Conditions for Unique Graph Embeddings

Bruce Hendrickson*

88-950
November 1988

Department of Computer Science
Cornell University
Ithaca, NY 14853-7501

*Supported by a Fellowship from the Fannie and John Hertz Foundation.

Conditions for Unique Graph Embeddings

Bruce Hendrickson*

29 November 1988

Abstract

The graph embedding problem is that of computing the relative locations of a set of vertices placed in Euclidean space relying only upon some set of inter-vertex distance measurements. This paper is concerned with the closely related problem of determining whether or not a graph has a unique embedding. Both these problems are NP-hard, but the proofs rely upon special combinations of edge lengths. If we assume the edge lengths are unrelated then the uniqueness question can be approached from a purely graph theoretic framework that ignores edge lengths. This paper identifies three necessary graph theoretic conditions for a graph to have a unique embedding in any dimension. Efficient sequential and NC algorithms are presented for each condition, although these algorithms have very different flavors in different dimensions.

*Supported by a Fellowship from the Fannie and John Hertz Foundation

1 Introduction

Consider a graph $G = (V, E)$ consisting of a set of vertices V and edges E, along with a real number $d(e_i)$ associated with each edge e_i. Now try to assign coordinates to each vertex so that the Euclidean distance between any two adjacent vertices is equal to the number associated with that edge. This is the *graph embedding problem.* It appears in situations where one needs to know the locations of various objects, but can only measure the distances between pairs of them. Surveying and satellite ranging are among the more obvious problems that can be expressed in this form [BR80,Whi84]. A less obvious but potentially more important application has to do with determining molecular conformations. It is possible to analyze the nuclear magnetic resonance spectra of a molecule to obtain pairwise inter-atomic distance information. Solving the graph embedding problem in this context would allow one to determine the three dimensional shape of the molecule, which is critically important in understanding the molecule's properties.

Unfortunately, the graph embedding (or *positioning*) problem is known to be very hard. Saxe [Sax79] has shown it to be NP-complete in one dimension and NP-hard for higher dimensions. In practice, this means that one is unlikely to find an efficient general algorithm to solve it. However, the graphs and edge lengths that Saxe uses in his proofs are very special and are highly unlikely to occur in practical problems. It may be possible to develop techniques that work well for examples of the problem that arise in practice.

If one wishes to determine the correct locations of atoms or satellites then it is essential that the graph embedding problem have a unique solution. (For our purposes translations, rotations and reflections of the entire space are not considered to be different solutions.) What properties must a graph possess to have only one satisfying configuration? That is the question this paper will address.

Saxe has shown this problem to be as hard as the original embedding problem, but again his proofs rely on very special graphs. In particular, he needs extremely unlikely combinations of edge lengths. As will become precise later, his graphs are very atypical. For *almost all* graphs it is possible to state necessary graph theoretic conditions for a unique embedding to exist. These conditions ignore the edge lengths and only consider the underlying unweighted graph. Certain unlikely combinations of distances require the *almost all* caveat, as will be explained in Sections 2, 3 and 4. By disregarding the edge lengths it is possible to make very strong probabilistic statements about whether or not the graph will have a unique embedding. It seems that no one has considered this problem before, so that results in this paper are entirely new.

How can a graph have multiple configurations? There are several distinct manners in which nonuniqueness can appear. First, the graph can be susceptible to continuous deformations, like the graph in Figure 1. The

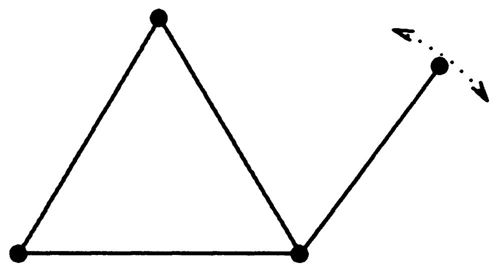

Figure 1: Example of a flexible graph in two dimensions.

rightmost vertex in this graph can pivot freely since it is underconstrained. A graph that can be continuously deformed while still satisfying all the constraints is said to be *flexible.* A nonflexible graph is *rigid.* Even a rigid graph can suffer from nonuniqueness. The graph in Figure 2 has two positionings in the plane. One half of the graph can reflect across the central two vertices. Continuous deformations and graph rigidity will be discussed in Section 2. Although graph rigidity is a well studied problem, the connections to the graph positioning problem have not been explored previously.

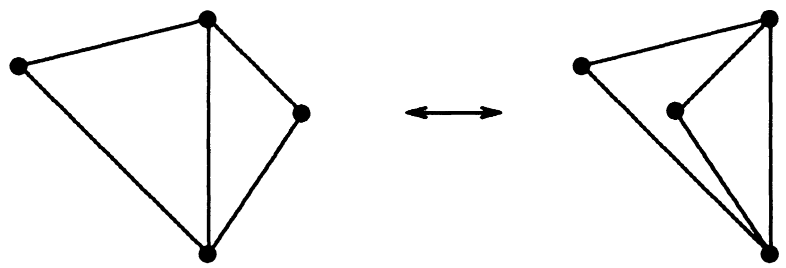

Figure 2: Example of a graph with two embeddings in the plane.

Discontinuous transformations like reflections will be covered in Sections 3 and 4. The definition of redundant rigidity in Section 4 and its importance in this context is entirely new.

The graph embedding problem can be posed in any dimension. Clearly the most practically interesting dimensions are two and three. Where possible this paper will present the most general results. However there are some substantial theoretical differences between two-space and three-space which will be elucidated in Section 2.

2 Graph Rigidity

A graph that has a unique embedding cannot be susceptible to continuous flexings. It must be rigid. Questions about the rigidity of graphs have occupied mathematicians for centuries. Among the more notable researchers to work in the field were Euler, Cauchy, Maxwell, Pascal and Tchebychef. More recently, structural engineers have been drawn to the problem because of novel building architectures like geodesic domes. The framework of a building can be thought of as a set of rigid rods, joined at their endpoints. One can consider the endpoints to be vertices of a graph and the rods to be edges of a fixed length. For the structure to bear weight, the corresponding graph must be rigid. For an old problem with an easy description the characterization of rigid graphs has proved to be difficult, and many important questions remain unanswered.

This section will develop the essentials of rigidity theory. Various workers from many fields have made contributions to the problem, and the result is a confusing lack of standard notation. The essential notation and basic results will be presented in Section 2.1. Characterizations of rigid graphs will be developed in Section 2.2, followed by algorithms in Section 2.3.

2.1 Basic Concepts

A mathematical analysis of rigidity requires a formal definition of the intuitive notion of a flexible structure. Everything in this section occurs in an arbitrary Euclidean dimension d.

Definition 2.1 *A* skeletal structure *consists of a graph $G = (V, E)$ and a positioning function p that maps vertices to points in Euclidean d-space. Such a structure, denoted (G, p), is called a* realization *of the graph G.*

Definition 2.2 *A* length preserving displacement *of a skeletal structure (G, p) consists of:*

a) an interval of real numbers $[\alpha, \beta], \alpha < 0 < \beta$, and

b) for every vertex a and every $\tau \in [\alpha, \beta]$ a point $p_\tau(a)$ in Euclidean d-space satisfying:

1. $p_0(a) = p(a)$,
2. *the function $p_\tau(a)$ is differentiable in τ,*
3. *(G, p_τ) is a skeletal structure, and*
4. *$|p_\tau(a) - p_\tau(b)| = |p(a) - p(b)|$ for every edge (a, b).*

A skeletal structure formalizes the intuitive notion of a positioned graph, while a length preserving displacement is a continuous flexing that preserves all the edge distances, parameterized by τ. It is convenient to think of τ as time. Note that a motion of the Euclidean space itself, a rotation or translation, satisfies the definition of a length preserving displacement. Such displacements are said to be *trivial.* In d-space there are d independent translations and $d(d-1)/2$ rotations.

