Algorithms for three-dimensional rigidity analysis and a first-order percolation transition

M. V. Chubynsky[1,*] and M. F. Thorpe[2,†]

[1]*Département de Physique, Université de Montréal, Case Postale 6128, Succursale Centre-Ville, Montréal, Québec, Canada H3C 3J7*
[2]*Department of Physics, Arizona State University, P. O. Box 871504, Tempe, Arizona 85287-1504, USA*

A fast computer algorithm, the pebble game, has been used successfully to analyze the rigidity of two-dimensional (2D) elastic networks, as well as of a special class of 3D networks, the bond-bending networks, and enabled significant progress in studies of rigidity percolation on such networks. Application of the pebble game approach to general 3D networks has been hindered by the fact that the underlying mathematical theory is, strictly speaking, invalid in this case. We construct an approximate pebble game algorithm for general 3D networks, as well as a slower but exact algorithm, the relaxation algorithm, that we use for testing the new pebble game. Based on the results of these tests and additional considerations, we argue that in the particular case of randomly diluted central-force networks on bcc and fcc lattices, the pebble game is essentially exact. Using the pebble game, we observe an extremely sharp jump in the largest rigid cluster size in bond-diluted central-force networks in 3D, with the percolating cluster appearing and taking up most of the network after a single bond addition. This strongly suggests a first-order rigidity percolation transition, which is in contrast to the second-order transitions found previously for the 2D central-force and 3D bond-bending networks. While a first order rigidity transition has been observed previously for Bethe lattices and networks with "chemical order," here it is in a regular randomly diluted network. In the case of site dilution, the transition is also first order for bcc lattices, but results for fcc lattices suggest a second-order transition. Even in bond-diluted lattices, while the transition appears massively first order in the order parameter (the percolating cluster size), it is continuous in the elastic moduli. This, and the apparent nonuniversality, make this phase transition highly unusual.

I. INTRODUCTION

In this paper, we examine some important issues involved in studies of rigidity of elastic networks in three dimensions. In general, the three-dimensional (3D) rigidity problem is much more difficult computationally than its 2D analog, which has been studied extensively using an efficient topological algorithm, the pebble game. In a way, this parallels the Ising model that is much used for the theory of phase transitions, in that exact solutions are available in 2D but not in 3D [1]. There is an important difference, however, that in rigidity, exact means an algorithmic solution, rather than an analytic one.

We start by giving an extended discussion of the issues involved in 3D rigidity that make it such a challenging problem. We then develop an exact algorithm for small systems (up to several hundred or perhaps thousand sites) where a solution for geometric quantities, such as identifying the rigid clusters and the stressed regions, as well as counting the number of floppy modes, can be obtained. This serves as a benchmark for a version of the much faster pebble game algorithm that we develop, and shows that, while the latter is approximate in general, for central-force randomly diluted lattices its errors are very small indeed and so that it may be regarded as operationally exact. We then use the pebble game to study bond and site rigidity percolation on fcc and bcc lattices. The transition appears to be massively first order

in all these cases, except for site percolation on the fcc lattice where it is likely second order.

We hope our critical assessment of this important problem will be useful for mathematicians and computer scientists who are trying to develop rigidity algorithms in 3D, for physicists interested in phase transitions, and for those involved in studying the rigidity of 3D systems in the laboratory. Rigidity theory has been applied to different systems, such as covalent network glasses [2–5], proteins [6,7], and random fiber networks [8,9] with relevance to paper [10] and the cytoskeleton [11], and we hope the algorithms in this paper will broaden the scope of systems that can be studied using rigidity theory.

A. Rigidity theory

An elastic network is a set of *sites* in space and pairwise *constraints* between these sites; a constraint is a statement that the distance between a certain pair of sites has to take on a certain value, and if this value is different, there is an associated potential energy cost. Constraints can be modeled as elastic springs, although whether these springs are harmonic or not is not important. Both static and dynamic properties of such an elastic network depend in principle on the details of the potential energy function. However, there are certain properties that depend only on the geometry of the network. Among them are the number of *floppy modes* (linearly independent infinitesimal motions that do not deform any constraints and thus do not cost energy); rigid cluster decomposition (a rigid cluster is a set of sites that move together as a rigid body in any floppy motion); some aspects

*mykyta.chubynsky@umontreal.ca
†mft@asu.edu

of stress (particularly, what constraints are stressed); and some other related properties. In fact, typically such properties are determined solely by the network *topology* (i.e., what sites are connected to what sites by constraints): most networks with different geometries but the same topology have the same number of floppy modes, same rigid clusters, etc., and only an infinitesimal fraction of *nongeneric* networks that are special in some way (for instance, having some constraints that are parallel) may differ in this respect from the rest. *Rigidity theory* [12–14] deals with properties of this type. Below, we outline some aspects of rigidity theory that will be useful to us, without detailed proof or explanation; for details, see the above references. We will assume that the networks we are dealing with are always *generic*. Generically, there is also no difference between rigidity with respect to infinitesimal and small enough finite deformations, and we will assume that this is the case as well.

It is, of course, always possible and straightforward in principle to analyze rigidity properties of a network by constructing a particular realization of the potential energy function consistent with the given topology and then, assuming infinitesimal displacements and thus the harmonic approximation, diagonalizing the dynamical matrix to find the zero-frequency (floppy) modes; likewise, stressed constraints can be obtained by relaxing the network, etc. Such methods are, however, relatively slow, often scaling (particularly in the case of dynamical matrix diagonalization) as the cube of the network size; also, they are subject to inevitable round-off errors, which may, for instance, make zero eigenvalues of the dynamical matrix slightly nonzero etc. Yet methods of this type are the only choice if one needs detailed information, such as the actual values of the frequencies of all modes, the eigenvectors of the dynamical matrix, etc. (assuming, of course, that the actual potential energy function, and not just the topology, is known). But if we are interested only in those properties that depend solely on the topology, clearly such methods are somewhat "irrational": they take some irrelevant information (all force constants and detailed geometry) as input, and produce a lot of unnecessary output (such as all the eigenvalues and eigenvectors of the dynamical matrix, while we only want the number of zero eigenvalues). Thus there may be more rational methods that deal with network topology directly, never constructing a physical realization in the process. Such methods may be faster and also, since the topology information is discrete, not subject to round-off errors. Indeed, such methods have been found, as discussed below.

Historically, the first and simplest of such methods (but, unfortunately, only approximate) is *Maxwell counting* [15], due to J. C. Maxwell. Consider a *d*-dimensional elastic network of *N* sites. Dimensionality *d* here (and elsewhere in this paper) refers to the dimensionality of space in which the sites can move, rather than the dimensionality of the network itself. It is possible that some of the constraints in the network cannot be satisfied simultaneously; it is useful, just for the sake of this consideration, to change their lengths so that they fit exactly and are not strained; it turns out that this does not affect the number of floppy modes in generic networks and so can be used for the floppy mode counting that we do here (but cannot be used to find stress, of course). If the

network has no constraints, all degrees of freedom correspond to floppy modes, whose number is thus dN. Each constraint, in the linear approximation, is some linear relation between the coordinates of sites in the system. Then, when a constraint is added to the network, it reduces the dimensionality of the space of allowed infinitesimal motions by 1, if the relation between the coordinates that this constraint represents is linearly independent from the rest. If one assumes that all constraints are like this (i.e., *independent*), then the number of floppy modes is

$$F_{\mathrm{Maxw}} = dN - N_c, \tag{1}$$

where N_c is the number of constraints and F_{Maxw} denotes the number of floppy modes in the Maxwell counting approximation. Of course, this result for the number of floppy modes is only approximate, since in reality not all constraints are independent. In a generic network, nonindependent constraints are those and only those that are inserted between two sites that are already mutually rigid even before the insertion—such a constraint does not restrict floppy motions further and in this sense is *redundant*; it does not decrease the number of floppy modes. So the correct result for F is actually

$$F = dN - N_c + N_R, \tag{2}$$

where N_R is the number of redundant constraints. The Maxwell counting approximation neglects N_R; a more accurate method would require a way to evaluate it. Note that if the lengths of constraints are restored (i.e., they are no longer required to fit exactly), then generically, a redundant constraint inserted in the network becomes strained and introduces stress. Thus (again generically) stress is present if and only if there is redundancy ($N_R \neq 0$).

Since the number of floppy modes in a network of size $N \geq d$ can never be less than $d(d+1)/2$ (the number of motions of a rigid body), then it is clear that if $dN - N_c < d(d+1)/2$ for the network or for any of its subnetworks of size $N_s \geq d$, then $N_R > 0$ (i.e., there must be some redundant constraints). A more interesting question is whether the converse is also true. In other words, if $N_R > 0$, is it true that there is at least one subnetwork of size $N_s \geq d$ for which $dN_s - N_{cs} < d(d+1)/2$ (N_{cs} being the number of constraints within the subnetwork)? In 2D ($d=2$), for generic networks the answer is yes, and this statement is known as the *Laman theorem* [16]. It is also conjectured to be true for a particular class of 3D networks, known as *bond-bending* networks. A bond-bending network is defined by a set of bonds; constraints then connect all first neighbors (*bond-stretching* or *central-force* constraints) and all second neighbors (*bond-bending* or *angular* constraints). The conjecture is a part of what is known as the *molecular framework conjecture* [14,17,18]; the origin of this name is the fact that bond-bending networks are a natural model for covalent molecules (as well as covalent disordered solids), since covalent bonds have strong bond-stretching and bond-bending interactions associated with them, and all other interactions are usually weaker. This conjecture is not proved rigorously; however, no counterexamples have been found in more than 20 years since its for-

mulation. For general (non-bond-bending) 3D networks there are violations of the molecular framework conjecture, as discussed in detail below. Note that for bond-bending networks, we are making a careful distinction between bonds and constraints: the set of bonds specifies what sites are considered first neighbors (those that are connected by a bond), and then constraints connect both first and second neighbors. On the other hand, in central-force networks that we consider in Sec. VI, there is one constraint per bond and we use "bonds" and "constraints" interchangeably.

The Laman theorem in 2D and the molecular framework conjecture for 3D bond-bending networks enable a convenient and fast approach to finding N_R exactly. Start with the "empty" network (all sites present but no constraints). Such a network obviously has $N_R=0$. Now, add constraints one by one, checking each of them for redundancy by testing all subnetworks to which the newly added constraint belongs. If the constraint is redundant, N_R is increased by 1; otherwise it is unchanged. Thus, N_R is known at all times during the network construction process. One very important caveat in the case of 3D bond-bending networks is that even if the final network being analyzed is bond bending, this is not necessarily true for the intermediate networks obtained during the construction process (and, in fact, cannot in general be true for all of them). For this reason, it is important to keep the networks as close to being bond bending as possible. That is, a constraint coinciding with a bond (a first-neighbor constraint) should always be inserted first, and all second-neighbor constraints induced by the just inserted first-neighbor constraint (i.e., those second-neighbor constraints that span the angles formed by the just inserted first-neighbor constraint with previously inserted first-neighbor constraints) should follow immediately, before any other first- or second-neighbor constraints are inserted. While this does not keep the network strictly bond bending at all times, deviations are as small as possible, and it is assumed (as a part of the molecular framework conjecture) that all intermediate networks obey the statement of the conjecture as well.

Another issue is rigid cluster decomposition. Rigid clusters in 2D and in 3D bond-bending networks have a useful property: they are always *rigid by themselves*, i.e., they remain rigid when separated from the rest of the network. A corollary of this is that rigid clusters in such networks are always *contiguous*: when moving from any site of the cluster to any other site belonging to it along the network constraints, it is always possible to choose a path such that only sites belonging to the same cluster are passed. Simply stated, rigid clusters always "come in one piece." These properties mean that, first of all, a rigid cluster of size $n \geq d$ with all redundant constraints removed will always have exactly $dn - d(d+1)/2$ constraints within itself; also, contiguity allows easy cluster mapping, by starting with an arbitrary group of d mutually rigid sites and then moving outward until the region that has emerged is fully surrounded by sites not rigid with respect to at least one of the d initial sites.

Regarding stress determination, an important fact is that in 2D and in 3D bond-bending networks, the set of all stressed constraints can be represented as the union of stressed regions, each of which is stressed by itself, i.e., remains stressed when separated from the rest of the network,

and has the property that it is possible to find a set of sites such that all constraints connecting sites within the set belong to the region and all constraints connecting sites at least one of which does not belong to the set do not belong to the region. In other words, using graph theory terminology, each stressed region is an *induced subgraph* of the graph whose vertices are the network sites and whose edges are the constraints.

An algorithm using the above ideas, known as the *pebble game*, was proposed first for 2D networks [19–21] and then for 3D bond-bending networks [21–23]. The idea is to relate the constraints to the degrees of freedom for all subnetworks simultaneously, by assigning pebbles to degrees of freedom and then matching those pebbles to constraints. The details of the 3D version of the algorithm are described in the next subsection.

From the description of the pebble game, it will become clear that it relies significantly on the three special properties of 2D and 3D bond-bending networks stated above: the molecular framework conjecture; the contiguity of rigid clusters and their being rigid by themselves; and stressed regions being induced subgraphs. In Sec. II we demonstrate that, unfortunately, all of these properties are violated in general in 3D networks that are not fully bond bending. Creation of a pebble-game-type algorithm that does not rely on these properties is problematic and a way to do this has not been found to date. While partial fixes (covering some but not all possible situations) are possible, in our generalization of the pebble game that we introduce in Sec. III, we choose to ignore these problems completely. This, of course, makes the algorithm only approximate, and to estimate the accuracy of the new pebble game, an exact but slower algorithm, the *relaxation algorithm*, is introduced in Sec. IV. Some general considerations on the accuracy of the pebble game and how this accuracy can be estimated using the relaxation algorithm are given in Sec. V. In Sec. VI, we look at one application of our algorithms. We consider central-force networks (i.e., those that have only first-neighbor constraints) obtained by randomly removing bonds (bond-diluted networks) or sites (site-diluted networks) from regular three-dimensional lattices. In this case, we show that the pebble game is essentially exact for the most interesting quantities, such as the number of floppy modes and the size of the largest rigid cluster, although some very small clusters may be misidentified. On the other hand, the pebble game is much less successful in some other cases, as we show in Sec. VII. We use the success of the pebble game for randomly diluted central-force networks to study *rigidity percolation* on such networks. An introduction to the subject of rigidity percolation is given in Sec. I C, after the description of the old pebble game algorithm.

