a convex polygonal region R. Then in time on the order of a polynomial in the length of the description of L and R, one can compute the description of a linkage M that contains L and has the following property: Given a point p in R, an initial position P_i of L in R, and a joint J of L, linkage L can be moved inside R to place J at p if, and only if, linkage M can be moved (without regard to R) to place J at p.

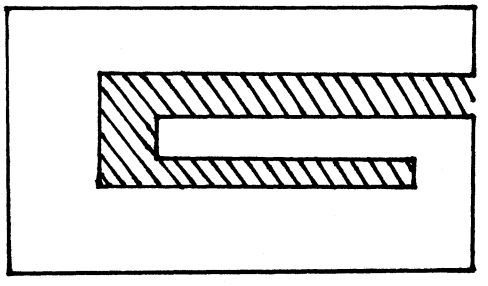


Figure 2.2: A Typical Nonconvex Region.

We can extend this result to include the case in which R is a bounded region, not necessarily convex, that can be divided into triangular subregions. (See Figure 2.2.)

We have already indicated how to restrict a joint to a triangular subregion, since its boundary is a convex polygon. The next step is to design a linkage that restricts a joint to the union of a set of such regions. This can be done by using techniques from Kempe [6]. There is, however, one more difficulty. Restricting the joints of a linkage to a nonconvex region does not guarantee that the links themselves will remain in the region. However, we are able to overcome this difficulty by partitioning the excluded portions of the convex hull of the region into triangles and then constructing mechanisms that exclude each link from each triangle. Again, the details appear in [4].

Unfortunately, we can show that even the reachability problem for linkages not restrained by regions is PSPACE-hard.

PSPACE-Hardness of the Unconstrained Reachability Problem for Linkages

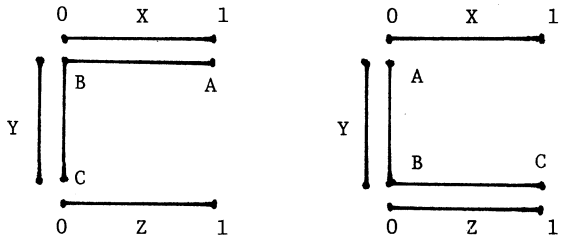
Our next result, whose proof is given in detail in [4], shows that removing boundaries does not help.

Theorem 2.2: Let L be a 2-dimensional linkage fixed to the plane at one or more joints but otherwise moving freely. Then the problem of deciding whether or not a designated joint of L can reach an arbitrary given point in the plane is PSPACE-hard.

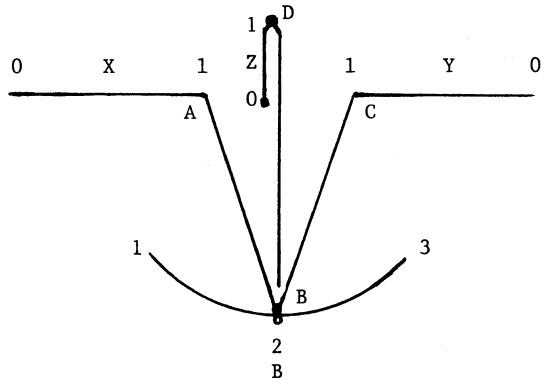
This result can be compared to Reif [9], where it is shown that the reachability problem for even a simple, tree-like linkage constrained by a 3-dimensional concave region is PSPACE-hard.

Our proof consists of showing that there are complex linkages that are capable of simulating Linear Bounded Automaton (LBA) computations and that the number of links needed to simulate a LBA on inputs of length n is linear in n. The PSPACE-hardness of the linkage reachabilty problem then follows from the fact that the acceptance problem for LBAs is PSPACE-complete.

The proof involves constructing linkages that compute the logical functions AND and NOT, and then from these, building up linkages that compute any Boolean function. (See Figure 2.3.) We then simulate a given LBA by using a linkage that acts as a pair of registers. One register is used to store an instantaneous description of the LBA at time t, the other to store the description at time t+1. (See Figure 2.4.) The linkages for Boolean functions are used to insure that information is correctly transferred from one register to the other.