For technical reasons the concept of a length preserving displacement is not the most convenient way to analyze a graph. It is easier to investigate a closely related infinitesimal property. This is a slight generalization, but it will prove to be essentially equivalent, as will be explained later. For now consider a flexible structure. As the structure flexes each vertex a has a well-defined velocity $\dot{p}(a)$.

Definition 2.3 *A* small displacement *of a skeletal structure consists of:*

> *a) a mapping* $\dot{p} : V \to \mathbf{R}^d$ *that assigns a velocity to each vertex, and*
>
> *b) a set of maps* $p_\tau : V \to \mathbf{R}^d$ *such that* $p_\tau(a) = p(a) + \tau\dot{p}(a)$ *for every vertex* a.

The set of all small displacements of a given skeletal structure constitute a $d|V|$ dimensional vectorspace. Figure 3 illustrates the difference between a length preserving displacement on the left, and a small displacement on the right. The movable vertex in the left picture is traveling in an arc, always maintaining the correct distance. In the right diagram the vertex continues to move in its initial direction.

Nothing in the definition of a small displacement ensures that the velocities assigned to the vertices actually correspond to a flexing that preserves edge lengths. A restraint on the velocities is required to enforce this property.

Definition 2.4 *A small displacement is* admissible *if for every edge* (a, b)

$$d/d\tau\{|p_\tau(a) - p_\tau(b)|^2\}_{\tau=0} \doteq (p(a) - p(b)) \cdot (\dot{p}(a) - \dot{p}(b)) = 0.$$

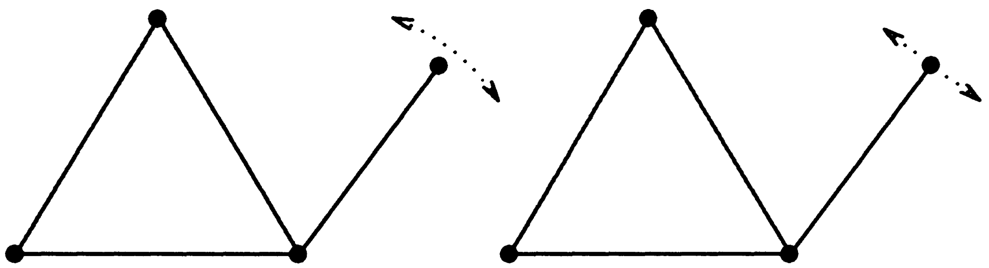

Figure 3: The difference between a length preserving displacement on the left, and a small displacement on the right.

Note that the form of the derivative requires that the difference in velocities between vertices a and b is perpendicular to the edge connecting them. Hence the distance between them won't change under an infinitesimal admissible small displacement. The motion depicted in the right diagram of Figure 3 is an admissible small displacement because the initial direction is length preserving.

The trivial translations and rotations of the Euclidean space satisfy the definition of admissibility, but they don't correspond to a flexing of the structure. This leads to the final definition.

Definition 2.5 *A skeletal structure is* rigid *if every admissible small displacement is trivial. Otherwise it is* flexible.

Some authors choose to call this property *infinitesimal rigidity* to emphasize the derivative in the definition of admissible small displacements.

To illustrate these definitions consider the closely related two dimensional structures depicted in Figure 4. All three have the same underlying graph, but their rigidity properties are very different. The leftmost structure is rigid. The middle one isn't. Although it has no length preserving

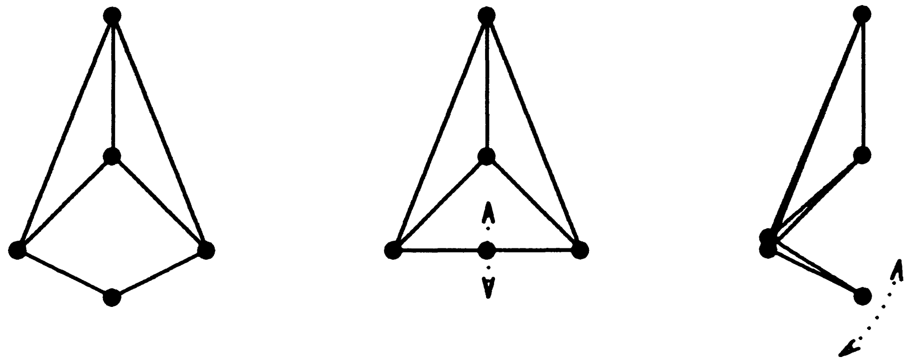

Figure 4: Similar structures with different rigidity properties.

displacements, the bottom center vertex can move infinitesimally in a direction perpendicular to the edges. In the rightmost structure, if the two vertices are made to overlap then the structure has a length preserving displacement. Clearly, rigidity is not an innate property of the graph, but depends upon the locations of the vertices as well. However, it seems that the center and rightmost structures are somewhat contrived, while the leftmost represents the more typical behavior. In this sense the underlying graph can be considered rigid. This characterization of graphs by their typical behavior is made possible by a theorem due to Gluck [Glu75].

Theorem 2.6 (Gluck) *If a graph has a single rigid embedding, then* almost all *its embeddings are rigid.*

Proof: A proof will be provided at the end of this section. □

Almost all in Theorem 2.6 means that the set of rigid positionings is open and dense in the space of all positionings. This is a very strong result. If a graph has a rigid realization then the set of embeddings that render the structure flexible has probability zero with respect to any Borel probability measure on the set of embeddings.

Theorem 2.6 is critical for a graph theoretic approach to the embedding problem. The structures made from a graph are either all flexible or almost all rigid. This allows for the characterization of graphs as either rigid or flexible without reference to the actual placements of the vertices. It also allows one to be somewhat cavalier in the distinction between rigid structures and graphs that have rigid realizations. Henceforth such graphs will be referred to as *rigid graphs.*

The relationship between length preserving displacements and admissible small displacements is well illustrated by the center structure in Figure 4. Clearly, any length preserving displacement is also an admissible small one, but as the figure indicates the converse is not always true. Special vertex positionings can be flexible according to Definition 2.5 even though they lack length preserving displacements. Roth [Rot81] has shown that these embeddings are very rare.

Theorem 2.7 *For a graph G, almost all flexible embeddings have length preserving displacements.*

Proof: See Roth [Rot81] for a proof. □

Flexible structures without length preserving displacements constitute a set of measure zero. So Definition 2.2, the intuitive notion of a flexible structure, is essentially equivalent to Definition 2.5, the formal one.

2.2 Characterizations of Rigid Graphs

How can a rigid graph be recognized? Clearly, graphs with many edges are more likely to be rigid than those with only a few. In some sense the edges are constraining the possible movements of the vertices. In d-space a set of $|V|$ vertices has $d|V|$ possible independent motions. Of these, d correspond to rigid translations and $d(d-1)/2$ to rotations, so the space of possible flexings has dimension $Q(d, |V|) \doteq d|V| - d(d+1)/2$. If each edge adds an independent constraint then Q edges are required to eliminate all nonrigid motions of the graph. This intuition is sound as the theorems in this section will demonstrate.