B. The pebble game algorithm for bond-bending networks

The 3D bond-bending version of the pebble game algorithm is as follows [22–24]. Starting with the empty network without constraints, three pebbles are assigned to each site, so that the total number of pebbles is equal to the total number of degrees of freedom, $3N$. A pebble can be free, or it can

cover one of the constraints associated with the site to which the pebble belongs. Initially, there are no constraints, so all pebbles are free. As constraints are added to the network, all independent (nonredundant) constraints (detected as described below) are covered by a pebble from either side and should remain covered at all times during the pebble game process. Since the number of independent constraints is $N_c - N_R$, then, according to Eq. (2), the number of free pebbles is equal to the number of floppy modes. A constraint can be covered at either end, and this allows freeing of pebbles. A pebble covering a constraint can be freed, if there is a free pebble available at the other end of the constraint; then that free pebble covers the constraint and the pebble covering it is released; this process may have to be repeated several times, if a pebble is free not at the end of the constraint, but at one of its neighbors, neighbors of neighbors, etc. As a consequence, checking whether freeing a pebble at a given site is possible starts at that site, then looks at what constraints the pebbles belonging to the site cover, and checks the site's neighbors connected to it by those constraints; if no free pebbles are found there, the procedure is repeated until a free pebble is found or until no unchecked sites connected to checked sites by constraints covered by pebbles belonging to checked sites remain. If the search for a pebble has failed, the region over which the search has proceeded (the failed pebble search region) is recorded, which is important for finding stressed regions, as described below.

Each newly inserted constraint is tested for independence in the following way. First, all three pebbles need to be freed at each of the two ends of the new constraint—this is always possible, unless the new constraint coincides with a previously inserted one, in which case the new constraint is obviously redundant and should not be tested. Then, with all six pebbles kept free, an attempt is made to free one more pebble at each neighbor of the ends of the new constraint in turn. In fact, even fewer checks are needed: for a bond-stretching (first-neighbor) constraint, just first neighbors of one end need to be checked (second neighbors need not be checked, even though they are connected with a second-neighbor constraint); for a bond-bending (second-neighbor) constraint, only the vertex of the angle that the constraint spans needs to be checked. If all of these attempts are successful, then the new constraint is independent and should be covered by one of the six pebbles at its ends. Otherwise, the constraint is redundant and should not be covered. As a reminder, constraints should be inserted in a particular order: a first-neighbor constraint is inserted first and then all second-neighbor constraints induced by it should be inserted immediately afterward, before another first-neighbor constraint is inserted. Just as the very similar 2D algorithm relies on the Laman theorem, as explained in detail in Ref. [20], the above procedure assumes the validity of its generalization, the molecular framework conjecture.

Whenever a redundant constraint is inserted, it will create additional stress in the network and may increase the set of constraints that are stressed. The redundancy is detected when a pebble search fails, and the region of the failed pebble search should be merged with such regions found previously to find the part of the network that is stressed. Once one failure to find the pebble is detected, there is no need to continue further checks in order to find the stressed region—the failed search regions will coincide for all neighbors for which the search fails. Regions of failed pebble search are defined as sets of sites passed when searching for a pebble, and all constraints connecting such sites are stressed. Of course, this implies an important property of bond-bending networks that we have already mentioned— that any stressed region is always an induced subgraph of the graph whose vertices are the sites and whose edges are the constraints.

During the pebble game, whenever a large stressed region is detected, the *tetrahedralization* procedure [24] (similar to the triangularization procedure in 2D [20,25]) is commonly used to convert the stressed region into an isostatic (rigid but unstressed) one. This speeds up further pebble searches significantly. We do not consider this procedure here and do not implement it in our treatment of general (non-bond-bending) 3D networks. This limits the network sizes we can routinely consider to perhaps 10^5 sites or so, while with tetrahedralization 10^6-site or even larger networks could be considered, so the implementation of this procedure in the future is desirable.

Once the network construction is finished, the information about the number of floppy modes and the stressed constraints is available. The next stage is rigid cluster decomposition. One thing to keep in mind is that, unlike in the usual *connectivity* percolation, a site can belong to several clusters simultaneously (imagine, for instance, two rigid objects sharing a common point, a *pivot*, or a common axis, a *hinge*); but choosing three mutually rigid sites identifies a cluster uniquely: any three chosen sites can belong simultaneously to at most one cluster. Bond-bending networks are special: among all clusters to which a given site belongs, there is always one and only one to which all of its neighbors also belong. For this reason, for a bond-bending network, rigid cluster decomposition can be given by specifying for each site the unique cluster to which this site belongs with all of its neighbors. Given the above, it is convenient to start mapping a rigid cluster by choosing a site and two of its first neighbors. Such three sites are always mutually rigid (indeed, they form an angle and bond angles are constrained in bond-bending networks) and thus specify a rigid cluster. A maximum number of pebbles are freed at the three chosen sites. It is always possible to free exactly six pebbles. After this, neighbors of the three chosen sites are picked in turn, and for each of such sites an attempt is made to free a pebble while keeping the six freed pebbles at the first three sites free. If freeing an extra pebble fails, the site being tested is rigid with respect to the first three; moreover, this is true for the whole region of the failed search. But if freeing an extra pebble succeeds, then the site is not rigid with respect to the first three sites. After all neighbors of the three initial sites are checked and if at least some are found rigid with respect to the initial sites, neighbors of the neighbors found rigid are checked, etc. The process continues until no unchecked neighbors of the sites deemed rigid with respect to the initial three sites remain. At this point mapping of the cluster is complete, since all clusters are contiguous. All sites such that all neighbors of them are found rigid with respect to the original three sites are assigned identical labels to specify the

cluster. Then another site is chosen among those that are not yet labeled, together with its two neighbors, and mapping another cluster starts. This continues until all sites are labeled, at which point rigid cluster decomposition is complete. Note that this procedure uses the contiguity of rigid clusters, but also, less explicitly, their being rigid by themselves, since a region of the network not rigid by itself may contain more than six free pebbles, and thus extra pebbles besides the six freed at the first three sites may be found.

After the rigid cluster decomposition process described above, for each bond in the network its end sites can have either identical or different labels. In the latter case, both ends of the bond are shared between the same two rigid clusters and such a bond is a hinge. It is impossible for two clusters in a bond-bending network to share just a single site, so pivots cannot exist in such networks [22]. Only first-neighbor constraints coinciding with bonds can be hinges; this can never happen to second-neighbor constraints. It is also impossible to have a hinge that does not coincide with any constraint (such hinges are known as *implied*). A hinge can be shared between at most two clusters. These limitations do not apply to non-bond-bending networks.

To avoid confusion, we should note that there is also a different variant of the 3D pebble game for bond-bending networks based on an equivalent representation of such networks as *body-bar* networks [3,22]. It is this variant that is used, for instance, in the FIRST software for protein rigidity analysis [6,7,21,26]. However, as the body-bar representation does not apply to non-bond-bending networks, the corresponding variant of the pebble game algorithm does not apply directly to such networks, and we do not consider it here.

C. Rigidity percolation

The concept of rigidity percolation was first introduced by Thorpe [2] in the context of covalent network glasses; it was subsequently studied soon after in more detail for central force networks by Feng and Sen [27] and by Feng, Thorpe, and Garboczi [28]. In network glasses, covalent bonds connecting atoms are strongly directional, meaning that there is a strong restoring force associated with changing both bond lengths and bond angles. At the same time, all other interactions are much weaker. For this reason, covalent glasses can be modeled as 3D bond-bending networks, for which the pebble game is exact. If the number of bonds per site is increased (in practice, by changing the chemical composition), the networks go from overall floppy (the *floppy phase*) to overall rigid (the *rigid phase*) as the rigidity percolation threshold is crossed, at which point a percolating rigid cluster emerges. The fraction of sites in the percolating cluster (which serves as the order parameter for the rigidity percolation transition) grows continuously, starting from zero at the threshold, and thus the transition is said to be continuous, or second order [3,5,29] (Fig. 1). One can also look at *stress percolation*—whether the set of bonds that are stressed percolates. The stress percolation in random networks occurs at the same point as rigidity percolation and the behavior is similar (Fig. 1), although there are models in which network self-organization leads to these thresholds being different
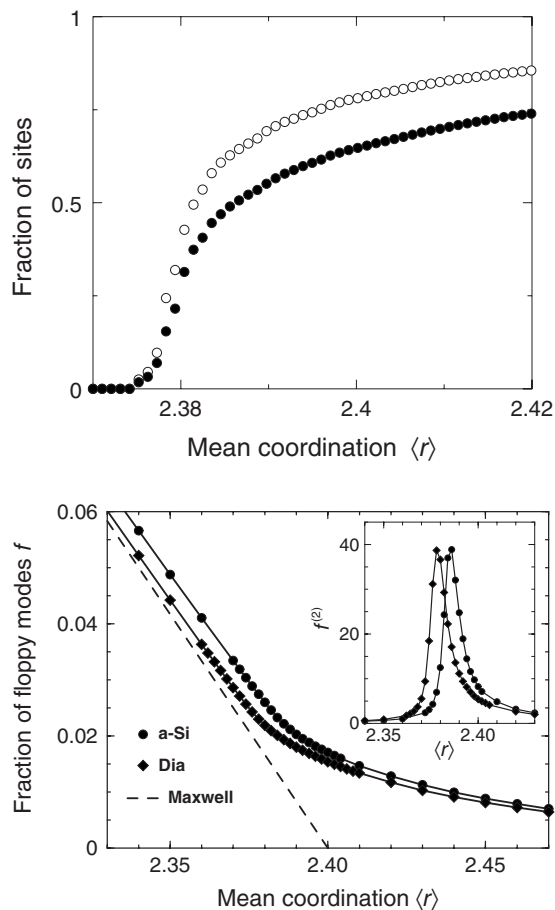

FIG. 1. Bond-bending networks in 3D. (Top) The fractions of sites belonging to the percolating rigid cluster (open circles) and the percolating stressed region (filled circles) as functions of mean coordination $\langle r \rangle$ for the case of random bond dilution of the diamond lattice. The results are averages over 11 networks of 125 000 sites each. Rounding near the transition is due to finite-size effects. (Bottom) The number of floppy modes per degree of freedom $f = F/3N$ for randomly bond-diluted amorphous silicon networks (circles) and diamond lattices (diamonds). The dashed line is the Maxwell counting result. The inset shows the second derivative of f with respect to $\langle r \rangle$. The upper panel is from Ref. [36]; the lower panel is adapted from Ref. [3].

[4,5,29–33]. Likewise, rigidity percolation in 2D central-force networks was considered (2D bond-bending percolation is equivalent to the usual connectivity percolation). The result is also a second-order transition [19,25]. But there are known cases in which the transition is first order, i.e., the fraction of sites in the percolating cluster jumps from zero to a nonzero value at the transition. This has been found for "pathological" Bethe lattices or random bond networks [3,34,35] (Fig. 2) and networks with chemical order [36], but up to now, there have been no cases where it would be observed for a regular randomly diluted network.

Besides the size of the percolating cluster, the order of the rigidity percolation transition can be found by looking at the behavior of the number of floppy modes F as a function of *mean coordination* $\langle r \rangle$ (the average number of bonds connecting a site to other sites). It has been suggested [35] that

FIG. 3. Example of a network (the double-banana graph) for which the generalization of the Laman theorem fails. The dashed line is a hinge around which the two bananas can rotate.
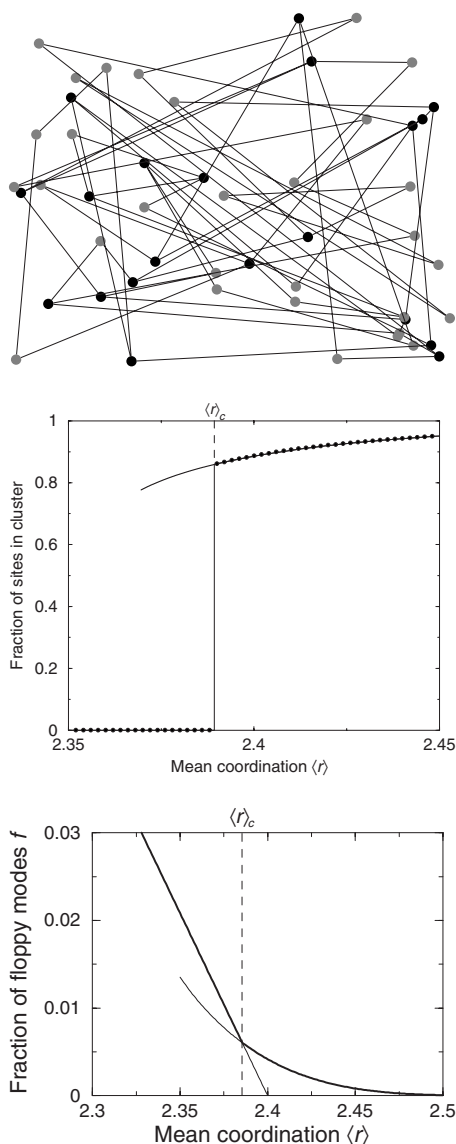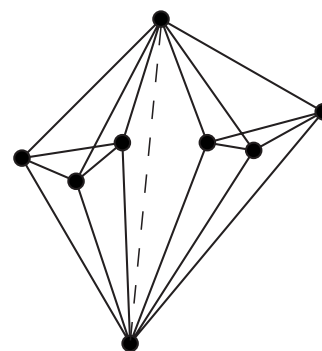
FIG. 2. (Top) Sketch of a random bond network: sites are connected at random, regardless of the distances. This network consists of two- and three-coordinated sites (gray and black, respectively). (Middle) The fraction of sites in the percolating rigid cluster as a function of $\langle r \rangle$ for a bond-bending random bond network consisting of two- and three-coordinated sites in 3D. The solid line is theoretical, the circles are the result of pebble game simulations. The transition occurs at $\langle r \rangle_c$. (Bottom) The theoretical number of floppy modes per degree of freedom, $f = F/3N$, for a bond-bending random bond network consisting of two- and three-coordinated sites in 3D. Note the break in the slope at the transition. These panels are adapted from Refs. [3,37].

$-F$ serves as an analog of the free energy of the system. When a system goes through a phase transition, the free energy is continuous, but in a first-order transition, its first derivative is discontinuous; in a second-order transition, the first derivative is continuous, but the second derivative shows a singularity. Indeed, in the case of a regular 3D bond-bending network (Fig. 1), the number of floppy modes is a continuous and smooth function of $\langle r \rangle$, but there is a cusp in the second derivative, which is consistent with a second-order transition; on the other hand, for the random bond network, where the transition is first order, there is a break in the first derivative (see the lower panel of Fig. 2).

There have also been some studies of 3D central-force elastic networks [28,38,39]. However, the networks used by Feng *et al.* [28] and Garboczi and Thorpe [38] were not large enough to draw definite conclusions about the nature of the rigidity transition. While Arbabi and Sahimi [39] used larger networks, they considered only physical properties like elastic moduli and force distributions; they were not concerned with geometric quantities, such as sizes of rigid clusters. Lack of a fast pebble-game-type algorithm made studies of 3D central-force rigidity extremely difficult. Since it was known that the pebble game is not exact for 3D networks that are not bond bending, it was assumed that the errors would make any applications of the pebble game approach unreliable. In Sec. VI, we show that this is not the case and then use the pebble game to study the rigidity percolation transition in both bond-diluted and site-diluted networks in 3D.

