Embedding Linkages on an Integer Lattice

Susan Landau

CMPSCI Technical Report 93-63
November, 1993

Embedding Linkages on an Integer Lattice

Susan Landau[*]
Department of Computer Science
University of Massachusetts
Amherst, Mass. 01003

Abstract

This paper answers the following question: Given an "erector set" linkage, what is the minimal ϵ one must adjust edges by to embed it on an integer lattice? Each edge length may change by ϵ, and angles are not fixed, but collinearity must be preserved. We show that the problem is $\mathcal{NP}$-complete, but that a restricted version can be done in polynomial time.

1 Introduction

Using links and joints, one can build an "erector set" type linkage in which angles are free to change, but lengths and collinearity properties are fixed. A problem in computer-aided design [8] raised the question of when such a linkage can be embedded in an integer lattice, assuming one is allowed to shift lengths by ϵ, but collinearity properties must be preserved. What is the minimal ϵ that will allow an embedding where each of the edges has its endpoints on integer vertices in the lattice?

We show here that the question of whether a linkage can be embedded on the integer lattice is $\mathcal{NP}$-complete, an unhappy situation. But all is not lost: for applications, it is quite reasonable to assume that lengths are bounded, as are the number of "co-incident" polygons. With these caveats, we show there is a polynomial time solution to the problem.

We note that the problem has some nice ties to number theory: the linkage can be embedded only if each of its links can, and this can happen

[*]Supported by NSF grant CCR-9204630.

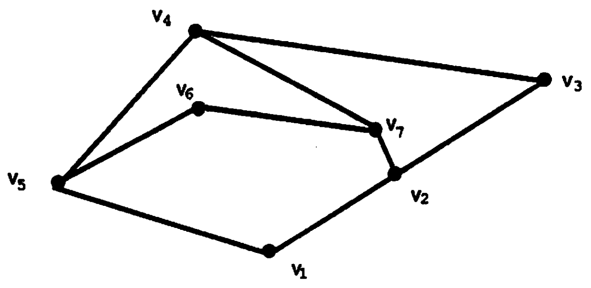

Figure 1: A Linkage with Eight Links and Seven Vertices.

if and only if each of the links has a length whose square can be written as the sum of two squares. We explore these connections further in the paper.

This paper is organized as follows: §2 Embedding Fixed Length Linkages and §3 What is the minimal ϵ?.

2 Embedding Fixed Length Linkages

A linkage $\mathcal{L}$ is a set of connected links $l_1, \ldots, l_n$. By abuse of notation, the lengths of each of the linkages will also be denoted by $l_1, \ldots, l_n$. Each link l_i has two endpoints, c_i and d_i. An arbitrary number of links may share a vertex. We say a linkage can be ϵ-embedded on an integer lattice if there is another linkage $\mathcal{L}'$ such that for each link l_i in $\mathcal{L}$ there is a link l'_i in $\mathcal{L}'$ where $|l_i - l'_i| \leq \epsilon$ and $\mathcal{L}$ and $\mathcal{L}'$ have the same topolgical structure, that is, collinearity and adjacency structure of $\mathcal{L}$ and $\mathcal{L}'$ are the same, and the vertices of $\mathcal{L}'$ lie on integer lattice points.

The problem we consider first is given a planar linkage $\mathcal{L}$, find the minimum ϵ such that $\mathcal{L}$ can be $\epsilon-$embedded on an integer lattice, and find such an embedding. We allow links to lie upon one another and to cross. In the usual way, the size of the problem is the size of the input, $\Sigma \log l_i$.

In this section we handle a slightly simpler problem: we consider the problem with a fixed ϵ. We make the following observation:

Theorem 2.1 *A linkage $\mathcal{L}$ with links of length $l_1, l_2, \ldots, l_n$ can be embedded on a two dimensional integer grid only if for each $i = 1, \ldots, n$, there is an integer n_i such that $\sqrt{n_i} \in [l_i - \epsilon, l_i + \epsilon]$ and n_i is expressible as the sum of two squares.*

Proof Obvious. ∎

We use the following well-known characterization as to which integers can be written as the sum of two squares.

