Proof of Theorem 2.3. If we apply Lemma 2.8 with $s=0$ then $C^s=\mathbf{I}$ and we see that $|F^T|$ is an integral multiple of $|\mathbf{I}|/2^{T(T+1)/2}$. Since $E^T=\mathbf{I}-F^T$ it follows that $|E^T|$ is also a multiple of this number. Now $e\in E^T$ so we have $|E^T|>0$ and thus $|E^T|\geq |\mathbf{I}|/2^{T(T+1)/2}$. By our assumption on g and by Lemma 2.9 we need $|E^T|\leq|\mathbf{I}|r$. Therefore $2^{T(T+1)/2}\geq r$ and so $T=\Omega(\sqrt{\log_2 r})$. □

Acknowledgments. We are indebted to a careful referee who found a serious mistake in our original Lemma 2.3, which claimed a stronger result. We thank Paul Beame for letting us include his results here. We also wish to thank Al Borodin and Dick Lipton for helpful comments on an early version of this manuscript.

REFERENCES

[AHU-74] A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA 1974.

[B-83] P. BEAME, personal communication, 1983.

[CD-82] S. A. COOK AND C. DWORK, *Bounds on the time for parallel* RAM's *to compute simple functions*, Proc. 14th ACM Symposium on Theory of Computing, 1982, pp. 231–233.

[CSV-82] A. K. CHANDRA, L. J. STOCKMEYER AND U. VISHKIN, *Complexity theory for unbounded fan-in parallelism*, Proc. 23rd Annual IEEE Symposium on Foundations of Computer Science, 1982, pp. 1–13.

[FW-78] S. FORTUNE AND J. WYLLIE, *Parallelism in random accesss machines*, Proc. 10th ACM Symposium on Theory of Computing, 1978, pp. 114–118.

[FSS-81] M. FURST, J. B. SAXE AND M. SIPSER, *Parity, circuits and the polynomial-time hierarchy*, Proc. 22nd Annual IEEE Symposium on Foundations of Computer Science, 1981, 260–270.

[GGKMRS-83] A. GOTTLIEB, R. GRISHMAN, C. P. KRUSKAL, K. P. MCAULIFFE, L. RUDOLPH AND M. SNIR, *The* NYU *ultracomputer—Designing a* MIMD *Shared Memory Parallel Machine*, IEEE Trans. Comput., C-32 (1983), pp. 175–189.

[Go-78] L. M. GOLDSCHLAGER, *A unified approach to models of synchronous parallel machines*, Proc. 10th ACM Symposium on Theory of Computing, 1978, pp. 89–94.

[HU-79] J. HOPCROFT AND J. ULLMAN, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979, pp. 314–318.

[K-77] D. J. KUCK, *A survey of parallel machine organization and programming*, Computing Surveys, 9 (1977), pp. 29–59.

[LS-81] R. LIPTON AND R. SEDGWICK, *Lower bounds for* VLSI, Proc. 13th ACM Symposium on Theory of Computing, 1981, pp. 300–307.

[PS-82] C. PAPADIMITRIOU AND M. SIPSER, *Communication complexity*, Proc. 14th ACM Symposium on Theory of Computing, 1982, 196–200.

[SV-81] Y. SHILOACH AND U. VISHKIN, *Finding the maximum, merging and sorting in a parallel computation model*, J. Algorithms, 2 (1981), pp. 88–102.

[ShV-82] ———, *An* $O(\log n)$ *parallel connectivity algorithm*, J. Algorithms, 3 (1982), pp. 57–67.

[SV-82] L. J. STOCKMEYER AND U. VISHKIN, *Simulation of parallel random access machines by circuits*, RC 9362, IBM T. J. Watson Research Center, Yorktown Heights, NY 1982; this Journal, 5 (1984), pp.409–422.

[V-82] U. VISHKIN, *Parallel-design space distributed—implementation space* (PDDI) *general purpose computer*, RC 9541, IBM T. J. Watson Research Center, Yorktown Heights, NY, 1982 Theoret. Comput. Sci., to appear.

[W-83] A. WIGDERSON, *Studies in computational complexity*, Ph.D. thesis, Princeton Univ., Princeton, NJ, May 1983.

