An efficient algorithm for testing the generic rigidity of graphs in the plane

1996 J. Phys. A: Math. Gen. 29 8079

(http://iopscience.iop.org/0305-4470/29/24/030)

J. Phys. A: Math. Gen. **29** (1996) 8079–8098. Printed in the UK

An efficient algorithm for testing the generic rigidity of graphs in the plane

C Moukarzel†

Höchstleistungsrechenzentrum, Forschungszentrum Jülich, D-52425 Jülich, Germany

Received 11 October 1995, in final form 4 July 1996

Abstract. Given a structure made up of n sites connected by b bars, the problem of recognizing which subsets of sites form rigid units is not a trivial one because of the non-local character of rigidity in central-force systems. Even though this is a very old problem of statics, no simple algorithms are available for it, so the most usual approach has been to solve the elastic equations, which is very time consuming for large systems. Recently an integer algorithm was proposed for this problem in two dimensions, using matching methods from graph theory and Laman's theorem for two-dimensional graphs. The method is relatively simple, but its time complexity grows as n^2 in the worst case, and almost as fast on practical cases, so that an improvement is highly desirable. In this paper we describe a further elaboration of that procedure, which relies upon the description of the system as a collection of rigid *bodies* connected by bars, instead of sites connected by bars. Sets of rigidly connected objects are replaced by a unique body, and this is done recursively as more rigid connections between bodies are discovered at larger scales. As a consequence of this 'rescaling transformation', our algorithm has much improved average behaviour, even when its worst-case complexity remains n^2. The time complexity of the body–bar algorithm is found to scale as $n^{1.12}$ for the randomly diluted triangular lattice, while the original site–bar version scales as $n^{1.9}$ for the same problem.

1. Introduction

Consider a structure‡ made of n sites connected by b bars in d dimensions. Determining the rigid properties of such systems constitutes a problem of obvious technological interest, which has already been under study for a long time. The first formal results date back to Maxwell [1], who discussed the connections between statics and geometry. Later theoretical work in the field [2, 4–6, 9, 18, 22–24], was carried out by mathematicians and is not widely known among physicists, although the concept of rigidity appears in many fields of physics, such as, for example, glasses [10, 20], critical phenomena [11] and granular materials [12] among others.

The physicists' treatment of this problem often reduced to a brute-force solution of the elastic equations because of the lack of a simple integer algorithm for the identification of the rigid clusters. In this work, recent progress in the design of such algorithms for the analysis of rigid properties of generic lattices is reported. To do so, let us start by briefly introducing some of the basics of rigidity. In this section the main ideas will be described qualitatively,

† Present Address: Instituto de Física, Universidade Federal de Rio Grande do Sul, Porto Alegre RS, Brazil; e-mail address: cristian@if.ufrgs.br

‡ The words *structure* or *lattice* are used in this work to describe a set of point-like objects (called joints or nodes) connected by bars (also called bonds or edges). Bars are only able to transmit forces in the direction of their axis. No regularity is assumed whatsoever. In the engineering context this is usually called a *framework*.

0305-4470/96/248079+20$19.50

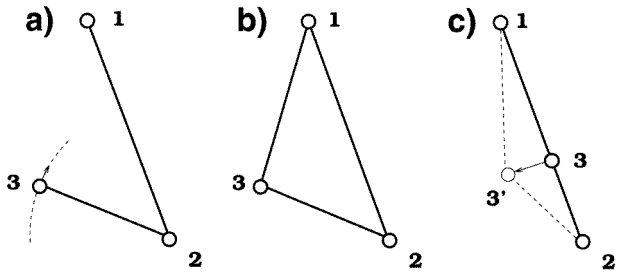


Figure 1. A simple illustration of the reasons for which a structure may fail to be infinitesimally rigid (cases (*a*) and (*c*)) is provided by a triangle. Case (*a*) is flexible since it admits a continuous deformation. Case (*b*) is both rigid (no continuous deformation is possible) and infinitesimally rigid (no infinitesimal deformation is possible). Case (*c*) on the other hand provides an example of a *degenerate* configuration, for which the structure is rigid (there is no continuous deformation leaving all bar lengths unchanged) but not infinitesimally rigid, since site 3 may be displaced by an infinitesimally small amount, and all bar lengths would remain unchanged to first order in the displacement. Failure to be infinitesimally rigid can be shown to be equivalent to the existence of zero-frequency modes in the linear approximation of the vibratory behaviour of the structure.

leaving more precise definitions, which will be made with the aid of the rigidity matrix, to the next section. The interested reader is referred to the recent literature for more complete descriptions [5, 6, 9], alternative approaches [18, 21] and recent results [22–24] in the field of rigidity.

A structure† is *flexible* if it admits a continuous deformation (a finite *flexing*) preserving all bar lengths, other than the trivial translations and rotations in Euclidean space. Otherwise it is *rigid*. Obviously if a structure admits a finite flexing, then it also admits an infinitesimal flexing. If a structure admits no non-trivial infinitesimal flexing, it is said to be *infinitesimally rigid*. Obviously a flexible structure is also infinitesimally flexible. The converse is not always true, since there may be special situations in which a structure is infinitesimally flexible but does not have finite flexings. Figure 1(*a*) shows an example of a structure that admits a finite flexing, and therefore is not rigid. If one more bar is added, the extra degree of freedom corresponding to the flexing may be eliminated. Figure 1(*b*) is an example of a structure which is both rigid and infinitesimally rigid. However, the same triangle has special combinations of site locations for which infinitesimal rigidity is lost. This is exemplified in figure 1(*b*), in which the three sites are aligned. As a consequence site 3 may be displaced by a small amount in the direction of the normal to bar 12, and all bar lengths remain unchanged *to first order in the displacement*. Therefore this structure is infinitesimally flexible, though it is rigid to second order in the displacement.

These situations for which rigidity does not imply infinitesimal rigidity are very rare. They only occur for a 'small' set of site locations, which are called *degenerate* configurations. The complement of this set, the *generic* configurations, form an open dense subset of space. For a generic configuration, infinitesimal rigidity is equivalent to rigidity. Configurations in which all site coordinates are randomly chosen are generic with probability 1 (a precise definition will be provided in the next section).

The importance of infinitesimal rigidity can be easily understood in physical terms. One can define a structure to be *statically rigid* if it is able to compensate, by means of

† For a definition see the second footnote on the title page of this paper.

suitable finite stresses in its bars, any equilibrated load applied to its nodes. It is a classical engineering result that static rigidity is equivalent to infinitesimal rigidity (see [9] for a proof). Therefore infinitesimal rigidity, and not rigidity, is the relevant concept in the linear approximation of elasticity. Case (a) in figure 1 is obviously not statically rigid. Case (c) is also not statically rigid because a load normal to bar 12 applied on site 3 cannot be compensated by finite stresses on the bars, which are all parallel to 12.

When studying the oscillatory properties of a structure in the harmonic oscillator approximation, infinitesimal flexibility is equivalent to the existence of degenerate, zero-frequency modes. Note that cases (a) and (c) in figure 1 both have zero-frequency modes, but for different reasons. In case (a) because there is no restoring force, and in case (c) because the restoring force is zero in the linear approximation.

Throughout this work we will be concerned with the property of infinitesimal rigidity, and therefore we drop the prefix 'infinitesimal' in what follows. As suggested by the example in figure 1, a structure can fail to be rigid for two different reasons. In the first place, because it has too few bars, or they are not correctly distributed. This has to do with the topology of the lattice, i.e. it depends only on its connectivity, and not on the geometry of the structure. We will say that a given structure is *generically rigid* if it has the required minimum number of 'correctly distributed' bars. Generic rigidity is a necessary condition for rigidity, but not a sufficient one. The importance of generic rigidity resides in the fact that a generically rigid structure will be rigid for generic site locations. Generic rigidity depends only on the number and location of the bars, and not on site locations. In other words, generic rigidity depends only on the topological properties of the structure.

Topological information about a structure is conveniently represented by means of a graph $G = (V, E)$. Each site of the lattice is associated with a node $a \in V$, while bars are associated with edges $ab \in E$. The nodes a and b are *adjacent* if the edge ab exists. The edge ab is said to be *incident* to nodes a and b and, conversely, the nodes a and b are incident to the edge ab. Graphs as described here contain no information about the geometry of the system (site locations), but only about its connectivity properties.