Theorem 2.2 *([5], pg. 299) A number n is the sum of two squares if and only if all prime factors of n of the form $4m + 3$ have even exponents in the prime factorization of n.*

For the remainder of this paper, the symbol p_i shall denote a prime congruent to 1 mod 4, and q_j shall denote a prime congruent to 3 mod 4. Let $N_2(n)$ be the number of distinct representations of n as the sum of two squares; we count two representations as distinct even when they differ only trivially, say by order, or by multiplication by -1. Then we have:

Theorem 2.3 *([5], pg. 242) Suppose $n = 2^\alpha \Pi p_i^{r_i} \Pi q_j^{s_j}$. Then $N_2(n) = 4\Pi(r_i + 1)$ if each of the s_j is even, and 0 otherwise.*

A natural question to ask is how large $N_2(n)$ can be in comparison with n? Then we have:

Theorem 2.4 *([5], pg. 270) $N_2(n) = O((\log n)^\Delta)$ is false for every constant Δ.*

Theorem 2.5 *([5], pg. 270) The maximum order for $N_2(n)$ is $2^{\frac{\log n}{2 \log \log n}}$.*

Finally, the average order of $N_2(n)$ is also known:

Theorem 2.6 *([5], pg. 270) $\frac{N_2(1)+N_2(2)+\ldots+N_2(n)}{n} = \pi$. More precisely,*

$$N_2(1) + N_2(2) + \ldots + N_2(n)n = \pi n + O(\sqrt{n}).$$

These last two theorems lead one to suspect that the question of embedding a linkage in an integer lattice is hard, and indeed, it is. We study a simple linkage first.

The simplest kind of linkage is a series of link segments; we call such a linkage a tree. Formally a linkage is a tree if it contains no cycles. Then we have:

Lemma 2.7 *A tree $\mathcal{T} = t_1 \ldots t_n$ can be embedded if and only if each one of its links can be embedded.*

Theorem 2.8 *Suppose the tree $\mathcal{T} = t_1 \dots t_n$ can be embedded on the integer lattice, and suppose $t_i^2 = 2^{\alpha_i}\Pi_j p_{ij}^{r_{ij}} \Pi_k q_{ik}^{s_{ik}}$. The number of distinct embeddings of $\mathcal{S}$ equals $4^n \Pi_{i,j}(r_{ij}+1)$.*

Proof The embedding of each link is independent of the embedding of previous links. The number of possible embeddings of t_i is $4\Pi(r_j+1)$. The result follows immediately. ∎

This result means that checking whether a tree can be embedded can be done in time $O(\Sigma R(t_i))$, where $R(t_i)$ is the time needed to compute whether t_i can be written as the sum of squares. At present, the fastest algorithm to check this is exponential time. (Indeed the problem is likely to be hard: in the appendix, we show that the problem of enumerating the representations of n as the sum of two squares is polynomial time equivalent to factoring an integer n with a bounded number of prime factors of the form $4k+1$, and no prime factors of the form $4k+3$.)

But embedding a linkage is hard not only for number theoretic reasons. Our intuition is that it is hard because of the complexity of interacting polygons. In any case, we show that the question of embedding a general linkage is $\mathcal{NP}$-complete. We do this via a series of lemmas.

The intuition behind the following three lemmas is that we are building a gadget to check the truth value of a clause. Given a $3-\mathcal{CNF}$ formula $\varphi = C_1 \wedge \ \dots \wedge C_r$ where each $C_i = l_{1i} \vee l_{2i} \vee l_{3i}$, and the l_{ji}'s are literals, we build a linkage $\mathcal{L}(\varphi)$ with the property $\mathcal{L}(\varphi)$ is embeddable if and only if φ is satisfiable. Think of the linkage being aligned on the $x-y$ co-ordinate plane, and l_{ji} is on the line $x=1$ as l_{ji} being true, l_{ji} is on the line $x=-1$ as l_{ji} being false.

