CONDITIONS FOR UNIQUE GRAPH REALIZATIONS*

BRUCE HENDRICKSON†

Abstract. The *graph realization 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 realization. Both these problems are NP-hard, but the proofs rely upon special combinations of edge lengths. If we assume the vertex locations are unrelated then the uniqueness question can be approached from a purely graph theoretic angle that ignores edge lengths. This paper identifies three necessary graph theoretic conditions for a graph to have a unique realization in any dimension. Efficient sequential and NC algorithms are presented for each condition, although these algorithms have very different flavors in different dimensions.

1. Introduction. Consider a graph $G = (V, E)$ consisting of a set of n vertices and m edges, along with a real number associated with each edge. 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 realization 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 [27, 39]. 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 [13]. Solving the graph realization problem in this context would allow one to determine the three-dimensional shape of the molecule, which is important in understanding the molecule's properties.

Unfortunately, the graph realization problem is known to be difficult. Saxe has shown it to be strongly NP-complete in one dimension and strongly NP-hard for higher dimensions [35] . 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.

This paper will address a closely related problem: when does the graph realization problem have a unique solution? (For our purposes translations, rotations and reflections of the entire space are not considered to be different realizations.) Clearly if the location of a satellite or an atom is to be determined unambiguously the solution to the realization problem must be unique.

Saxe has shown this uniqueness problem to be as hard as the original realization problem, but again his proofs rely on very special graphs. In particular, he needs

* SIAM J. Comput., 21(1):65–84, February 1992.

† Mathematics and Computational Science Department, Sandia National Laboratories, Albuquerque, NM 87185. Research performed while the author was at Cornell University, supported by a fellowship from the Fannie and John Hertz Foundation

special combinations of edge lengths, implying specific algebraic relations among the coordinates of the vertices. This paper will address the more typical behavior of graphs.

A *realization* of a graph G is a function p that maps the vertices of G to points in Euclidean space. The combination of a graph and a realization is called a *framework* and is denoted by $p(G)$. A realization is *satisfying* if all the pairwise distance constraints are satisfied. Consider a set $\mathcal{S}$ with nonzero measure. A subset $\mathcal{T}$ of $\mathcal{S}$ is said to contain *almost all* of $\mathcal{S}$ if the complement of $\mathcal{T}$, $\{q \in \mathcal{S} | q \notin \mathcal{T}\}$, has measure zero. A realization is said to be *generic* if the vertex coordinates are algebraically independent over the rationals. This computationally unrealistic requirement is actually stronger than we truly need. We just have to avoid several specific algebraic dependencies. However, the set of generic realizations is dense in the space of all realizations, and almost all realizations are generic.

Restricting ourselves to generic realizations will greatly simplify our analysis. It will allow us to ignore the edge distances and base our analysis solely on the underlying graph. The results we develop will apply to graphs in almost all realizations. However, non-generic realizations might have different properties.

How can a framework have multiple realizations? There are several distinct manners in which nonuniqueness can appear. First, the framework can be susceptible to continuous deformations, like the one in Fig. 1. The rightmost vertex in this graph

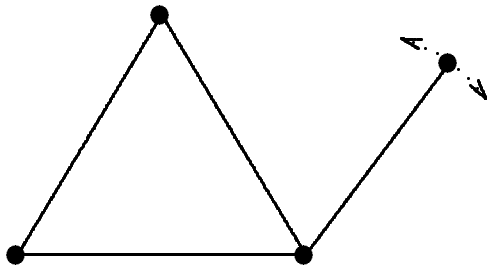


FIG. 1. *A flexible framework in two dimensions.*

can pivot freely since it is underconstrained. A framework that can be continuously deformed while still satisfying all the constraints is said to be *flexible*; otherwise it is *rigid*. Even a rigid framework can suffer from nonuniqueness. The rigid framework in Fig. 2 has two realizations in the plane. One half of the graph can reflect across the

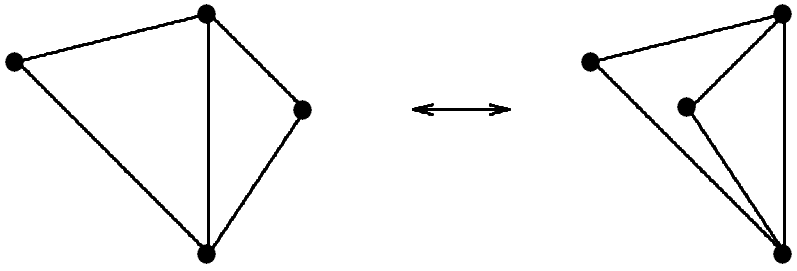


FIG. 2. *A graph with two realizations in the plane.*

central two vertices. Continuous deformations and graph rigidity will be discussed in

§ 2. Although graph rigidity is a well studied problem, the connections to the graph realization problem have not been well explored. Discontinuous transformations like reflections will be covered in §§ 3 and 4. The definition of redundant rigidity in § 4 and its importance in this context is entirely new.

The graph realization 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 § 2. These will lead to completely different algorithms for these different dimensions.

2. Graph Rigidity. A graph that has a unique realization cannot be susceptible to continuous flexings. It must be rigid. Questions about the rigidity of graphs have occupied mathematicians for centuries. 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 framework 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.

Section 2.1 will develop the essentials of rigidity theory, stressing the importance of the rigidity matrix. A more complete discussion can be found in some of the references [2, 3, 33, 11]. § 2.2 will present sequential and parallel algorithms for rigidity testing.

2.1. Basic Concepts. A mathematical analysis of rigidity requires a formal definition of our intuitive notion of a flexible framework. Everything in this section occurs in an arbitrary Euclidean dimension d.

A *finite flexing* of a framework $p(G)$ is a family of realizations of G, parameterized by t so that the location of each vertex i is a differentiable function of t and $(p_i(t)-p_j(t))^2 = constant$ for every $(i,j) \in E$. Thinking of t as time, and differentiating the edge length constraints we find that

$$(v_i - v_j) \cdot (p_i - p_j) = 0 \quad \text{for every } (i,j) \in E, \tag{1}$$

where v_i is the instantaneous velocity of vertex i. An assignment of velocities that satisfies Equation 1 for a particular framework is an *infinitesimal motion* of that framework. Clearly the existence of a finite flexing implies an infinitesimal motion, but the converse is not always true. However, for generic realizations infinitesimal motions always correspond to finite flexings [33].

The infinitesimal motions of a framework constitute a vector space. Note that a motion of the Euclidean space itself, a rotation or translation, satisfies the definition of a finite flexing. Such finite flexings are said to be *trivial*. In d-space there are d independent translations and $d(d-1)/2$ rotations. If a framework has a nontrivial infinitesimal motion it is *infinitesimally flexible*. Otherwise it is *infinitesimally rigid*. As noted above, for generic realizations infinitesimal motions correspond to finite flexings.

Since we are considering only generic realizations we will drop the prefix and refer to frameworks as either rigid or flexible.

We would like to be able to determine whether a particular framework is rigid or flexible. Conveniently, this is substantially a property of the underlying graph as the following theorem indicates [18].