II. NON-BOND-BENDING 3D NETWORKS

For non-bond-bending networks in 3D, unfortunately, the molecular framework conjecture and other statements crucial for the application of pebble-game-type algorithms are not true in general. In this section we present a few known counterexamples. Some of these were published before (see, e.g., Ref. [22]).

Figure 3 shows an example of a network for which the generalization of the Laman theorem fails. This is the infamous *double-banana graph* [12]. For all subnetworks with $N_s \geq d = 3$, $dN_s - N_{cs} \geq d(d+1)/2$ and so there should be no redundant constraints, $N_R = 0$. Since there are $N = 8$ sites and $N_c = 18$ constraints, there should be $F = 3 \times 8 - 18 = 6$ floppy modes—exactly the number that a rigid body has (three translations and three rotations), so the network should be rigid. It is obvious that this is not the case, as the two "bananas" can rotate around the axis they share. Thus, there is one internal floppy mode in addition to the six rigid body motions, so $F = 7$ and then, according to Eq. (2), $N_R = 1$—there is one redundant constraint.
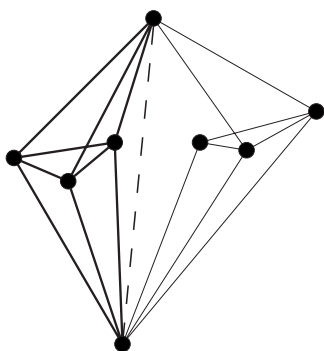
FIG. 4. Example of a network with a rigid cluster (shown with thinner lines) that is no longer rigid when taken in isolation from the rest of the network. The dashed line is a hinge.

Likewise, rigid clusters are no longer necessarily rigid by themselves or even contiguous. Figure 4 shows the same network as in Fig. 3, except one constraint is missing. There is still one internal floppy mode, as in Fig. 3, but as N_c is less by 1, $N_R = 0$, which agrees with the generalization of the Laman theorem. Yet the part of the network shown with thin lines is a rigid cluster, despite not being rigid by itself, when detached from the rest of the network that rigidifies it. A straightforward application of the rigid cluster decomposition procedure described above may fail to detect this rigid cluster. Figure 5 shows an even more extreme example of a *noncontiguous* rigid cluster [22]. The three bananas in the figure are normal, contiguous rigid clusters. But, in addition to that, the sites marked 1, 2, and 3 also form a rigid cluster, being mutually rigid with no other sites in the network rigid with respect to all three. This cluster is, of course, noncontiguous, and there is no way a rigid cluster decomposition procedure similar to the one described above can detect it, as it marks clusters in a contiguous fashion.

There is one common feature in the networks shown in Figs. 3–5. In all three cases, there are implied hinges (shown in the first two figures with dashed lines). As a reminder, a hinge is a straight line that for rigid clusters sharing exactly two sites goes through these two sites; it is the axis of rotation around which the clusters can rotate with respect to each other. In bond-bending networks, all hinges coincide with constraints (i.e., are *explicit*), as mentioned above, but in general, this is not necessarily the case, as Figs. 3–5 illustrate. In Figs. 3 and 4, there are two rigid clusters, and they
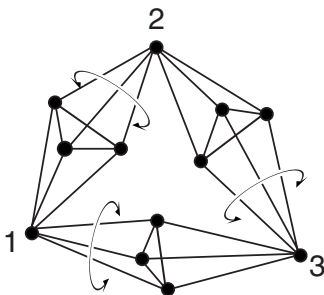

FIG. 5. Example of a network with a noncontiguous rigid cluster consisting of sites marked 1, 2, and 3.

share an implied hinge. In Fig. 5, there are three implied hinges, each shared by the noncontiguous cluster 1-2-3 with one of the three bananas.

In fact, it turns out that problems with obeying the generalization of the Laman theorem (or the molecular framework conjecture), as well as with contiguity of rigid clusters and their being rigid by themselves, are always due to implied hinges. A network not having implied hinges has no such problems, and if implied hinges are placed explicitly as constraints, the problems are eliminated as well. This can be checked explicitly for the networks in Figs. 3–5. If the hinge is placed explicitly as a constraint in Fig. 3, there are now $N_c = 19$ constraints, and the condition $dN - N_c \geq d(d+1)/2$ is now violated, so the redundant constraint is now predicted correctly to exist. When the hinge is placed explicitly in Fig. 4, it becomes part of the rigid cluster drawn with thin lines, and this cluster then becomes rigid by itself. Finally, in Fig. 5, once the hinges are placed explicitly, the rigid cluster 1-2-3 becomes contiguous. The general statement that all problems with floppy mode counting and rigid cluster decomposition are due to implied hinges is related to the so-called *Dress conjecture* [40] in rigidity theory. The Dress conjecture actually gives the exact count of the number of floppy modes once all implied hinges are identified. But since the implied hinges still need to be found first, unfortunately, unlike the molecular framework conjecture, the Dress conjecture does not give rise to a straightforward approach to floppy mode counting or finding rigid clusters, and at this time, there is no topological algorithm of the pebble game type that would do that. Of course, there can be much more complicated cases than those shown in Figs. 3–5—whole hierarchies of bananas within bananas within bananas—and a way to take all such cases into account has not been found to date.

There is also a complication related to stress determination. The pebble game finds the set of stressed constraints as the union of sets of constraints in regions of failed pebble search. Each such failed pebble search region is by construction an induced subgraph, since all constraints connecting sites through which the failed pebble search has proceeded are assumed to belong to the set. Moreover, for the pebble search to fail over some region, this region needs to be stressed by itself, when isolated from the rest of the network. This is because all other stress-inducing redundant (i.e., not covered by a pebble) constraints, besides the one being tested, are irrelevant to how the pebble search proceeds, and this includes any redundant constraints outside the failed search region. But in non-bond-bending networks the full set of stressed constraints may not even be representable as the union of sets of edges of induced subgraphs stressed by themselves. Consider the network consisting of two bananas with one bridging constraint between them, as in Fig. 6. All constraints are stressed, with the exception of the bridge (a thinner line in the figure). Note that any subset of the set of stressed constraints (other than the full set) cannot be considered a stressed region, because it would not be stressed in isolation. For instance, when the two bananas are separated from each other, each banana separately will not remain stressed. So the whole set of stressed constraints is a single stressed region here. But this stressed region is not an in-
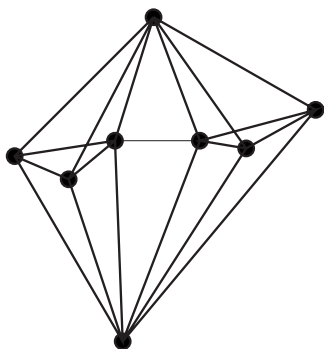
FIG. 6. Example of a network in which the stressed region is not an induced subgraph, since the thin constraint is the only one that is not stressed.

duced subgraph: it is impossible to find a set of sites such that all constraints but the bridge connect the sites in the set, but at least one of the bridge ends is outside the set. A pebble game procedure based on failed pebble search regions will not be able to identify this stressed region correctly. Note that, in this case, there are no implied hinges. So whereas implied hinges are the reason for all problems with floppy mode counting and rigid cluster decomposition, this is not so for stress. But note also that if the "problematic" bridge constraint is removed, the hinge *will* appear. It can be argued that, whenever there are problems in the pebble game determination of stress, they are either due to implied hinges (or uncovered explicit hinges—see Sec. V), or such a hinge would appear if one constraint were removed.

III. THE PEBBLE GAME FOR NON-BOND-BENDING NETWORKS

The lack of an exact pebble-game-type algorithm for general 3D networks has been a significant impediment to studying such networks. In Sec. I, we have seen that there are many properties of bond-bending networks that are useful for the pebble game algorithm and that do not hold in general for networks that are not bond bending. Violation of some of these properties, such as the possibility to specify rigid cluster decomposition by labeling sites, is a mere inconvenience. Crucial, however, is the absence of implied hinges in bond-bending networks and their presence in general non-bond-bending networks, and, as a consequence, the violation of the molecular framework conjecture, of the properties of contiguity of rigid clusters and their being rigid by themselves, as well as of the induced subgraph property of stressed regions. These properties are essential for the pebble game approach and it is not known how to avoid using them in a pebble-game-type algorithm.

But even though we know that in some cases application of the pebble game approach would be wrong, a reasonable question to ask is just how wrong such an algorithm would be in various cases of interest. In other words, are the errors frequent and significant or are they rare and negligible? In much of the remainder of the paper, we will try to answer this question. In this section, we describe the pebble game

algorithm we are going to use, which is mostly a straightforward generalization of the algorithm for bond-bending networks described above, except that it does not rely on some of those properties of bond-bending networks that no longer hold for non-bond-bending ones. This algorithm needs to be compared with the exact result, and for this reason in the next section we introduce a "physics-based" approach, the relaxation algorithm. It has all the disadvantages mentioned before, such as slowness and round-off errors (although it is likely faster than straightforward diagonalization; see the discussion at the end of Sec. IV)—but if the latter are brought under control, the approach is potentially exact and can be used for testing the pebble game.

The first part of the pebble game algorithm, in which the redundant constraints are counted and stress is detected, is very similar to the bond-bending case, but some details differ. Just as for bond-bending networks, a constraint is tested for independence by first freeing six pebbles at its ends and then attempting to free an extra pebble at the neighbors of an end of the constraint in turn. Obviously, since the subdivision into first- and second-neighbor constraints is no longer present, a particular order in which constraints are inserted can no longer be enforced—although for networks with partially bond-bending character (for instance, a bond-bending network with some angular constraints missing), following the order (a first-neighbor constraint inserted first and all associated second-neighbor constraints immediately afterward) as much as possible could be beneficial and reduce errors (but see a counterexample to this in Sec. VII). Another difference is that, again, for the same reason that there is no longer a strict subdivision into different types of constraints, *all* neighbors of at least one of the ends of the constraint being tested should be checked by trying to free a pebble. But it is still unnecessary to check the neighbors of *both* ends—this basically has to do with the fact that any subnetwork of more than two sites containing two given sites and rigid by itself (thus having only six associated free pebbles) always includes at least some neighbors of both of these sites.

There is an important difference regarding the stressed region determination. It is no longer true that all regions of failed pebble search for each of the neighbors of an end of the constraint being tested are going to coincide. So, even when failure to free an extra pebble is detected for one of the neighbors, the procedure should still be repeated for all of the other neighbors and the *intersection* (not the union) of the regions of failed search is the new stressed region. Of course, we should remember that even this more complicated procedure is not completely error-free: for instance, we still assume that stressed regions are induced subgraphs, but, as explained in the previous section, this is not necessarily true. The justification for the fact that the intersection of the failed search regions should be taken is as follows. Imagine a network consisting of just those constraints that are covered by a pebble. As only independent constraints are covered, such a network will have no redundancy and thus no stress. When a new constraint is inserted, a stressed region appears if this constraint is redundant. Any constraint from such a region can be removed without changing rigidity but making the stressed region unstressed, while any constraint from outside
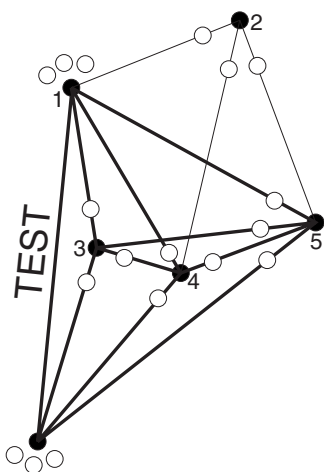
FIG. 7. Example of a network for which different failed pebble search regions do not coincide and the intersection of the regions needs to be taken to identify the stressed region correctly. The constraint marked "TEST" is being inserted. Six pebbles are freed at its ends (shown). No other free pebbles are present, so search for the seventh free pebble cannot succeed. An attempt is made to free the seventh pebble at each neighbor of site 1. When this is done at the site marked 2, sites 3, 4, and 5 are all passed when searching for a pebble. But when this is done at one of the sites marked 3, 4, and 5, site 2 is not passed, since the constraints leading to this site are not covered by pebbles belonging to either of the sites 3, 4, and 5. Clearly, only thick constraints are stressed, and thus the stressed subgraph should not include site 2, so the intersection of failed search regions needs to be taken to identify the stressed region correctly.

the region will add one floppy mode but the stress will remain. So removing a constraint from inside the stressed region and freeing the associated pebble should make this pebble available to every neighbor of the ends of the newly inserted constraint (as this constraint should now become independent)—in which case the site to which the pebble belongs is part of all pebble search regions; conversely, the pebble from any constraint from outside the region should not become available to at least one of the neighbors—and then the site to which the pebble belongs is not part of at least one of the pebble search regions. An example of a network where failed search regions differ and clearly an intersection of those regions needs to be taken is shown in Fig. 7. Of course, if implied hinges are present or would appear upon removal of a single constraint, this procedure may not work correctly, as for the examples in Figs. 3 and 6.

Regarding rigid cluster decomposition, one serious issue is choosing a starting set of three mutually rigid sites for each cluster. In the bond-bending case, we started with a triple consisting of an arbitrary site (having at least two neighbors) and two of its first neighbors knowing that they always form a mutually rigid set. Six free pebbles were then collected at these three sites. In the general case, unfortunately, not every angle is rigid, i.e., not every triple consisting of a site and its two neighbors is a mutually rigid set. If we still do the same, i.e., choose an angle and collect as many pebbles as possible at the three sites forming it, then
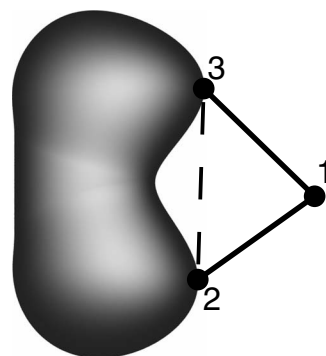

FIG. 8. Sketch of the simplest ("trivial") implied hinge. Constraints 1-2 and 1-3 are present, but there is no constraint between sites 2 and 3. A body on the left denotes a rigid cluster that is rigid by itself. The triple 1-2-3 is also a rigid cluster, but is not rigid by itself. Line 2-3 is an implied hinge. In many cases, this is the most frequent situation involving an implied hinge. It will not cause errors in the floppy mode count or stress determination, but the cluster 1-2-3 may be missed by the rigid cluster decomposition procedure.

sometimes it may be possible to collect more than six pebbles. It may be so because the set is not mutually rigid; but it may also be so when the set is mutually rigid, but only because the rigid cluster that it is a part of is rigidified from outside. So, if we rely on the number of freed pebbles to determine if the angle is rigid, we may fail to identify some of the rigid clusters. The simplest example is in Fig. 8. The three explicitly marked sites in this figure, 1, 2, and 3, form a rigid cluster, but since it is not rigid by itself, it will always be possible to collect seven pebbles at these sites, and thus this cluster will be missed. In this particular case, it is easy to detect the error: if one inserts a constraint between sites 2 and 3, this constraint is redundant, and so sites 2 and 3 are mutually rigid and then all three sites are mutually rigid. However, even if the failure is detected in such a way (and it is not always possible), it is not obvious in general how to proceed from there. Do we keep all seven pebbles free? Do we free only six of the pebbles? Both of these choices are potentially problematic. So we have chosen to limit ourselves to the test based on the number of freed pebbles. That is, we choose an angle and try to free as many pebbles as possible at the first site and then at both of its neighbors keeping the previously freed pebbles free. If only six pebbles are freed, the angle is deemed rigid and we proceed exactly as in the bond-bending case. Otherwise, the angle is deemed nonrigid and another angle is chosen instead. This means that we are going to miss rigid clusters such as 1-2-3 in Fig. 8, which, however, is of minor importance for most purposes. Some other clusters can be missed, too, but, at least in the examples we consider later in this paper, this is much more rare. Note that this problem can still only appear due to implied hinges (or uncovered explicit hinges—see Sec. V), as we show when we analyze possible errors of the pebble game in more detail in Sec. V. We should note that it is also possible to err on the other side, i.e., to misidentify a nonrigid region as rigid, for instance, in the double-banana case (Fig. 3), where the whole network is deemed rigid by the pebble game, but there is, in fact, a hinge. Once a rigid angle

is found, the associated rigid cluster is mapped as in the bond-bending case. Then another rigid angle is chosen, and so on, until all rigid angles are assigned to clusters.