Lemma 2.9 *For each $C_i = l_{1i} \vee l_{2i} \vee l_{3i}$, where the l_{ji}'s are literals, the gadget G_i with the central axis drawn on the y-axis can have the vertex M_i on the line $x=1$ if and only if l_{1i} or l_{2i} is on the line $x=1$.*

Proof Obvious. ∎

Lemma 2.10 *With conditions as above, the gadget H_i can be embedded on the integer lattice. P_i can be on the line $x=1$ if and only if $l_{1i} \vee l_{2i}$ is true.*

Proof The first statement is clear. The second statement follows from the fact that of necessity P_i and M_i have the same x value. ∎

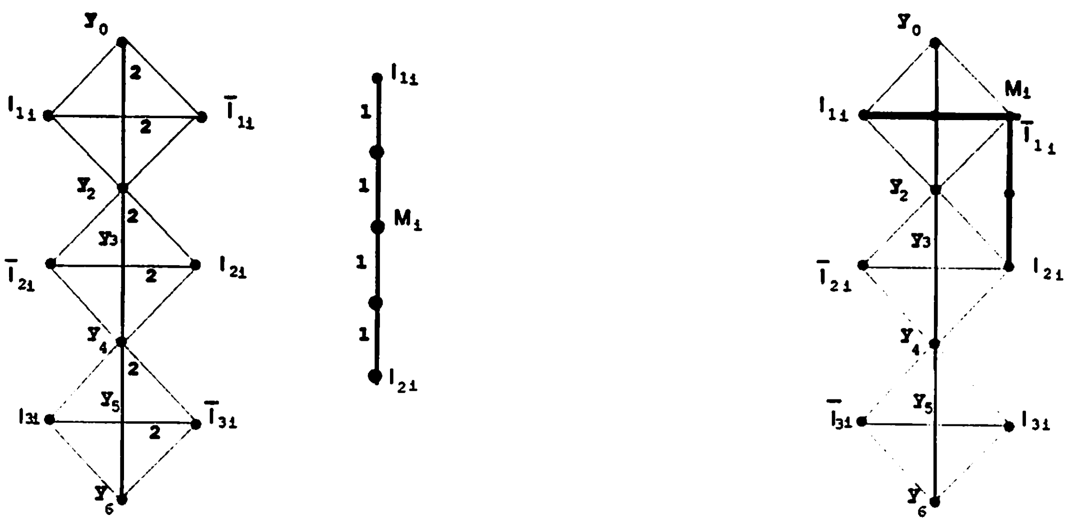

Figure 2: Gadget G_i, on the right, consists of the composition of the two gadgets on the left.

Lemma 2.11 *The gadget K_i can be embedded on the integer lattice. The point Q_i can be on the line $x = 1$ if and only if C_i evaluates to true.*

Proof Obvious. ∎

Theorem 2.12 *Linkages is strongly $\mathcal{NP}$-complete.*

Proof We do this via reducing 3-Satisfiability to Linkages.

Let φ be a $3 - \mathcal{CNF}$ formula with variables $x_1, \ldots, x_n$, and clauses $C_1, \ldots, C_r$.

We construct the linkage as follows.

For each clause C_i we will build a gadget K_i with three variables corresponding to the variables of C_i. We will line these gadgets up underneath one another, the bottom of K_i connecting to the top of K_{i+1}. Furthermore, note that the size of a gadget for a clause is small.

We will need to ensure that all appearances of a variable have the same truth value; this we will do by connecting appearances of the same variable, say in clause K_i and K_j with a straight rod. This will force all appearances of the same variable to have the same x coordinate. Observe that we have at most $3/2r$ rods, each of length no more than $6r$.

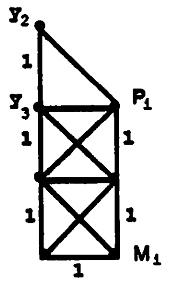

Figure 3: Gadget H_i consists of the above combined with Gadget G_i, where points with the same label coincide.

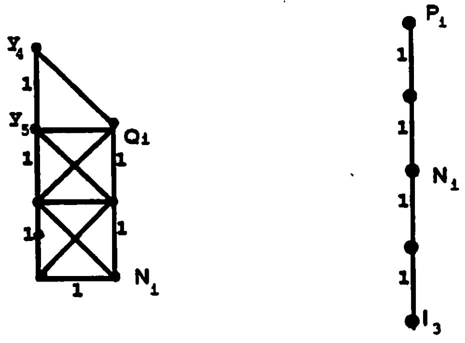

Figure 4: Gadget K_i, consisting of the above combined with Gadget H_i, where points with the same label coincide.