THEOREM 2.1 (GLUCK). *If a graph has a single infinitesimally rigid realization, then all its generic realizations are rigid.*

This theorem is critical for a graph theoretic approach to the realization problem. The frameworks built from a graph are either all infinitesimally flexible or almost all rigid. This allows for the characterization of graphs as either rigid or flexible according to the typical behavior of a framework, without reference to a specific realization. It also allows us to be somewhat cavalier in the distinction between rigid frameworks and graphs that have rigid realizations. Henceforth such graphs will be referred to as *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 n vertices has nd possible independent motions. However a d-dimensional rigid body in d-space has d translations and $d(d-1)/2$ rotations. (If the body has dimension $d' < d$ then it has only $d'(2d - d' - 1)/2$ rotations. This corresponds to a framework with only $d'+1$ vertices.) The total number of allowed motions is the number of total degrees of freedom, nd, minus the number of rigid body motions. For convenience we will call this quantity $S(n,d)$, where

$$S(n,d) = \begin{cases} nd - d(d+1)/2 & \text{if } n \geq d \\ n(n-1)/2 & \text{otherwise.} \end{cases}$$

If each edge adds an independent constraint then $S(n,d)$ edges should be required to eliminate all nonrigid motions of the graph. This intuition is sound as the theorems in this section will demonstrate.

Any realization of a flexible graph has a nontrivial infinitesimal motion. An infinitesimal motion is a solution for velocities in Equation 1. The matrix of this set of equations is the *rigidity matrix.* It has m rows and nd columns. Each row corresponds to an edge while each column corresponds to a coordinate of a vertex. Each row has $2d$ nonzero elements, one for each coordinate of the vertices connected by the corresponding edge. The non-zero values are the differences in the coordinate values for the two vertices. For example, consider the graph K_3, the complete graph on three vertices, positioned in $\mathbb{R}^2$. If the realization maps the vertices to locations $(0,1)$, $(-1,0)$ and $(1,0)$, the rigidity matrix would be:

$$\begin{array}{c} \\ e_{1,2} \\ e_{1,3} \\ e_{2,3} \end{array} \begin{array}{c} \begin{array}{cccccc} v_1^x & v_1^y & v_2^x & v_2^y & v_3^x & v_3^y \end{array} \\ \begin{pmatrix} 1 & 1 & -1 & -1 & 0 & 0 \\ -1 & 1 & 0 & 0 & 1 & -1 \\ 0 & 0 & -2 & 0 & 2 & 0 \end{pmatrix} \end{array}$$

The rank of the rigidity matrix is closely related to the rigidity behavior of the framework, as this section will elucidate.

THEOREM 2.2. *A framework $p(G)$ is rigid if and only if its rigidity matrix has rank exactly equal to $S(n,d)$.*

Proof. All infinitesimal motions must be in the null space of M since the rigidity matrix expresses all constraints on the infinitesimal velocities. By construction, $S(n,d)$ is the size of the rigidity matrix minus the number of trivial infinitesimal motions. If the null space of M contains any nontrivial infinitesimal motions then the rank must be less than $S(n,d)$. □

So the question of whether a framework is flexible can be reduced to a question about the rank of the rigidity matrix. The framework is rigid if and only if the rank of the rigidity matrix is maximal, $S(n,d)$.

THEOREM 2.3. *Every rigid framework $p(G)$ has a rigid subframework with exactly $S(n,d)$ edges.*

Proof. The rigidity matrix has rank $S(n,d)$ and each of its rows corresponds to an edge. Simply discard redundant rows and the corresponding edges until only S remain. □

COROLLARY 2.4. *For a framework $p(G)$, if $m > S(n,d)$ then there is linear dependence among the rows of the rigidity matrix.*

Proof. The maximum rank of the rigidity matrix is $S(n,d)$. □

Dependence among rows in the rigidity matrix can be expressed in terms of a matroid [30, 15]. For our purposes it will be sufficient to say that a set of edges is *independent* if their rows in the rigidity matrix are linearly independent in a generic realization. A rigid graph has $S(n,d)$ independent edges.

THEOREM 2.5. *If a framework $p(G)$ with exactly $S(n,d)$ edges is rigid, then there is no subgraph $G' = (V', E')$ with more than $S(n',d)$ edges, where $n' = |V'|$.*

Proof. Since there are only $S(n,d)$ edges, their rows in the rigidity matrix must all be independent by Theorem 2.3. But if G' has $|E'| > S(n',d)$, then by Corollary 2.4 there must be linear dependence among these edges which is a contradiction. □

Theorems 2.3 and 2.5 say that a rigid graph with n vertices must have a set of $S(n,d)$ *well distributed* edges, where well distributed means that no subgraph with n' vertices has more than $S(n',d)$ edges. This requirement is often referred to as *Laman's condition* after G. Laman [28] who first articulated the two-dimensional version. This condition is necessary for a graph to be rigid in any dimension. It is sufficient in one dimension where $S = n - 1$. It is straightforward to show that this is equivalent to requiring the graph to be connected. Laman was able to show that it is also sufficient in two dimensions where $S = 2n - 3$.

THEOREM 2.6 (LAMAN). *The edges of a graph $G = (V, E)$ are independent in two dimensions if and only if no subgraph $G' = (V', E')$ has more than $2n' - 3$ edges.*

COROLLARY 2.7. *A graph with $2n - 3$ edges is rigid in two dimensions if and only if no subgraph G' has more than $2n' - 3$ edges.*

This was the first graph theoretic characterization of rigid graphs in two-space. Several equivalent characterizations have since been discovered [36, 24, 30, 37, 12].

Unfortunately, for all its intuitive appeal Laman's condition is not sufficient in higher dimensions. A three-dimensional counterexample is depicted in Fig. 3. Although this graph has the required 18 well distributed edges it is still flexible. The top and bottom halves can pivot about the left and right-most vertices.

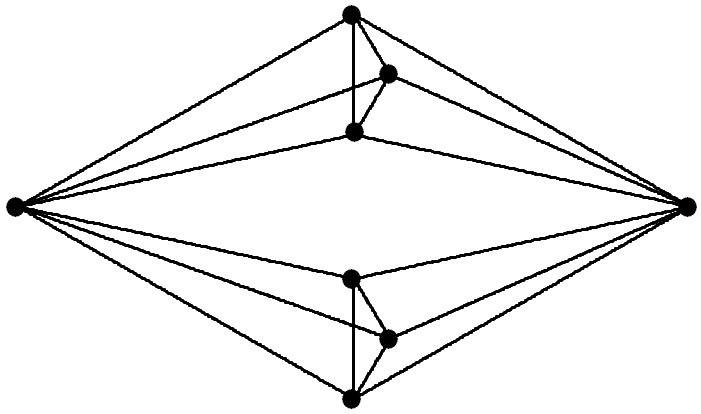


FIG. 3. *A flexible graph in three-space that satisfies Laman's condition.*

The problem with Fig. 3 is that its edges are not independent in the sense of Theorem 2.2. The rows of the rigidity matrix are linearly dependent. Expressing this independence graph theoretically has proved to be a very difficult problem. No general characterization of rigid graphs in three dimensions is known, although the problem has been considered by many researchers, and several special cases have been solved. Cauchy proved that triangulated planar graphs (those with all $3n - 6$ edges) are generically rigid in three-space [6]. Fogelsanger recently generalized this result to include complete triangulations of any two-manifold in three-space [14]. Another class known to be rigid is complete bipartite graphs with at least five vertices in each vertex set [5, 38]. However, the characterization of general graphs remains open.