IV. THE RELAXATION ALGORITHM FOR EXACT RIGIDITY ANALYSIS

In order to test the accuracy of the pebble game algorithm described in the previous section, we need a way to do exact rigidity analysis. In this section, we describe one possible method, which we call the relaxation algorithm. Like more straightforward methods, such as numerical diagonalization of the dynamical matrix or singular value decomposition (SVD) of the rigidity matrix [41], our approach is not an integer algorithm and thus potentially subject to round-off errors (although the method incorporates several consistency checks which make any errors in the final result unlikely). Unlike such straightforward methods, the relaxation algorithm also relies on some facts from rigidity theory, in particular, the Dress conjecture is used to find the number of floppy modes. Also, unlike for the diagonalization and SVD procedures and similar to the pebble game, the actual eigenmodes (including the eigenvectors corresponding to the floppy motions) are not obtained; on the other hand, rigid cluster decomposition is easier to obtain using the relaxation algorithm. Based on this, it can be said that the relaxation algorithm is "intermediate" between the straightforward approaches and the pebble game.

Suppose we are given a network topology for which rigidity properties need to be obtained. Consider a particular realization of that topology, i.e., an elastic network (modeled as a network of harmonic springs) with specified equilibrium positions of sites and whose connectivity is consistent with the given topology. For the first stage in the relaxation algorithm, used to obtain the rigid cluster decomposition and the number of floppy modes, assume that the natural lengths of the springs are chosen to fit exactly between the sites at specified positions, so that initially the network is in equilibrium and unstrained, thus being at the energy minimum. Now, displace all sites by infinitesimal amounts in random directions. In general, the network will no longer be in equilibrium. If the network is now relaxed using, for instance, the conjugate gradient algorithm [42], then after the relaxation is complete, the network is again in equilibrium. However, generally speaking, the positions of sites will not coincide with the initial equilibrium positions. This is because the equilibrium is not unique: any displacement from the initial equilibrium corresponding to a floppy motion will leave the energy unchanged and thus still equal to its minimum value of zero. Thus we can expect the final configuration (after relaxation) to differ from the initial one (before displacing the sites) by a $3N$-dimensional vector that belongs to the space of floppy motions. Since floppy motions do not change distances between mutually rigid sites, then for any pair of mutually rigid sites, the final distances will be the same as the initial distances. On the other hand, since the displacement was chosen at random, it is very unlikely that for a pair of sites that are not mutually rigid, the distances will be the same at the beginning and at the end. Thus the procedure

described above allows finding all mutually rigid pairs of sites. All displacements have to be sufficiently small (ideally, infinitesimal): otherwise, the system can jump from one local minimum to another. If initial displacements are infinitesimal, they will remain infinitesimal during and after relaxation.

If the initial position of site i is $\mathbf{r}_i$ and the infinitesimal displacement from that position is $\mathbf{u}_i$, then to the lowest order in $\{\mathbf{u}_i\}$, the change in the distance between sites i and j is

$$\delta r_{ij} = \frac{(\mathbf{r}_j - \mathbf{r}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i)}{|\mathbf{r}_j - \mathbf{r}_i|}. \tag{3}$$

Spring constants can be chosen arbitrarily, as the final result does not depend on them. It is convenient to choose them so that the spring constant for the spring between sites i and j is equal to $|\mathbf{r}_j - \mathbf{r}_i|^2$. Then the total energy is

$$U = \frac{1}{2} \sum_{\langle ij \rangle} [(\mathbf{r}_j - \mathbf{r}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i)]^2, \tag{4}$$

where the sum runs over all pairs of sites that have a constraint between them. After relaxation, for each pair of sites $\{k, l\}$, the quantity

$$\delta_{kl} = (\mathbf{r}_l - \mathbf{r}_k) \cdot (\mathbf{u}_l - \mathbf{u}_k) \tag{5}$$

can be used to determine if the distance between these sites has changed: if this quantity is zero, then the distance has not changed and these sites are mutually rigid; otherwise the distance has changed and the sites are not mutually rigid. Note that since U is quadratic and δ_{kl} linear in displacements $\{\mathbf{u}_i\}$, these displacements no longer have to be infinitesimal when using these "linearized" equations: indeed, rescaling all displacements by some arbitrary factor does not change the results. This has an advantage that in the actual implementation of this procedure on a computer, one does not have to worry if the displacements are "small enough" (which would be the case if linearization were not done).

In practice, the computer precision is always limited, of course, so in an actual implementation, the values of δ_{kl} for rigid pairs found numerically will be very small but nonzero. The solution is introducing a *cutoff*: pairs of sites for which δ_{kl} is below the cutoff are deemed rigid, and those for which δ_{kl} is above the cutoff are not rigid. However, another complication is possible: if the realization of the topology is accidentally very close to a nongeneric one or a special initial displacement was chosen, some of the nonrigid pairs may have the corresponding value of δ_{kl} below the cutoff. For this reason, at least two realizations are always run; the absolute values of δ_{kl} are summed up and then it is determined if a gap of at least two decades containing no values of δ_{kl} has formed. If it has, the cutoff is chosen inside the gap and the procedure is finished. This situation is illustrated in Fig. 9. If the gap has not formed, another realization is run and the absolute values of δ_{kl} are added to the previously obtained sums. If the gap is now present, then the procedure is finished, if not, the current run is abandoned, and a new run is started. A new run is also started if, in any of the relaxations, a certain low energy is not achieved in a predetermined number of steps. At most three runs are done; if none are suc-
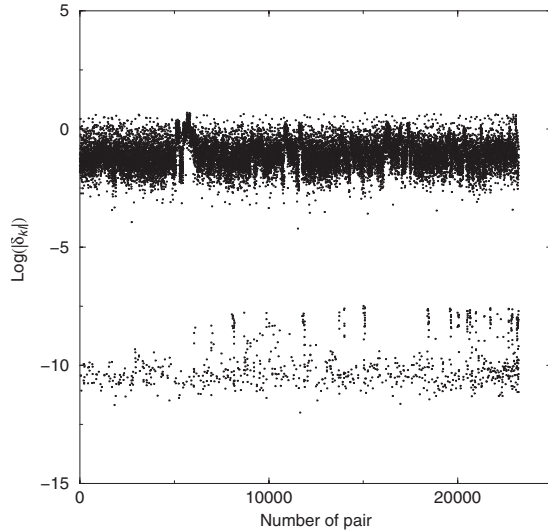
FIG. 9. For the relaxation procedure described in the text, the base-10 logarithm of the sum over two realizations of the absolute value of the quantity δ_{kl} in Eq. (5) for all pairs of sites of a network with 216 sites. The gap between "zero" and "nonzero" values is clearly seen. Pairs with values below the gap are mutually rigid, those with values above the gap are not.

cessful, the relaxation procedure has failed; it can then be repeated using a higher precision. Even despite always doing at least two realizations and even when the self-consistency checks described below succeed, there is still a very small chance of an error. In cases of doubt, for instance, when a discrepancy with the pebble game is detected, the procedure can be done as many times as desired, and lower final energy tolerances and larger gap sizes can be set.

Once all pairs of mutually rigid sites are found, the next step is finding all hinges, including implied ones. First, make a list of sites such that among their neighbors, not all are mutually rigid—only such sites can be hinge end points. From this list, choose all pairs of mutually rigid sites. For each such pair $\{A,B\}$, select an arbitrary site C rigid with respect to both A and B. Then go through the list of all other sites rigid with respect to both A and B; if *any* of such sites is *not* rigid with respect to C, then A-B is a hinge (an implied one if there is no constraint A-B in the network).

If all implied hinges are added to the network explicitly as constraints, the configuration of rigid clusters in the network and the number of floppy modes are not affected. But, as follows from the Dress conjecture, all rigid clusters become contiguous. So once all implied hinges are identified, it is convenient to add them to the network as constraints and then mark rigid clusters labeling each rigid angle formed by network constraints (including the just added implied hinges) so that angles belonging to the same cluster are labeled identically. Recall that sites forming an angle (or any triple of sites for that matter) can belong to at most one rigid cluster, so that each rigid angle will be assigned just one label. Because of contiguity of rigid clusters, such labeling retains the full rigidity information; it is always possible to traverse the network between any two mutually rigid sites by going through angles assigned the same label. This can be used as

a self-consistency check for the algorithm. Constraints that do not form any labeled (i.e., rigid) angles are not rigid with respect to any other sites and form single-constraint clusters on their own; likewise, isolated (disconnected) sites are considered single-site clusters. While finding all mutually rigid pairs of sites does not involve any assumptions from the rigidity theory, the decomposition into rigid clusters, as well as the self-consistency check using this decomposition, do rely on the Dress conjecture.

The next stage is finding the number of floppy modes. The procedure, as described below, is just a convenient interpretation of the Dress conjecture that gives the number of floppy modes based on the number of (explicit and implied) hinges [40]. Intuitively, it follows from the assumption that once implied hinges are added, all clusters become rigid by themselves and redundant constraints in each of them can be counted separately and then added up. Any cluster of three sites or more should have six floppy modes when isolated from the rest of the network; then, if it contains n sites and c constraints, the number of redundant constraints [according to Eq. (2) with $N=n$, $N_c=c$ and $F=6$] is $6-3n+c$. Obviously, the numbers of redundant constraints should be non-negative for all clusters and this serves as another self-consistency check for the algorithm. After the numbers of redundant constraints are found for each cluster, these numbers are added up. Note that, even though some constraints (namely, hinges) belong to two or more clusters simultaneously, they are included in the count of constraints for each cluster they belong to when obtaining the numbers of redundant constraints for these clusters. The total number of redundant constraints N_R is then used in Eq. (2) to obtain the number of floppy modes F.

Counting redundant constraints in each rigid cluster gives information on the presence or absence of stress within that cluster. However, even if the presence of stress is detected, the information obtained so far does not indicate where exactly it is located within the cluster. Remember that we have specifically chosen the lengths of constraints so that they fit exactly, so there is no stress in our networks after relaxation even when it has to be present generically. To locate stress, we need to carry out another relaxation procedure, this time with constraints that do not fit exactly. In the spirit of the first relaxation procedure, we choose constraints with *infinitesimal* misfits. That is, just as in the first procedure, sites are first assigned random positions in space, $\mathbf{r}_i$; if, according to the given connectivity table, there is a constraint between sites i and j, its length is chosen equal to $l_{ij}^0 = |\mathbf{r}_j - \mathbf{r}_i| + \Delta_{ij}$, i.e., there is an infinitesimal misfit Δ_{ij}. As the misfits are infinitesimal, it is expected that displacements from the initial position, $\mathbf{u}_i$, will also remain infinitesimal at all times during the subsequent relaxation procedure, as well as in the relaxed network. In the lowest order in $\mathbf{u}_i$, the deformation of the constraint between sites i and j is

$$\delta l_{ij} = \frac{(\mathbf{r}_j - \mathbf{r}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i)}{|\mathbf{r}_j - \mathbf{r}_i|} - \Delta_{ij}. \tag{6}$$

If we choose the spring constant equal to $|\mathbf{r}_j - \mathbf{r}_i|^2$ and introduce $\epsilon_{ij} = \Delta_{ij} |\mathbf{r}_j - \mathbf{r}_i|$, the energy is

$$U = \frac{1}{2} \sum_{\langle i,j \rangle} [(\mathbf{r}_j - \mathbf{r}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i) - \epsilon_{ij}]^2. \qquad (7)$$

As in the first stage of the relaxation algorithm, we displace all sites at random initially, although now this is not really important, since constraint lengths are themselves random and do not fit exactly between sites. After relaxation with the potential (7) is done, the quantities

$$\delta_{ij} = (\mathbf{r}_j - \mathbf{r}_i) \cdot (\mathbf{u}_j - \mathbf{u}_i) - \epsilon_{ij} \qquad (8)$$

are used to find stressed constraints: if $\delta_{ij} = 0$, then the constraint between sites i and j is unstressed, otherwise it is stressed. In practice, a cutoff between "zero" and "nonzero" values is again introduced. Similarly to the first relaxation procedure, since U is quadratic and δ_{ij} linear in ϵ_{ij} and $\mathbf{u}_i$, these two latter quantities do not have to be infinitesimal when using these "linearized" equations.

Finally, we analyze the computational speed of the relaxation algorithm. In theory, the conjugate gradient algorithm converges to the exact minimum after the number of steps equal to the number of degrees of freedom, which is $3N$ for a network of N sites, or $O(N)$. Each of these steps requires the evaluation of the gradient of the potential, which takes $O(N)$ floating-point operations (flops), so that the relaxation procedure proper takes $O(N^2)$ flops overall. In practice, away from the rigidity percolation transition the number of steps required may be significantly smaller; on the other hand, very close to the transition a somewhat larger number of steps may be needed, since because of the round-off errors the convergence is not perfect after $3N$ steps. Another potentially costly part of the algorithm is finding hinges. Formally, this part requires $O(N^3)$ operations; but in practice, for typical network sizes (up to a few thousand sites), it is usually much faster than the relaxation proper, which in part is explained by the fact that only integer operations are involved. In fact, it may be possible to optimize this part so that in practice, it is never (or rarely) $O(N^3)$.