Finally, we add one more straight rod of length $6r$ with vertices $Q_1, \ldots, Q_r$ on it. This will assure that all clauses are true simultaneously.

It is clear that if the formula can be satisfied that the linkage is embeddable.

Suppose the linkage is embeddable. Call the side on which the rod connecting the Q's "true." This induces an assignment to the boolean variables, and all occurrences of the same variable have the same assignment. Thus at least one variable from each clause is "true," and the formula is satisfiable.

Clearly the construction is in polynomial time. ∎

This result should be compared with a theorem by Hopcroft, Joseph and Whitesides [7]. They show that the ruler problem – the question of whether a carpenter's ruler consisting of a sequence of n links of integer lengths, $l_1, \ldots, l_n$, hinged together at the endpoints, and allowed to fold so that angles of either 0° or 180° are formed, can be aligned so that its length is k – is $\mathcal{NP}$–complete. Their proof was via a reduction from the Partition problem. That problem is not strongly NP complete. In fact, it is shown via dynamic programming that if the lengths are bounded, then the problem can be done in polynomial time [7].

In our case, Theorem 2.12 shows that lengths have little to do with the situation. However we can restrict linkages in two ways. The first restriction is that the linkage must lie on an $m \times m$ integer grid. (This is equivalent to saying all of the lengths l_i must be small.) The second restriction is that there are only a bounded number of polygons in the linkage. For many applications that is a very reasonable assumption. With these restrictions, testing whether the linkage can be embedded on the integer lattice can be done in polynomial time.

We begin by showing that the question of whether a polygon can be embedded on an $m \times m$ grid can be answered in polynomial time. There are potentially exponentially many embeddings, so this result is not obvious. A result of Edmund Landau's will help even further by giving a bound on the number of ways each link can lie. Then:

Theorem 2.13 *(E. Landau, [4] pg. 22)* $N_2(x) = b\frac{x}{\sqrt{\log x}} + o(\frac{x}{\sqrt{\log x}})$, *where* $b = \frac{1}{2}\Pi_{q\equiv 3 \pmod 4}(1-q^{-2})^{1/2} \sim 0.764\ldots$.

Now we are ready to show how to test whether a polygon with bounded edge lengths can be embedded on an $m \times m$ gird.

Theorem 2.14 *Suppose a linkage $\mathcal{L}$ is a polygon $\mathcal{P} = l_1 \ldots l_n$ with $L = max_i l_i^2$. Determining whether $\mathcal{P}$ can be embedded in an $m \times m$ integer grid can be determined in $O(m^2nL) \leq O(m^4n)$ steps.*

Proof We will use the following algorithm to check if $\mathcal{P} = l_1 \ldots l_n$ can be embedded on the integer lattice.

input: $\mathcal{P} = l_1 \ldots l_n$; m;
output: yes (if polygon can be embedded on an $m \times m$ integer grid),
no otherwise.

Step 1: Check that each link $l_i \leq \sqrt{2}m$. If not, the polygon cannot be embedded. Next factor each length $l_i^2 = 2^{\alpha_i}\Pi_j p_{ij}^{r_{ij}} \Pi_k q_{ik}^{s_{ik}}$. Check that each

s_{ik} is even. If not, linkage cannot be embedded.

```
Step 2: For $k, l \in \{0, \ldots m\}$ do:{{
        Build an $m+1 \times m+1$ array $G^{kl}$ with a "1" in the $(k,l)$ grid point, and 0's elswhere.
        For $i = 1$, to $n-1$, do:
            {Build an $m+1 \times m+1$ array $T$ which gets a 1 in the $(s,t)$ grid point if and only if there is a $(u,v)$ in $G^{kl}$ with $(u,v) = 1$ and $(s,t)$ is reachable from $(u,v)$ by $l_i$. All other entries of $T$ get 0's. $G^{kl} \leftarrow T$.}
            For $i = n$, $T$ gets a 1 in the $(k,l)$ entry if and only if there is a $(u,v)$ in $G^{kl}$ with $(u,v) = 1$ and $(k,l)$ is reachable from $(u,v)$. All other entries of $T$ get 0's. $G^{kl} \leftarrow T$.}}
Step 3: If there is a $(k,l)$ entry of $G^{kl}$ equal to 1, $\mathcal{P}$ can be embedded; otherwise not.
```

We first prove correctness, then analyze running time.