Recent work by Tay, Whiteley and Graver [37] has brought such a characterization almost within reach. However, it is difficult to see how this possible solution could lead to an efficient algorithm. Any straightforward implementation of their approach would have a worst case exponential time behavior.

2.2. 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 [1].

2.2.1. Rigidity Algorithms in Two Dimensions. In two dimensions Laman's condition characterizes rigidity, but in its original form it gives a poor algorithm. It involves counting the edges in every subgraph, of which there are an exponential number. Sugihara discovered the first polynomial time algorithm for determining the independence of a set of edges in two dimensions [36]. Imai presented an $O(n^2)$ algorithm for rigidity testing using a network flow approach [24]. This time complexity was matched by Gabow and Westermann using matroid sums [15]. In this section we will develop a

new $O(n^2)$ algorithm based on bipartite matching. Besides any intrinsic interest, this new algorithm will be needed in § 4 when we need to test for a stronger graph condition.

We will first need to introduce a particular bipartite graph, $B(G)$, generated by our original graph $G = (V, E)$. The bipartite graph has the edges of G as one of its vertex sets, and two copies of the vertices of G for the other. Edges of $B(G)$ connect the edges of G with the two copies of their incident vertices. More formally, $B(G) = (V_1, V_2, \mathcal{E})$, where $V_1 = E$, $V_2 = \{q_1^1, q_1^2, \ldots, q_n^1, q_n^2\}$, and $\mathcal{E} = \{(e, q_i^1), (e, q_i^2), (e, q_j^1), (e, q_j^2) \ : \ e = (v_i, v_j) \in E\}$. $B(G)$ has $2n + m$ vertices and $4m$ edges, where n and m are respectively the number of vertices and edges in G. A simple example of the correspondence between G and $B(G)$ is presented in Fig. 4 for the graph K_3.

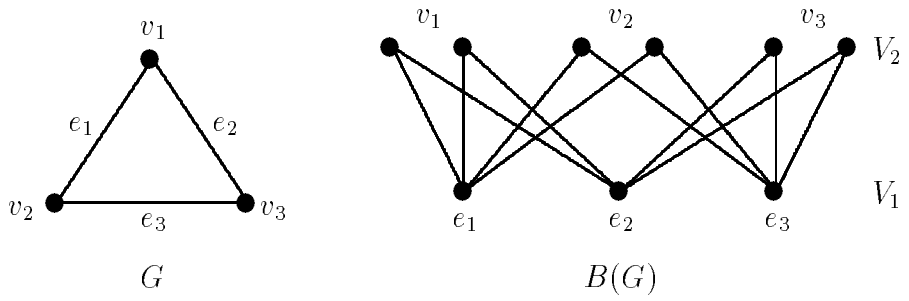


FIG. 4. *The correspondence between G and $B(G)$.*

This bipartite graph leads to an alternate form of Laman's condition, expressed in the following theorem. As above, a set of edges is said to be *independent* if the corresponding rows in the rigidity matrix are linearly independent in a generic realization.

THEOREM 2.8. *For a graph $G = (V, E)$ the following are equivalent:*

A. The edges of G are independent in two dimensions;

B. for each edge (a, b) in G, the graph $G_{a,b}$ formed by adding three additional edges between a and b has no subgraph G' in which $m' > 2n'$;

C. for each edge (a, b), the bipartite graph $B(G_{a,b})$ generated by $G_{a,b}$ has no subset of V_1 that is adjacent only to a smaller subset of V_2.

D. for each edge (a, b), the bipartite graph $B(G_{a,b})$ generated by $G_{a,b}$ has a complete matching from V_1 to V_2.

Proof. The equivalence of A and B is a restatement of Laman's condition. The equivalence of B and C is a straightforward consequence of the construction of $B(G_{a,b})$. Property D is equivalent to C by Hall's theorem from matching theory. Assertions C and D were first discovered in a slightly different form by Sugihara [36]. □

Our algorithm will be based upon the characterization in Theorem 2.8D. The basic idea is to grow a maximal set of independent edges one at a time. Denote these *basis* edges by $\hat{E}$. A new edge is added to the basis if it is discovered to be independent of the existing set. If $2n-3$ independent edges are found then the graph is rigid. Determining whether a new edge is independent of the current basis can be done quickly using the bipartite matching characterization.

Assume we have a (possibly empty) set of independent edges $\hat{E}$. Combined with

the vertices of G these form a graph $\hat{G}$, which generates a bipartite graph $B(\hat{G})$. Note that $|\hat{E}| = O(n)$ and so $B(\hat{G})$ will have $O(n)$ edges. We wish to determine if another edge, e, is independent of $\hat{E}$. Adding e to $\hat{G}$ produces $\bar{G}$ and $B(\bar{G})$. By characterization D, e is independent of $\hat{E}$ if and only if there is a complete bipartite matching in B after any edge in $\bar{G}$ is quadrupled. Actually, only e needs to be quadrupled as the following Lemma demonstrates.

LEMMA 2.9. *If a complete matching exists when e is quadrupled then e is independent of $\hat{E}$.*

Proof. Assume the matching succeeds but e isn't independent of $\hat{E}$. Then there must exist some edge in $\hat{E}$ whose quadrupling causes G', a subgraph of $\bar{G}$, to have $m' > 2n' - 3$. Since the edges of $\hat{E}$ are independent this bad subgraph must include e. But this bad subgraph has the same number of edges it had when e was quadrupled. Since the matching succeeded when e was quadrupled we have a contradiction. □

Determining whether a new edge can be added to the set of independent edges is now reduced to the problem of trying to enlarge a bipartite matching. This is a standard problem in matching and it is performed by growing Hungarian trees and looking for 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. Growing a Hungarian tree takes time proportional to the number of edges in the bipartite graph.

LEMMA 2.10. *If $\hat{E}$ is independent and a corresponding matching in $B(\hat{G})$ is known, then determining whether a new edge is independent requires $O(n)$ time.*

Proof. By Lemma 2.9, testing for independence of e requires just enlarging the matching in $B(\hat{G})$ to include the four copies of e. This involves growing four Hungarian trees in a bipartite graph of size $O(n)$. □

This gives a two-dimensional rigidity testing algorithm that runs in time $O(nm)$. Build a maximal set of independent edges one at a time by testing each edge for independence. Each test involves the enlargement of a bipartite matching requiring $O(n)$ time. If the matching succeeds the edge is independent and is added to the basis. Otherwise it is discarded.

To improve this to $O(n^2)$ we need to make use of failed matchings to eliminate some edges from consideration. Define a *Laman subgraph* as a subgraph with n' vertices and $2n' - 3$ independent edges. A matching will fail precisely when the new edge lies in a subgraph which already has $2n' - 3$ independent edges. No edge can be added between vertices in this subgraph, so it is a waste of time even to try. By avoiding these unnecessary attempts we can improve the performance of our algorithm. To accomplish this we will need some further insight into the bipartite matching.