These estimates of the computational cost of the relaxation algorithm should be compared to those for diagonalization and SVD. Since the computational cost of both diagonalization and SVD is $O(N^3)$ (if the number of constraints is comparable to the number of degrees of freedom) [43], it is likely in view of the above analysis that the relaxation algorithm is faster (although direct tests need to be done to find out if this is really so for typical problems and network sizes). The advantage of the more traditional approaches is, as mentioned before, the fact that they also find the floppy modes themselves, not just their number; on the other hand, rigid cluster decomposition is problematic. In fact, even if a traditional approach is used, we suggest combining it with a procedure similar to ours to find rigid clusters. That is, once floppy modes are obtained, their linear combination with random coefficients can be used as the analog of the outcome of relaxation; then the values of δ_{kl} can be calculated using Eq. (5) and the rest of the procedure for finding rigid clusters is the same as described above. Note also that in the case of diagonalization and SVD, a cutoff still needs to be defined between "zero" and "nonzero" eigenvalues or singular values;

but note that there are only $O(N)$ of these values, whereas there are $O(N^2)$ values of δ_{kl} used to define the cutoff in the rigidity part of the relaxation algorithm, and the more values are used to define the cutoff, the more confident one can be that the cutoff is chosen correctly. This is another advantage of the relaxation algorithm, although, unfortunately, it does not apply to the stress determination part, since in this case, the number of values of δ_{ij} from Eq. (8) is equal to the number of constraints and thus is $O(N)$.

To conclude this section, we should mention a certain similarity between the relaxation algorithm for rigid cluster decomposition and the FRODA algorithm for generating internal motions of proteins [21,44]. In both algorithms, the initial configuration is distorted in some way and then relaxed to generate a new configuration. The details are, of course, different: FRODA uses a different relaxation procedure that makes use of special properties of protein networks; also, FRODA takes care of finite atom sizes by avoiding hard sphere van der Waals overlaps.

V. ERRORS IN THE PEBBLE GAME: GENERAL CONSIDERATIONS

We are now in a position to analyze the correctness of the approximate pebble game described in Sec. III in different situations. We start with some general considerations.

First of all, consider what configurations can give rise to pebble game errors. As mentioned before, according to the Dress conjecture, all violations of the molecular framework conjecture statement and the associated properties of contiguity of rigid clusters and their being rigid by themselves are due to implied hinges. However, it is important to remember that the pebble game in essence does constraint counting not only (or even not quite) for the network being analyzed itself. First, any constraint not covered by a pebble is "invisible" to the pebble game in the sense that pebble searches will occur in the same way and with the same outcome as if this constraint was not present in the network. Essentially, the pebble game looks at the network from which all uncovered constraints are removed. An uncovered *explicit* hinge is thus exactly the same as an implied hinge for the pebble game. For this reason, both implied hinges and uncovered explicit hinges can cause pebble game errors and should be considered on an equal footing. We analyze the case of an implied or uncovered explicit hinge in the next subsection. Besides that, note that during the pebble game, the network is built gradually, constraint by constraint, and the counting of redundant constraints is done during the whole construction process. For this reason, we should also consider a situation where a hinge was present at some time during the construction process but does not exist anymore at the end. This is considered in Sec. V B.

A. Implied and uncovered explicit hinges

When an implied or an uncovered explicit hinge is present, it may affect both the floppy mode count and the rigid cluster decomposition (as in Fig. 3), or it may only affect the latter (as in Fig. 4). To find out what determines the

difference, recall that, according to the Dress conjecture, when obtaining the number of floppy modes F, redundant constraints are counted in each cluster separately (after all implied hinges are added) and then summed up. When counting is done within a given cluster, the hinge contributes to the number of redundant constraints (that is, its addition changes this number) if it is stressed when this cluster (with the hinge included explicitly) is taken separately from the rest of the network (with a caveat mentioned at the end of the paragraph). So the contribution of a hinge to the total number of redundant constraints is equal to the number of times this hinge is found stressed when different clusters sharing this hinge are taken in isolation. On the other hand, the hinge always contributes 1 to the total number of constraints. So the net contribution of the hinge to F is the number of times the hinge is found stressed minus 1. Any implied or uncovered explicit hinge has to be found stressed at least once. If it is found stressed once, the net contribution is zero. If it is found stressed more than once, the net contribution is nonzero. In the pebble game, on the other hand, implied hinges do not contribute to either the total number of constraints or the number of redundant constraints, so the net contribution is always zero; as mentioned above, uncovered explicit hinges are equivalent to implied ones for floppy mode count purposes, so the net contribution is zero for them as well. The conclusion is then that when an implied or uncovered explicit hinge is found stressed once, the floppy mode count of the pebble game is correct. This is the case in Fig. 4, where the hinge is found stressed when it is considered with the left banana, but not when it is considered with the right banana. On the other hand, when a hinge is found stressed more than once, the pebble game count is incorrect (always lower than the correct one, never higher). In Fig. 3, indeed, the hinge is found stressed twice, i.e., it is found stressed when considered with each of the two bananas. Occasionally, the considerations of this paragraph may *overestimate* the number of redundant constraints and thus the pebble game error due to implied hinges, if there are several hinges within the same rigid cluster and the same stressed region. For instance, if there are two hinges within the same stressed region, then according to the above considerations, the contribution of these hinges to the number of redundant constraints in the cluster should be equal to 2; but it is still possible that the actual constraint count for this region will indicate that it has just one redundant constraint.

B. "Former" hinges

We now consider the case of an implied or uncovered explicit hinge that existed at some time during the construction process but does not exist in the final network.

The first possibility is an implied hinge that became a covered explicit hinge when a constraint coinciding with it was inserted. Note that the end points of an implied hinge are mutually rigid, so when a constraint connecting these points is added, it will always be redundant and normally will not be covered by a pebble. Exceptionally, the pebble game may deem this constraint independent incorrectly and cover it because of errors due to hinges elsewhere in the clusters shar-

ing the hinge. But in this case the end results of the pebble game are the same as if the constraint was inserted and covered early enough so there was never an implied hinge in its place, so this situation will not cause any errors.

The remaining possibility is a hinge that ceased to exist altogether when the rotation around it was locked later in the network construction process. Consider for simplicity a hinge that is shared (and has always been shared) by just two clusters (the consideration can easily be extended to the case of more than two clusters sharing a hinge). Then just one locking constraint is sufficient to "destroy" the hinge. As such a constraint removes a floppy mode, it is always independent. But the pebble game may find it either independent or redundant. If the hinge in question is of the kind that causes errors in the floppy mode count, then the floppy mode associated with the rotation around the hinge is ignored and both clusters sharing the hinge are considered as a single rigid cluster by the pebble game and so the locking constraint is (incorrectly) found redundant. Note that this will correct the error in the pebble game count. If, on the other hand, the hinge does not cause a floppy mode count error, the locking constraint is viewed by the pebble game as rigidifying the region and removing a floppy mode and is found independent. In this case there was no error in the floppy mode count and it is not introduced. In either case, the resulting floppy mode count is correct. As for the rigid cluster decomposition, in the first case (the locking constraint deemed redundant) the two clusters sharing the hinge are (incorrectly) deemed mutually rigid even before the locking constraint is inserted and, of course, remain mutually rigid afterward, which after locking is the correct result; in the second case, when the locking constraint is deemed independent, this constraint is covered by a pebble and then for the pebble game this is equivalent to the situation where the hinge was never present (which is especially obvious for rigid cluster decomposition, since it is done when the network building is finished, at which point there is "no evidence" that the hinge was ever present).

So the conclusion is that there are no problems in either the floppy mode count or the rigid cluster decomposition due to hinges that were there but are no longer present at the end. But the stress determination may still be wrong in such cases, as, e.g., for the network in Fig. 6; this is considered in detail in Sec. V D.

C. Detecting pebble game errors using relaxation

The conclusion of the above consideration is that only implied and uncovered explicit hinges in the final network cause errors in the rigid cluster decomposition and/or floppy mode count; moreover, whether the floppy mode count is in error depends on the number of times the hinge inserted explicitly is found stressed when it is considered separately with each of the clusters sharing it. Since the relaxation algorithm finds explicit and implied hinges and also counts redundant constraints within each cluster, we can obtain some information on the possible error in the pebble game floppy mode count even without running the pebble game.

Consider each hinge (explicit or implied) and determine the number of clusters among those that share this hinge that

have a nonzero number of redundant constraints n_r. Having a nonzero n_r is a *necessary* condition for the hinge to be found stressed when considered with this cluster. But it is not a sufficient condition, since a nonzero number of redundant constraints only indicates that stress is present somewhere in the cluster, and this does not necessarily include the hinge. For this reason, if the number of clusters sharing the hinge and having nonzero n_r is zero, the hinge is unstressed; it is then explicit and, moreover, is covered, so it can never spoil finding the number of floppy modes or the configuration of rigid clusters. If the number of clusters with nonzero n_r is 1, the actual number of times the hinge is found stressed is either 0 or 1, and then the hinge is either not "dangerous" at all (in the first case), or can affect the rigid cluster decomposition (in the second case, provided that it is implied or inserted late enough to be uncovered); it cannot affect the floppy mode count. Finally, if the number of clusters with nonzero n_r is 2 or higher, it is possible that the hinge is stressed two or more times, and then the number of floppy modes *may* be affected. The maximum possible error in the number of floppy modes due to the hinge is the number of clusters with nonzero n_r minus 1. The advantage of the described procedure (as opposed to the straightforward comparison of the pebble game results with the relaxation results) is that the error determined in this way is at least as high as the maximum possible error (where the maximum is taken over all possible orders of constraint insertion). In other words, this is the worst-case scenario estimate.

D. Stress determination

As mentioned above, there may be problems in stress determination if a single constraint locks a mutual rotation of two clusters around a hinge, as in Fig. 6. Unfortunately, since in such cases the hinge is not actually present, it is impossible to find such situations, except by doing stress determination through relaxation and then comparing directly with the pebble game result. Another case where problems with finding stress can arise is when the floppy mode count fails, such as in the double-banana case. We should note that problems with stress determination can only arise due to hinges or "former" (i.e., locked) hinges that can create problems for the floppy mode count, but not those that can only affect rigid cluster decomposition. This is because the stress determination procedure is designed to be correct if there are no errors in the floppy mode count for either the given network itself or any network obtained from it by removing a single constraint.

E. Random bond networks

While in general, to use the results of this section one needs to run the relaxation algorithm on the network of interest to find if there are any dangerous hinges, there is one class of networks for which the result is obvious. These are random bond networks [Fig. 2] in the thermodynamic limit. In random bond networks, sites are assigned certain coordination numbers and then are connected at random, regardless of the distance between them, consistent with the assigned coordinations. In the thermodynamic limit, such networks

have no finite loops and thus no finite clusters other than single bonds and sites (and, when an infinite cluster is present, triangles sharing an implied hinge with the infinite cluster, as in Fig. 8). Thus the only dangerous hinges possible in such networks are trivial ones (Fig. 8). As triangles are not stressed, such a hinge is found stressed only once (namely, when considered with the infinite cluster) and thus its presence cannot affect the floppy mode count or the sizes of the infinite rigid cluster and the infinite stressed region. The worst error possible in this case is missing a few rigid triangles.

In networks with loops, larger finite clusters are possible and the arguments given for random bond networks do not apply, of course. In the next two sections, we look at two classes of such networks.

VI. RANDOMLY DILUTED CENTRAL-FORCE NETWORKS

In this section, we consider a particular class of 3D networks, randomly bond-diluted central-force networks, using the algorithms considered in the previous sections. In particular, we study rigidity percolation on such networks. We first apply the relaxation algorithm to the networks in order to ascertain the possibility of their study with the faster pebble game algorithm, whose use allows studying much larger networks than would be feasible with the relaxation algorithm. At the end of the section, we briefly consider site-diluted networks, with some of the results markedly different from those for bond-diluted networks.

A. Maxwell counting

We start with the straightforward Maxwell counting in order to guide our search for the rigidity transition.

In a network of N sites, the number of degrees of freedom is $3N$. In central-force networks, the number of constraints is equal to the number of bonds and is $\langle r \rangle N/2$, where $\langle r \rangle$ is the mean coordination. Then

$$F_{\text{Maxw}} = 3N - \langle r \rangle N/2. \tag{9}$$

This reaches zero at $\langle r \rangle = 6$. Even though Maxwell counting is not exact, it is assumed that in reality, the number of floppy modes becomes small when $\langle r \rangle$ approaches this value and a percolating rigid cluster emerges somewhere around this point (indeed, in 2D central-force and 3D bond-bending networks, the transition occurs very close to the point at which the Maxwell counting result turns zero).

Given that the transition is expected to be located at $\langle r \rangle \approx 6$, to study it by bond-diluting a regular lattice we need a lattice with the coordination number exceeding 6. Thus the body-centered cubic (bcc) lattice with the coordination of $z = 8$ and the face-centered cubic (fcc) lattice with the coordination of $z = 12$ are natural choices. In a regular lattice with coordination number z, the total number of bonds is $Nz/2$. If bond dilution is done so that fraction p of the bonds remain, this gives the number of bonds $N_B = Nzp/2$, and since each bond is shared between the two sites that it connects, the mean coordination is

$$\langle r \rangle = 2N_B/N = zp. \qquad (10)$$

Then at the transition we expect $p \approx 1/2$ for fcc and $p \approx 3/4$ for bcc lattices. Note, by the way, that Eq. (10) is valid even for finite networks and even nonrandom ones, if p is indeed interpreted as the actual fraction of present bonds and not as the probability that a given bond is present.

One should keep in mind that the approaches to studying rigidity described in this paper are applicable only to generic networks. Regular lattices like fcc and bcc are not generic, of course, as they have parallel bonds, all bonds are of the same length, etc. Any results described here are therefore applicable not to the diluted regular lattices themselves, but rather to networks topologically equivalent to them, but distorted by introducing bond length disorder. This is also true for the older results for the diluted diamond lattice bond-bending networks shown in Fig. 1.

B. Using the relaxation algorithm

We first study small bond-diluted bcc and fcc networks using the relaxation algorithm, doing both rigidity and stress runs (the latter just for those networks where the former detected any redundancy). For fcc lattices, we use networks of 500 sites, for all numbers of bonds between 1460 ($\langle r \rangle = 5.84$ or $p \approx 0.4867$) and 1490 ($\langle r \rangle = 5.96$ or $p \approx 0.4967$). For each number of bonds, we generate 100 different networks; this gives a total of 3100 networks. Even though usually much larger networks can be analyzed easily, this is a particularly difficult case for the relaxation algorithm, since close to the transition, there are large regions (taking up most of the network) that are isostatic (i.e., rigid but unstressed, with constraints exactly balancing degrees of freedom) or nearly isostatic, and it is very hard computationally to distinguish an exactly isostatic region from one lacking just a single constraint and thus having one floppy mode spread over thousands of degrees of freedom: in the former case, the region is rigid but with extremely low effective elastic moduli (vanishing in the thermodynamic limit); in the latter case, it is flexible but with the motion limited to a subspace of dimensionality one in a space with thousands of dimensions. For this reason, in rare cases the relaxation algorithm has failed to converge. That is, in the rigidity runs, out of 3100 networks, relaxation has failed for 39, or just over 1%. All these cases were rerun using more computationally intensive quadruple-precision arithmetic; all 39 runs succeeded. Likewise, in the stress runs, relaxation has failed in 23 cases, and again, all succeeded using quadruple precision. At the lowest bond number (1460), all 100 networks have only small rigid clusters and no stress; at the highest bond number (1490), all 100 networks have a rigid cluster taking up almost all the network and 99 out of 100 networks have a stressed region likewise taking up most of the network. In other words, the chosen range of mean coordinations indeed contains the rigidity transition. In the bcc case, networks of 686 sites were used, with numbers of bonds between 2040 ($\langle r \rangle \approx 5.9475$ or $p \approx 0.7434$) and 2060 ($\langle r \rangle \approx 6.006$ or $p \approx 0.7507$), again with 100 networks for each number of bonds, for the total of 2100 networks. In this case, all runs,

both for rigidity and for stress, succeeded without using quadruple-precision arithmetic. Again, at the smallest bond number, only small rigid clusters are present and no stress in all networks; at the highest bond number, all 100 networks have a rigid cluster and a stressed region taking up most of the network.