Definition 2.8 *The* rigidity matrix M *of a structure* (G, p) *has* $|E|$ *rows and* $d|V|$ *columns. Row* i*, corresponding to edge* $e_i = (v_k, v_l)$*, has* $2|d|$ *nonzero elements.*

$$M_{i,j} = \begin{cases} x_m(v_k) - x_m(v_l) & \text{if } j = d(k-1) + m \\ x_m(v_l) - x_m(v_k) & \text{if } j = d(l-1) + m \\ 0 & \text{otherwise} \end{cases}$$

Where $x_m(v)$ *is the* m^{th} *coordinate of vertex* v*; and* $1 \leq m \leq d$.

The rigidity matrix will appear repeatedly in what follows. It is important because it is the matrix of the system of equations for $\dot{p}$,

$$(p(a) - p(b)) \cdot (\dot{p}(a) - \dot{p}(b)) = 0, \ \forall (a, b) \in E.$$

That is, these equations can be expressed as $M\dot{p} = 0$. The rank of the rigidity matrix is a crucial quantity as the following theorem indicates.

Theorem 2.9 *The following assertions are equivalent:*

A. The skeletal structure (G, p) *is rigid.*

B. If for some velocity assignment $\dot{p}$*,* $(p(a) - p(b)) \cdot (\dot{p}(a) - \dot{p}(b)) = 0$ *for all edges* (a, b)*, then* $\dot{p}$ *is trivial.*

C. The rigidity matrix M *of* (G, p) *has rank* $Q(d, |V|)$.

Proof: The equivalence of A and B follows directly from the differentiation of $\{|p_\tau(a) - p_\tau(b)|^2\}_{\tau=0}$ in Definition 2.4. As observed above, there is a space of dimension $Q(d, |V|)$ of trivial velocities. And admissible small velocities must be in the null space of M since the rigidity matrix expresses all the constraints induced by the edges. If the null space of M is of rank $Q(d, |V|)$ then the only admissible small displacements are the trivial ones. This establishes the equivalence of B and C. □

Theorem 2.10 *Every rigid skeletal structure (G, p) has a rigid substructure with exactly $d|V| - d(d+1)/2$ edges.*

Proof: This follows from Theorem 2.9, C. The rigidity matrix has rank $Q(d, |V|)$ and each of its rows corresponds to an edge. Simply discard redundant rows and the corresponding edges until only Q remain. □

Corollary 2.11 *For a structure (G, p), if $|E| > Q(d, |V|)$ then there is linear dependence among the rows of the rigidity matrix.*

Proof: The maximum rank of the rigidity matrix is $Q(d, |V|)$. □

Theorem 2.12 *A rigid skeletal structure (G, p) with $Q(d, |V|)$ edges has no substructure G' with $|V'| \geq d$ vertices and more than $Q(d, |V'|)$ edges.*

Proof: Since there are only $Q(d, |V|)$ edges, their rows in the rigidity matrix must all be independent by Theorem 2.10. Now assume the existence of a subgraph $G' = (V', E')$ with $|E'| \geq Q(d, |V'|)$. By Corollary 2.11 there must be linear dependence among these edges which is a contradiction. □

Theorems 2.10 and 2.12 say that a rigid graph with $|V|$ vertices must have a set of $Q(d, |V|)$ *well distributed* edges, where *well distributed* means that no subgraph with $|V'|$ vertices has more than $Q(d, |V'|)$ edges. This requirement is often referred to as *Laman's condition* after G. Laman [Lam70] who first articulated the two dimensional version where $Q = 2|V| - 3$. This condition is necessary for a graph to be rigid in any dimension. Laman was able to show that in two dimensions it is also sufficient. This was the first graph theoretic characterization of rigid graphs in 2-space.

Theorem 2.13 (Laman) *A graph $G = (V, E)$ is rigid in two-space if and only if it contains a subgraph $G' = (V, E')$ with $|E'| = 2|V'| - 3$ and no subgraph G'' of G' has more than $2|V''| - 3$ edges.*

Laman's condition is also sufficient in one dimension where $Q = d-1$. It is easy to show that this is equivalent to requiring the graph to be connected.

Unfortunately, for all its intuitive appeal Laman's condition isn't sufficient in higher dimensions. A three dimensional counterexample is depicted in Figure 5. Although this graph has the required $3|V| - 6$ well distributed

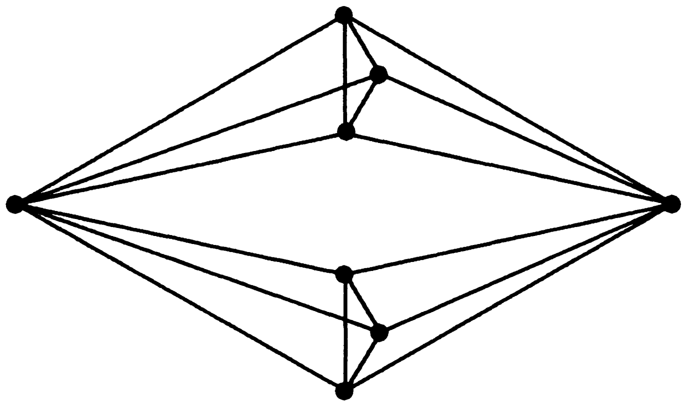

Figure 5: A flexible graph in 3-space that satisfies Laman's condition.

edges it is still flexible. The top and bottom halves can pivot about the left and right-most vertices.

The problem with Figure 5 is that its edges are not independent in the sense of Theorem 2.9 C. The rows of the rigidity matrix are linearly dependent. Expressing this independence graph theoretically has proved to be a very difficult problem. No characterization of rigid graphs in three dimensions is known, although the problem has been considered by many researchers. Several classes of graphs have been shown to be rigid in 3-space. These include triangulated planar graphs [Cau13] and complete bipartite graphs with at least five vertices in each component [BR80,Whi84]. But the characterization of general graphs remains open.

Recent work by Tay, Whiteley and Graver [TW85] has brought a characterization tantalizingly within reach. However, it is difficult to see how this possible characterization could lead to an efficient algorithm. Any straight-forward implementation of their approach would have a worst case exponential time behavior.

It is now possible to prove Theorem 2.6.

Theorem 2.6 (Gluck) *If a graph G has a single rigid embedding, then* almost all *its embeddings are rigid.*

Proof: By Theorem 2.9 a skeletal structure (G, p) is rigid if and only if the rigidity matrix has a set of $Q(d, |V|)$ independent rows. If it does, then the determinant of some $Q(d, |V|)$ by $Q(d, |V|)$ submatrix is nonzero. But this determinant can be expressed as a polynomial in the coordinates of the positions of the vertices. Since this polynomial is not identically zero, the positionings that make it zero constitute a set of measure zero. □

A realization of a rigid graph G will be flexible only if the determinant of the normally independent rows and columns of the rigidity matrix happens to be zero. That is, the coordinates of the vertices satisfy a very special algebraic relationship. To avoid this problem, and for future convenience, it is useful to make the following definition.

Definition 2.14 *A realization (G, p) is* generic *if the vertex coordinates are algebraically independent over the rationals. Furthermore, a set of edges in a graph are* independent *if their columns in the rigidity matrix are linearly independent in a generic realization.*

Generic realizations of rigid graphs are always rigid. They also possess other nice nondegeneracy properties that will be of use later, like having no $k+1$ vertices lying in a k dimensional subspace. Strictly speaking, we should be referring to a graph as either *generically rigid* or *generically flexible*, but in the interest of typographic brevity the the *generic* will be assumed. Although the requirement of algebraic independence is actually stronger then we will need, the set of generic realizations is open and dense $\mathbf{R}^{d|V|}$. Non-generic realizations constitute a set of measure zero.

2.3 Algorithms for Rigidity Testing

In one-space, rigidity is equivalent to connectivity. There are simple connectivity algorithms that run in time proportional to the number of edges in the graph [AHU74].