A graph contains enough information about a structure if we are only willing to discuss its generic properties, i.e. those which are valid for 'almost all' sets of site locations, except for those few degenerate configurations, which will be ignored. It is only meaningful to do this if degenerate configurations, and therefore lattices which are generically rigid but not rigid, are 'exceptional'. A context in which this is justified is when site locations (an assignment of site locations to the nodes of a graph is called a *realization* of the graph) are randomly chosen. This ensures that degenerate configurations have zero probability of appearing, i.e. degenerate realizations are a zero measure set. In view of this we would like to determine, from topological information only, whether a given graph is generically rigid. A first condition which must be satisfied is that the number b of bars be sufficiently large. Since n points in d dimensions have dn degrees of freedom and each bar restricts one degree of freedom, b has to be at least equal to $dn - d(d+1)/2$†, where $d(d+1)/2$ is the number of distance-preserving linear transformations in d dimensions (d translations and $d(d-1)/2$ rotations). However, a correct global count of bars is not enough to ensure rigidity, since bars could be 'crowded' on certain subsets of the graph, while others have less bars than needed to make them rigid. If a certain subgraph has more bars than necessary, some of them are *redundant*, and this subgraph is *overconstrained*. Bars which are not redundant are said to be *(generically) independent*. A sufficient condition for (generic) rigidity is that the graph possesses $dn - d(d+1)/2$ (generically) independent bars. We see that the key

† We are assuming $n \geqslant d$.

point is that of being able to identify independent bars. The basic theoretical tool for doing this in two dimensions is provided by a theorem due to Laman [2].

Theorem 1 (Laman). The edges of a bar and joint graph $G = (V, E)$ are generically independent in two dimensions if and only if no subgraph $G' = (V', E')$ has more than $2n' - 3$ edges.

Laman's theorem constituted the first graph-theoretic characterization of rigidity in two dimensions, but in its original form it would give a very bad algorithm since it requires testing all possible subgraphs, of which there is an exponentially large number. There are some equivalent restatements of this theorem [7, 9], some of which give rise to polynomial-time algorithms. Using one such equivalence due to Sugihara, Hendrickson [13] has recently proposed an algorithm for testing generic rigidity of two-dimensional graphs, which is simple enough to admit an on-lattice implementation [17]. Roughly described, Hendrickson's algorithm consists in adding edges one by one to the graph and matching [14, 15] them to the nodes. If the matching succeeds, the new edge is independent and is left on the system. If the matching fails, then (a) the set of edges visited during the failed search is mutually

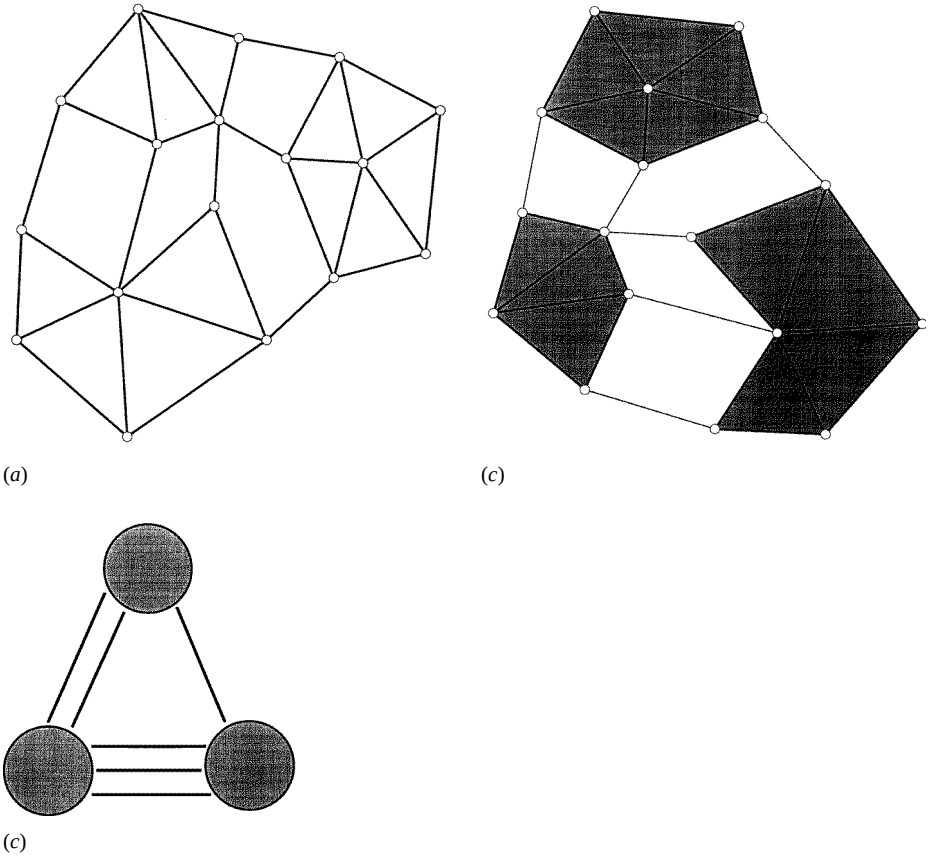


Figure 2. In the *bar–joint* representation (*a*), each site of the lattice is associated with a node of the graph, and bars are represented by edges. If sub-sets of rigidly connected sites (*b*) are combined into *bodies* and represented by a node, we get the *body–bar* representation (*c*). The resultant structure is a *multigraph* since several edges can connect a given pair of nodes. The process of replacing a set of rigidly connected nodes by one node is called *condensation*.

rigid, and (b) the last edge is redundant. This algorithm has a worst-case time complexity that scales as $O(n^2)$. One factor of n arises as edges are added one at a time, while the degree to which the computational time is greater than n is determined by the typical size of the search that must be performed in order to match each added edge. If $O(n)$ sites are mutually rigid, this size is of order n, and the algorithm is of order n^2. However, in this case considerable time is spent in searching over edges that have been previously identified as mutually rigid. There is thus room for improvement, and this work is devoted to the description of such an improved algorithm.

To avoid the need to search over edges which have already been identified as mutually rigid, it is natural to combine mutually rigid edges into clusters or 'bodies' [8, 12, 20, 21, 23, 24]. To develop this idea into a workable algorithm, we note that any bar and joint graph can be considered as being composed of bodies and bars, where eventually some of the bodies may be 'trivial' bodies with just one site. Each body is represented by a node in a *multigraph*. An example of this is shown in figure 2. Each body, composed of several sites rigidly connected by bars, is shown as a node in the multigraph of figure 2(c). One of the advantages of the body–bar representation is that the number of elements in the graph is smaller, and may be reduced each time a cluster of rigidly connected bodies is identified, by replacing them by just one node in a process we will call *condensation*.

Of course, we now have to demonstrate that the idea is sound, and for that we must restate the relevant theoretical results [13] in terms of bodies and bars. We start in section 2 by introducing the *rigidity matrix*, which will help us to more precisely define the concepts of infinitesimal rigidity, generic rigidity, generic configurations and of redundant and independent bars, and also to discuss the necessary conditions for generic rigidity. In section 3 we express Laman's theorem in the body–bar language, while section 4 shows how Hendrickson's algorithm is generalized for this case. Section 5 discusses some practical implementation details. Also in this section the performance of the body–bar algorithm introduced here is compared to that of the previously existing joint–bar version. We will see that the body–bar algorithm has much better scaling properties for the two examples analysed, even when its worst-case behaviour is the same as that of the joint–bar algorithm, that is, $O(n^2)$.