NOT-Gate made from three straight-line motion devices. A, B and C are constrained to segments X, Y and Z respectively.



AND-Gate
B is at 1 when A=0 and C=1
B is at 2 when A=1 and C=1
B is at 3 when A=1 and C=0

Figure 2.3: An AND Linkage and a NOT Linkage.

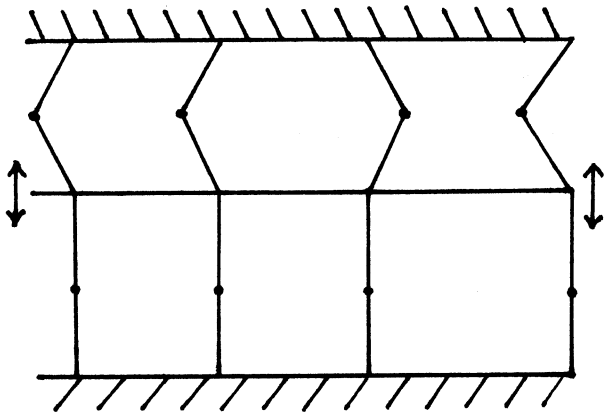


Figure 2.4: A Register Linkage.

In comparing our result to Reif's, a comment should be made. Reif's result is for 3-dimensional space, and it requires a constraining region that is not all of 3-space. However, Reif's linkage is very simple. In fact, a minor modification of his proof allows the linkage to be an arm. In contrast to this, our linkage is 2-dimensional and does not involve a constraining region, but it is much more complex than an arm.

3. Folding a Carpenter's 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 are allowed to rotate freely about their joints and to cross over one another. (See Figure 1.2.) Throughout, 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?

Using a reduction from the NP-complete PARTITION problem (see Garey and Johnson [4]), it is easy to show that the RULER FOLDING problem is NP-complete.

Theorem 3.1: 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 "half-sized" 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. □

It is also easy to see that a ruler can always be folded into length 2m, where m is the length of the longest link. (In fact, 2m is the least upper bound for the minimum folding length.) Using this result, it can be shown that the RULER FOLDING problem, like the PARTITION problem, is solvable in pseudo-polynomial time by a dynamic programming scheme. 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 of a carpenter's ruler in time O(n*m). What's more, if the lengths of the links are polynomially related, then the dynamic programming scheme is polynomial.

Having listed these basic results about folding rulers, we return to the original problem of moving arms.

4. Moving an Arm in a 2-Dimensional Region

The remainder of our paper is concerned with moving arms, which can be thought of as rulers that have one endpoint, A_0, pinned down. We continue to use the notation of Section 3, but we allow the lengths l_i to be non-integral.

Unrestricted Movement

The next theorem, whose simple proof we omit, shows that it is easy to find the points that can be reached by the free end of an arm placed in the plane.

Theorem 4.1: Let $L_1, \ldots, L_n$ be an arm positioned in 2-dimensional space, and let r be 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 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 $l_i - \sum_{j \neq i} l_j$.

Restricted Movement

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

Example of a hard reachability problem for an arm: We want to know whether the arm shown in Figure 4.1 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 folded 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.

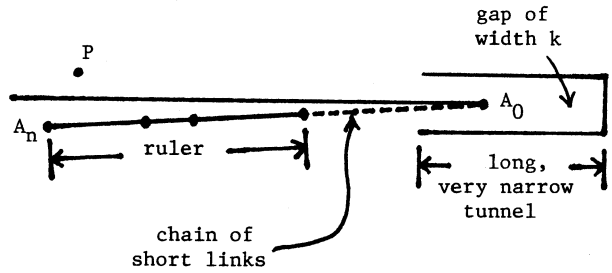


Figure 4.1: A Point That is Hard to Reach.