Step 1 is a necessary condition for an embedding. The difference between a "line" $l_1 \ldots l_n$ and a polygon $l_1 \ldots l_n$ is that the polygon must connect.

We make several observations. The linkage $l_1 \ldots l_{i+1}$, $i \neq n$ can be embedded "beginning" at (k,l) if and only if there is a grid point (u,v) such that the linkage $l_1 \ldots l_i$ can be embedded beginning at (k,l) and ending at (u,v) and the link l_{i+1} can be embedded beginning at (u,v) and ending at (s,t). The last link of the polygon needs to connect to l_1; the polygon can be embedded "beginning" at (k,l) if and only if after the n^{th} time step, there is a pair (k,l) with G^{kl} having a "1" in the (k,l) square.

Now we are ready to analyze the running time. The first step takes $O(l_i^{1/2}) < O(L)$ time per link, thus $O(nL)$ time in total. Let us consider the running time of Step 2 for a single starting point (k,l).

At each computation of G^{kl}, at most $N_2(l_i^2)$ grid points are marked. Now we know that $N_2(l_i^2) = .764 \ldots \frac{l_i^2}{\sqrt{\log l_i^2}} < \frac{L}{\log l_i^2} < L$. Thus the time to compute a single G^{kl} is bounded by L. For each starting point (k,l), we compute n G^{kl} arrays; this takes $O(nL)$ steps. There are, of course, m^2 different choices for the starting point; thus the entire computation is bounded by $O(m^2 nL) \leq O(m^4 n)$ steps since $L = max_i l_i^2 \leq 2m^2$. ∎

Note: There are some symmetries one can take advantage to speed up this computation (e.g., one needs only try one-quarter of the grid as a possible starting point). Furthermore, the linkage can be embedded in an $m \times m$ grid only if the linkage can be embedded in a $2m \times 2m$ grid starting at the point (m,m). If one is concerned with embedding in an $O(m \times m)$ grid, then one can check whether the polygon is embeddable in a $2m \times 2m$ grid in $O(m^2L)$ steps, by using the point (m, m) as a starting point.

In order to show how to embed a linkage in an $m \times m$ integer grid, it will be helpful to view the linkage as a graph. Normally graphs have straight edges only between a pair of vertices; in our case, the linkage may have an internal vertex with links:

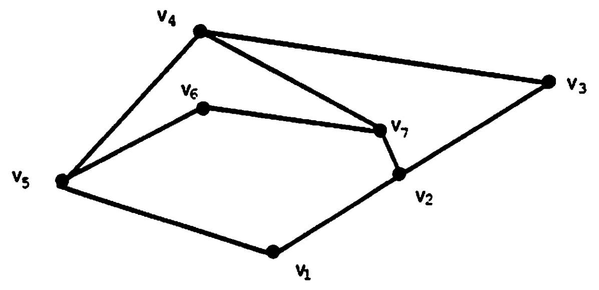

Figure 5: A Linkage with Eight Links and Seven Vertices.

where the edge with vertices v_1, v_2 and v_3 lie on a straight line. This is not a serious problem.

A linkage is complex precisely because polygons may share an edge, or set of edges; the question is how to place them on the grid so that all the polygons can be embedded compatibly. In our next theorem, we will show how to test whether a linkage with a bounded number of polygons can be embedded in an $m \times m$ grid. We will build a depth-first spanning tree of the linkage. The linkage may have more than two vertices on an edge; each vertix v_i which is internal to a link will be labelled v_i^- in the depth-first search tree.

Theorem 2.15 *Suppose a linkage $\mathcal{L}= l_1 \ldots l_n$ has at most e back edges per biconnected component of the dfs tree of $\mathcal{L}$. Let L $=max_i l_i^2$. Determining whether $\mathcal{L}$ can be embedded in an $m \times m$ integer grid can be determined in $O(nL^e m^{2e}) \leq O(nm^{4e})$ steps.*

Proof We begin by building a dfs tree for $\mathcal{L}$. Observe that to embed the linkage, it suffices to consider the embeddings of each biconnected component separately, for then the linkage can be viewed as a tree, which we know is a much simpler question. The rest of the proof will consider the issue of embedding biconnected components.