2. Rigidity matrix for two-dimensional body–bar systems

In what follows we will only consider two-dimensional structures, unless explicitly stated otherwise. Consider figure 2, in which the relationship between a bar–joint and a bar–body representation is explicitly drawn. Any subset of rigidly connected bars and sites can be replaced by a *body*, a rigid object that in two dimensions has three degrees of freedom, two displacements and one rotation. We call all remaining bars *external*. Each body has a set of joints on its surface, to which external bars are incident. We let x_i be the location of joint i belonging to body a. An infinitesimal motion is a set of instantaneous velocities $\{v_i\}$, one for each joint, which leave all bar lengths unaltered. This condition is written as

$$(x_i - x_j) \cdot (v_i - v_j) = 0 \tag{1}$$

for every bar ij with $i \in a$, $j \in b$. We now express the velocities of the joints in terms of the velocities of the body to which they belong. For this we select an arbitrary point x_a for each body a, and say that the velocity v_i of any joint i of a body is equal to the velocity v_a of x_a plus a rotational component, which is $\omega_a \wedge (x_i - x_a)$, where ω_a is the angular velocity (a vector normal to the plane) of body a, and $\wedge$ indicates vector product:

$$v_i = v_a + \omega_a \wedge (x_i - x_a). \tag{2}$$

Without loss of generality, we can choose all those arbitrary reference points x_a to be at the origin and obtain

$$v_i = v_a + \omega_a \wedge x_i. \tag{3}$$

Now let us rewrite (1) by using (3), so that

$$(x_i - x_j) \cdot \{(v_a - v_b) + \omega_a \wedge x_i - \omega_b \wedge x_j\} = 0. \tag{4}$$

A little algebra shows that (4) can be reduced to

$$(x_i - x_j) \cdot (v_a - v_b) + (x_j \wedge x_i) \cdot (\omega_a - \omega_b) = 0. \tag{5}$$

This set of equations can be formally written as

$$\tilde{M}\boldsymbol{V} = 0 \tag{6}$$

where $\boldsymbol{V}$ is the vector of velocities and contains three entries for each body. Each row in $\tilde{M}$ corresponds to a bar ij. Conditions (1), when written in the bar–joint representation, also give rise to a matrix equation of the form (6). In the bar–joint representation only the first term in (5) occurs, each row of the rigidity matrix is associated with the vector $(x_j - x_i)$, and there are four non-zero elements per row. In the body–bar case, each row of the rigidity matrix is associated with the 'line–bound vector' $(x_j - x_i, x_i \wedge x_j)$ and there are 6 non-zero elements per row since $x_i \wedge x_j$ is a pseudoscalar in two dimensions. Line–bound vectors represent a force acting along a line [5] and, in contrast to vectors, are not translationally invariant. They are only invariant under translations in the direction $(x_i - x_j)$, meaning that a force can be moved along its line of action without changing its effect. Line–bound vectors have three independent components in 2D, two of them are needed to determine the vector and the third one locates its line of action in the plane.

We will consider the general case of multigraphs formed by n bodies and m point-like (or 'trivial') bodies, as in figure 4 later, which we denote as $G(n, m)$. The general equations (5) or (6) hold for $G(n, m)$ with the additional constraint that the angular velocity $w_a = 0$ for each of the m point-like bodies. These additional constraints reflect the fact that rotation is irrelevant for them, so their angular velocities can be arbitrarily fixed, thus reducing the dimension of $\boldsymbol{V}$. Counting the number of degrees of freedom, we then find (3n + 2m). The space of solutions of (6) has at least dimension 3, since two rigid translations and a rotation of the system as a whole leave all bar lengths unchanged. This means that the rank of $\tilde{M}$ cannot be larger than $K(n, m) = 3n+2m-3$. The system is said to be (infinitesimally) rigid if the rigidity matrix has this maximal rank $K(n, m)$, which means that the *only* infinitesimal motions are the Euclidean rigid transformations.

A *realization* is an assignment of site locations to all nodes of a graph. For certain (degenerate) realizations, the rank of the rigidity matrix may be accidentally lowered by the existence of algebraic dependencies between the node coordinates (degeneracies). This will only happen if a determinant is zero, and given that determinants of the rigidity matrix are polynomials in the site coordinates, degenerate realizations must satisfy a finite number of polynomial equations. Therefore the subset of configurations for which the rigidity matrix attains the maximum possible rank over all sets of coordinates constitutes an open dense subset. We say that a realization is *generic* if all site coordinates are algebraically independent over the rationals. Therefore at a generic configuration the rank of the rigidity matrix attains its maximum value over the site coordinates.

A set of edges is *independent* if their associated rows in $\tilde{M}$ are linearly independent in an algebraic sense. This means that an edge will be independent if and only if by removing it the rank of the rigidity matrix is decreased by one. Cases (a) and (b) in figure 3 are independent, but case (a) is flexible, while case (b) is infinitesimally rigid.

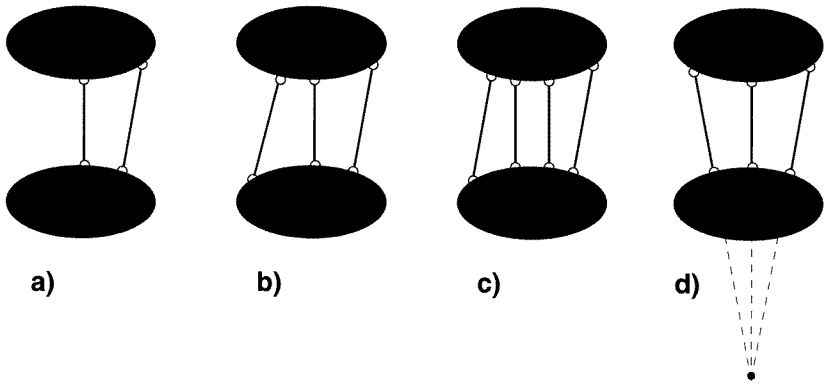


Figure 3. A simple body–bar structure for which: (*a*) the bar set is independent, and the structure is flexible; (*b*) the bar set is independent, and the structure is rigid; (*c*) the bar set is dependent due to an excess of bars, and the structure is rigid; and (*d*) the bar set becomes dependent at a degenerate configuration, and the structure becomes infinitesimally flexible there. In this last case, the existence of a common intersection point for three bars is a non-generic configuration at which the rank of the associated rigidity matrix is reduced from its generic value of 3 to a non-generic value of 2. The extra eigenvector is identified as an infinitesimal relative rotation of the two bodies around the common intersection point.

If the removal of a given edge e does not alter the rank of $\tilde{M}$, e is *dependent* or *redundant*. Dependencies can arise because there are too many bars (for example the edge set in figure 3(*c*)) or because of degenerate configurations. (for example three bars with a common point as in figure 3(*d*)). The difference is that case (*c*) is infinitesimally rigid while case (*d*) is not. The dependency produced by the coincident intersection point has reduced the rank of the rigidity matrix from its maximum value of 3 to a value of 2. There is therefore one extra infinitesimal motion, which can be identified as a relative rotation of the two bodies around the intersection point of the bars. A similar reasoning holds for the example shown in figure 1(*c*).

Once the atypical character of degeneracies is recognized, we are justified in ignoring them, and concentrate on generic properties only. Therefore we define generic rigidity by saying that: *a structure is generically rigid if its rigidity matrix attains its maximum rank $K(n, m)$ at a generic configuration*. The relevance of generic properties is ensured by a theorem due to Gluck [3].

Theorem 2 (Gluck). If a graph has a single infinitesimally rigid realization, then all its generic realizations are rigid.

In other words,

At a generic realization, a structure is infinitesimaly rigid if and only if it is higher-order rigid if and only if its multigraph is generically rigid.

In addition to the property of generic site locations already defined, we will assume the following *generic incidence condition* to hold: *no three bars can be incident to the same joint of a* non-trivial *body*. This condition means that bars incident to the same body are located on generic lines. In what follows, whenever we refer to generic realizations of multigraphs we will be assuming generic joint locations as well as generic incidence. Notice that multiple incidences are allowed if they occur on trivial (single-site) bodies. An example of a multigraph that is generic in the sense required here is shown in figure 4.

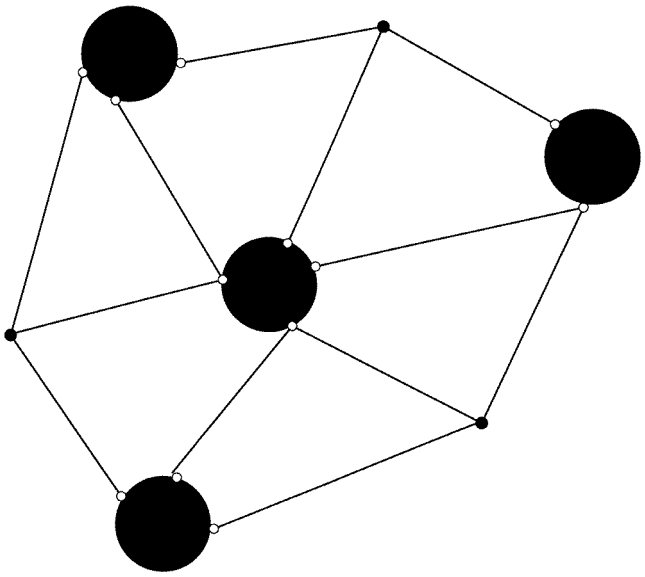


Figure 4. A multigraph that satisfies the generic incidence condition. No surface joint has more than two incident bars. Point-like bodies can have an arbitrary number of incident bars.