[Y-81] A. YAO, *The enthropic limitations on* VLSI *computations*, Proc. 13th ACM Symposium on Theory of Computing, 1981, pp. 308–311.

SECTION 1 & 2 only.

Vol. 14, No. 2, May 1985

005

(3)

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

JOHN HOPCROFT†, DEBORAH JOSEPH‡ AND SUE WHITESIDES§

Abstract. The *mover's problem* is the following: can an object in 3-dimensional space be moved from one given position to another while avoiding obstacles? It is known that the general version of this problem involving objects with movable joints is PSPACE hard, 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. Finally, we show that 148 circles suffice to cover the boundary of the reachable region of a joint in an arm enclosed in a circle and that the boundary can be computed in polynomial time.

Key words. robotics, manipulators, mechanical arms, algorithms, polynomial time

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* 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. Polynomial time algorithms (Lozano-Perez and Wesley [3], Reif [5], Schwartz and Sharir [6]) are known in the case where X is a rigid 2- or 3-dimensional polyhedral object, and R is a region described by linear constraints.

A more difficult problem, which is related to problems in robotics, assumes that the object X has joints and is hence nonrigid. Schwartz and Sharir [7] give a polynomial time algorithm, the degree of the polynomial being exponential in the number of joints, for moving X from position P_i to P_f within a region R. Unfortunately, an algorithm with running time polynomial independent of the number of joints is unlikely, as Reif [5] 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.

This paper investigates variants of the mover's problem that we believe are of interest. We begin in §§ 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 §§ 4–6 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

* Received by the editors August 1, 1983, and in revised form January 19, 1984. This work was supported in part by the Office of Naval Research under contract N00014-76-C-0018, the National Science Foundation under grant MCS81-01220, and an NSF Postdoctoral Fellowship, and a Dartmouth College Junior Faculty Fellowship.

† Computer Science Department, Cornell University, Ithaca, New York 14853.

‡ Computer Science Department, University of Wisconsin, Madison, Wisconsin 53706.

§ School of Computer Science, McGill University, Montreal, Quebec, Canada H3A 2K6.

points. Also, we show that circles covering the boundary of the set of points reachable by a joint can be computed in polynomial time.

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 $L_1, \cdots, L_n$ 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 $A_0, \cdots, A_n$ and for $1 \leqq i \leqq n$, we let l_i denote the length of link L_i. (See Fig. 2.1.) We define the Ruler Folding problem to be the following:

Given: Positive integers $n, l_1, \cdots, l_n$, and k.

Question: Can a carpenter's ruler with lengths $l_1, \cdots, l_n$ 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?

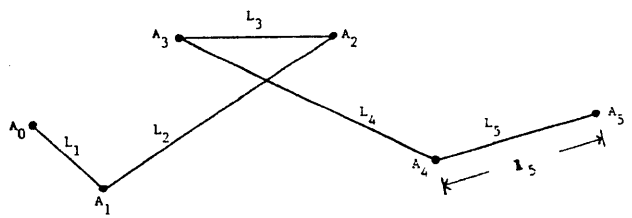


FIG. 2.1. *A typical ruler with five links.*

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 $l_1, \cdots, l_n$, there is a subset $S' \subseteq S$ such that

$$\sum_{l_i \in S'} l_i = \sum_{l_j \in S - S'} l_j.$$

THEOREM 2.1. *The* RULER FOLDING *problem is* NP*-complete.*

Proof. Given an instance of the PARTITION problem with $S = \{l_1, \cdots, l_n\}$, let $d = \sum_{i=1}^{n} l_i$. Then the desired subset S' of S exists if and only if a ruler with links of length $2d, d, l_1, \cdots, l_n, d, 2d$ (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* $l_1, \cdots, l_n$ *can always be folded into length at most* $2m$, *where* $m = \max\{l_i | 1 \leqq i \leqq n\}$.