We consider first embedding a single polygon. We use the algorithm of Theorem 2.14. We need to do more work; we want to know where all the links of the polygon can be placed if the head of l_1 is at (k,l). That is easy to compute.

For each beginning point (k,l), and each link l_i in the polygon, we compute the matrix G_i^{kl}, which has a "1" in the (s,t) entry iff the links $l_1 \ldots l_i$ can have l_1 embedded at (k,l) and l_i ending at (s,t). If there is a successful embedding of the polygon $l_1 \ldots l_n$, at the end of the algorithm, the matrix G_n^{kl} has a "1" in the (k,l) entry iff there is an embedding with link l_1 placed at (k,l).

Next we run the algorithm "backwards" , beginning at G_n^{kl} and ending at G_1^{kl}, recomputing the G_i^{kl}. An entry in G_i^{kl} will have a "1" in the (s,t) position if and only if

1. there is a grid point (u,v) with G_{i+1}^{kl} having a 1 in the (u,v) position,
2. l_{i+1} can go from (u,v) to (s,t), and
3. G_i^{kl} has a 1 in the (s,t) position. (There is a "1" in the s,t position of G_i^{kl} if there is a polygon with l_1 at (k,l) and l_i at (s,t).)

In this way, we find where the links of the polygon $\mathcal{P} = l_1 \ldots l_n$ can lie.

Now the situation grows more complicated. We need to check if the polygon we have computed embeddings for can be compatibly embedded with another polygon of the biconnected component. We can compute the matrices G_i^{kl} for each of the other polygons; the question is how to combine them. We make two observations:

- Each new "polygon" can be uniquely identified by its back edge in the depth first traversal. This back edge is what "closes" the section of the tree into a polygon.
- There are $m^2 N_2(L)$ choices for the way this back edge (u,v) may lie. An alternative upper bound is m^4.

To check if the new polygon has a compatible embedding with the linkage, we identify the polygon by its back edge u,v. Then we repeat the previous algorithm for all possible positions of (u,v), and check whether any lead to an embedding. Thus what we do is travel the dfs tree, computing all possible embeddings of each of the associated back edges for of the individual polygons. There are e of them. Hence we have a running time of $O(n(m^2L)^e) \leq O(nm^{4e})$ steps.

∎

We can, of course, extend these results to higher dimensions. We start with three dimensions. Theorem 2.1 generalizes in the obvious way to:

Theorem 2.16 *A linkage $\mathcal{L}$ with links of length $l_1, \ldots, l_n$ can be embedded on a three dimensional integer grid only if for each $i = 1, \ldots, n$, there is an integer n_i such that $\sqrt{n_i} \in [l_i - \epsilon, l_i + \epsilon]$, and n_i can be written as the sum of three squares.*

It is well-known which integers may be written as the sum of three squares.

Theorem 2.17 *([5], pp. 310-311) An integer n can be written as the sum of three squares if and only if $n \neq 4^\alpha(8m + 7)$.*

Corollary 2.18 *Let $\mathcal{L}= l_1 \ldots l_n$ be a linkage with a bounded number of polygons. Then determining whether L can be embedded on a three dimensional $m \times m \times m$ grid can be determined in polynomial time.*

We leave dimensions four and higher as an exercise for the reader. Note: Lagrange showed that every integer can be written as the sum of four squares ([5], pg. 369).

3 What is the Minimal ϵ?

We begin with trees in two dimensions. By Theorem 2.1 and Lemma 2.7, a tree $T= t_1 \ldots t_n$ is embeddable whenever each of its links is the square root of an integer that can be written as the sum of two squares. Clearly each link s_i of length $s_i = d_t d_{t-1} \ldots d_1 . \bar{d}_1 \bar{d}_2 \ldots$, then each link is no more than 1/2 away from the square root of a perfect square. Can it be as much as 1/2 away from an integer which can be written as sum of two squares? Can it be the case that there is an integer m such that there are no integers between m^2 and $(m + 1)^2$ which can be written as the sum of two squares?