Throughout this work we will assume all multigraphs $G(n, m)$ to contain at least one non-trivial body or two point-like ones, that is, $n + m/2 \geqslant 1$. Excluded then are the graph with no nodes and those with just one trivial (point-like) body. Under this condition the following results follow almost trivially from our discussion of the rigidity matrix.

Theorem 3. Every rigid multigraph $G(n, m)$ has a subset of $K(n, m)$ independent bars.

Theorem 4. If a multigraph $G(n, m)$ has more than $K(n, m)$ bars, some are dependent.

Theorem 5. If a multigraph $G(n, m)$ with exactly $K(n, m) = 3n + 2m - 3$ bars is rigid, then there is no subgraph G' with more than $K(n', m')$ bars.

In the process of obtaining the bar–body representation from the bar–joint representation, we have replaced subsets of rigidly connected sites by bodies with three degrees of freedom. This does not change the number of independent infinitesimal motions of the system, since these subsets, being rigid, had three degrees of freedom each. This means that the dimension of the null space of the rigidity matrix is the same in the bar–joint and bar–body representations. A consequence of this is the following.

Theorem 6. A set E of external edges is dependent in the body–bar representation if and only if it is dependent in the corresponding joint–bar representation.

Proof. The removal of an independent edge causes the rank of the rigidity matrix to change, increasing by one the number of independent infinitesimal motions. Assuming a given edge $e \in E$ to be dependent in one representation but not in the other would then conflict with our discussion above. □

Another result which we need for later use is the following.

Theorem 7. If an edge e incident to a non-trivial (trivial) body a and to some other body $b \neq a$ is dependent, then there exist at least three (two) other edges $\{e', e'', e'''\}$ ($\{e', e''\}$) incident to a which are also dependent.

Proof. Let a be a non-trivial body and assume that an edge e incident to a is dependent. Take the row of $\tilde{M}$ corresponding to e, which under the hypothesis can be expressed as a linear combination of other rows of $\tilde{M}$, and consider its three components associated with the three degrees of freedom of a. These are the three components of a line–bound vector, and are independent degrees of freedom in a generic realization so that at least three other line–bound vectors (rows of $\tilde{M}$) are needed to express it as a linear combination. The demonstration for the case in which a is a trivial body is similar, except that vectors (two degrees of freedom) take the place of line–bound vectors, and therefore only two other edges are needed. □

We have introduced the rigidity matrix $\tilde{M}$ through a discussion of infinitesimal rigidity, but it also appears in the context of static rigidity, describing how external loads are resolved into stresses on bars. Dependent subgraphs can then be identified to be those that can sustain equilibrated internal stresses even in the absence of external loads.

3. Laman's theorem for body–bar systems

A general graph-theoretic characterization of rigidity for linkages of bodies in n-space was first provided by Tay [8]. We will derive a similar result in two dimensions, which holds for the case of mixed multigraphs, and which we need for our algorithm. Mixed multigraphs are those including point-like bodies as well as non-trivial ones. Our derivation is based on Laman's theorem for bar–joint systems.

Theorems 3 and 5 imply that a rigid system must have a set of $K(n, m)$ well distributed bars, where the meaning of 'well distributed' is that no subgraph has 'too many' bars. This condition is known, in the context of bar–joint systems, as Laman's condition, and is a *necessary* condition for rigidity in *any* space dimension (in a suitably generalized form). The converse, i.e. $K(n, m)$ well distributed bars $\Rightarrow$ rigid does not hold in any dimension above 2. A counterexample in three dimensions, due to Whiteley, is shown in figure 5. This structure satisfies $b' \leqslant 3n' - 6$ for all subgraphs with $n' > 2$, yet it is dependent and therefore non-rigid.

Laman was able to show [2] that the 'correct distribution' of bars is a sufficient condition for rigidity in two dimensions (theorem 1 above). Here we translate his result to the body–bar case and show that

Theorem 8. The edges of a multigraph $G(n, m) = G(V_n, V_m, E)$ are independent in 2D if and only if there is no submultigraph $\tilde{G}$ with more than $3\tilde{n} + 2\tilde{m} - 3$ edges.

Proof. The demonstration of necessity, i.e. independent $\Rightarrow$ well distributed, follows trivially from theorem 4 above. In order to demonstrate sufficiency, we will transform the body–bar graph to a bar–joint graph and use Laman's theorem to show that, if a multigraph is dependent, there is a subset of it with too many bars, i.e. a bad submultigraph.

Assume that a multigraph $G = (V_n, V_m, E)$ contains a subset E' of dependent bars. Let $G' = (V_n', V_m', E')$ be the submultigraph of G defined by restricting the edge set to E' and the node sets to those nodes incident to some edge in E'. For each non-trivial body $a \in V_n'$, let the cardinality g_a be the number of joints on its surface to which bars $ab \in E'$ are incident. According to theorem 7, in order for a non-trivial body a to belong to V_n', at least four of its incident bars must belong to E'. This together with the condition of generic incidence (section 2) implies $g_a \geqslant 2$. A body of cardinality $g_a \geqslant 2$ can be replaced by an isostatic bar–joint graph G_a made up of $2g_a - 3$ well distributed 'internal' bars and g_a point-like bodies, these last taking the place of surface joints. Doing this for each of the n' non-trivial bodies in G' leads to the *expanded* graph G_E.

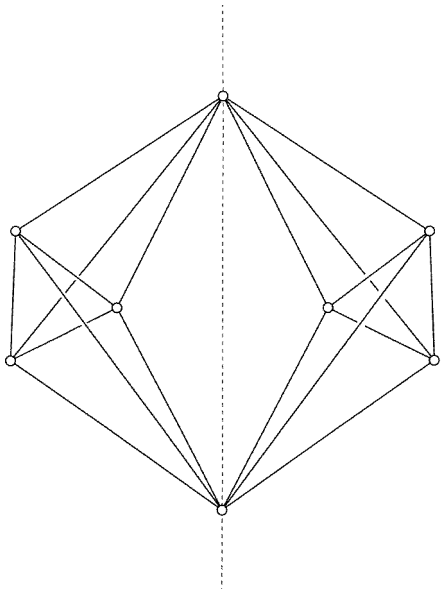


Figure 5. The graph in this figure has no rigid realization in three dimensions, even when it has all the required $18 = 3 \times 8 - 6$ bars, and no subgraph of it violates Laman's condition $b' \leqslant 3n' - 6$.

Theorem 6 implies that E' is also dependent in the expanded graph G_E, and since Laman's theorem in its original form applies to it, there must be a subgraph $\tilde{G}_E = (\tilde{V}, \tilde{E})$ of G_E with $\tilde{b}$ bars and $\tilde{m}$ joints such that $\tilde{b} > 2\tilde{m} - 3$.

We will in general have $\tilde{b} = \tilde{b}_e + \tilde{b}_i$, where $\tilde{b}_e$ is the number of external edges and $\tilde{b}_i$ the number of internal edges in $\tilde{E}$. $\tilde{E}$ cannot be entirely formed by internal edges since they are well distributed, and therefore it must contain one or more external edges. Two cases are possible:

(a) There are no internal bars in $\tilde{E}$.

If this is the case none of the nodes of $\tilde{G}_E$ can be the surface joint of a body. The reason for this is theorem 7. At least three bars incident to a point are dependent if one of them is. However, if none of them is internal the point cannot be a surface joint since at most two incident bars are allowed per surface joint in generic multigraphs. This means that no bar in $\tilde{E}$ is incident to a subgraph G_a. In this case the bad subgraph is entirely formed by point-like bodies and the result follows immediately since Laman's theorem is a particular case of this one (with $n = 0$).

(b) There are one or more internal bars in $\tilde{E}$, or equivalently, there is at least one edge $e \in \tilde{E}$ incident to G_a for some body a.