In two dimensions Laman's condition is necessary and sufficient. However, as given in Theorem 2.13 this condition involves looking at every subgraph. Since there are an exponential number of these, this does not lead to a very efficient algorithm. Various equivalent characterizations have been discovered. They all depend upon the existence of $2|V| - 3$ independent edges. Several characterizations of independence are collected in the following theorem.

Theorem 2.15 *For a graph $G = (V, E)$ the following are equivalent:*

> *A. The edges of G are independent;*
>
> *B. no subgraph $G' = (V', E')$ has more than $2|V'| - 3$ edges;*
>
> *C. for each edge (a,b), the graph formed by adding three additional edges between a and b has no subgraph in which $|E'| > 2|V'|$;*
>
> *D. for each edge (a, b), the graph formed by adding three additional edges between a and b contains two edge disjoint matchings of edges to vertices.*

Proof: The equivalence of A and B is Laman's theorem. Property C is easily shown to be equivalent to B. And the equivalence of D and C is a variant of Hall's Theorem, a standard result in matching theory. Assertions C and D were first discovered in a slightly different form by Sugihara [Sug80]. □

Corollary 2.16 *For a graph with* $2|V|-3$ *edges, properties B, C and D from Theorem 2.15 are equivalent to two dimensional rigidity.*

Each of the characterizations B through D can be used to test for the independence of edges in a two dimensional graph. But as discussed above, property B generates a poor algorithm. The best algorithm for independence testing comes from characterization D. This condition can be reduced to a sequence of bipartite matchings for which there are efficient algorithms. Let the graph from property D formed by quadrupling edge (a, b) be denoted by $\overline{G}_{a,b} = (V, \overline{E})$. Construct the bipartite graph $B_{a,b} = (V_1, V_2, \mathcal{E})$ as follows: let $V_1 = \overline{E}$, $V_2 = \{p_1, q_1, \ldots, p_n, q_n\}$, and $\mathcal{E} = \{(e, p_i), (e, q_i), (e, p_j), (e, q_j) | e = (v_i, v_j) \in \overline{E}\}$. This bipartite graph has the edges of $\overline{G}$ as one of its edge sets, and two copies of the vertices of $\overline{G}$ for the other. Edges of $B_{a,b}$ connect the edges of $\overline{G}$ with their incident vertices. Hence, a complete matching in $B_{a,b}$ corresponds to two edge disjoint matchings of edges to vertices in $\overline{G}$. So by property D two dimensional independence testing can be reduced to a set of bipartite matchings, one for each edge of G.

Bipartite matching is a well studied problem. The best algorithm runs in time $O(rs^{1/2})$ where r and s are respectively the number of edges and vertices in the bipartite graph [HK73,Gal83]. This technique involves Hungarian trees and augmenting paths. The basic idea is to look for a path from an unmatched vertex in V_1 to an unmatched vertex in V_2 that alternates between edges that are not in the current matching and edges that are. When such an augmenting path is found the matching can be enlarged by changing the unmatched edges in the path to become matching edges, and vice versa. These paths can be found by growing Hungarian trees from the unmatched vertices in V_1. These trees grow along the unmatched edges from the starting vertex to its neighbor set in V_2. Matching edges are followed back to V_1 and unmatched edges back to V_2. If an unmatched vertex in V_2 is ever encountered an augmenting path has been identified. Forming these trees requires time $O(r)$. For a complete description of this approach see any good book on graph algorithms [GM84].

A straight-forward way to test for independence would be to run the bipartite matching algorithm for each of the $|E|$ graphs $B_{a,b}$ formed by quadrupling an edge (a, b). This would require time $O(|E||V|^{3/2})$ since the number of vertices and edges in the bipartite graph are both $O(|V|)$. A better approach that actually identifies a maximal set of independent edges would try to add edges one at a time, keeping independent ones and discarding the rest. At a typical step, the algorithm will have identified k independent edges and it will try to add a new edge (a, b). Say these $k+1$ edges comprise a subgraph G' of G. If (a, b) is not independent of the other k edges then by characterization B it must fall in a subgraph of G' with too many edges. When this edge is quadrupled it will lead to a violation of property C and so the matching from property D will fail. The key point is that only edge (a, b) need be quadrupled. It would be redundant to quadruple the other k edges since the dependence will be identified by the single matching quadrupling (a, b). This gives an algorithm that runs in time $O(|E||V|^{3/2})$, but it is more efficient than the previous approach since dependent edges are discarded and the matchings involve smaller bipartite graphs. If $2|V| - 3$ independent edges are identified then the graph is rigid.

This algorithm can be further improved by taking advantage of the matching that has already been identified in demonstrating the independence of the current set of k edges. Trying to add the new edge (a, b) will involve adding four vertices to V_1 corresponding to the quadrupling of (a, b). Growing the Hungarian trees from these new vertices will require time $O(|V|)$ which is cheaper than performing a new matching from scratch by a factor $|V|^{1/2}$. The entire algorithm will take time $O(|E||V|)$. This is better than the current best algorithm due to Imai which runs in time time $O(|E|^2)$ [Ima85]. Most graphs of interest have $O(|V|)$ edges in which case the asymptotic running times of both algorithms is $O(|V|^2)$.

There are also fast parallel bipartite matching algorithms. In particular, Karp, Upfal and Wigderson [KUW86] have discovered a random NC algorithm. The class NC is the set of problems that can be solved in polylogarithmic time using a polynomial number of processors. It is a standard measure of *good* parallel algorithms, although it's applicability is more theoretical than practical. Unfortunately, this algorithm for two dimensional rigidity testing involves a sequence of matchings. Although the matchings can be done quickly in parallel, the full algorithm is not in random NC. However, as we will see there is another way of approaching rigidity that generates fast random sequential and parallel algorithms in any dimension.

Since Laman's theorem only characterizes rigidity in the plane, the bipartite matching approach does not generalize to higher dimensions. In fact, since there is no graph theoretic characterization of rigidity for $d \geq 3$, there is no deterministic graph algorithm to distinguish between rigid and flexible graphs. A different approach is possible that relies instead upon Theorem 2.6. Since this theorem is valid in all dimensions, the following discussion is applicable to all spaces. If the graph is rigid then almost any set of positions for the vertices will yield a rigid structure. Once these locations are selected it is a straight-forward manner to determine the rigidity of the structure using Theorem 2.9. Simply construct the rigidity matrix M and determine its rank. If the rank is $d|V| - d(d+1)/2$ then the graph is rigid. A lower rank indicates that the structure is flexible. Unless the selection of vertex coordinates was extremely unlucky the graph must be flexible as well. So even without a graph theoretic characterization an efficient practical randomized algorithm for rigidity exists.

In principle the rank of the rigidity matrix, M, can be determined by Gaussian elimination which will require $O(mn^2)$ time, where $m = |E|$ and $n = |V|$. When used to determine the rank, Gaussian elimination can suffer from numerical instability so a more sophisticated numerical technique is probably better. A QR decomposition with column pivoting is about twice as expensive as Gaussian elimination, but it is more numerically stable [GVL83]. Performing this calculation on M^T will identify a maximal independent set of rows of M, corresponding to a maximal set of independent edges in the graph. In principle the rank will be determined by the appearance of zeros below the diagonal in the course of computing the decomposition. In practice this technique can suffer from roundoff errors.