5. Moving an Arm Inside a Circle

In this section, we consider motion problems for an arm confined to move on or inside a circle C. We have algorithms with time complexity on the order of a polynomial in n, the number of links, that solve the following problems for a given initial configuration of the arm with A_0 fixed inside or on C:

1. Decide whether the arm can be moved to a given final configuration, and if so, how;
2. Given a point p on or inside C, decide whether the arm can be moved so that the free end A_n reaches p, and if so, how;
3. Describe, for each j, $0 \leq j \leq n$, the set S_j of all points inside or on C that A_j can reach.

After discussing problems 2 and 3 briefly, we will return to problem 1, which we will examine at greater length. Detailed accounts of the solutions to all these problems can be found in [3] and [5].

A result that is important in the solutions of both problems 2 and 3 is that the set R_j of points on C that joint A_j can reach consists of at most two arcs of C. Furthermore, these arcs can be found quickly, as can a reachable configuration for links $L_1,\ldots,L_j$ that places A_j at any given point in R_j. This fact can be used to reduce problem 2 to problem 1 as follows. Except for a special case that can be handled separately, it turns out that if there are reachable configurations of the arm that place A_n at p, then in some such configuration, there is a joint A_j on C connected to A_n by a line of links containing at most one non-straight joint. To check whether this happens, compute the R_j's, look for an appropriate straight line or "elbow" reaching from p back to a non-empty R_j, and if this is successful, then compute a reachable configuration from A_o to A_j that places A_j at the appropriate point in R_j. In this way, a reachable configuration that places A_n at p can be computed if such a configuration exists, so problem 2 reduces to problem 1.

The heart of the solution to problem 3 is to show that there is a constant number of circles (independent of n) whose union covers the boundary of S_j. Some of the circles in the covering collection correspond to natural bounds on minimum and maximum distances that A_j can move from A_o and from the center of C. Each of the remaining circles is centered at the endpoint of some R_i and has radius equal to the length of a line of links from A_i to A_j that is either straight or has one folded joint. Here again, computing the R_i's plays a role.

We now sketch a solution to 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, the arm can be moved 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", and the length of this sequence is on the order of the cube of the number of links.

While there are many ways to define a basic unit of motion, it is obviously desirable to use a definition that neither limits the configurations that can be reached nor complicates the algorithms and proofs. With this in mind, we have chosen to define a simple motion of an arm to be a continuous motion during which at most four joint angles change; changing angles must be connected by straight lines of links and can only increase or decrease during the motion. (The angle at A_o between L_o and a reference line is considered to be a joint angle.)

The orientation of a link L_i is defined as follows. If the arc of C that lies to the left of the straight line through L_i, viewed from A_{i-1} toward A_i, is shorter than the arc on the right, then L_i is said to have left orientation. Right orientation is defined in a similar manner, and a link lying on the diagonal is said to have both left and right orientation. (Look ahead to Figure 5.3.)

An obvious necessary condition for being able to move the arm from one given configuration to another is that it be possible to reorient each link whose orientation differs in the two configurations. Since a link must be moved to the diagonal in order to be reoriented, it is necessary that its endpoints be able to move far enough off C to allow this to happen. Hence we need to compute c_i and d_i, the minimum and maximum distance that A_i can be moved off C by arbitrary motions of the arm. Distance will be measured along a radius of C, so $0 \le c_i \le d_i \le d/2$, where d is the diameter of C. The following normal form lemma, which we state but do not prove, shows that it is easy to compute the c_i's. We will also use this lemma again when we give our algorithm for changing configurations.

Lemma 5.1: Any arm for which $\sum_{i=1}^{n} l_i \ge c_o = d_o$ can be moved from its initial configuration to a normal form in which links $L_1,\ldots,L_j$ extend along a radius of C, and joint A_{j+1} and all its successors lie on C, where A_j is the last joint whose preceeding links have lengths summing to less than d_o. Furthermore, this normal form can be reached by a sequence of O(n) simple motions that can be computed in O(n) time. (See Figure 5.1.)