THEOREM 2.11. *In a bipartite graph $(V_1, V_2, \mathcal{E})$, if a Hungarian tree fails to find an alternating path then it spans a minimal subgraph which violates Hall's theorem. That is, it identifies a minimal set of k vertices in V_1 with fewer than k neighbors.*

Proof. This is a simple consequence of Hall's theorem. □

LEMMA 2.12. *If the new edge, e, is tripled instead of quadrupled, generating a graph $\underline{G}$ from $\hat{G}$, then $B(\underline{G})$ has a complete matching.*

Proof. Assume the contrary. Then there is some subgraph G' of $\bar{G}$ with $m' > 2n'$. Remove the three copies of e from this subgraph and quadruple one of the other edges. This altered subgraph still has $m' > 2n'$, but it is the graph generated by quadrupling an edge in $\hat{G}$. But since the edges of $\hat{G}$ are assumed to be independent this is a contradiction. □

LEMMA 2.13. *If edge e fails the matching test, then the failing Hungarian tree spans a set of edges of $\hat{E}$ that form a Laman subgraph.*

Proof. By Lemma 2.12 when e is quadrupled the first 3 copies of it can be matched. By Theorem 2.11 when the fourth fails it spans a set of vertices of V_1 adjacent to a smaller set from V_2. Discarding the four copies of e leaves a set of k elements of V_1 adjacent to no more then $k+3$ vertices from V_2. By the construction of the bipartite graph this is a set $\hat{E}'$ of k edges of $\hat{E}$ incident upon no more than $(k+3)/2$ vertices. That is, $m' \geq 2n' - 3$. Since the edges of $\hat{E}$ are independent we must have equality. □

We will need the following result to analyze the running time of our algorithm.

LEMMA 2.14. *Let $G = (V, \hat{E})$ be a graph whose edges are independent. If two Laman subgraphs of G share an edge then their union is a Laman subgraph.*

Proof. Let the subgraphs be (V', E') and (V'', E'') with union $(\bar{V}, \bar{E})$. Let $\bar{m} = m' + m'' - l$ and $\bar{n} = n' + n'' - k$. Since the subgraphs share at least one independent edge, $l \leq 2k - 3$. Hence,

$$\begin{aligned} \bar{m} &= 2n' + 2n'' - 6 - l \\ &\geq 2n' + 2n'' - 2k - 3 \\ &= 2(n' + n'' - k) - 3 \\ &= 2\bar{n} - 3. \end{aligned}$$

Since the edges are independent we must have equality. □

We are now ready to present our algorithm. We will maintain the appropriate bipartite graph with a matching of all the independent edges discovered so far. We will also keep a collection of all the Laman subgraphs that have been identified, represented as linked lists of independent edges. The algorithm is outlined in Fig. 5.

By Lemma 2.14 we know that no edge need be in more than one subgraph. By merging whenever a new Laman subgraph is found, we guarantee that the total number of elements in all the subgraphs is kept to $O(n)$. This ensures that the marking and merging operations can be done in $O(n)$ time. As above, checking for independence of (u, v) requires $O(n)$ time. Each time an edge is checked it results in either a new basis edge or a merging of components, so this can only happen $O(n)$ times. Hence the total time for the algorithm is $O(n^2)$.

```
basis ← ∅
For Each vertex v
    Mark each vertex in a Laman subgraph with v, and unmark all others
    For Each edge (u, v)
        If u is unmarked Then
            If (u, v) is independent of basis Then
                add (u, v) to basis
                create Laman subgraph consisting of (u, v)
            Else a new Laman subgraph has been identified
                Merge all Laman subgraphs that share an edge
                Mark each vertex in a Laman subgraph with v
```

FIG. 5. *An $O(n^2)$ algorithm for two-dimensional graph rigidity.*

2.2.2. Rigidity Algorithms in Higher Dimensions. For dimensions greater than two there are no graph theoretic characterizations of rigidity, so there are no good combinatoric algorithms to test for it. One approach would involve a symbolic calculation of the rank of the rigidity matrix by symbolically constructing the determinant. However, the determinant can have an exponential number of terms, so this requires an exponential amount of time. A different approach is possible which relies instead upon Theorem 2.1. Since this theorem is valid in all dimensions, the following discussion is applicable to all spaces. If the graph is rigid then almost any realization will generate a rigid framework. Simply select a random realization for the graph. Once these vertex locations are selected it is a straightforward matter to determine the rigidity of the framework using Theorem 2.2. Just construct the rigidity matrix M and determine its rank. If the rank is $S(n, d)$ then the graph is rigid. A lower rank indicates that the framework is flexible. Unless the selection of vertex coordinates was extremely unlucky the underlying graph will be flexible as well. So even without a graph theoretic characterization an efficient practical randomized algorithm for rigidity exists.

To determine the rank of M we suggest using a QR decomposition with column pivoting, requiring $O(mn^2)$ time [19]. This is more numerically stable than Gaussian elimination, but not as costly as a singular value decomposition. A QR factorization has several advantages over an SVD in this application. Performing a QR on M^T will identify a maximal independent set of rows of M one at a time, corresponding to a maximal set of independent edges in the graph. This ability to identify independent rows will be needed in § 4. Also, the rigidity matrix is quite sparse, having only $2d$ nonzeros in each row. To save time and space, sparse techniques could be used for large problems. There are sparse QR algorithms, but none for SVD [8, 16, 17].

There are also efficient parallel algorithms for finding the rank of a matrix. Ibarra, Moran and Rosier [23] discovered an algorithm that runs in $O(\log^2 m)$ time on $O(m^4)$ processors. This means that rigidity testing is in random NC for any dimension. 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 a *good* parallel algorithm, although its applicability is more theoretical than practical.

3. Partial Reflections. Even rigid graphs can have multiple realizations as was shown in Fig. 2. This discontinuous flavor of nonuniqueness has not been well studied, probably because it is not relevant to structural engineers. Buildings can only deform continuously. For the graph realization problem these discontinuous transformations must be considered. This section and the next will be concerned with multiple realizations that do not arise from flexibility. These are cases in which there are two or more noncongruent realizations that satisfy all the distance requirements, but there is no continuous flexing of the framework to transform one to another while maintaining the constraints. Whereas flexible graphs have an infinite number of potential configurations, the number of realizations of a rigid graph is finite, although possibly exponential in the size of the graph.

A two-dimensional example of the simplest type of discontinuous transformation is depicted in Fig. 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 Fig. 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. We will say that a framework in d-space allows a *partial reflection* if a separating set of vertices lies in a $(d-1)$-dimensional subspace.

The realizations in which more than d vertices lie in a $(d-1)$-dimensional subspace are not generic. So for almost all frameworks, partial reflections only occur when there is a subset of d or fewer vertices whose removal separates the graph into two or more unconnected pieces, that is, when G is not vertex $(d+1)$-connected. This gives us the following well known result.

THEOREM 3.1. *A rigid graph positioned generically in dimension d will have a partial reflection if and only if it is not vertex $(d+1)$-connected.*

The connectivity of a graph is an important property, and it has been well studied. There are well known $O(m)$ time algorithms for vertex two-connectivity, also known as *biconnectivity* [1]. Avoiding partial reflections in two dimensions requires a vertex three-connected (or *triconnected*) graph. Hopcroft and Tarjan [22] were the first to discover an $O(m)$ time algorithm to find triconnected components. Miller and Ramachandran [31] have recently proposed a parallel algorithm to identify triconnected components in $O(\log^2 n)$ time with $O(m)$ processors. This places triconnectivity in NC.

Four-connected components are more difficult, but Kanevsky and Ramachandran [25] have recently found an $O(n^2)$ time algorithm. They also discovered a parallel implementation of their algorithm that runs in $O(\log n)$ time using $O(n^2)$ processors. So the problem of partial reflections is in NC in both two and three dimensions.