It can be very difficult to determine the rank of a matrix unambiguously since there will generally be some residual nonzero elements due to round-off. Instead of relying on the appearance of zeros, the algorithm could try to add the new column and then estimate the resulting condition number [Bis88]. A large condition number would imply a nearly singular matrix, meaning that the new column is nearly linearly dependent upon the others and should be discarded. A safer, but still more expensive approach to find the rank of the rigidity matrix would use a singular value decomposition. The singular values provide a good measure of how nearly singular the matrix is. Very small values would correspond to residual roundoff errors instead of true nonzeros.

Even if the graph is rigid so the rigidity matrix is full rank, numerical problems can arise. If the randomly selected embedding happens to be close to one from the set of measure zero in which the structure is flexible, then numerical precision will be a serious problem. This will be true no matter what technique is used to determine the rank. To be safe, the algorithm could select a new set of random vertex locations if the rank is deficient in the first attempt. A second or third failure would greatly decrease the risk of falsely declaring a graph to be flexible.

The rows of the rigidity matrix have only $2d$ nonzeros each, so sparse techniques should be employed for large problems. Finding the rank of a general sparse matrix is more complicated than the corresponding dense problem. The straight-forward approaches will cause many of the initial zeros in the matrix to become nonzero in an intermediate computation, greatly increasing the memory and computational requirements. Sparse techniques try to minimize this *fill*. There is no known sparse singular value decomposition, so sparse QR is the method of choice [CEG86,GH80].

There are also efficient parallel algorithms for finding the rank of a matrix. Ibarra, Moran and Rosier [IMR80] discovered an algorithm that runs in $O(m^4)$ time on $O(\log^2 m)$ processors, placing rank finding in NC. This means that rigidity testing is in random NC for any dimension.

3 Reflections

Even rigid graphs can have multiple embeddings as was shown in Figure 2. This discontinuous flavor of nonuniqueness has not been well studied, probably because it is not relevant to structural engineers. A geodesic dome can only deform continuously. For the graph embedding problem these discontinuous transformations must be considered. This section and the next will be concerned with multiple embeddings that do not arise from flexibility. These are cases in which there are two or more noncongruent embeddings that satisfy all the distance requirements, but there is no continuous flexing of the graph to transform one to another while maintaining the constraints.

An important difference between flexibility and the discontinuous transformations that will be considered here has to do with the number of acceptable positionings for the graph. A flexible graph has an infinite number of embeddings, but a nonflexible graph can only have a finite number. In principle this makes a listing of all satisfying embeddings possible, although the actual number could be large. In fact, it could be exponential in the size of the graph.

The simplest type of discontinuous transformation is depicted in Figure 2. The right half of this graph is able to fold across the line formed by the two middle vertices. When can this type of nonuniqueness occur? As in Figure 2 there must be a few vertices about which a portion of the graph can be reflected. These vertices form a *mirror*. There must be no edges between the two halves of the graph separated by this mirror. For the general d dimensional problem, the mirror vertices must lie in a $d-1$ dimensional subspace.

Definition 3.1 *A structure in d-space has a* reflection *if a separating set of vertices lie in a $d-1$ dimensional subspace.*

If one considers the set of all possible vertex locations, those in which more than $d+1$ vertices lie in a $d-1$ dimensional subspace constitute a set of measure zero. These degenerate placements are a special case of nongeneric realizations. For graphs generated from random vertex locations the probability of $d+1$ vertices falling in a $d-1$ dimensional subspace is zero. So for almost all structures, reflections only occur when there is a subset of d or fewer vertices whose removal separates the graph into two or more unconnected pieces. More formally we have the following theorem.

Theorem 3.2 *A rigid graph positioned generically in dimension d will have no reflections if and only if it is $d+1$-connected.*

The connectivity of a graph is a very important property, and it has been well studied. There are well known $O(m)$ time algorithms for biconnectivity, where m is the number of edges in the graph [AHU74]. Avoiding reflections in two dimensions requires a triconnected graph. Hopcroft and Tarjan [HT73] were the first to discover an $O(m)$ time algorithm to find triconnected components, making the two dimensional reflection problem resolvable in linear time. Miller and Ramachandran [MR87] have recently proposed a parallel algorithm to identify triconnected components in $O(\log^2 n)$ time with $O(m)$ processors, where n is the number of vertices. This places triconnectivity in NC.

Four-connected components are more difficult, but Kanevsky and Ramachandran [KR] have recently found an $O(n^2)$ time algorithm. They also discovered a parallel implementation of their algorithm that runs in $O(\log^2 n)$ time using $O(n^2)$ processors. So the problem of reflections is in NC in both two and three dimensions.

For k greater than 4, the question of k-connectivity is less well understood. Consequently the reflection problem is more difficult in spaces of dimension greater than three. A simplistic approach would involve deleting each vertex in turn and running an algorithm for $(k-1)$-connectivity. This idea can be applied recursively with a base case of Kanevsky and Ramachandran's 4-connectivity algorithm. For a fixed k this gives an NC algorithm. Test for 4-connectivity in parallel all subgraphs formed by deleting $k-4$ vertices. But for any moderate k this approach is not very practical. Currently, the best sequential algorithms for general k-connectivity are randomized algorithms that run in time $\Omega(n^{5/2})$ [B+82,LLW86]. The lack of good deterministic algorithms is not particularly troubling for the reflection problem since the guarantees of uniqueness are probabilistic anyway.

4 Redundant Rigidity

Rigidity and $d+1$-connectivity are necessary but not sufficient to ensure that a graph has a unique embedding. A two dimensional example of a rigid, 3-connected graph with two embeddings is shown in Figure 6. To

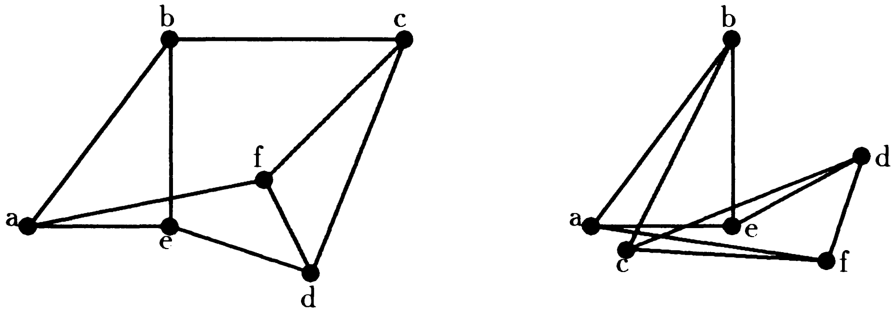

Figure 6: Two embeddings of the same rigid triconnected graph in the plane.

understand this nonuniqueness, consider Figure 7. Edge (a,f) has been removed from the original graph. This resultant graph is now composed of a quadrilateral *bcde* with two attached triangles *abe* and *cdf*. The quadrilateral is not rigid, so this new graph can flex. The flexing will lift vertex d up until it crosses the line between c and e as depicted in the center picture of Figure 7. Eventually vertex c can swing all the way around to the right. As the graph, moves the distance between vertices a and f varies. When

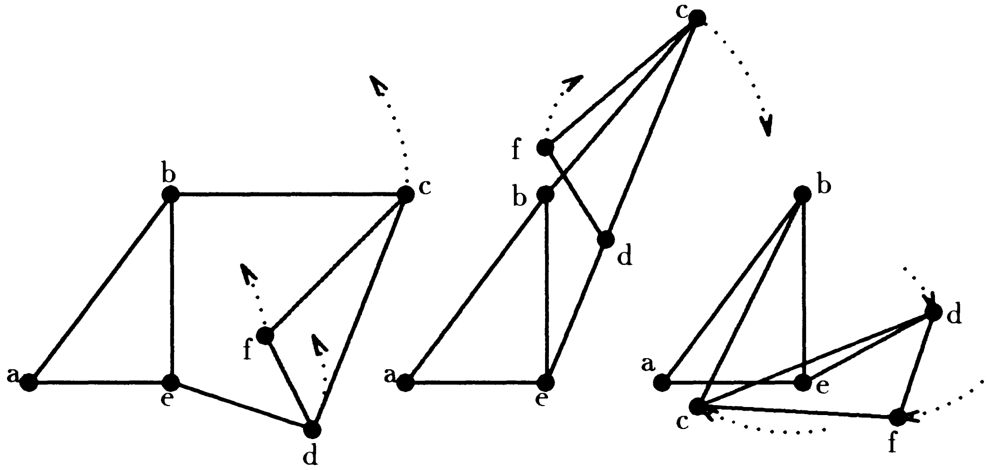

Figure 7: Intermediate stages in the construction of Figure 6. See text for explanation.

vertex c swings far enough around, this distance becomes the same as it was originally, as shown in the rightmost picture of Figure 7. Now the missing edge can be replaced to yield a new embedding.