Theorem 3.1 *(Bambah and Chowla, [2]) Let s_i be the i^{th} integer which can be expressed as the sum of two squares. Then $s_{n+1} - s_n = O(s_n^{1/4})$.*

This theorem will answer the case for embedding trees.

Theorem 3.2 *Let $\mathcal{T} = t_1 \dots t_n$ be a tree. Then the minimal ϵ such that $\mathcal{T}$ can be ϵ-embedded in the integer lattice can be determined in $O((t_1)^{3/4} + \dots + (t_n)^{3/4})$ steps.*

Proof By Theorem 2.1, it suffices to consider each link separately. Suppose link t cannot be embedded. This means t^2 cannot be expressed as the sum of two squares.

Let t' be the integer closest to t^2; we have $|t' - t^2| \leq 1/2$. By Theorem 3.1, there is an integer r such that $|r - t'| < O((t')^{1/4})$ such that r can be expressed as the sum of two squares. It takes time $O(t^{1/2})$ to deterministically test if t' can be embedded (that is the time it takes to factor (t')). Thus it takes $O(t^{3/4})$ time to find the closest integer to t which can be expressed as the sum of two squares. The minimal ϵ is the maximum of the epsilons. The time to compute the minimal ϵ for the entire tree is $O((t_1)^{3/4} + \dots + (t_n)^{3/4})$.
∎

The problem of embedding a lingkage is $\mathcal{NP}$-complete, thus the best we can realistically hope for is an exponential time algorithm for the problem. The obvious algorithm of trying all possibiities is single exponential, so we have no more to say about the problem of embedding arbitrary linkages. However, there is a nice solution to linkages with a bounded number of polygons limited to an $m \times m$ grid. There are two cases to consider, one where an upper bound ϵ is given, the other not. They are handled similarly, with the latter treated first.

We begin with the following generalization of Theorem 2.14:

Theorem 3.3 *Suppose a linkage $\mathcal{L}$ is a polygon $\mathcal{P} = l_1 \dots l_n$. Let $L = max_i l_i^2$. Determining the minimal ϵ for which $\mathcal{L}$ can be ϵ-embedded can be determined in $O(nm^4 \log m)$ steps.*

Proof We will do the same algorithm as we did for Theorem 2.14, but we will do it many more times.

Without loss of generality, we can assume we are interested in computing an $\epsilon \leq cm$ for some constant $0 < c \leq \sqrt{2}$.

We do binary search to find the minimal ϵ such that $\mathcal{L}$ can be ϵ-embedded. Clearly $\mathcal{L}$ can be embedded with $\epsilon = \sqrt{2}m$. We check if $\mathcal{L}$ can be embedded with $\epsilon = \frac{\sqrt{2}m}{2}$. In this way, in $O(\log m)$ rounds, we will find the minimal ϵ by which the polynomial can be embedded, if there is one less than $\sqrt{2}m$.

Suppose we are checking whether the polygon can be ϵ_i-embedded. We create the array G^{kl} just as we did in the proof of Theorem 2.14, except that we mark *all* the grid points reached by a linkage of length in the interval $[l_i - \epsilon_i, l_i + \epsilon_i]$.

How long does this procedure take? Although at each step of the algorithm, there are potentially more grid points marked than in Theorem 2.14, the marking algorithm still has the same upper bound as it did previously, which is $O(m^2)$ per array. The only difference is that we may have to run the entire algorithm as much as $O(\log m)$ times. Thus we get the running time of $O(nm^4 \log m)$ steps.

∎

Theorem 2.15 also generalizes easily.

Theorem 3.4 *Suppose a linkage $\mathcal{L}= l_1 \ldots l_n$ with $L = max_i l_i^2$. Suppose $\mathcal{L}$ has at most e back edges per biconnected component of the dfs tree. Determining the minimal ϵ for which $\mathcal{L}$ can be ϵ-embedded in an $m \times m$ integer grid can be done in $O(nm^{4e} \log m)$ steps.*

Proof The proof is a quite natural combination of the ideas of Theorems 2.15 and 3.3. ∎

We observe that since $s_{n+1} - s_n = O(s_n^{1/4})$, the case with relative error can be handled in a similar way.