If this edge e has both ends incident to the same G_a, then we have identified the bad sub-multigraph as being formed by body a and this edge with both ends connected to it, since $1 = b > 3n - 3 = 0$. Then assume that no edge $e \in \tilde{E}$ has both ends connected to the same G_a. For each body a to which some edge $e \in \tilde{E}$ is incident, let $\tilde{G}_a$ with $\tilde{g}_a$ nodes and $\tilde{b}_a$ edges, be the subgraph of G_a contained in $\tilde{G}_E$. Let $\tilde{n}_e$ be the number of such bodies, and $\tilde{m}_e$ the number of joints in $\tilde{G}_E$ other than surface joints. The condition that $\tilde{G}_E$ be a bad subgraph is then rewritten

$$\tilde{b}_e + \sum_{a=1}^{\tilde{n}_e} \tilde{b}_a > 2\left(\tilde{m}_e + \sum_{a=1}^{\tilde{n}_e} \tilde{g}_a\right) - 3. \tag{7}$$

Theorem 7, together with the generic incidence condition, ensure that $\tilde{g}_a \geqslant 2$. Since G_a

are independent by construction Laman's condition then implies $\tilde{b}_a \leqslant 2\tilde{g}_a - 3$. Using this and (7) we get

$$\tilde{b}_e + \sum_{a=1}^{\tilde{n}_e}(2\tilde{g}_a - 3) > 2\left(\tilde{m}_e + \sum_{a=1}^{\tilde{n}_e}\tilde{g}_a\right) - 3 \tag{8}$$

or

$$\tilde{b}_e > 2\tilde{m}_e + 3\tilde{n}_e - 3 \tag{9}$$

which finishes the demonstration for generic body–bar graphs. □

Non-generic graphs, i.e. those with more than two bars incident to a joint of a body (we call this 'multiple incidence') can also be handled by transforming them into an equivalent generic graph in the following form. Notice that there is no limitation on the number of bars incident to point-like bodies. Thus we can simply replace each multiple-incidence joint of a body by an auxiliary structure made of a point-like node connected to the body by two new bars, like in figure 6. The graph obtained by transforming in this way all multiple-incidence joints is generic in the sense required here, and therefore the extended Laman's theorem (theorem 8 above) applies to it. It is easy to see that this transformed graph has equivalent rigid properties.

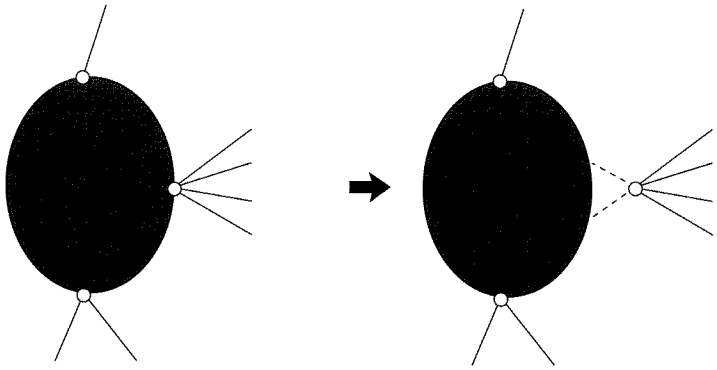
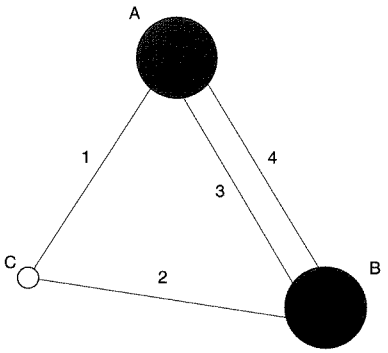


Figure 6. A graph with a multiple-incidence joint (a surface joint with more than two incident bars) is not generic. An equivalent graph (with the same rigid properties) that is generic can be obtained by detaching the surface joint and connecting it to the body with two auxiliary bars (dashed lines).

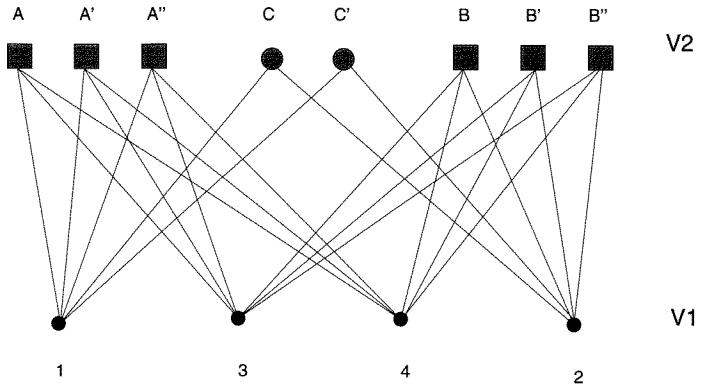
4. The algorithm

Laman's theorem constitutes a graph-theoretic characterization of rigidity, but a naive implementation of it, namely checking all possible subgraphs, would give a very poor algorithm. Sugihara [7] and later Hendrickson [13] have used a reformulation of Laman's theorem to propose efficient (polynomial-time) algorithms for this problem. This section follows Hendrickson's approach, adapting his arguments to the body–bar case where needed.

We first define the bipartite graph $B(G)$ generated by a graph $G(V_n, V_m, E)$ in the following way: $B(G)$ is composed of two sub-sets of nodes $V1$ and $V2$, and a set of edges connecting nodes of $V1$ with nodes of $V2$. There are no edges between nodes in the same subset (that is what defines a bipartite graph). The first subset $V1$ is the set of edges E of



(*a*)



(*b*)

Figure 7. (*a*) The original graph $G(V_n, V_m, E)$ has $V_n = \{A, B\}$ (non-trivial bodies), $V_m = \{C\}$ (point-like bodies) and $E = \{1, 2, 3, 4\}$ (edges). (*b*) The bipartite graph $B(G)$ derived from G has as node set the union of $V1$ and $V2$. $V1$ is the edge set of G, while $V2$ is made of three copies of V_n plus two copies of V_m. Elements of $V1$ and $V2$ are connected by an edge in $B(G)$ if they are incident in G.

G, while $V2$ is composed of three copies of the set of non-trivial nodes V_n plus two copies of the set of point-like nodes V_m. Edges of $B(G)$ connect the edges (ab) of G ($V1$ nodes) to all copies ($V2$ nodes) of the bodies a and b to which (ab) is incident. An example of this is shown in figure 7. We now briefly describe some concepts from graph theory which we need for our algorithm. The reader is referred to the literature on the subject [14, 15] for detailed accounts.

A *matching* $\mathcal{M}$ of a graph is a subset of edges, no two of which share a node. Edges in $\mathcal{M}$ are said to be *matched*, and edges not in $\mathcal{M}$ *unmatched*. Nodes incident to a matched edge are *covered*, while the rest are *exposed*. If $(ab) \in \mathcal{M}$, nodes a and b are *mates*. A matching $\mathcal{M}$ is *maximum* if it has the largest possible number of edges. A matching is *perfect* or *complete* if no node is exposed. Obviously, a perfect matching is also maximum. The *matching problem* (finding maximum matchings) is a classic in graph theory, and has many practical applications. A *path* is a chain of edges of the form $\{(ab)(bc)(cd)\cdots\}$, and it is *alternating* if matched and unmatched edges follow each other in the succession. An alternating path $\{(ab)(bc)\cdots(xy)\}$ is an *augmenting path* if both a and y are exposed

nodes. The name is due to the obvious fact that any such path allows one to increment the number of edges in the matching by one, simply by interchanging *matched* $\leftrightarrow$ *unmatched* along the path. Moreover, the problem of finding a maximum matching can be reduced to that of finding augmenting paths, since

Theorem 9. $\mathcal{M}$ is maximum $\iff$ there are no augmenting paths.