For k greater than 4, the question of k-connectivity for a general k is less well understood. Consequently the partial reflection problem is more difficult in spaces of dimension greater than three. There are randomized algorithms for general k-connectivity

that run in time proportional to $n^{5/2}$ [4, 29]. Recently, Cheriyan and Thurimella have described an algorithm with a time complexity of $O(k^3n^2)$, which reduces to $O(n^2)$ for a fixed k [7]. There are also NC algorithms that run in time $O(k^2 \log n)$ [26].

4. Redundant Rigidity. Rigidity and $(d+1)$-connectivity are necessary but not sufficient to ensure that a graph has a unique realization. A two-dimensional example of a rigid, triconnected graph with two satisfying realizations is given in Fig. 6.

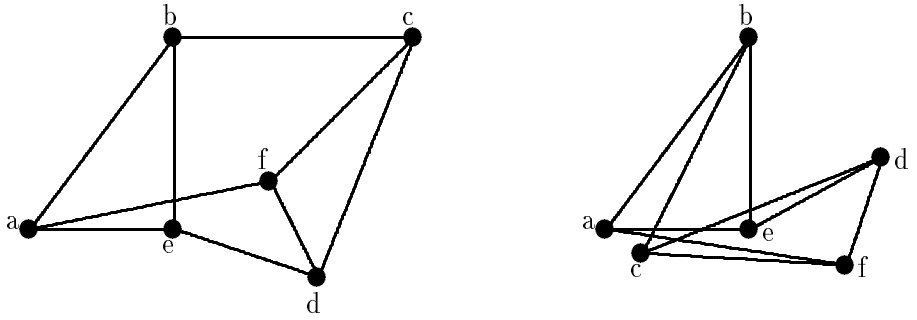


FIG. 6. *Two realizations of a rigid triconnected graph in the plane.*

To understand this nonuniqueness, consider Fig. 7. Edge (a, f) has been removed

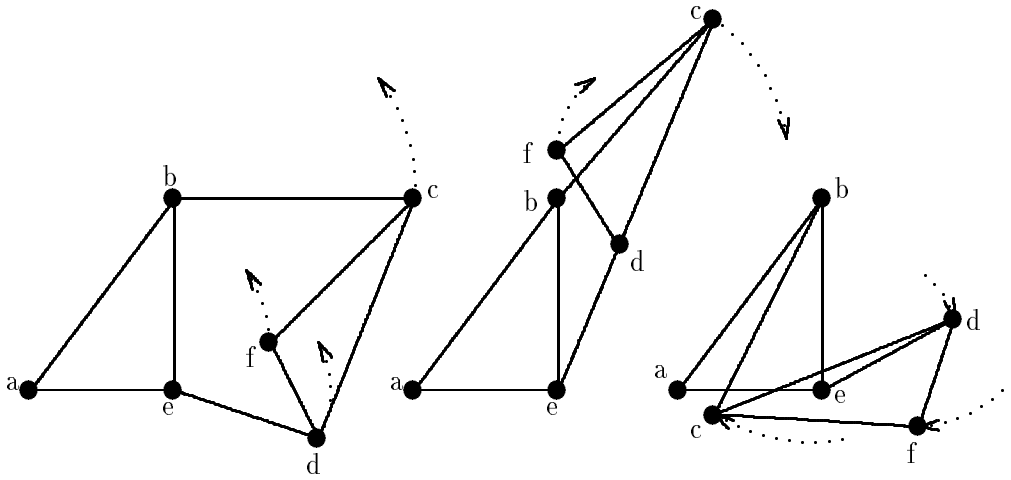


FIG. 7. *Intermediate stages in the construction of Fig. 6.*

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

The fundamental problem with the graph in Fig. 6 is that the removal of a single edge makes it flexible. We will define an edge of a framework to be *infinitesimally redundant* if the framework remaining after its removal is infinitesimally rigid. A framework is *infinitesimally redundantly rigid* if all its edges are infinitesimally redundant. Redundant bracing is a familiar concept to engineers who wish to build frameworks with additional strength and failure tolerance properties, but this precise formulation and its significance with regards to the unique realization problem are entirely new.

Infinitesimal redundant rigidity is clearly a more restrictive property than infinitesimal rigidity, but the two properties have many similarities. For generic realizations infinitesimal motions always correspond to finite flexings. So as with rigidity, since we are only interested in generic realizations we can drop the prefix and refer to frameworks as redundantly rigid. The following theorem is a trivial consequence of Theorem 2.1.

THEOREM 4.1. *If a graph has a single infinitesimally redundantly rigid realization, then all its generic realizations are redundantly rigid.*

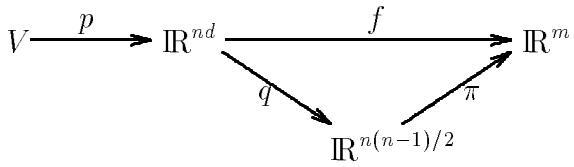
As with Theorem 2.1 for rigid graphs, Theorem 4.1 says that either none of a graph's realizations are redundantly rigid, or almost all of them are. *Almost all* again means that the set of counter examples has measure zero. This blurs the distinction between a redundantly rigid framework and its underlying graph. Graphs with redundantly rigid realizations will be referred to as *redundantly rigid graphs.*

In Fig. 6 the lack of redundant rigidity led to multiple realizations. This is usually true, and the proof will be the main result of this section. Intuitively, a flexible framework can move around, but it must always end up back where it started. That is, the path it traces in nd-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. However, there are some graphs for which this argument fails. Consider the triangle graph, K_3. It has only one realization, but if an edge is removed it becomes flexible. To understand which graphs need to be redundantly rigid to have unique realizations we will need to carefully investigate the space of satisfying realizations for flexible graphs. This will require an incursion into differential topology, and is the subject of the next section.

5. The Necessity of Redundant Rigidity. The proof that flexible graphs typically move in closed loops will rely upon some special properties of the graph realization problem. Given a framework $p(G)$ there is a *pairwise distance function* $q : \mathbb{R}^{nd} \to \mathbb{R}^{n(n-1)/2}$ that maps vertex locations to squares of all the pairwise vertex distances. That is,

$$q(p(v_1), \ldots, p(v_n)) = (\ldots, |p(v_i) - p(v_j)|^2, \ldots).$$

For the molecule problem we are only interested in a specific set of pairwise distances; namely, those corresponding to the edges of G. These can be obtained from $R^{n(n-1)/2}$ by a simple projection, π. We will define an *edge function*, f, to be the composition of these two operations, $f = \pi \circ q$. These functions are described by the following commutative diagram.



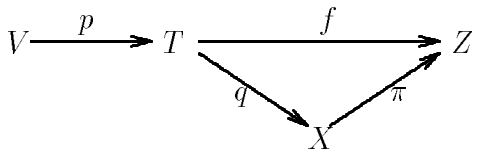
The functions f and q have many nice properties. We will say a function is *smooth* at a point x if it has continuous partial derivatives of all orders at x. The functions f and q are everywhere smooth. Also, the Jacobian of f is twice the rigidity matrix introduced in § 2.1.