The critical problem with the graph in Figure 6 is that the removal of a single edge makes it flexible.

Definition 4.1 *An edge, e, of a structure is* redundant *if the structure remaining after the removal of e is rigid. A structure is* redundantly rigid *if all its edges are redundant.*

Redundant rigidity is clearly a more restrictive property than rigidity, but the two properties have many similarities as the following theorem indicates.

Theorem 4.2 *If a graph has a redundantly rigid embedding, then almost all of its embeddings are redundantly rigid.*

Proof: This is a straight-forward consequence of Gluck's result, Theorem 2.6. □

As with Theorem 2.6 for rigid graphs, Theorem 4.2 says that either none of a graph's positionings are redundantly rigid, or almost all of them are. *Almost all* has the same meaning here that it did in Theorem 2.6. This blurs the distinction between a redundantly rigid structure and its underlying graph. Graphs with redundantly rigid embeddings will be referred to as *redundantly rigid graphs.*

In Figure 6 the lack of redundant rigidity led to multiple embeddings. This is true in general, and the proof will be the main result of this section. Intuitively, a flexible structure can move around, but it must always end up back where it started. That is, the path it traces in $d|V|$-space will be a loop. If the removal of an edge allows the graph to flex, then the distance corresponding to that edge must be a multivalued function as the flexing completes its loop. To prove this formally will require an incursion into differential topology. The results needed here can all be found in the first chapter of Guillemin and Pollack's excellent introductory text [GP74].

The proof that flexible graphs typically move in closed loops will rely upon some special properties of the graph embedding problem. Given a graph G, a selection of vertex coordinates induces a set of edge lengths. There is an *edge function* $f : \mathbf{R}^{d|V|} \to \mathbf{R}^{|E|}$ that maps vertex locations to squares of edge lengths. That is,

$$f(p(v_1), \ldots, p(v_{|V|})) = (\ldots, |p(v_i) - p(v_j)|^2, \ldots), \quad (v_i, v_j) \in E$$

The embedding problem is really the problem of finding the inverse of the edge function. Of course, this inverse is multivalued because edge lengths are invariant to translations, rotations and reflections of the entire space. Two embeddings will be considered *congruent* if all pairwise distances between vertices are the same under the two positionings. We will only be interested in the inverse of f modulo congruencies.

The function f has many nice properties. It has continuous derivatives of all orders at all points. Also, the Jacobian of f is the rigidity matrix introduced in Section 2.2. Additional properties are presented in the following lemmas and theorems.

Definition 4.3 *Let $f : \mathbf{R}^k \to \mathbf{R}^j$ have continuous partial derivatives of all orders. A point $x \in \mathbf{R}^k$ is a* regular point *if the Jacobian of f is full rank at x. A point $z \in \mathbf{R}^j$ is a* regular value *if every point in the preimage of z is a regular point.*

Non-regular points of the edge function are points at which the determinant of the rigidity matrix is zero. This implies a polynomial relationship among the coordinates of the vertices, so these points are nongeneric. All generic points are regular points.

Lemma 4.4 *Let $G = (V, E)$ be a graph in which all of the edges are independent. If (G, p) is a generic realization of G then no edge length is at a minimum or maximum allowable value.*

Proof: Assume edge $e_{i,j}$ takes on the maximum value allowed by the other edge lengths. Then the vertices are in locations that solve the optimization problem

$$max\{|p(v_i) - p(v_j)|^2 : |p(v_k) - p(v_l)|^2 = d_{k,l}^2, \quad (v_k, v_l) \in E\}.$$

A necessary condition at the solution point is that the gradient of the objective function is a linear combination of the gradients of the constraints [Fle81]. But this implies that the rigidity matrix is not of full rank, so (G, p) is not a generic realization. A parallel argument works for minimum values. □

Lemma 4.5 *Let $G = (V, E)$ be a graph in which all of the edges are independent, and let (G, p) be a generic realization of G in dimension d, inducing a point z in $\mathbf{R}^{|E|}$-space. There is an open neighborhood about z in which every point corresponds to a possible realization of G.*

Proof: No edge length can be extremal by Lemma 4.4. Say the nearest embedding that makes for an extremal edge length is at a distance δ from p in $\mathbf{R}^{d|V|}$-space. Since the edges are independent, for any edge e there is a flexing of the graph that alters the length of e while preserving all the remaining edges. Since the length of edge e is not extremal it can be shortened or lengthened. All the edge lengths can be continuously and independently varied until an extremal length is encountered. As long as no edge is changed by more than $(\frac{\delta}{d|V|})^{\frac{1}{2}}$ extremal lengths are avoided. The combination of all of these flexings defines a sphere in $\mathbf{R}^{|E|}$ about z in which every point corresponds to a realization of G. $\square$

Lemma 4.6 *Let G be a graph in which all of the edges are independent. Then for almost all positionings $p(V)$, $f(p(V))$ is a regular value.*

Proof: By Lemma 4.5 there is an open neighborhood about $f(p(V))$ of possible realizations of G. By Sard's theorem (see [GP74]), the set of regular values is open and dense in $\mathbf{R}^{|E|}$. So for almost all p, $f(p(V))$ is a regular value. $\square$

Theorem 4.7 *Let G be a graph in which all of the edges are independent. Then for almost all positionings $p(V)$, the preimage of $f(p(V))$, that is $\mathcal{P} \doteq \{x \in \mathbf{R}^{d|V|} \mid f(x) = f(p(V))\}$, is a manifold.*

Proof: For almost all p, $\mathcal{P}$ is the inverse image of a regular value, by Lemma 4.6. By the Inverse Function Theorem of differential topology $\mathcal{P}$ must be a manifold. $\square$

Theorem 4.8 *Let $G = (V, E)$ be a connected graph with realization (G, p). Then the preimage of $f(p(V))$ modulo congruencies, $\mathcal{P}$, is compact.*

Proof: $\mathcal{P}$ is compact because it is closed and bounded in $\mathbf{R}^{\mathbf{d}|\mathbf{V}|}$. It is bounded because G is connected. It is closed because its complement is open. To see this last assertion, consider a point $y \in \mathbf{R}^{\mathbf{d}|\mathbf{V}|}$ that is not in $\mathcal{P}$. Some edge constraint must be violated at y by some length δ. The sphere around y of radius $\delta/2$ defines an open neighborhood that is in the complement of $\mathcal{P}$. This shows that the complement of $\mathcal{P}$ is open, finishing the proof. $\square$