Then a maximum matching is found by repeatedly discovering and inverting augmenting paths. A particularly simple case is the matching of bipartite graphs, i.e. those whose node set V can be partitioned into two subsets $V1$ and $V2$ such that no edge is incident to two nodes in the same subset. The search for augmenting paths can be efficiently done by growing *Hungarian trees* from exposed nodes. They are built by breadth-first search (BFS) in the following way. Consider figure 8, in which thick lines represent edges in the current matching. We start the search for augmenting paths from an exposed node $v1 \in V1$. Take node 1 for example. Following all unmatched (thin) edges incident from $v1$, go to nodes $v2$ in $V2$. Since $v1$ is exposed it is incident to unmatched edges only. In our example of figure 8 these edges lead to A, B. If any of these $v2$ nodes is exposed, we have found an augmenting path. If not, their mates in $v1$ are sent to a *queue* Q for further inspection. These mates are $2, 4$ in this case. Nodes $v2$ are marked *visited*. Once all neighbours of 1 have been exhausted without finding an exposed node, the next element $v1$ in queue Q is taken. Say it is node 2. All unmatched edges incident to 2 are followed to $v2$ in $V2$ and, if some of them lead to an exposed node, the search is over. Otherwise, and if $v2$ was not visited before, it is marked visited and its mate in $V1$ is sent to Q. In our example this would result in node 3 being sent to Q. The procedure proceeds in this way until either an exposed node in $V2$ is found, or Q is depleted. In our example, after taking node 4 from Q we would find node D exposed. The matching can then be enlarged to cover node 1 by inverting the path $D \rightarrow 4 \rightarrow B \rightarrow 1$. If on the other hand Q is depleted without finding an exposed node in $V2$, there is no augmenting path from 1. We say in this case that v_1 cannot be matched. Now we are ready to discuss some results needed for our algorithm. Using the relation between G and $B(G)$ we can give an equivalent formulation for the extended Laman's theorem, which will provide the basis for our algorithm.

Theorem 10. The following are equivalent:

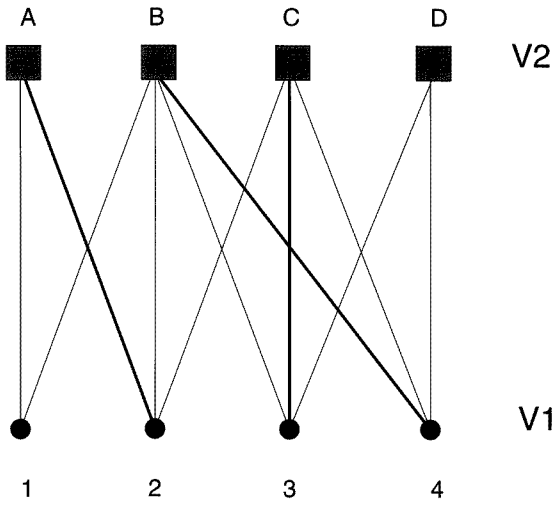


Figure 8. A matching $\mathcal{M}$ (thick lines) can be enlarged to cover node 1 if an augmenting path is found. Starting from 1 all unmatched (thin) edges are followed to nodes in $V2$, and from them all matched edges are followed back to nodes in $V1$, in a breadth-first manner. Repeating this procedure, an exposed node D is found in $V2$. If thin and thick edges are now interchanged along the path $1 \rightarrow D$, the enlarged matching is obtained.

(A) The edges of G are independent in two dimensions.
(B) For every edge ab in G, the multigraph G_{ab} formed by quadrupling ab has no subgraph with $b' > 3n' + 2m'$.
(C) For each edge ab in G, the bipartite graph $B(G_{ab})$ has no subset of $V1$ that is adjacent only to a smaller subset of $V2$.
(D) For each edge ab in G, $B(G_{ab})$ has a complete matching from $V1$ to $V2$.

Proof. The equivalence of (A) and (B) is a trivial consequence of Laman's theorem in its extended form. The equivalence of (B) and (C) is an immediate consequence of the way in which G_{ab} is constructed. If such a subset exists, one would have b' nodes of $V1$ connected only to $3n' + 2m' < b'$ nodes of $V2$. To prove that (C) and (D) are equivalent, it is easy to first see that (C) is necessary for (D). To prove that (D) is necessary for (C), assume that no complete matching exists. Then there exists at least one exposed node x in $V1$. Do a breadth first search (BFS) in the following way. Starting from x, follow all *unmatched* edges from $V1$ to the nodes in $V2$, and from them all *matched* edges back to $V1$ and so on. For each new node in $V1$ found (including x), increment in 1 a variable k_1, and for each new node in $V2$ do the same with k_2. Both k_1 and k_2 start from zero. Each new node in $V2$ leads automatically to a new node in $V1$, because the hypothesis implies that no exposed node can be found. Therefore when the BFS comes to an end (because all $V2$ nodes have been visited once) we have that $k_1 = k_2 + 1$ and we have identified a subset of k_1 nodes of $V1$ adjacent to only $k_1 - 1$ nodes of $V2$. □

Theorem 3 means that, in order to recognize whether a given graph is rigid, we have to count the number of independent bars it has, or equivalently, be able to detect how many of them are dependent. Our rigidity testing algorithm will be based upon theorem 10(D), and consists in adding edges e one at a time to a set of independent edges $\hat{E}$, and testing whether this enlarged $\hat{E}'$ is independent. If this is the case, e it is definitively added to $\hat{E}$, otherwise e is identified as a dependent edge (e is not independent of $\hat{E}$) and removed from the graph.

Adding a new edge e to $\hat{E}$ produces the graph G, and its associated bipartite graph $B(G)$. We know by theorem 10(D) that e is independent of $\hat{E}$ if and only if a complete matching from $V1$ to $V2$ exists in $B(Gab)$ when any edge ab in $\hat{E}' = e \cup \hat{E}$ is quadrupled. Fortunately only the last edge e needs be quadrupled, as the following result due to Hendrickson [13] demonstrates.

Lemma 11. Add a new edge e to an independent edge set $\hat{E}$. If a complete matching in the sense of theorem 10 exists when e is quadrupled, then $\hat{E}'$ is independent.

Proof. Assume $\hat{E}'$ is not independent. Then there must exist some edge $e' \in \hat{E}'$ whose quadrupling causes some subgraph G' of G to have 'too many edges'. This subgraph *must* include e since $\hat{E}$ is an independent set. However, this subgraph would have the same number of edges if e is quadrupled instead of e', therefore a complete matching would not be possible if e is quadrupled. □

Then in order to determine if a new edge e is independent of a set $\hat{E}$ we have to enlarge a bipartite matching to cover the four copies of the new edge e in $V1$. If this cannot be done, e is redundant. We have then an algorithm which enables us to identify redundant bars. However, we also need a means to identify rigid clusters in the system. The advantage of the body–bar representation over the original formulation in terms of joints is that each new rigid cluster which is identified may be replaced by one node, therefore reducing the size of the system. We will now see how these rigid clusters are detected. For this we need the following results [13].

Theorem 12. If three copies of a new edge e are added to an independent set $\hat{E}$ generating a graph G'', then $B(G'')$ has a complete matching from $V1$ to $V2$.

Proof. Assume there is no matching when e is tripled. Then there is some subgraph $\tilde{G}$ of G'' for which $\tilde{b} > 3\tilde{n} + 2\tilde{m}$. Remove the three copies of e and quadruple any of the other edges. This graph has the same number of edges as $\tilde{G}$ so it can also not be matched. However, this is a contradiction if $\hat{E}$ is assumed to be independent. □

Theorem 13. If the edge e is dependent and e is quadrupled, the failing Hungarian tree spans a minimal subset of edges in $\hat{E}$ which form an isostatic subgraph.

Proof. Since we added four copies of the new edge, the number of $v1$ nodes visited by the failing Hungarian tree is $b' + 4$, where b' of them belong to E. The number of $v2$ nodes visited is by construction of the bipartite graph $3n' + 2m'$. However, in a previous demonstration, we saw that when a BFS fails, it visits k_1 nodes in $V1$ and k_2 nodes in $V2$ with $k_1 = k_2 + 1$. Therefore, $b' + 4 = 3n' + 2m' + 1$, or $b' = 2m' + 3n' - 3$, which is the condition for an isostatic subgraph. This subgraph is minimal, since removing any of its edges would make the matching possible, by freeing one $v2$ node. □

The algorithm then proceeds as follows: Starting from a (possibly empty) set $\hat{E}$ of independent edges, test new edges e one by one by adding them to $\hat{E}$ and trying to enlarge an already existing matching of $B(G_e)$, which contains four copies of e in $V1$.