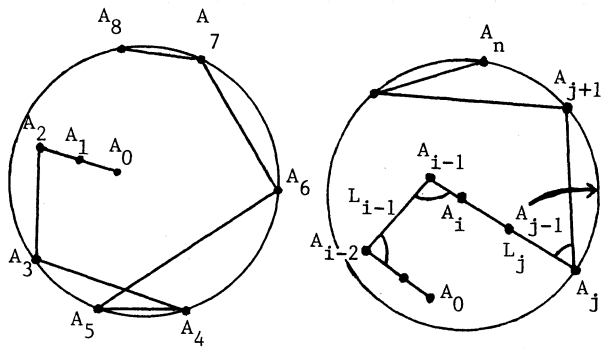


Figure 5.1: A Simple Move Toward Normal Form. Figure 1a shows an arm in normal form, with L_1 and L_2 extended along the radius through A_0 while A_3, the first joint that can reach C, and all its successors lie on C. Figure 1b shows a typical simple move used in reaching normal form. The locations of A_{i-2} and its predecessors and the locations of A_j and its successors are held fixed. Only the angles at A_{i-2}, A_{i-1}, A_{j-1}, and A_j are changing while L_j is rotated about A_j, moving A_{j-1} toward C but away from A_{i-2} (so A_{i-2} does not fold). The move ends when A_{i-1} straightens or A_{j-1} reaches C.

The normal form lemma shows that each successive A_i can get closer to the circle by an amount $\mathbf{1}_i$ until the circle is reached, so $c_i = \max\{c_{i-1} - \mathbf{1}_i, 0\}$. Computing the d_i's is slightly more complicated and involves computing the maximum distance t_i that each A_i can move off C if it is constrained only by the tail of the arm (i.e., if $L_1,\ldots,L_i$ are removed). It is straightforward to compute each d_i from t_i and d_{i-1}. Figure 5.2 indicates that $t_i = d/2$ unless there is a long link after A_i. It also illustrates a simple but crucial algorithm for moving the tail with a small number of simple motions so that the distance between A_i and C increases or decreases in a monotone fashion. Now we are ready to show how to reorient links, which we do in the next lemma.

Lemma 5.2: A link L_i can be reoriented if, and only if, at least one of the following inequalities holds:

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

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

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

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

Proof: Since L_i must lie on a diagonal at some time during reorientation, the above condition is obviously necessary: 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 condition is also sufficient and that reorientation can be done quickly, start by moving the arm to normal form. Then if inequality i) holds, move A_{i-1} to a position distance d_{i-1} from C, keeping the tail in normal form. It is possible to do this in $O(n^2)$ additional simple motions. If inequality iii) holds, move A_{i-1} to a position distance d_{i-1} from C, again using $O(n^2)$ simple motions and keeping the tail in normal form. 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}. This takes at most n-i simple motions of the type illustrated in Figure 5.2.

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

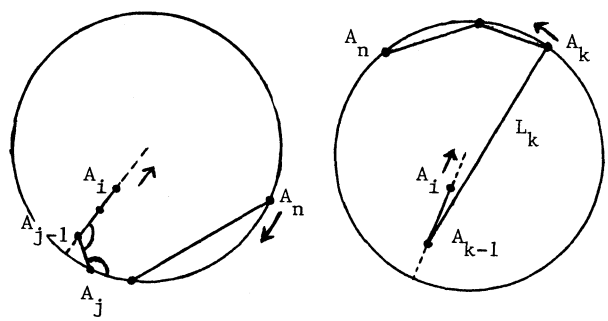


Figure 5.2: Moving the Tail. In Figure 2a, the tail is kept in normal form while A_i is moved away from (or toward) C by simple motions. $A_j,\ldots,A_n$ move along the radius while $A_j,\ldots,A_n$ move around C. Only the angles at A_{j-1} and A_j are changing. Note that if the tail were reattached, the simultaneous combination of this motion with a rotation of L_i about A_1 would still be a simple motion. In Figure 2b, A_i will reach distance t_i from C when A_{k-1} folds, preventing further travel toward the center of C.