Acknowledgements This problem was posed by Maharaj Mukherjee at the Stony Brook Computational Geometry Workshop. Thanks also to Neil Immerman, for helpful insights into $\mathcal{NP}$-completeness, and to Hendrik W. Lenstra, Jr., for pointing out an error in an earlier version of this paper.

References

[1] A. Aho, J. Hopcroft and J. Ullman, *The Design and Analysis of Computer Algorithms,*, Addison Wesley, 1974.

[2] R.P. Bambah and S. Chowla, On Numbers which can be Expressed as the Sum of Two Squares, *Proc. Nat. Inst. Sci. India (1947)*, pp. 101-103.

[3] W. Bosma and M.P.M. van der Hulst, "Primality proving with cyclotomy," Proefschrift, Universiteit van Amsterdam, Amsterdam 1990.

[4] E. Grosswald, *Representations of Integers as the Sums of Squares*, Springer-Verlag, 1985.

[5] G. H. Hardy and E. M. Wright, *An Introduction to the Theory of Numbers*, Oxford University Press, 1971.

[6] J. Hopcroft, D. Joseph, and S. Whitesides, "Movement Problems for 2-Dimensional Linkages", pp. 282-303, in *Planning, Geometry and Complexity,*" by J. Schwartz, M. Sharir and J. Hopcroft, Ablex, 1987.

[7] J. Hopcroft, D. Joseph, and S. Whitesides, "On the Movement of Robot Arms in 2-Dimensional Bounded Regions," pp. 304-329, in *Planning, Geometry and Complexity,*" by J. Schwartz, M. Sharir and J. Hopcroft, Ablex, 1987.

[8] M. Mukherjee and G. Nagy, Collinearity Constraints on Spatial Subdivision Algorithms with Finite Precision Arithmetic, *Proceedings Fifth International Symposium on Spatial Data Handling,* pp. 425-431.

[9] R. Schoof, Elliptic Curves over Finite Fields and Computation of Square Roots Mod P, *Math. Comp. 44 (1985)*, pp. 483-494.

A Appendix

In this section, we show that:

Theorem A.1 *Let n be a positive integer with a bounded number of prime factors of the form $4k+1$, and no prime factors of the form $4k+3$. Then the following problems are polynomial time equivalent:*

1. *Enumerating the representations of n as the sum of two squares.*
2. *Factoring n.*

Proof We first observe that if n has a bounded number of prime factors of the form $4k+1$ and no prime factors of the form $4k+3$, then the number of representations of n as the sum of squares is polynomial in $\log n$. This is because if

$$n = 2^{\alpha}\Pi_{i=1}^{d} p_i^{a_i},$$

then

$$N_2(n) = 4\Pi_{i=1}^{d}(r_i+1) \leq 4((\max r_i)+1)^d.$$

But the maximum r_i is less than or equal to $\log r$, thus

$$N_2(n) \leq (\log n + 1)^d.$$

Since d is bounded, the number of representations of n as a sum of squares is polynomial in $\log n$.

Now we prove that if (1) is polynomial time reducible to 2. Since we can factor, the only issue is how to find a representation of each $4k+1$ prime p as a sum of two squares, $p = x^2 + y^2$. This way is easy due to a result of Schoof [9], who gave a polynomial time algorithm for finding square roots mod p. Thus one can quickly find an x such that $x^2 + 1 \equiv 0 \pmod{p}$, and therefore $x^2 + 1 = mp$ for some integer m. But then $\gcd(x+i, p)$ in $Z[i]$ gives an $a + bi$, where $p = a^2 + b^2$. Hence, (1) can be reduced to (2) in polynomial time.

We say two representations of n as the sum of two squares, $a^2 + b^2 = c^2 + d^2 = n$ are inequivalent, if $a \neq \pm c, \pm d, b \neq \pm d, \pm c$, with $\gcd(a,b) = \gcd(c,d) = 1$. Euler was the first to observe that two inequivalent representations of n as the sum of two squares leads to a factorization of n. A more modern treatment is in Bosma's thesis [3]; using the language of complexity, he shows that the reduction is polynomial time. ∎