- If all four copies can be matched, the new edge e is independent. Remove the three extra copies and add e to the set of independent edges $\hat{E}$.
- If the matching fails for the fourth copy of e (the first three can always be matched), e is dependent and the failing Hungarian tree identifies a rigid subset of nodes. Remove all four copies.

The rigid subgraph G_a identified by a failed search may be *condensed* to a unique body A. This condensation step simply amounts to a deletion of all bars and bodies visited during the failed search, and replacing them by a unique body. Therefore the amount of work involved in one condensation is equivalent to that needed for one BFS. All bars incident to G_a are, after the condensation, incident to body A. Condensation is only possible because we are able to handle bodies, and is a key step in our algorithm.

Bars whose both ends are found to be connected to the same rigid body are not tested because they are obviously dependent. This means that the same subset of nodes is not condensed twice, so that each condensation involves at least one node which has never been condensed. Therefore there will be at most n condensations. The condensation step requires just deleting all bars and bodies in the failed Hungarian tree, therefore the total amount of work needed for condensations is at most $O(n)$, since there are at most $O(n)$ bars in $\hat{E}$, and no bar is condensed twice.

A new bar leads either to a condensation or to a new element in $\hat{E}$. Both happen at most $O(n)$ times, so that at most $O(n)$ tests are needed. Since each test involves growing four Hungarian trees, each one taking (at most) a time proportional to the number $O(n)$ of edges in $\hat{E}$, the algorithm has a worst-case time complexity of $O(n^2)$.

This theoretical bound is the same as for Hendrickson's algorithm [13]. It is not difficult to understand why the worst-case complexity of the body–bar algorithm cannot be better than $O(n^2)$. If no condensations ever occur, our algorithm is the same as Hendrickson's. However, this would only happen if there are no redundant bars, since each redundant bar identifies a rigid subgraph and leads to a condensation. We can see that while the joint–bar

algorithm has its worst-case performance in all cases in which there is long-range rigidity†, the body–bar algorithm can only be pushed towards $O(n^2)$ behaviour in the improbable case of long-range rigidity without redundancies (long-range isostatic rigidity). This situation is not frequent in physical systems with disorder, where rigidity is always redundant. Therefore we expect the body–bar version here introduced to perform much better than the original joint–bar version on practical applications such as rigidity percolation [16, 17], glasses [10, 20] or granular materials [12].

5. Implementation and performance of the algorithm

The body–bar algorithm described in this work has been recently applied [16] to study rigidity percolation on site diluted-triangular lattices. In this work we will present the comparison of performance between this algorithm and the original site-and-joint version, for bond and site dilution on triangular lattices. Bonds or sites are present on the lattice with probability p. Initially random numbers are assigned to bonds in the following manner: In the bond dilution case, each bond `ij` is assigned a random number `brn(ij)`. For site dilution, sites `i` are given a random number `srn(i)` and afterwards bonds are assigned `brn(ij) = max(srn(i),srn(j))`. In both cases, bonds are sorted in order of increasing `brn` and tested in that order. The rest of the procedure is the same for bond or site dilution. This scheme allows one to exactly detect the percolation point [16], but is not the only possibility. For example, one could fix p to a certain value and, starting from an empty system, test all bonds for which `brn` $< p$ in arbitrary order. The time-complexity of the algorithm depends on the order in which bonds are tested, although its results do not.

The graph data structure is stored using based (pointer) variables because, in contrast to the regular lattice which originates it, the multigraph has no regularity, and its size changes during the procedure, which makes static allocation of memory impractical. A new bond is tested by adding four copies of it ($V1$ nodes) to the bipartite graph and attempting to match them to the bodies (three copies of each exist on the graph) or sites (two copies of each) in $V2$. If the four copies are matched, the new bond is marked as being independent and its three additional copies are removed from the graph. If the fourth copy is not matched, the new bond is marked redundant. The set of bonds covered in the last search is in this case a rigid subgraph. This rigid subgraph is *minimal*, which means that if any of its edges is removed then the matching would be possible. This subgraph identifies then the subset of E upon which e is dependent. The concept of dependence can be recast to mean that a self-stress is possible, therefore the failing BFS identifies the set of edges of G which would be stressed if the new edge e is say, too long or too short. This feature of the algorithm is very important in, for example, rigidity percolation [16], since it provides a means to identify the stress-carrying part of a rigid cluster.

Each time a rigid subgraph is identified a routine `condensation` is called, which replaces all its elements (enclosed bars and bodies) by a single body, putting three copies of it in the graph. All external bars (bars incident to an enclosed body from a non-enclosed body) are now incident to the new body. At a practical level the only difference with Hendrickson's original algorithm is this condensation step. If the replacement of rigid objects by one node is not done, one has the original bar–joint algorithm. One must in this case [13] mark all enclosed objects with the new rigid label to avoid the need to test bonds

† By long-range rigidity we mean that $O(n)$ sites are rigidly connected, and therefore the size of most BFSs is $O(n)$, since rigidly connected subgraphs contain no free (unmatched) $v2$ nodes. This situation pushes the joint–bar algorithm towards its worst-case behaviour, $O(n^2)$, when edges are tested in random order.

both ends of which connected to the same rigid graph.

As mentioned in section 3, multiple incidence joints must be replaced by an auxiliary structure in order to have an equivalent graph which is generic. In fact it is necessary to do this only in the bond-diluted case, since for site dilution no multiple incidence points are possible. For simplicity we always use auxiliary structures for incidence points to bodies, although they are only necessary if the number of incident bars is larger than two (section 3). In this way the procedure is much simpler (otherwise the number of incident bars should be checked after each condensation or addition of a new edge) while the performance is not seriously affected.

We saw already that rigid clusters will only be detected if a bond is tested on them, that is, if a bond is found to be redundant. In other words, the algorithm naturally identifies self-stressed (hyperstatic) regions, while isostatic rigid clusters would go unnoticed if no bond is tested on them. In the study of rigidity percolation [16], one is interested in detecting the exact concentration p_c of bonds at which an isostatic rigid connection between two sides of the system first appears. This is not automatically provided by the algorithm. A possible way to do it would be as follows: after testing each edge e, test a *fictitious* bond connecting opposite sides of the sample. If the fictitious bond is independent, then these sides are not rigidly connected. Otherwise the first time that the fictitious bond is found dependent, a rigid connection is identified between the two sides of the system. This method doubles the number of tests (BFSs) needed. We now describe a better option, which allows the detection of the rigidity percolation point without extra effort [16].

Two rigid bus-bars are assumed to exist on the upper and lower edges of the sample. These are represented as two bodies B_1 and B_2, and their corresponding three copies are set in V_2. Next a fictitious bond f connecting the two bus-bars is added to the graph. A node $v_f \in V1$ represents this fictitious bond in the bipartite graph $B(G)$, and is adjacent to three copies of B_1 and B_2. This node v_f (just one copy of it) is matched to one of these nodes before starting to test any other edges. Next edges in the system are tested in order of increasing random number `rn`. The first time that an *isostatic* rigid connection exists between the bus-bars, because of the existence of this bond f that already restricts one relative degree of freedom between the busses, a dependent subgraph will be found including the fictitious bond. Thus the method to detect isostatic percolation is simply checking, at each failed matching, whether the fictitious bond v_f has been visited during the last search. If so, the last added edge e is independent (the number `rn` associated with this bond is p_c), and the subgraph visited in the failed search is exactly the *elastic backbone*†. All bars in this subgraph except f are *cutting bonds*, that is, bonds whose individual removal would produce the loss of rigidity (called *red bonds* in scalar percolation). At the percolation point the fictitious bond is removed.

Performance comparison

Both the body–bar and the bar–joint algorithms have a worst-case time complexity that scales as $O(n^2)$. However, we argued that the body–bar representation has a more convenient average-case behaviour, since searches are now done in a graph of much reduced number of elements whenever there are rigidly connected subsets of sites. In this section these differences are quantified. In order to do so we take as a test case the randomly diluted triangular lattice. Total CPU times are measured as a function of p for the bond- and site-dilution cases. Bonds are assigned a random number `rn` as described in section 5, and tested

† The *spanning cluster* is the subset of edges rigidly connected to both B_1 and B_2. The subset of it that would be stressed if a pair of forces is applied between the busses is called the *elastic backbone*.