A remarkable observation is that in all these runs, for both fcc and bcc lattices, only very small and very large rigid clusters are observed, but never those of intermediate size. That is, in fcc networks, only clusters of 11 and fewer sites or 431 and more sites are found; in bcc networks, only clusters of 1, 2, 3 sites or at least 668 sites are found. As for stressed regions, small ones are never observed: the smallest regions ever found are of size 388 for fcc and 632 for bcc networks. (As we discuss in the Appendix, small stressed regions are possible but so rare that it is not surprising we have not seen them in these runs.) This is a strong indication that the rigidity transition is first order: instead of the average cluster size growing gradually as the transition is approached, before a percolating cluster (that first takes up a small part of the network) arises, here no gradual growth is observed; the percolating cluster emerges suddenly, upon a single bond addition, and immediately takes up much of the network. Likewise, stress is extremely rare in the floppy phase (not present at all in our relaxation runs) and arises suddenly after a single bond is added, again, immediately taking up most of the network.

As we have mentioned, a first-order transition was found previously in so-called random bond networks. The reason why the transition is first order in that case has to do with the already mentioned property of these networks: there are no finite rings and thus finite rigid clusters (other than single bonds and single sites and perhaps triangles). Without finite clusters, the infinite cluster has to emerge suddenly: there is no diverging correlation length typical of second-order transitions as the threshold is approached. Of course, rings are certainly present in regular lattices like bcc and fcc, but still, is the situation similar here in some way? In the Appendix, we analyze this question in detail using a computational procedure for generating maximally rigid configurations of a certain size. The result is that, in the bcc case, the situation is indeed somewhat similar: besides the clusters of sizes 1, 2, and 3, the smallest possible cluster has size 90 if it stands alone or 84 if sharing a hinge with another rigid cluster (and these are extremely rare; the frequency per site of observing such a cluster is estimated in the Appendix to be very roughly $\sim 10^{-30}$); but, in the fcc case, clusters of all sizes are possible, but the probability of actually observing a cluster of a given size decreases very rapidly as the size increases.

We can now address the question of the accuracy of the pebble game algorithm when applied to bond-diluted central-force bcc and fcc networks. In the bcc case, when besides the percolating cluster, only clusters of size up to 3 are observed, obviously the only possible kind of hinge is a trivial one shared between the percolating cluster and a cluster of size 3 (a triangle), as shown in Fig. 8. Indeed, such hinges are rather frequent: a total of 4822 are observed in the 1276 percolating networks (out of the total of 2100 networks), including two that are shared with the percolating cluster by *two* triangles. All of these hinges are implied: indeed, tri-

angles of bonds do not exist in bcc lattices, so rigid triangles actually consist of two real bonds and an implied hinge. Such trivial hinges are also by far the most frequent kind in the fcc case, although in this case, both implied and explicit ones are possible: there are 15 179 implied and 3971 explicit trivial hinges in 1288 percolating networks, including respectively 11 and 5 shared by two triangles. Besides these, there are also a very small number (13 implied and 1 explicit) of hinges shared by the percolating cluster with a *four-site* cluster (a tetrahedron); also, the two largest observed "small" clusters (of sizes 10 and 11) share a few hinges with triangles. While the total number of hinges is large, none of them can affect the floppy mode count of the pebble game. Such hinges also cannot affect determination of the size of the percolating cluster: indeed, since neither the triangle nor the tetrahedron is a stressed object, they cannot rigidify the percolating cluster and it has to be rigid by itself (cf. Fig. 4, where the left banana, although not stressed by itself, becomes stressed if the hinge is inserted explicitly and thus has the potential to rigidify the right, "incomplete" banana); thus the pebble game will have no problem finding the percolating cluster correctly. The only possible type of error is then failure to identify a rigid triangle or, very rarely, tetrahedron. Since so few types of errors are possible, these errors can easily be taken care of, if needed; but if we are interested only in the number of floppy modes and the percolating cluster size (the quantities relevant for determining the order of the transition), we need not do this. Note also that the above numbers for problematic explicit hinges are likely overestimations, as the rigid cluster decomposition part of the relaxation algorithm overestimates the number of stressed hinges, as explained in Sec. V. The actual number can be found if required.

Obviously, since clusters of all sizes are possible in the fcc case, other situations involving larger "small" clusters are possible in principle. For example, one can ask if configurations of the double-banana type that would violate the floppy mode count of the pebble game can occur. A method described in the Appendix allows generation of such configurations, even if they are extremely rare in reality. It turns out that the standard double-banana graph (like the one shown in Fig. 3) that consists of two clusters (bananas) of size 5 is not possible in a fcc network. However, larger configurations of the same type are possible. The smallest one consists of two bananas of size 8 each, as shown in Fig. 10. In the Appendix, the frequency of a single eight-site rigid cluster is estimated. It is rather low, only 3×10^{-5} per site. To have a double-banana-type configuration, two such clusters need to be located next to each other and in a certain relative orientation; the probability of this will be roughly $(3 \times 10^{-5})^2 \sim 10^{-9}$ per site—a very small number, and even this is probably an overestimation. Another possibility for a double-banana-type configuration is the percolating cluster sharing a hinge with a small cluster, the smallest possibility being a six-site cluster (an octahedron) with the hinge connecting two opposite vertices. Octahedra are extremely rare in the rigid phase: in our relaxation runs, we see none, and in a much larger pebble game study discussed in the Appendix, only a few. Based on that study, in the Appendix we give a crude estimate for the frequency of hinges of this kind—at most 10^{-8} per site and probably less—again, extremely rare.
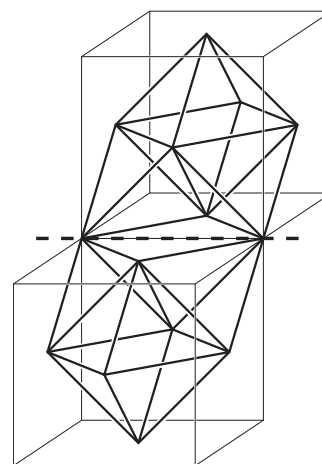

FIG. 10. Smallest graph of the double-banana type that can exist in the diluted fcc network. The dashed line is the hinge. The outlines of two unit cells are shown for clarity with thin lines.

Regarding stress determination, as we have mentioned, finding hinges for a particular network alone does not detect all possible errors in finding stress using the pebble game because of configurations with locked hinges, like the one in Fig. 6. However, since we have never seen any hinges that cause problems for the floppy mode count in any of the networks we have analyzed, the chance of having a locked hinge of this kind is very small—there are no hinges to lock, to start with. For this reason, we do not expect to see errors in finding stress either.

C. The pebble game analysis

With the relaxation algorithm, we can study only rather small networks. While this study gives strong indications that the rigidity transition in 3D central-force bond-diluted networks is first order, using larger networks is desirable, in particular, to reduce finite-size effects. Based on our results obtained using the relaxation algorithm, we can be confident that the pebble game results are going to be accurate in this case, even though in general the algorithm is only approximate. We have also confirmed this explicitly, by applying the pebble game to the same networks that we have analyzed using the relaxation algorithm, as described in the previous subsection. The results of this comparison (among other results described below) are shown in Figs. 11 (for the bcc lattice) and 12 (for the fcc lattice). In the upper panels, we plot the average number of floppy modes obtained using the relaxation algorithm (pluses) and the pebble game (circles). In the lower panels, we show the average sizes of the largest rigid cluster (pluses for the relaxation algorithm, circles for the pebble game) and of the only stressed region ($\times$'s for the relaxation algorithm, squares for the pebble game). All pluses are inside circles and all $\times$'s are inside squares, indicating that the pebble game and the relaxation results coincide. In fact, the results were checked network by network; very minor and rare discrepancies (two networks out of 3100 for fcc, none for bcc) turned out to be due to round-off errors in the relaxation algorithm, rather any problems with
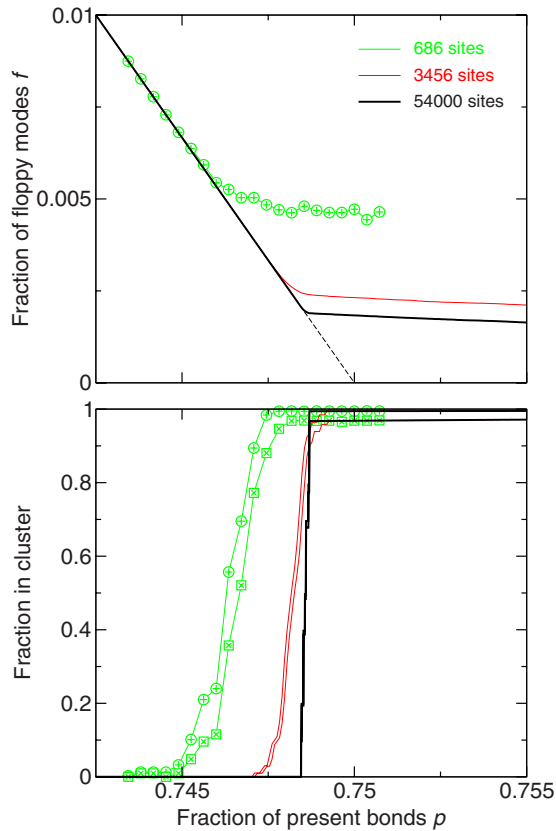
FIG. 11. (Color online) (Top) Number of floppy modes per degree of freedom, $f=F/3N$, for bond-diluted central-force bcc networks, for three different sizes: small (686 sites; symbols, green online); medium (3456 sites; the thin line without symbols, red online); and large (54 000 sites; the thick line). For the smallest size, the results obtained by both the pebble game (circles) and the relaxation algorithm (pluses inside the circles) are shown; the same realizations are used in both cases. For the other two sizes, the pebble game was used. The dashed line is the Maxwell counting result. (Bottom) Fraction of sites in the largest rigid cluster and fraction of bonds in the only stressed region for bond-diluted central-force bcc networks, for three different sizes. The line thicknesses and the color scheme (in the online version) are the same as in the top panel. For each size, the top line represents the largest rigid cluster and the bottom line, the stressed region. For the smallest size, the results obtained by both the pebble game (circles for the largest rigid cluster, squares for the stressed region) and the relaxation algorithm (pluses and ×'s, respectively) are shown. For the other two sizes, the pebble game was used. In both panels, the results for the small and medium sizes are averages over 100 networks, with intervals between data points equal to one bond; the results for the largest size are averages over ten networks, with intervals between data points equal to one bond in the vicinity of the transition and ten bonds elsewhere.

the pebble game (as we confirmed by using quadruple precision on these networks that eliminated the discrepancies).

Having ascertained the accuracy of the pebble game, we can now apply it to study larger networks. For the relaxation study, networks were generated independently at each mean coordination. Here instead, networks are built gradually and intermediate stages are used for obtaining results as well. We
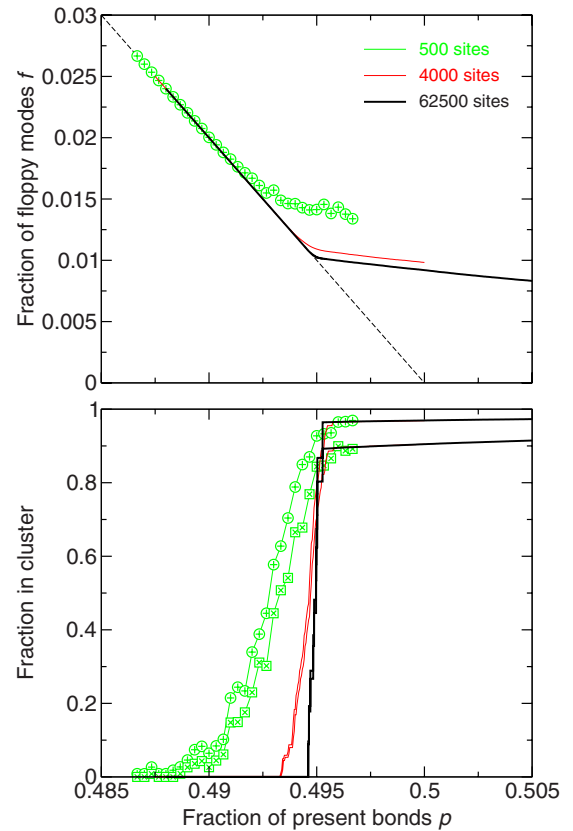
FIG. 12. (Color online) Same as in Fig. 11, for bond-diluted fcc networks. The sizes used are 500 (small), 4000 (medium), and 62 500 sites (large).

first remove all bonds from the full fcc or bcc lattice and then place them back one by one randomly while testing each for redundancy with the pebble game. Rigid cluster decomposition is done after every bond addition close to the transition, but can be done less frequently away from it. In this way, we can analyze the whole sequence of networks with different p in a single pebble game run, which is yet another advantage of the pebble game compared to other algorithms.

The results for the bcc and fcc lattices are shown in Figs. 11 and 12, respectively. Lattice sizes used are 3456 and 54 000 sites for bcc and 4000 and 62 500 sites for fcc. In both cases, the percolation transition is clearly seen as a jump in the size of the percolating rigid cluster and percolating stressed region, as well as a break in slope in the number of floppy modes (which coincides with the Maxwell counting result below the transition, as there are no redundant constraints, but deviates immediately above the transition). The jump gets sharper as the network size increases. This is consistent with the rigidity transition being first order. The transition occurs at $p \approx 0.7485$ in the bcc case and at $p \approx 0.495$ in the fcc case. Both values are quite close to the Maxwell counting estimates ($p=3/4$ and $1/2$, respectively) as expected. Note that there is a slight discrepancy between our value in the bcc case and that obtained by Arbabi and Sahimi [39] ($p=0.737\pm0.002$), which is probably due to the fact that their simulations were on undistorted nongeneric lattices.
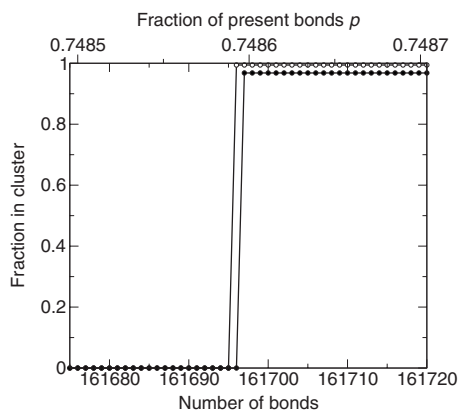
FIG. 13. Fraction of sites in the percolating rigid cluster (open circles) and fraction of bonds in the only stressed region (filled circles) as a function of the number of bonds in the network, for a single bond-diluted bcc network of 54 000 sites built incrementally, bond by bond. Note that the interval between adjacent data points is one bond.