Proof. Place link L_1 into the interval $[0, 2m]$ with A_0 at 0. Having placed links $L_1, L_2, \cdots, L_{i-1}$ 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 $p \leqq l_i \leqq m$. Hence A_i can surely be placed to the right of A_{i-1}. □

Using this result, we can now give an $O(m^2 * n)$ dynamic 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 $L_1, \cdots, L_n$, compute the maximum link length m. Then, for each k, $1 \leqq k \leqq 2m$, construct a table with rows numbered 0 to n and columns numbered 0 to k. For a given k, a T is placed in row i, column j if the linkage $L_1, L_2, \cdots, L_{i-1}$ fits in $[0, k]$ with the endpoint A_i at integer j. Row 0 is filled in by writing a T in each column j. Once row $i-1$ has been filled in, fill in row i by writing a T in each column j for which the linkage $L_1, \cdots, L_i$ 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 some sense, the best upper bound for the minimum folding length.

Example 2.1. *A ruler with minimum folding length* $2m - \varepsilon$. Consider a ruler which has $n = 2k-1$ links $L_1, \cdots, L_n$. (See Fig. 2.2.) Suppose that links with odd subscripts have length m and that links with even subscripts have length $m - \varepsilon$, where $\varepsilon = m/k$. It is easy to check that this ruler cannot be folded into length less than $2m - \varepsilon$.

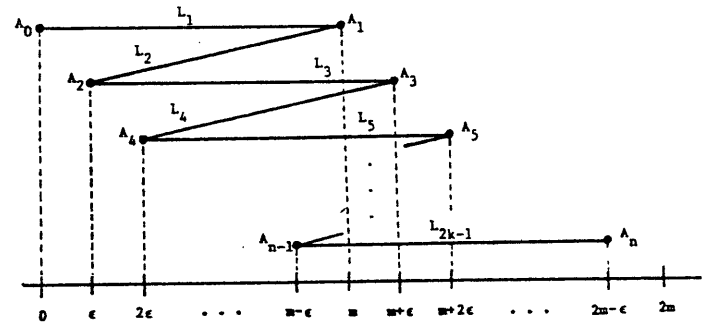


FIG. 2.2. *The ruler of Example* 2.1.

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 $L_1, \cdots, L_n$ be an arm positioned in 2-dimensional space, and let $r = \sum_{i=1}^{n} l_i$, 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 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 during its motions, then determining whether A_n can reach some given point p is difficult. The following example suggests that a reduction of RULER FOLDING can be used to show that this problem is NP-hard even for *walls* consisting of a few straight line segments.

Example 3.1. *A hard decision problem.* 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.

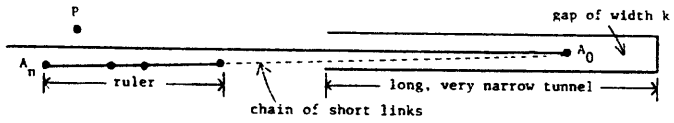


FIG. 3.1. *A point that is hard to reach.*

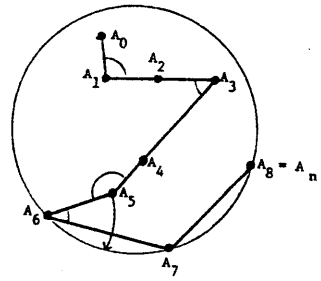
We would like to find natural classes of regions for which questions concerning the movement of arms are decidable in polynomial time. Certainly the simplest such region is the inside of a circle, since there are no corners in which an "elbow" might be caught. We believe that studying motions inside 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.

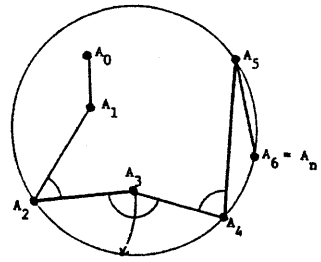
Simple motions. A definition of a "simple motion" is needed in order to make clear the sense in which our algorithms for moving an arm are polynomial. This definition should not limit the positions the arm can reach nor should it complicate the algorithms and proofs. With these considerations in mind, 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.

Figure 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 counterclockwise, 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 $\sum_{i=1}^{j} l_i \geqq d_0$. Then the arm is in *normal form* if and only if $L_1, \cdots, L_j$ contains at most one bent joint, and for each $k, j \leqq k \leqq n$, A_k is on the circle. Moreover, if $L_1, \cdots, L_j$ is not a straight line of links, the bend is at joint A_{j-1}. (See Fig. 4.2.) In any event, $L_1, \cdots, L_{j-1}$ lie on a radius.

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 $L_j, L_{j+1}, \cdots, L_n$ 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