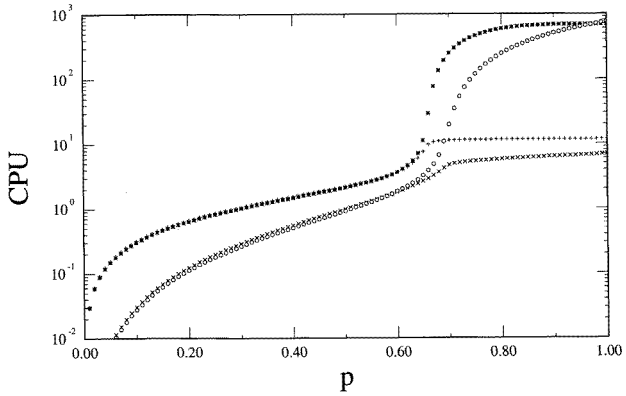


Figure 9. Total CPU times, in seconds, needed to test all present bonds of a randomly diluted triangular lattice of size $L = 128$. These measurements were done both on site- and bond-diluted lattices, using the body–bar ($\times$, site dilution; $+$, bond dilution) and the joint–bar algorithms ($\circ$, site dilution; $*$, bond dilution) on a SPARC10 workstation.

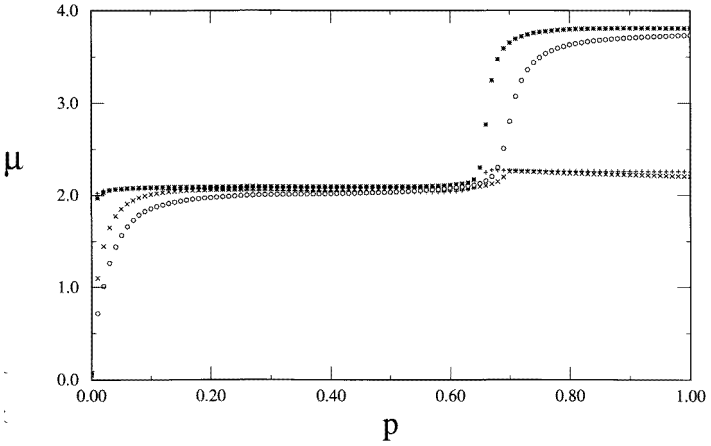


Figure 10. Graph showing that CPU times given in figure 9 scale with system size as L^{μ} with $\mu = 2 + \theta(p)$. The exponent $\theta(p)$ determines how the size of a typical search scales with size (see text). Symbols are the same as for figure 9. From the data in this plot we see that in the bar–joint algorithm, this size grows almost as fast as n in the rigid phase, while for the body–bar algorithm described in this work this size remains almost constant. This improved behaviour is due to 'condensation' of rigidly connected clusters in the body–bar algorithm.

sequentially in order of increasing `rn`. In figure 9 we see a plot of CPU times required on typical lattice of linear size $L = 128$ for both algorithms in the above mentioned cases. As expected, the body–bar algorithm is more advantageous only in the rigid phase, since the search for an exposed node takes place in a region which is typically of the size of rigidly connected clusters. Each such region is represented by one node in the body–bar case, and therefore searched over in just one step. At larger scales, further collections of bodies are found to be rigidly connected and therefore replaced by one body, so that the size of the typical searches is kept almost O(1).

In section 1 we mentioned that the time-complexity of this procedure is of order L^2

(number of bonds tested) times the size of the typical search for an augmenting path, which generally scales as L^{θ}, with θ a function of p. The overall CPU time then scales as L^{μ} with $\mu = 2 + \theta(p)$. We estimate this (p-dependent) exponent by measuring CPU times for sizes $L = 256, 128, 64$ and 32. Averages were done over 10^2 to 10^4 samples. Figure 10 shows the value of the exponent μ for the four cases under consideration, as a function of p. We see that the body–bar algorithm has a time complexity that scales approximately as $n^{1.12}$, while the bar–joint algorithm scales approximately as $n^{1.9}$, which is not much better than the theoretical worst-case limit n^2.

Acknowledgments

My interest in the problem of rigidity was sparked by P M Duxbury, and many of the ideas in this work were inspired by discussions with him. An anonymous referee provided helpful advice concerning the presentation of material in sections 1 and 2. I thank the DOE under contract DE-FG02-90ER45418 and the PRF for financial support. I also acknowledge support from the Conselho Nacional de Desenvolvimento Científico e Tecnológico, CNPq, Brazil.

References

[1] Maxwell J C 1864 On reciprocal figures and diagrams of forces *Phil. Mag.* Series 4 **27** 250–61
[2] Laman G 1970 On graphs and rigidity of plane skeletal structures *J. Eng. Math.* **4** 331-340
[3] Gluck H 1975 Almost all simply connected closed surfaces are rigid *Geometric Topology (Lecture Notes in Mathematics 438)* (Berlin: Springer) pp 225–39
[4] Asimov L and Roth B 1978 The rigidity of graphs *Trans. Am. Math. Soc.* **245** 279–88; 1979 The rigidity of graphs II *J. Math. Anal. Appl.* **68** 171–90
[5] Crapo H 1979 Structural rigidity *Struct. Topology* **1** 26–45
[6] Whiteley W 1979 Realizability of polyhedra *Struct. Topology* **1** 46–58
[7] Sugihara K 1980 On redundant bracing in plane skeletal structures *Bull. Electrotech. Lab.* **44** 376–86
[8] Tay T S 1985 Rigidity of multigraphs, I: linking rigid bodies in n-space *J. Combinat. Theory* B **36** 95–112
[9] Tay T S and Whiteley W 1985 Generating isostatic frameworks *Struct. Topology* **11** 21–69
[10] Thorpe M F 1985 Rigidity percolation *Physics of Disordered Materials (Institute for Amorphous Studies Series)* (New York: Plenum)
[11] Feng S and Sen P N 1984 Percolation on elastic networks: new exponent and threshold *Phys. Rev. Lett.* **52** 216
Feng S 1985 Percolation properties of granular elastic networks in two dimensions *Phys. Rev.* B **32** 510; 513
Day A, Tremblay R and Tremblay A-M 1986 Rigid backbone: a new geometry for percolation *Phys. Rev. Lett.* **56** 2501
Hansen A and Roux S 1989 Universality class of central-force percolation *Phys. Rev.* B **40** 749
Obukov S P 1995 First order rigidity transition in random rod networks *Phys. Rev. Lett.* **74** 4472
[12] Guyon E *et al* 1990 Non-local and non-linear problems in the mechanics of disordered systems: application to granular media and rigidity problems *Rep. Prog. Phys.* **53** 373–419
[13] Hendrickson B 1992 Conditions for unique graph realization *Siam J. Comput.* **21** 65–84
[14] Papadimitriou C H and Sleiglitz K 1982 *Combinatorial Optimization: Algorithms and Complexity* (Englewood Cliffs, NJ: Prentice-Hall) ch 10
[15] van Leeuwen J 1990 *Handbook of Theoretical and Computer Science* (Amsterdam: Elsevier) ch 10
[16] Moukarzel C and Duxbury P M 1995 Stressed backbone of random central-force systems *Phys. Rev. Lett.* **75** 4055
[17] Jacobs D J and Thorpe M F 1995 Generic rigidity percolation: the pebble game *Phys. Rev. Lett.* **75** 4051
[18] Crapo H On the generic rigidity of structures in the Plane *Adv. Appl. Mater.* to appear
[19] Franzblau D 1995 Combinatorial algorithm for a lower bound on frame rigidity *SIAM J. Disc. Math.* **8** 388–400
[20] Franzblau D 1995 Computing degrees of freedom of a molecular frame: when is greediness sufficient? *Preprint* DIMACS, Rutgers University

[21] Gabow H and Westerman H H 1988 Forests, frames and games: algorithms for matroid sums and applications *Proc. 20th Symp. on the Theory of Computing* (New York: Association for Computing Machinery) 47–421
[22] Tay T S and Whiteley W 1985 Recent advances in generic rigidity of structures *Struct. Topology* **9** 31–8
[23] White N and Whiteley W 1987 The algebraic geometry of bar-and-body frameworks *SIAM J. Algebr. Discr. Meth.* **8** 1–32
[24] Whiteley W 1988 Matroid unions and rigidity *SIAM J. Discr. Math.* **1** 237–55