Theorem 4.9 *Let G be a connected graph with $d|V| - d(d+1)/2 - 1$ independent edges. Then for almost all positionings $p(V)$, the structure (G, p) will be flexible, and the flexing will follow a closed loop in $d|V|$-space.*

Proof: By Theorem 2.9, G is flexible, so almost all tis realizations will be flexible by Gluck's Theorem. By the Preimage Theorem from differential topology the preimage of $f(p(V))$ will be of dimension one. And by Theorems 4.8 and 4.7 this preimage will be a compact manifold. All compact manifolds of dimension one are diffeomorphic to the circle. $\square$

Corollary 4.10 *If G is a flexible graph, then for almost all positionings $p(V)$, the structure (G, p) will be flexible, and the flexing can follow a closed loop in $d|V|$-space.*

Proof: This is a straight-forward generalization of Theorem 4.9. There are three cases to consider. First, if the graph is not connected then it is obvious. Second, if the edges are all independent then this is an easy extension of the previous theorem. Finally, if some of the edges are dependent then the allowable flexes are determined by a maximal set of independent edges, which is again a straight-forward extension of Theorem 4.9. $\square$

Theorem 4.11 *If a graph G is not redundantly rigid then for almost all positionings $p(V)$ the realization (G,p) will have multiple embeddings.*

Proof: Assume G is rigid. The other case is trivial. The removal of some edge $e_{i,j}$ between vertices i and j makes the resulting graph G' flexible. By Corollary 4.10 for almost all p, (G',p) can flex in a closed loop, diffeomorphic to the circle. As it flexes the distance between vertices i and j will vary. Since this flexing is a loop every possible value for this distance will occur multiple times, except for the extremal values. Realizations that induce extremal values constitute a set of measure zero. So for almost all positionings there are at least two distinct points in $d|V|$-space that satisfy all the constraints. $\square$

Theorem 4.11 means that the example in Figure 6 was not a fluke. Redundant rigidity is a necessary condition for unique embedibility.

How difficult is it to test for redundant rigidity? A simplistic approach would use the algorithm for rigidity repeatedly, removing one edge at a time. This would require time $O(|E|K)$, where K is the time required for testing rigidity. This approach parallelizes easily by simply running the $|E|$ different problems on independent sets of processors. Since rigidity testing was shown to be in random NC, redundant rigidity is as well.

In one dimension redundant rigidity is equivalent to edge biconnectivity; that is, removal of any edge leaves the graph connected. This property can be determined by finding the cut points of the graph, requiring $O(|E|)$ time [AHU74].

For the two dimensional case a modification of the rigidity testing algorithm described in Section 2.3 can be employed. The rigidity algorithm grows a set of independent edges one at a time by checking them against the existing independent set. If a new edge e is found to be independent of the existing set, then it is added. Independence is determined by the success of a particular bipartite matching. If the matching fails then there must be some dependence among the edges. Identifying and utilizing these dependencies will lead to an efficient redundant rigidity algorithm. This will require some further insight into the rigidity algorithm.

Claim 4.12 *In the rigidity algorithm of Section 2.3, if the new edge (a, b) were tripled instead of quadrupled then a complete matching in the bipartite graph $B_{a,b}$ would always exist.*

Proof: Assume the claim is false. Then by Hall's theorem from matching theory the addition of the three new vertices to V_1 must create a subset of V_1, $\mathcal{V}_1$, that is adjacent to a smaller subset of V_2, $\mathcal{V}_2$. If the three new vertices are replaced by three copies of one of the other vertices in $\mathcal{V}_1$ then this violation of the Hall property would still exist. That would correspond to quadrupling one of the edges in the set that is already known to be independent. But this graph must have a complete matching which is a contradiction. □

This property ensures that at least three of the four vertices added to V_1 can be matched. Only a single vertex can go unmatched. This is important because of the following general property of bipartite matching.

Theorem 4.13 *Let $B = (V_1, V_2, \mathcal{E})$ be a bipartite graph with a matching from V_1 to V_2 involving all but one vertex from V_1, denoted by u. Also let $\mathcal{V}_1$ be the subset of V_1 that is in the Hungarian tree built from v. Then if any vertex from $\mathcal{V}_1$ is deleted from B, the resultant graph will have a complete matching.*

Proof: The removal of a vertex w from $\mathcal{V}_1$ creates an unmatched vertex in V_2 that is reachable from u along an alternating path. □

Theorem 4.14 *In the rigidity algorithm, assume a new edge, (a, b), is not found to be independent of the current set of k independent edges. Let $\mathcal{V}_1$ be the set of vertices of V_1 that are in the Hungarian tree of the failed matching. Then if (a, b) replaces any of the edges in $\mathcal{V}_1$ the resulting set of k edges is still independent.*

Proof: By Claim 4.12 three of the four new vertices will match. When the Hungarian tree is built on the fourth, Theorem 4.13 guarantees that a complete matching will exist if any of the vertices of $\mathcal{V}_1$ is removed. The removal leaves a set of $k-1$ independent edges. The new edge must be independent of these $k-1$ because the appropriate bipartite graph has a complete matching. □

Theorem 4.14 gives an efficient algorithm for redundant rigidity testing that is a simple extension of the rigidity algorithm. An edge is not independent of the current set if the bipartite matching fails. When this happens the Hungarian tree identifies precisely which edges are dependent. All these edges are redundant because any of them could be replaced by the new edge. After all the edges have been tested, if there are $2|V|-3$ independent ones and they are all redundant, then the graph is redundantly rigid. Otherwise it is not. This takes essentially no more effort than testing for rigidity, so two dimensional redundant rigidity can be determined in $O(|E||V|)$ time.

As was discussed in Section 2.3, the algorithms for rigidity in the plane are very different from those in higher dimensions. However, a higher dimensional redundant rigidity algorithm has many similarities to the planar one. As in the two dimensional case the basic idea is to construct an independent set of edges, while determining which of them are redundant. This can be decided by building the set of independent edges one at a time. If a new edge is not independent of the current set, then before discarding him determine which of the current edges he could replace. All of these replaceable edges are redundant. If a full contingent of independent edges is found and they can each be replaced, then the graph is redundantly rigid.

This is where the similarities between two and three dimensions ends. In the higher dimensional case the determination of edge independence is more difficult since there is no graph theoretic characterization. Begin by positioning the vertices randomly and constructing the rigidity matrix M. The rigidity of the structure can be determined by performing a QR factorization on M^T to find its rank. This procedure will form an independent set of edges one at a time, where independence refers to linear independence of the corresponding columns of M^T. A new column is added if it is linearly independent of the current set of k columns; otherwise it is discarded. A discarded edge e is linearly dependent upon some set of current edges. It can be identified by the absence of non-zeros values below position k. As in the two dimensional case, the edge e could replace any of the edges in the linear combination which forms it.

How difficult is it to determine which of the current columns contribute to the linear combination? Assume the algorithm has identified k independent columns of M^T. Place these columns together to form a $d|V| \times k$ matrix, A_k. The QR factorization has been proceeding on these columns as they are identified, so there is a $k \times k$ unitary matrix Q_k and a $d|V| \times k$ upper triangular matrix R_k satisfying $Q_k R_k = A_k$. If a new column b of M^T is linearly dependent upon the columns of A_k then there must be a vector c satisfying $A_k c = Q_k R_k c = b$, or alternately $R_k c = Q_k^T b$. In the course of the QR factorization the column b has been overwritten with $Q_k^T b$, so it is easy to solve the upper triangular system for c. The nonzero elements of c identify which columns of A_k contribute to the linear combination composing b, that is, which columns are redundant. How much work does this take? There are $O(|E|)$ triangular systems to solve, each of which requires $O(k^2)$ operations, where k is always $O(|V|)$. So the total additional time is of the same order as the QR factorization itself, $O(|E||V|^2)$. As in the two dimensional case, the redundant rigidity of a graph can be determined by modifying the rigidity algorithm without incurring substantial increased cost.