The realization problem is really that of finding the inverse of the edge function. Of course, this inverse is multivalued because edge lengths are invariant under translations, rotations and reflections of the entire space. Two realizations will be considered *equivalent* if all pairwise distances between vertices are the same under the two realizations. That is, two realizations are equivalent if they map to the same point under q. We will be interested only in the inverse of f modulo equivalences. More formally, define the *realization set* of $p(G)$ to be $\pi^{-1}f(p(G))$, the set of nonequivalent, satisfying realizations for the graph realization problem generated by $p(G)$. For $p(G)$ to be a unique solution to the realization problem it is necessary and sufficient that this realization set consist of a single point. Our goal in this section is to investigate the structure of the realization set. Our first result is the following theorem.

THEOREM 5.1. *If a graph, G, is connected, then the realization set of $p(G)$ is compact.*

Proof. The realization set is a subset of $\mathbb{R}^{n(n-1)/2}$. It is bounded since the graph is connected, and it is trivially closed. □

Although every point in $\mathbb{R}^{nd}$ corresponds to a realization, the image of $\mathbb{R}^{nd}$ under q does not cover $\mathbb{R}^{n(n-1)/2}$. Define this image to be a space $W \subset \mathbb{R}^{n(n-1)/2}$. The space W has a natural topology and measure inherited from the larger Euclidean space. For technical reasons we will restrict our consideration of realizations to those in which not all the vertices lie in a hyperplane. Call this subset of realizations T. The space T is a dense, open subset of $\mathbb{R}^{nd}$. Define X to be the subset of points in W that are images of points in T under q. If the graph has d or fewer vertices then X is empty. Otherwise, X is a dense, open subset of W, with a nice structure, as we will see shortly. Define Z to be the image of X under π. This gives us the following structure.



We will need the following notation from differential topology. Say the largest rank the Jacobian of a function $g : A \to B$ attains in its domain is k. A point $x \in A$ is called a *regular point* if the Jacobian of g at x has rank k. A point $y \in B$ is a *regular value* if every point in the preimage of y under g is a regular point. If a point or value is not

regular, it is *singular*. Note that for the edge function singular points are not generic. A *j-dimensional manifold* is a subset of some large Euclidean space that is everywhere locally diffeomorphic to $\mathbb{R}^j$.

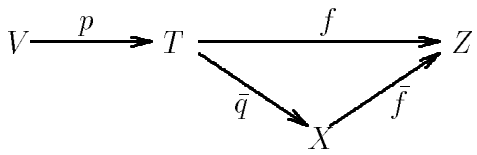
Consider the following procedure for identifying equivalent realizations, which is defined for any realization in T. Select a set of $d+1$ vertices from $p(G)$ whose affine span is all of $\mathbb{R}^d$. Translate the realization so that the first of these vertices is at the origin. Next rotate about the origin to move the second of these vertices onto the positive x_1 axis. Now rotate, keeping the first two vertices fixed, to move the third to the (x_1, x_2) plane so that the x_2 coordinate is positive. Continuing this process in the obvious way gives a smooth mapping that makes $d(d+1)/2$ of the vertex coordinates zero. Finally, if the $d+1^{st}$ vertex has its $d+1^{st}$ coordinate less than zero, reflect the vertices through the hyperplane defined by the $x_1, \ldots, x_{d-1}$ axes.

This procedure maps all equivalent realizations to a single one. This single realization can be described by its remaining variable coordinates, of which there are $nd - d(d+1)/2$. Since each of these remaining coordinates can vary continuously, the realization can be considered to be a point in $\mathbb{R}^{nd-d(d+1)/2}$. This defines a coordinate chart for X. Note that the sequence of operations performed on the original realization is smooth and invertible. If a different set of $d+1$ initial vertices was selected a different coordinate chart would have been generated. Since these coordinate transformations are smooth and invertible, on regions of intersection the two charts are diffeomorphic. The union of all such charts gives a differentiable structure to our space X. This construction provides a diffeomorphism between each open set of a collection that covers X and $\mathbb{R}^{nd-d(d+1)/2}$, giving us the following theorem.

THEOREM 5.2. *If the graph has at least $d+1$ vertices, then X is a smooth manifold of dimension $nd - d(d+1)/2$.*

The dimension of this manifold is a quantity that will come up frequently so it will be convenient to reintroduce the following notation: $S(n,d) = nd - d(d+1)/2$. This function was first defined in § 2.1 as the maximal rank of the rigidity matrix of a graph with n vertices positioned in d-space.

The procedure described above gives us an alternate way in which to view the space X. The sequence of translations, rotations and reflections constitute a function $\bar{q}$ that maps an entire set of equivalent realizations to a single one. The remaining variable coordinates uniquely define a point in X. Considering these to be the independent variables, the mapping from X to Z becomes more complicated than a simple projection. We will define this function to be $\bar{f}$, giving us the following commutative diagram.



This function $\bar{f}$ is closely related to the edge function f. In fact, $\bar{f}$ is everywhere smooth in X, and the rank of the Jacobian of $f(x)$ is the same as that of $\bar{f}(\bar{q}(x))$. So the singular

values of f are the same as the singular values of $\bar{f}(\bar{q})$. If we designate the number of independent edges of G by k then the rank of these Jacobians is almost always k. The following is a special case of a well known theorem due to Sard [34].

THEOREM 5.3 (SARD). *The set of singular values of f has k-measure zero.*

LEMMA 5.4. *If Z' is a subset of Z with k-measure zero, then for almost all realizations p, $f(p) \notin Z'$.*

Proof. The singular points of f constitute an algebraic variety in $\mathbb{R}^{nd}$ with dimension less than nd. Hence, the regular points of f can be covered by a countable number of open neighborhoods in such a way that the rank of the Jacobian of f is maximal within each neighborhood. Consider one of these neighborhoods $\mathcal{R}$, and let its image under f be $\mathcal{Z}$. By the implicit function theorem from analysis there is a submersion from $\mathcal{R}$ to $\mathcal{Z}$. That is, on this neighborhood f is diffeomorphic to a projection from $\mathbb{R}^{nd}$ to $\mathbb{R}^k$. Since Z' has k-measure zero its inverse image under this submersion must have (nd)-measure zero in $\mathcal{R}$. The countable union of these sets of (nd)-measure zero yields a preimage for Z' with (nd)-measure zero. □

These last two results imply the following theorem.

THEOREM 5.5. *For almost every realization p, $f(p)$ is a regular value.*

All this has been leading up to the following crucial result.

THEOREM 5.6. *For almost every realization p, the realization set of $p(G)$ restricted to X is a manifold.*

Proof. Almost all realizations map to regular values of f and hence of $\bar{f}(\bar{q})$. The preimage of a regular value is a submanifold of X by the implicit function theorem from differential topology [20]. □

If the graph is flexible then this manifold describes the allowed flexings. At any point in the manifold, the tangent space is exactly the null space of the Jacobian of $\bar{f}$. To show that flexings typically move in closed loops (actually, one-manifolds diffeomorphic to the circle), we will need the flexings to remain entirely in our manifold X. This can be ensured if the graph has *enough* independent edges. Enough means more than can be independent in a lower dimensional space as the following theorem demonstrates.

THEOREM 5.7. *If G has more than $S(n, d-1)$ independent edges, then for almost all realizations $p(G)$, the realization set of $p(G)$ stays within X.*

Proof. Assume the theorem is false. By the definition of X this means that the realization set must include a point at which all the vertices lie in a $(d-1)$-dimensional hyperplane. When this happens the edges can only constrain infinitesimal motions within the hyperplane. The rows of the rigidity matrix describe these infinitesimal constraints, so when the vertices lie in a hyperplane the rank of the rigidity matrix can be no larger than $S(n, d-1)$. Since there are more than $S(n, d-1)$ independent edges, this implies that $f(p(G))$ is a singular value, but by Theorem 5.5 this cannot be the case for almost all realizations. □

We can finally prove that flexings typically move in closed loops.