Note that the jumps in the cluster sizes, as presented in Figs. 11 and 12, are not infinitely sharp—there is a slight rounding off. But this is simply because these results are the averages over several realizations, and the transitions occur at slightly different points in different networks (a finite-size effect). But looking at each realization individually, it turns out that in each case (and for both medium and large networks) both the rigidity and the stress transitions happen upon a *single bond addition*. In the floppy phase, the largest rigid cluster size is very small, usually around ten or less for fcc (the maximum observed in the ten realizations for the largest network size was 19) and always two for bcc, and there are rarely (never in these runs) any stressed bonds; then suddenly, with a single bond added, a huge cluster taking up more than 90% of the network emerges; and then, again with a single bond added (and it is nearly always the bond immediately following the one causing the rigidity transition), a huge stressed region, again occupying around 90% of the network, appears. Thus both the rigidity and the stress transitions occur in the most dramatic manner possible, with an enormous jump in the order parameter upon a single bond addition. This is illustrated in Figs. 13 for the bcc and 14 for the fcc case. In each case, the results for a single realization are shown. Note that the interval between adjacent data points is one bond.

The width of the rigidity transition in terms of p thus appears to be $O(1/N)$. Normally, first-order transitions are not as sharp, as different parts of the system undergo transitions at slightly different points, resulting in a width that, while decreasing with growing N, does so more slowly than $O(1/N)$. Another interesting point is the absence of the hysteresis problem commonly associated with numerical studies of first-order transitions: in thermal phase transitions, for instance, as the temperature is changed to drive the system across the transition, it takes a long time for the system to reach the new phase and equilibrate and so the transition is delayed. With no equilibration required and the pebble game being exact, there is no hysteresis here, of course.
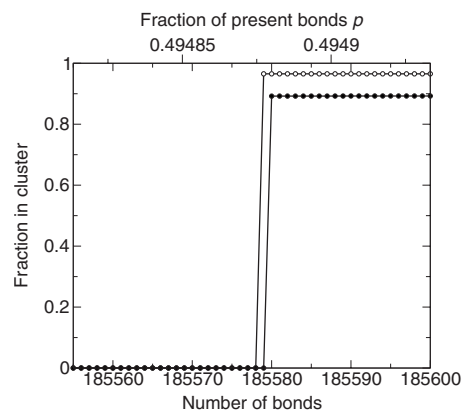

FIG. 14. Same as in Fig. 13, for a single bond-diluted fcc network of 62 500 sites.

We should also note that previous studies for bond-diluted fcc [28] and bcc [39] networks have indicated that the elastic moduli change continuously at the rigidity transition, without a jump (Fig. 15). We can thus say that the transition is geometrically first order but physically second order, as the geometric order parameters such as the size of the percolating cluster jump at the transition, but the physical quantities such as the elastic moduli do not.

D. Site-diluted networks

We now describe briefly the results of a similar study for site-diluted networks. Some of the results are radically different from those for the bond-diluted case.

In site-diluted networks, a certain number of sites are deleted with all associated constraints, but all remaining sites retain all connections to other remaining sites. One can still define the mean coordination $\langle r \rangle$ as the mean number of remaining neighbors of a site, averaged over all remaining sites. In the Maxwell counting approximation, the rigidity transition is still at $\langle r \rangle = 6$. If sites are deleted at random and the fraction of remaining sites is p, then on average a fraction p of neighbors of each present site remain, so $\langle r \rangle = zp$, where z is the coordination number of the full lattice. This is the
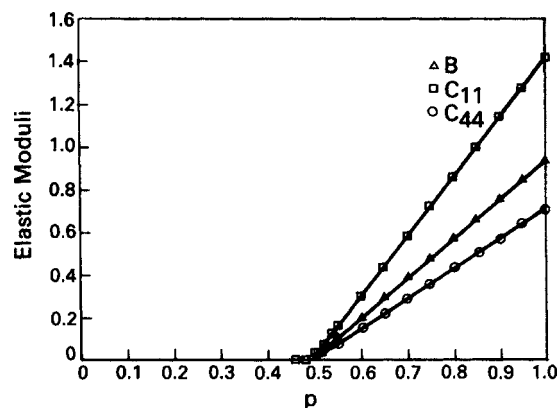

FIG. 15. Elastic moduli for the bond-diluted fcc network (adapted from Ref. [28]).

same relation as Eq. (10) for bond-diluted networks, except in that case the relation was exact even for finite networks, whereas in the site-diluted case it is exact only in the thermodynamic limit, but should still be good for large enough networks. Based on Maxwell counting, the transition should occur at $p \approx 1/2$ for the fcc and $p \approx 3/4$ for the bcc case, just as for bond dilution.

Similarly to the bond dilution case, we first use the relaxation algorithm. In the bcc case, we consider 1000 realizations on site-diluted lattices, originally of 686 sites each, all with 510 sites present ($p \approx 0.743$). At this point, both percolating and nonpercolating networks are present. There is no qualitative difference from the case of bond dilution: still, only very small (up to three sites) or very large (at least 459 sites) clusters are present. This is to be expected: we know that clusters of sizes above three and below 84 cannot exist in principle, and this does not depend on the dilution procedure, of course; nonpercolating clusters of size 84 or larger, on the other hand, are still expected to be very rare, although not as rare as in the bond case, for reasons explained in the Appendix in the discussion of the fcc case. Of course, no dangerous hinges, other than the trivial ones shared by the percolating cluster and a triangle, are possible, so no problems in the pebble game are expected, as far as the floppy mode count and the percolating cluster size are concerned.

In the fcc case, on the other hand, the situation is very different. We consider 1000 realizations on site-diluted lattices, originally of 500 sites each, all with 235 sites present ($p = 0.47$). Both percolating and nonpercolating networks are present, as well as many percolating in just one or two directions. In contrast to the case of bond dilution, we now see many clusters of all sizes, not just very small and very large. This is what one might expect in case of a second-order transition. Some statistics in comparison to the case of bond dilution is given in the Appendix.

Given that many clusters of all sizes are present in the fcc site-diluted case, one might naively expect many dangerous hinges with bananas on all length scales. Fortunately, this turns out not to be the case. There are still numerous hinges rigidifying triangles and tetrahedra that are also present in the bond-diluted case (although now they are often shared with a medium-sized cluster, not necessarily with the percolating cluster, which was very rare in the bond-diluted case). As before, the presence of these at most means that the pebble game may fail to detect some triangles or tetrahedra—any larger clusters, including the percolating cluster, are not affected, nor is the floppy mode count. Besides these, only very few other hinges are present. In the same 1000 500-site networks, the first part of the relaxation algorithm detects 45 hinges involving clusters both of which are larger than a tetrahedron and thus with the potential to introduce more significant errors than missing a triangle or a tetrahedron. Of these, 11 can influence the floppy mode count. But this is still an overestimate, as we know, and needs to be confirmed by analyzing stress. This more careful analysis brings the number of dangerous hinges down from 45 to 18, or roughly one per 13 000 present sites, of which none can affect the floppy mode count. The largest affected cluster is still fairly small, consisting of 24 sites, so the percolating cluster can never be misidentified. Most misidenti-
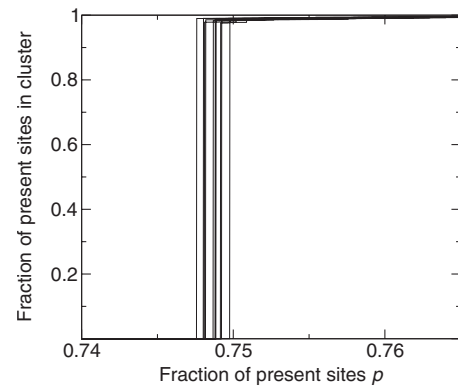

FIG. 16. Fraction of sites (among those still present, i.e., undeleted) in the largest rigid cluster for the site-diluted central-force bcc networks. Results for ten realizations on networks initially consisting of 54 000 sites are plotted separately. The step between adjacent data points is one site.

fications are, in fact, fairly benign—14 out of 18 involve missing a single triangle from a larger cluster and two more involve missing a single tetrahedron. Given this, we can still safely use the pebble game—chances of the floppy mode count or the percolating cluster size being affected are very slim.

The pebble game results confirm the expectations based on the relaxation study. The largest rigid cluster size as a function of the fraction of sites present is shown in Fig. 16 for the bcc lattice and in Fig. 17 for the fcc lattice. In the bcc case, we again see very sharp jumps, with the largest cluster size changing from two to almost the whole network upon addition of a single site, consistent with a first-order transition. But in the fcc case, the largest cluster size grows gradu-
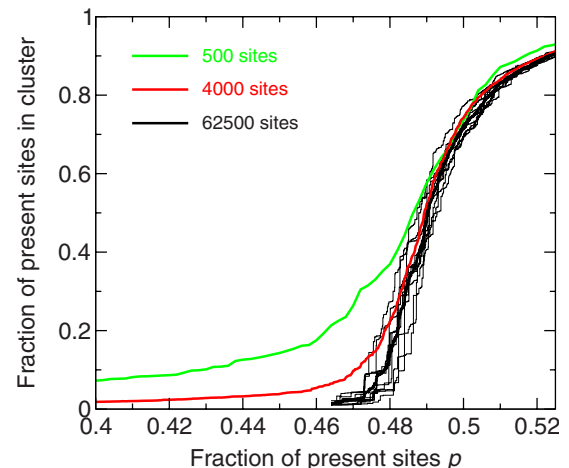

FIG. 17. (Color online) Fraction of sites (among those still present, i.e., undeleted) in the largest rigid cluster for the site-diluted central-force fcc networks. For the smallest (500 sites initially) and medium (4000 sites) networks, the average over 100 realizations is plotted; for the largest (62 500 sites) networks, both the average over ten realizations (the thick black line) and the results for the individual realizations (thin lines) are plotted. The step between adjacent data points is one site in all cases.

ally when sites are added, with only very small jumps present in individual realizations, which is typical of a second-order transition.

E. Discussion

Systems that we have considered in this section differ just in the underlying lattice and/or type of disorder (site vs bond). It is usually thought that these properties are irrelevant when determining general properties of a phase transition, such as its order, or, in the case of a second-order transition, the critical exponents, unless long-range interactions are introduced. For instance, in the Potts model [45], the order of the transition depends on the dimensionality and the number of states, but not on the lattice type. It is certainly possible to have a *tricritical point* separating regions of first- and second-order transitions, but this is usually observed when there are several competing interactions whose relative strengths can be varied. This is not the case here, and thus the situation we observe is highly unusual and counter to our intuitive expectations based on universality.

At the same time, some degree of nonuniversality should perhaps be expected in rigidity percolation based on previous experience. Indeed, in two dimensions rigidity of bond-bending networks is the same as their connectivity in the sense that any sites that are connected are mutually rigid. But since it is well known that critical exponents of rigidity percolation are different from those of connectivity percolation [19,46], this means that central-force and bond-bending networks are in different universality classes, even though the only difference between these networks is in short-range second-neighbor interactions that are present in bond-bending networks and missing in central-force ones and so, based on universality arguments, they should belong to the same universality class. This is especially striking for random bond networks for which the connectivity percolation transition is second order but the rigidity transition is first order. We also note that Head *et al.* [47] have shown that elastic critical exponents in 2D central-force and random rod networks are different, another sign of nonuniversality.

Of course, it should be remembered that claims based on numerical simulations can rarely be made with absolute certainty. While our results in Fig. 17 give a strong indication that the rigidity transition is second order in the case of fcc site dilution, a first-order transition rounded due to finite-size effects can never be ruled out completely. Our claim that in the other three cases the transition is first order looks even stronger given how sharp the transition is; but even in this case, surprises are possible.

VII. A COUNTEREXAMPLE: CHAINS OF EDGE-SHARING TETRAHEDRA

In the previous section, we have shown that, in the particular case of randomly diluted central-force networks, there are virtually no errors in the pebble game, except insignificant ones, such as missing a small cluster. In particular, there are hardly any double-banana-type configurations similar to those shown in Figs. 3 and 10. However, such configurations may be quite frequent in certain cases.
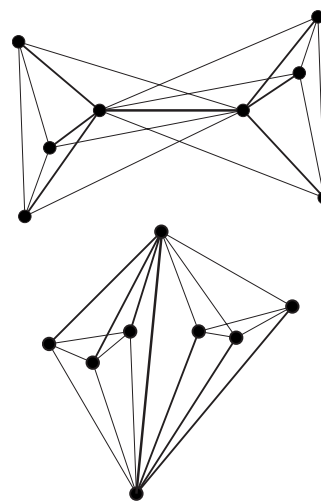

FIG. 18. Top: two connected fourfold coordinated sites and their neighbors in a bond-bending network with both central-force (thick lines) and bond-bending (thin lines) constraints shown. This is equivalent to a double-banana graph with the hinge added explicitly (bottom).

For instance, consider two neighboring fourfold coordinated sites in a bond-bending network. It is easy to realize that a network consisting of such a pair of sites with their neighbors and associated central-force and bond-bending constraints is topologically equivalent to the double-banana graph plus the explicit hinge (Fig. 18). Specifically, each banana is formed by the constraints associated with a particular site of the pair, and the constraint connecting these sites is the hinge. If one inserts constraints in an arbitrary order during the pebble game, it is possible that the hinge is inserted last (after all other central-force and angular constraints), in which case it will not be covered by a pebble and according to the analysis in Sec. V, will cause an error in the pebble game floppy mode count, as well as rigid cluster decomposition (just as if the hinge were implicit). This is why it is important to insert constraints in a proper order for the pebble game to be correct for bond-bending networks.