We need to make one more observation about reorienting links before we can give the algorithm for changing configurations. Suppose L_i is a link that can be reoriented. Then starting from any initial configuration of the arm, we can reorient L_1, 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 Figure 5.3.)

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.

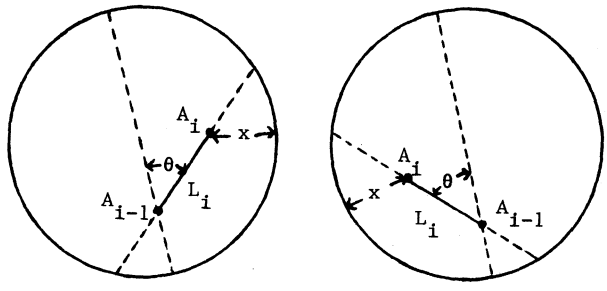


Figure 5.3: Changing Orientation. In Figure 3a, L_i is being moved toward the diagonal through A_i. L_i has left orientation and forms an angle θ with the diagonal. A_i is distance x from C. In Figure 3b, L_i is shown at the corresponding moment after it has passed the diagonal. Again L_i forms an angle θ with the diagonal, and A_i is distance x from C, but now, L_i has right orientation.

If each link that must be reoriented satisfies the necessary and sufficient condition given in Lemma 5.2, then the following algorithm can be used to move the arm to its desired final configuration with $O(n^3)$ simple motions that can be computed in $O(n^3)$ time.

Algorithm for Changing Configurations:

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). Then rotate L_i about A_{i-1} to put A_i in final position (n-i simple motions). 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 insures that the distance from A_i to C 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 C changes in a monotone fashion 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 which links must be reoriented, and check that the reorientability condition is satisfied.

6. Conclusions

In this paper we have investigated several restrictions of the mover's problem involving the movement of linkages and arms in 2-dimensional regions. In summary, we have shown that:

i) The problem of moving an arm in a 2-dimensional region can be polynomially reduced to the problem of moving a more complex linkage that is not constrained by a region. Unfortunately, the latter problem is PSPACE-complete;

ii) Deciding whether or not an arm can be folded to have length k is NP-complete;

iii) The problem of (ii) is solvable in pseudo-polynomial time. That is, if all of the links of the arm are known to be shorter than some given length, then there is a polynomial time (linear) algorithm;

iv) Because of (ii), the reachability problem is at least NP-hard for an arm in a 2-dimensional, nonconvex region;

v) We have a polynomial time algorithm for deciding how to move an arm in a circular region.

Clearly, one of the major open problems relating to this work is to give a polynomial time algorithm for deciding how to move arms in arbitrary convex regions. We conjecture that this can be done and believe that the ideas of Reif [9] and Schwartz and Sharir [10,11] together with those presented in our paper for moving arms within a circular region provide useful techniques for approaching this problem.

References

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

[2] Hilbert, D. and F. Cohn-Vossen. Geometry and the Imagination. Chelsea Publishing Company, New York, New York, 1952.

[3] Hopcroft, John, Deborah Joseph and Sue Whitesides. On the Movement of Robot Arms in 2-Dimensional Bounded Regions. Computer Science Department, Cornell University, TR 82-486, April 1982.

[4] Hopcroft, John, Deborah Joseph and Sue Whitesides. On the Mover's Problem for 2-Dimensional Linkages. Computer Science Department, Cornell University, TR 82-415, 1982.

[5] Hopcroft, John, Deborah Joseph and Sue Whitesides. Determining the Points of a Circular Region Reachable by Joints of a Robot Arm. Computer Science Department, Cornell University, TR 82-416, 1982.

[6] Kempe, A. B. On a general method of describing plane curves of the nth degree by linkwork. Proc. London Math. Soc. 1876, Vol.7, pp. 213-216.

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

[8] Peaucellier, M., Correspondance. Nouvelles Annales de Mathematiques 1864, Series 2, Vol. 3, pp. 414-415.

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

[10] 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.

[11] 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.