As was noted in Section 2.3, the rigidity matrix consists mostly of zeros. For large problems this property should be exploited by using sparse matrix techniques. The only real modification to the rigidity algorithm required to verify redundant rigidity was a sequence of triangular solves. These can be done sparsely, so the entire algorithm can be implemented in a sparse setting. An algorithm very similar to this has been described by Coleman and Pothen [CP87].

5 Conclusion

Three necessary conditions for a graph to have a unique embedding in d dimensions have been derived. They are, in order of appearance,

1. rigidity,
2. $d+1$-connectivity, and
3. redundant rigidity.

The first condition is a trivial consequence of the third so there are really only two independent criteria. However, flexibility leads to a very different kind of nonuniqueness than lack of redundant rigidity so it is useful to treat them independently. Algorithms for testing each of these three conditions have been presented that deal solely with the underlying graph, ignoring the edge lengths. The price for this convenience is that only probabilistic guarantees of success are possible. But the counter-examples are very rare. They constitute a set of measure zero over the space of all possible vertex positionings. Once the actual vertex coordinates are determined it is possible to double check that the graph in question actually satisfies the three criteria. It is a straight-forward matter to determine the rank of the rigidity matrix and to look for separating sets of vertices that lie in a $d-1$ dimensional hyperplane. In this way the three conditions can be verified deterministically.

Efficient algorithms for testing each of these three conditions have been presented. For dimensions greater than two, the algorithms for rigidity and redundant rigidity are probabilistic. Each of these three requirements can also be verified rapidly in parallel. There are random NC algorithms for each property.

Establishing necessary conditions for a graph to have a unique embedding makes it possible to prune the initial graph before attempting the difficult task of finding coordinates for the vertices. If the entire graph does not have a unique embedding then it would be impossible to assign unambiguous coordinates to the vertices. Instead portions of the graph that do satisfy the three criteria can be identified and positioned. Not only does this alleviate the confusion of a poorly posed problem, but since the cost of finding the embedding can grow exponentially with the size of the graph, it should be possible to save time by embedding a sequence of smaller subgraphs instead of the original full one.

Following this idea to its logical conclusion, even if the original graph has a unique embedding it might be possible to position chunks of it at a time and then piece them together. Since the cost of the embedding grows rapidly with the problem size this could lead to a substantial reduction of computational effort. To accomplish this it would be necessary to find induced subgraphs that have unique embeddings. One would need to characterize, perhaps probabilistically, sufficiency conditions for a graph to have a unique positioning. This paper has developed some necessary conditions, and it is tempting to speculate that they are also sufficient.

Acknowledgements

The ideas in this paper have been developed and refined in innumerable discussions with Tom Coleman. A large debt is also due to Bob Connelly for his patient attempts to teach me some of the finer points of rigidity theory. Thanks also go to Jim Renegar for help with differential topology.

References

[AHU74] Alfred V. Aho, John E. Hopcroft, and Jeffrey D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley, 1974.

[B+82] M. Becker et al. A probabilistic algorithm for vertex connectivity of graphs. *Information Processing Letters*, 15(3):135–136, 1982.

[Bis88] Chrisian Bischof. *QR Factorization Algorithms for Course-Grained Distributed Systems.* PhD thesis, Cornell University, Department of Computer Science, 1988.

[BR80] E. D. Bolker and B. Roth. When is a bipartite graph a rigid framework? *Pacific Journal of Mathematics*, 90(1):27–44, 1980.

[Cau13] A. L. Cauchy. Deuxième memoire sur les polygones et les polyèdres. *Journal de l'Ecole Polytechnique*, 19:87–98, 1813.

[CEG86] Thomas F. Coleman, Anders Edenbrandt, and John R. Gilbert. Predicting fill for sparse orthogonal factorization. *Journal of the ACM*, 33(3):517–532, July 1986.

[CP87] Thomas F. Coleman and Alex Pothen. The null space problem II. Algorithms. *SIAM Journal on Algebraic and Discrete Methods*, 8(4):544–563, October 1987.

[Fle81] R. Fletcher. *Practical Methods of Optimization: Volume 2, Constrained Optimization.* John Wiley and Sons, 1981.

[Gal83] Z. Galil. Efficient algorithms for finding maximal matchings in graphs. preprint, 1983.

[GH80] A. George and M. T. Heath. Solution of sparse linear least squares problems using Givens rotations. *Linear Alg. Appl.*, 34:69–83, 1980.

[Glu75] Herman Gluck. Almost all simply connected closed surfaces are rigid. In *Geometric Topology*, pages 225–239. Lecture Notes in Mathematics No. 438, Springer-Verlag, Berlin, 1975.

[GM84] Michel Gondran and Michel Minoux. *Graphs and Algorithms.* John Wiley & Sons, 1984.

[GP74] Victor Guillemin and Alan Pollack. *Differential Topology.* Prentice-Hall, Inc., 1974.

[GVL83] Gene H. Golub and Charles F. Van Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, MD, 1983.

[HK73] J. E. Hopcroft and R. M. Karp. An $n^{5/2}$ algorithm for maximum matchings in bipartite graphs. *SIAM Journal on Computing*, 2:225–231, 1973.

[HT73] J. E. Hopcroft and R. E. Tarjan. Dividing a graph into triconnected components. *SIAM Journal on Computing*, 2(3):135–158, 1973.

[Ima85] Hiroshi Imai. On combinatorial structures of line drawings of polyhedra. *Discrete Applied Mathematics*, 10:79–92, 1985.

[IMR80] O. H. Ibarra, S. Moran, and L. E. Rosen. A note on the parallel complexity of computing the rank of order n matrices. *Information Processing Letters*, 11(4,5):162, 1980.

[KR] Arkady Kanevsky and Vijaya Ramachandran. Improved algorithms for graph four-connectivity. Submitted to JCSS Special Issue on FOCS 87.

[KUW86] R. M. Karp, E. Upfal, and A. Wigderson. Constructing a maximum matching is in random NC. *Combinatorica*, 6(1):35–48, 1986.

[Lam70] G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.

[LLW86] N. Linial, L. Lovasz, and A. Wigderson. A physical interpretation of graph connectivity, and its algorithmic applications. In *Proceedings of the 27th IEEE Annual Symposium on Foundations of Computer Science*, 1986.

[MR87] G. L. Miller and Vijaya Ramachandran. A new graph triconnectivity algorithm and its parallelization. In *Proceedings of the Nineteenth ACM Annual Symposium on Theory of Computing*, pages 335–344, New York, NY, 1987.

[Rot81] B. Roth. Rigid and flexible frameworks. *American Mathematical Monthly*, 88:6–21, 1981.

[Sax79] James B. Saxe. Embeddability of weighted graphs in k-space is strongly NP-hard. Technical report, Computer Science Department, Carnegie-Mellon University, 1979.

[Sug80] Kokichi Sugihara. On redundant bracing in plane skeletal structures. *Bulletin of the Electrotechnical Laboratory*, 44:376–386, 1980.

[TW85] Tiong-Seng Tay and Walter Whiteley. Generating isostatic frameworks. *Topologie Structurale*, 11:21–69, 1985.

[Whi84] Walter Whiteley. Infinitesimal motions of a bipartite framework. *Pacific Journal of Mathematics*, 110(1):233–255, 1984.