THEOREM 5.8. *If a graph, G, is connected, flexible, and has more than $d+1$ vertices, then for almost all realizations $p(G)$ the realization set of $p(G)$ contains a submanifold that is diffeomorphic to the circle.*

Proof. Generate a new graph G' from G by arbitrarily adding additional edges until G' has $S(n,d)-1$ independent edges. The realization set of $p(G')$ must be a subset of the realization set of $p(G)$. Since $n > d+1$ it is easy to show that the number of independent edges is now greater than $S(n,d-1)$. By Theorems 5.6 and 5.1 we know that for almost all realizations the realization set of $p(G')$ is a compact manifold of dimension one. It is a well known result from differential topology that such manifolds are diffeomorphic to the circle. □

This finally leads us to the main result of this section.

THEOREM 5.9. *If G is not redundantly rigid and G has more than $d+1$ vertices, then almost all realizations of G are not unique.*

Proof. Assume the only interesting case, that G is rigid. Then the graph G must have $S(n,d)$ independent edges, and there is some edge e_{ij} of G whose removal generates a flexible graph G'. By Theorem 5.8, for almost all realizations p the realization set of $p(G')$ contains a submanifold diffeomorphic to the circle. The distance between vertices i and j will be a multivalued function for almost every point on this circle. The only distances that might not be multivalued are the extremal ones. When a flexing reaches a realization that induces an extremal value between i and j the derivative of $d_{i,j}^2$ is zero in the direction of the flex. In this case the realization is not generic [32]. So almost all realizations do not induce extremal edge lengths. □

Theorem 5.9 means that the example in Fig. 6 was not a fluke. Redundant rigidity is a necessary condition for unique realizability.

5.1. Algorithms for Redundant Rigidity. 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 approach parallelizes easily by simply running the m different problems on independent sets of processors. Since rigidity testing was shown to be in deterministic or random NC for all dimensions, redundant rigidity is as well.

In one dimension redundant rigidity is equivalent to edge two-connectivity. This property can be determined by looking for cut points of the graph, requiring $O(m)$ time [1].

For the two-dimensional case a simple modification of the rigidity testing algorithm described in § 2.2 can be employed. The rigidity algorithm grows a basis set of independent edges one at a time by checking them against the existing independent set. If a new edge 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.

As in § 2.2 we will denote by $B(G)$ the bipartite graph constructed from $G=(V,E)$. The current set of independent, *basis* edges is $\hat{E}$, generating a subgraph $\hat{G}=(V,\hat{E})$. When a new edge, e, is to be tested for independence, four copies of it are added to $\hat{G}$ generating $\bar{G}$ with its corresponding bipartite graph $B(\bar{G})$. As we saw in § 2.2, if a complete bipartite matching exists in $B(\bar{G})$ then e is independent of $\hat{E}$. For our current purposes we are interested in dependent edges and how they contribute to redundant rigidity. Dependent edges fail to have complete matchings in $B(\bar{G})$. However, if we

triple e instead of quadrupling it, generating $\underline{G}$ and $B(\underline{G})$, then Lemma 2.12 guarantees that $B(\underline{G})$ always has a complete matching. So only a single vertex in $B(\bar{G})$ can go unmatched. This is important because of the following general property of bipartite matching.

THEOREM 5.10. *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 v. 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 resulting 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 v along an alternating path. □

Theorem 5.10 identifies which vertices of a bipartite graph can be removed to result in a perfect matching. For our purposes, these are vertices in $B(\bar{G})$, which correspond to edges of $\bar{G}$. If any of these edges of $\bar{G}$ is removed then the new edge e will be independent of the remaining basis edges. That is, e can replace any of these edges identified by the Hungarian tree, leaving the number of basis edges unchanged. More formally, we have the following theorem.

THEOREM 5.11. *In the rigidity algorithm, assume a new edge, e, is found to be not independent of the current set of k independent edges. Let $\mathcal{V}_1$ be the subset of vertices of V_1 that are in the Hungarian tree of the failed matching. Then if e replaces any of the edges in $\mathcal{V}_1$ the resulting set of k edges is still independent.*

Theorem 5.11 gives an efficient algorithm for redundant rigidity testing. An edge is not independent of the current basis 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. In the $O(n^2)$ algorithm from § 2.2 a Laman subgraph is identified by this Hungarian tree. Hence, any edge in the Laman subgraph is redundant. When the algorithm is finished, if there is a basis edge that has not been merged into a larger Laman subgraph then it is not redundant and the graph is not redundantly rigid. Note that if the full graph is not redundantly rigid then the Laman subgraphs identified by this procedure are redundantly rigid components. This takes essentially no more effort than testing for rigidity, so two-dimensional redundant rigidity can be decided in $O(n^2)$ time.

In dimensions greater than two there is no graph theoretic characterization of redundant rigidity. As in § 2.2 an algorithm will have to randomly position the vertices and then examine the rigidity matrix. Like the two-dimensional case, the basic idea will be to build a set of independent edges one at a time, and then determine which of them are redundant. Every time a new edge fails to be independent it supplies information about the redundancy of some of the independent edges. If a full set of redundant, independent edges are found then the graph is redundantly rigid.

Begin by positioning the vertices randomly and constructing the rigidity matrix M. The rigidity of the framework 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. A new column is added if it is linearly independent of the current set of k columns; otherwise it is discarded. A discarded column, corresponding to an edge e,

can be expressed as a linear combination of some set of the independent columns. The discarded column could replace any of the columns in the linear combination which forms it, without altering the span of the independent set.

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 an $nd \times k$ matrix, A_k. The QR factorization has been proceeding on these columns as they are identified, so there is a $k \times k$ orthogonal matrix Q_k and a $nd \times k$ 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 $O(m)$ triangular systems to solve, each of which requires $O(k^2)$ operations, where k is always $O(n)$. So the total additional time is of the same order as the QR factorization itself, $O(mn^2)$. 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 § 2.2, 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 is 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 [9].

6. Conclusion. Three necessary conditions for almost all realizations of a graph to be unique in d dimensions have been derived. They are, in order of appearance, rigidity, $(d+1)$-connectivity, and 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 think of them independently. Efficient 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 there are combinations of edge lengths for which these conditions aren't necessary. But these counter-examples are very rare. For almost all realizations, a graph that violates one of these conditions will have multiple satisfying realizations.

Establishing necessary conditions for a graph to have a unique realization 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 realization then it would be impossible to assign coordinates unambiguously. Instead portions of the graph that do satisfy the necessary 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 realization can grow exponentially with the size of the graph, it should be possible to save time by positioning 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 realization it might be possible to position subgraphs first and then piece them together. Since the running time grows rapidly with problem size this could lead to a substantial reduction of computational effort. In fact, an approach to the molecule problem using precisely this approach has recently been proposed [21]. For this approach to be infallible we would need to develop sufficiency conditions for a graph to have a unique realization. Unfortunately, the necessary conditions developed in this paper are not sufficient. Connelly has identified a class of bipartite graphs that satisfy the conditions presented here, while still allowing multiple realizations in high dimensional spaces [10]. There are no graphs in this class in one or two dimensions, and $K_{5,5}$ is the only example in three-space. A complete characterization of uniquely realizable graphs remains an open problem. In fact, it is also unknown whether uniqueness itself is a generic property. That is, if a single generic realization of a graph is unique, are almost all realizations unique?

Acknowledgments. The ideas in this paper have been developed and refined in innumerable discussions with Tom Coleman and Bob Connelly.

REFERENCES

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

[2] L. Asimow and B. Roth, *The rigidity of graphs*, Trans. Amer. Math. Soc., 245 (1978), pp. 279–289.

[3] ——, *The rigidity of graphs, II*, J. Math. Anal. Appl., 68 (1979), pp. 171–190.

[4] M. Becker, W. Degenhardt, J. Doenhardt, S. Hertel, G. Kaninke, W. Keber, K. Mehlhorn, S. Näher, H. Rohnert, and T. Winter, *A probabilistic algorithm for vertex connectivity of graphs*, Inform. Process. Lett., 15 (1982), pp. 135–136.

[5] E. D. Bolker and B. Roth, *When is a bipartite graph a rigid framework?*, Pacific J. Math., 90 (1980), pp. 27–44.

[6] A. L. Cauchy, *Deuxième memoire sur les polygones et les polyèdres*, J. l'Ecole Polytechnique, 19 (1813), pp. 87–98.

[7] J. Cheriyan and R. Thurimella, *On determining vertex connectivity*, Tech. Rep. UMIACS-TR-90-79, CS-TR-2485, Dept. of Computer Science, University of Maryland at College Park, 1990.

[8] T. F. Coleman, A. Edenbrandt, and J. R. Gilbert, *Predicting fill for sparse orthogonal factorization*, J. Assoc. Comput. Mach., 33 (1986), pp. 517–532.

[9] T. F. Coleman and A. Pothen, *The null space problem II. Algorithms*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 544–563.

[10] R. Connelly, *On generic global rigidity*, in Applied Geometry and Discrete Mathematics, the Victor Klee Festschrift, DIMACS Series in Discrete Mathematics and Theoretical Computer Science, Volume 4, P. Gritzmann and B. Sturmfels, eds., AMS and ACM, 1991, pp. 147–155.

[11] H. Crapo, *Structural rigidity*, Topologie Structurale, 1 (1979), pp. 26–45.

[12] ——, *On the generic rigidity of plane frameworks*, tech. rep., Bat 10, Institut National de Recherche en Informatique et en Automatique, Cedex, France, 1988.

[13] G. M. Crippen and T. F. Havel, *Distance Geometry and Molecular Conformation*, Research Studies Press Ltd., Taunton, Somerset, England, 1988.

[14] A. Fogelsanger, *The generic rigidity of minimal cycles*, PhD thesis, Cornell University, Dept. of Mathematics, Ithaca, NY, May 1988.

[15] H. N. Gabow and H. H. Westermann, *Forests, frames and games: Algorithms for matroid sums and applications*, in Proc. 20th Annual Symposium on the Theory of Computing, Chicago, 1988, pp. 407–421.
[16] A. George and M. T. Heath, *Solution of sparse linear least squares problems using Givens rotations*, Linear Algebra Appl., 34 (1980), pp. 69–83.
[17] J. Gilbert, *Predicting structure in sparse matrix computations*, Tech. Rep. TR 86-750, Dept. of Computer Science, Cornell University, Ithaca, NY, 1986.
[18] H. Gluck, *Almost all simply connected closed surfaces are rigid*, in Geometric Topology, Lecture Notes in Mathematics No. 438, Springer-Verlag, Berlin, 1975, pp. 225–239.
[19] G. H. Golub and C. F. Van Loan, *Matrix Computations*, Johns Hopkins University Press, Baltimore, MD, 1983.
[20] V. Guillemin and A. Pollack, *Differential Topology*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1974.
[21] B. Hendrickson, *The Molecule Problem: Determining Conformation from Pairwise Distances*, PhD thesis, Cornell University, Dept. of Computer Science, Ithaca, NY, 1990. Technical Report 90-1159.
[22] J. E. Hopcroft and R. E. Tarjan, *Dividing a graph into triconnected components*, SIAM J. Comput., 2 (1973), pp. 135–158.
[23] O. H. Ibarra, S. Moran, and L. E. Rosen, *A note on the parallel complexity of computing the rank of order n matrices*, Inform. Process. Lett., 11 (1980), p. 162.
[24] H. Imai, *On combinatorial structures of line drawings of polyhedra*, Discr. Appl. Math., 10 (1985), pp. 79–92.
[25] A. Kanevsky and V. Ramachandran, *Improved algorithms for graph four-connectivity*, in Proc. 28th IEEE Annual Symposium on Foundations of Computer Science, Los Angeles, October 1987, pp. 252–259.
[26] S. Khuller and B. Schieber, *Efficient parallel algorithms for testing k-connectivity and finding disjoint s-t paths in graphs*, Tech. Rep. TR 90-1135, Dept. of Computer Science, Cornell University, Ithaca, NY, June 1990.
[27] K. Killian and P. Meissl, *Einige grundaufgaben der räumlichen trilateration und ihre gefährlichen örter*, Deutsche Geodätische Komm. Bayer. Akad. Wiss., A61 (1969), pp. 65–72.
[28] G. Laman, *On graphs and rigidity of plane skeletal structures*, J. Eng. Math., 4 (1970), pp. 331–340.
[29] N. Linial, L. Lovász, and A. Wigderson, *A physical interpretation of graph connectivity, and its algorithmic applications*, in Proc. 27th IEEE Annual Symposium on Foundations of Computer Science, Toronto, October 1986.
[30] L. Lovasz and Y. Yemini, *On generic rigidity in the plane*, SIAM J. Algebraic Discrete Meth., 3 (1982), pp. 91–98.
[31] G. L. Miller and V. Ramachandran, *A new graph triconnectivity algorithm and its parallelization*, in Proc. 19th Annual Symposium on Theory of Computing, New York, 1987, pp. 335–344.
[32] B. Roth, *Questions on the rigidity of structures*, Topologie Structurale, 4 (1980), pp. 67–71.
[33] ———, *Rigid and flexible frameworks*, Amer. Math. Monthly, 88 (1981), pp. 6–21.
[34] A. Sard, *The measure of the critical values of differentiable maps*, Bull. Amer. Math. Soc., 48 (1942), pp. 883–890.
[35] J. B. Saxe, *Embeddability of weighted graphs in k-space is strongly NP-hard*, in Proc. 17th Allerton Conference in Communications, Control and Computing, 1979, pp. 480–489.
[36] K. Sugihara, *On redundant bracing in plane skeletal structures*, Bull. Electrotech. Lab. 44 (1980), pp. 376–386.
[37] T.-S. Tay and W. Whiteley, *Generating isostatic frameworks*, Topologie Structurale, 11 (1985), pp. 21–69.
[38] W. Whiteley, *Infinitesimal motions of a bipartite framework*, Pacific J. Math., 110 (1984), pp. 233–255.
[39] W. Wunderlich, *Untersuchungen zu einem trilaterations problem mit komplaneren standpunkten*, Sitz. Osten. Akad. Wiss., 186 (1977), pp. 263–280.