Another spectacular case is a chain of *edge-sharing tetrahedra*. Consider a covalent network consisting of atoms of valence 4 (such as Si or Ge) and 2 (Se, Te, S, or O). Suppose there is perfect chemical order, i.e., each atom of valence 4 is always next to an atom of valence 2, and vice versa. Suppose also that there are both central-force and angular constraints associated with atoms of valence 4. Then an atom of valence 4 with all its associated constraints forms a rigid object that can be thought of as a rigid tetrahedron, where the atom itself is at its center and the four neighbors (all of which are atoms of valence 2) are the vertices. Then the network can be represented as a system of connected tetrahedra. These tetrahedra can be corner sharing [Fig. 19(a)] or edge sharing [Fig. 19(b)]. Suppose now that angular constraints at atoms of valence 2 are weaker and can be neglected in a crude approximation (this is often the case when these are oxygen atoms [48]). Then in the edge-sharing case one ends up with chains of edge-sharing tetrahedra that can rotate with respect to each other around a common edge. Two tetrahedra with a
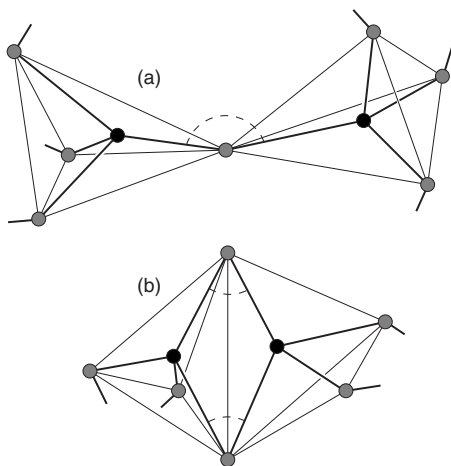
FIG. 19. Fragments of networks consisting of atoms of valence 2 and 4. (a) A pair of corner-sharing tetrahedra. (b) A pair of edge-sharing tetrahedra. In both cases, thick lines are first-neighbor constraints and thin lines are second-neighbor (angular) constraints; black atoms have valence 4 and have all associated angular constraints present, while gray atoms have valence 2 and their angular constraints are missing, so that angles marked with dashed arcs are not constrained. It is implied that these pairs of tetrahedra are connected to the rest of the network, as shown by short black lines stemming from atoms of valence 2. In particular, in the edge-sharing case, each of the two tetrahedra shares an edge with yet another tetrahedron, and thus a chain of edge-sharing tetrahedra will be formed.

common edge are topologically equivalent to a double-banana graph, with the common edge being the hinge. So a chain of edge-sharing tetrahedra is actually a chain of bananas. There are as many floppy modes (in addition to rigid body motions of the whole chain) as there are hinges (or one less than there are tetrahedra in the chain). In the worst-case scenario, about a half of all hinges may end up being uncovered during the pebble game, and then the pebble game will miss about one-half of the floppy modes. Note that here the hinges are *second* neighbor constraints, so it is safer to insert second-neighbor constraints first to make sure they are covered—a situation opposite to that with purely bond-bending networks. Of course, the worst-case scenario is unlikely—in reality, in the case of random insertion, it can be estimated numerically that, in long chains, about 13.5% of the floppy modes are going to be missed.

VIII. CONCLUSION

In this paper, we have described an extension of the pebble game algorithm for rigidity analysis that was used previously for the special class of bond-bending networks in 3D. The algorithm is applicable to general 3D networks, but is approximate: there are networks for which there are errors in the number of floppy modes, rigid cluster decomposition, and/or finding stress. We have also introduced a slower but exact algorithm, the relaxation algorithm. Unlike the pebble game, it is not an integer algorithm (it involves floating-point operations), but it has a number of built-in consistency checks, so errors due to rounding are unlikely in the final result. The relaxation algorithm can be used for comparison with the pebble game using a few representatives of a particular class of networks, before the latter algorithm is used more extensively.

Using the relaxation algorithm and other considerations, we have argued that, for randomly diluted central-force networks, the pebble game algorithm is essentially exact, as far as the percolating cluster size, stressed bonds, and the number of floppy modes are concerned; errors are possible, but extremely rare. Applying the pebble game to bond-diluted networks, we conclude that the rigidity percolation transition on such networks is first order, in contrast to bond-bending networks in 3D and central-force networks in 2D. In fact, the transition is actually first order geometrically, but second order physically, as it is known from previous work that the elastic constants change continuously at the transition. On the other hand, for site-diluted networks, the order of the transition depends on the lattice type: first order for bcc and second order for fcc. The dependence of the order of the transition on the lattice type and the disorder type would be highly unusual and even though the evidence we present is rather strong, further research is needed to confirm our results.

At the same time, there are networks for which the pebble game is less successful. In particular, this is so for bond-bending networks, when the proper order of constraint insertion is not obeyed, and partially bond-bending networks, one example of which, chains of edge-sharing tetrahedra, is discussed in the paper. What makes the difference between the "good" and the "bad" networks? In randomly diluted central-force networks, medium-sized and large nonpercolating clusters are relatively rare, even in site-diluted fcc nets in which they are much more frequent than in the other three cases considered here—much of the network is in clusters of sizes below four or in the percolating cluster. It is even more rare for two of such medium or large clusters to touch at exactly two places and form a hinge. But in bond-bending and partially bond-bending networks, every site of coordination at least four with its angular constraints present is associated with a cluster of at least size five, moreover, the vicinity of such a site is always stressed. If at the same time the average coordination of the network is low so the network is floppy overall, there will be many implied or stressed explicit hinges between such medium-sized clusters that can lead to errors in the pebble game. Arguably, such situations are less frequent than those in which such hinges are rare, but in each case tests should be run before the pebble game algorithm is used.

Of course, ideally one would desire an exact integer algorithm for rigidity analysis. Efforts to design such an algorithm have not paid off so far, and the authors would argue based on this experience that developing a polynomial-time pebble-game-type algorithm, while very interesting for the computer science and mathematical rigidity theory community, would be useless in practice: it is unlikely that such an algorithm would be sufficiently fast to beat the relaxation algorithm which is already appropriate for many purposes.

APPENDIX: SMALL CLUSTERS IN THE CENTRAL-FORCE DILUTED bcc AND fcc NETWORKS

Here we look in more detail at possible small rigid clusters and stressed regions in central-force diluted bcc and fcc networks. As we have mentioned, the reason the rigidity transition is first order in the previously considered case of random bond networks (RBNs) has to do with the absence of finite rings and thus finite rigid clusters (other than single sites, bonds, and triangles) in these networks. While the same clearly cannot be true for regular bcc and fcc lattices, it is interesting to find out if there are any similarities.

We first find what cluster sizes are possible. For this, we use a computational procedure outlined below. Its advantage compared to just looking at rigid cluster decomposition of many networks is that, even if clusters of certain size are extremely rare and would never be seen in reality, they can still be found with our approach. The problem of generating rigid clusters resembles that of generating lattice animals in ordinary, connectivity percolation [49].

Basically, the idea of the approach is to generate the most rigid configuration of a given size through an optimization procedure. First, note that a rigid cluster with n sites should contain at least $3n-6$ constraints (if it is rigid by itself), or $3n-7$ constraints, if it shares a hinge with some other rigid cluster and is rigidified by it. So, if rigid clusters of size n are possible, the maximum possible number of constraints (where the maximum is taken over all possible graphs with n sites on the full lattice) should be at least $3n-6$ (or respectively $3n-7$). Then the purpose is to maximize the number of constraints (equal to the number of bonds in central-force networks) for given n. This is done as follows. First of all, note that, if the positions of the n sites on the lattice are fixed, the optimum bond configuration consistent with those fixed positions will be the one containing all possible bonds connecting the sites in these fixed positions. Then the idea is to move sites around on the lattice and while doing so try to maximize the number of possible connections between these sites. This can be done using an analog of the simulated annealing procedure [50], where the "energy" that needs to be minimized is the negative number of connections for the given configuration of sites (as this is the quantity being optimized). If the optimization is done carefully, we can be sure that the optimum number of bonds for given n is reached. If this optimum number is less than $3n-7$, then a rigid cluster with n sites is not possible (we neglect the possibility of a cluster being rigidified by more than one external

rigid cluster: at least in the bond-diluted case, rigid clusters of medium and large size, other than the single percolating cluster, are rare, so two clusters coming together to rigidify a third one is even more rare; even in the fcc site case, when medium-sized clusters are not rare, we have not seen such a situation). If the optimum number is exactly $3n-7$, then a stand-alone rigid cluster of size n is not possible, but a cluster rigidified from the outside (say, by the percolating cluster) and sharing a hinge with it (a situation similar to that in Fig. 4) is possible. If the optimum number of bonds is exactly $3n-6$, then a stand-alone rigid cluster is possible, but any such cluster will be isostatic (stress is impossible). Finally, if the optimum number is $3n-5$ or higher, then even a stressed region of size n is possible. One caveat is that even a network with, say, $3n-6$ bonds may be floppy, if it contains a stressed region. Such situations are, however, easy to detect: start with the smallest n, find the maximum number of bonds, then go to $n+1$, etc.; if the difference between the maximum number of bonds and the number of degrees of freedom ($=3n$) ever goes *down* with increasing n, this means that the subnetwork actually became floppier with increasing size and this would require more careful checking.

Using the above procedure, we have found possible sizes of rigid clusters and stressed regions for both fcc and bcc networks. For bcc lattices, clusters of size two (single bonds) are, of course, possible, but stand-alone clusters of size three are not, because the bcc lattice does not contain triangles. However, clusters of size three rigidified by another cluster are possible (indeed, we have observed many clusters of size three sharing a trivial implied hinge with the percolating cluster, as we have discussed in Sec. VI B). Clusters of sizes 4, 5, etc. (either stand alone or not) are not possible, however; the next possible non-stand-alone cluster size is 84 and the next possible stand-alone cluster size is 90. The smallest possible stand-alone stressed region has size 96. So even though, unlike in RBNs, rings of small sizes are possible, small rigid clusters (other than trivial single sites, single bonds, and triangles) are still not possible, as in RBNs. Of course, in RBNs clusters of any finite size above three are not possible, and here clusters of size 84 and higher can exist at least in principle. But in practice, such clusters are extremely rare, as we show below.

For fcc lattices, the situation is different. It turns out that rigid clusters of all sizes are possible (except stand-alone clusters of size five are not possible, but non-stand-alone clusters of this size are still possible). However, using the relaxation algorithm, we have seen that the frequency of clusters decreases very rapidly with their size: there are a lot of clusters of size two and three, clusters of size four are much more rare, and there are almost no clusters of a larger size. Stand-alone stressed regions, on the other hand, start at size ten and these are much more rare (we give a frequency estimate below). Non-stand-alone stressed regions are essentially double-banana situations, and we discuss them separately, in Sec. VI B and later in this appendix.

Obviously, the optimization procedure described above not only answers the question about the possibility of a cluster of a certain size, but also produces a realization of such a cluster when the answer is positive. If the optimization procedure is carried out many times starting from different ini-

TABLE I. Frequencies per site of different cluster sizes in 500-site bond-diluted fcc networks obtained both with the relaxation algorithm (using the 1812 nonpercolating networks out of a sample of 3100 networks with numbers of bonds between 1460 and 1490) and with the pebble game (using the 9 242 577 nonpercolating networks out of a sample of 10^7 networks with 1470 bonds). This is compared to the theoretical estimates obtained using Eq. (A1), with quantities c_n obtained using the numerical simulated annealing procedure described in the text, and with the bond number $n_B = 3n - 6$.

Cluster size n	Relaxation		Pebble game		c_n	Theoretical frequency
	No. of clusters	Frequency	No. of clusters	Frequency		
3	747196	8.2×10^{-1}	3824801279	8.3×10^{-1}	8	9.4×10^{-1}
4	24665	2.7×10^{-2}	125596870	2.7×10^{-2}	2	2.8×10^{-2}
5	0	0	0	0	0	0
6	67	7.4×10^{-5}	318927	6.9×10^{-5}	1	19×10^{-5}
7	78	8.6×10^{-5}	329476	7.1×10^{-5}	8	18×10^{-5}
8	26	2.9×10^{-5}	134915	2.9×10^{-5}	28	7.4×10^{-5}
9	5	5.5×10^{-6}	36639	7.9×10^{-6}	80	25×10^{-6}
10	1	1×10^{-6}	13145	2.8×10^{-6}		
11	1	1×10^{-6}	7334	1.6×10^{-6}		

tial configurations and using different random number sequences in the simulated annealing, eventually the set of final configurations will reproduce the complete set of possible rigid clusters. This allows an estimate of probabilities of certain clusters.

We consider the case of bond dilution first and then make a comparison to site dilution at the end of this appendix. For simplicity, we will consider the situation when all clusters of a given size n have the same number of bonds n_B equal to the maximum possible one. In fcc networks, for instance, this will be the case for all clusters of sizes smaller than ten (the smallest stressed region). We also consider stand-alone clusters only at this point. Suppose the total number per site of possible rigid cluster configurations of size n is c_n (this includes all configurations related by symmetry). Then the frequency per site of clusters of size n, ν_n, can be estimated as

$$\nu_n \approx c_n p^{n_B}, \qquad (A1)$$

where p is the fraction of present bonds in the network. This involves three assumptions: (a) such clusters are rare so "excluded volume" effects can be neglected; (b) the cluster frequency decreases rapidly with increasing cluster size, so that the probability of clusters of size larger than n can be neglected when calculating the probability of clusters of size n; (c) there is no percolating cluster.

Table I gives the frequencies observed in simulations for fcc networks using both the relaxation algorithm and the pebble game, compared to the estimates obtained using Eq. (A1) with c_n calculated using the simulated annealing procedure described above and assuming $n_B = 3n - 6$, which is true for $n < 10$ (for this reason, we do not give estimates for $n = 10$ and above). For the simulation results, we use just nonpercolating networks, since Eq. (A1) ignores the possibility of existence of the percolating cluster. We also note that nearly all clusters in nonpercolating networks stand alone, which is another assumption behind Eq. (A1). For the relaxation study, we use the sample of 3100 500-site networks

described in Sec. VI B. Of these, there are 1812 nonpercolating networks, with a total of 906 000 sites. Thus, to obtain the frequency per site, the total number of observed clusters of a given size is divided by 906 000. For the pebble game study, we also use 500-site networks, but a much higher number of them: 10^7 networks, of which more than 9×10^6 are nonpercolating (the much higher percentage of nonpercolating networks compared to the sample used in the relaxation study is simply due to the fact than in the pebble game runs, we fix the bond number at 1470, which is slightly below the transition, whereas in the relaxation runs, the number of bonds varies between 1460 and 1490 and this spans the whole transition region). The much better statistics we are able to obtain with the pebble game illustrates its advantage compared to the relaxation algorithm. At the same time, it can be seen that the frequencies obtained using these two methods are essentially the same, apart from the statistical noise; this is yet another confirmation of the accuracy of the pebble game algorithm. For the theoretical estimate, we use $p = 0.49$; this corresponds to 1470 bonds, which is the same number as used in the pebble game runs and close to the average over the relaxation runs. It is seen that the frequency tends to be overestimated somewhat, especially for larger sizes, so rare larger clusters, hinges, etc., may even be more rare than the estimates below suggest.

We can now use our procedure to estimate the frequencies of finite clusters of size larger than three in bcc networks. We use $p = 0.745$. The smallest stand-alone cluster has size 90 and $3 \times 90 - 6 = 264$ bonds; the optimization procedure gives $c_{90} = 27$. Then $\nu_{90} \approx 27 \times 0.745^{264} \approx 5 \times 10^{-33}$. The next possible size is 94, with 276 bonds; $c_{94} = 720$; then $\nu_{94} \approx 4 \times 10^{-33}$. These probabilities are extremely small; obviously, the existence of finite clusters of size larger than three can be neglected for any practical purposes, and in this respect, bcc networks are exactly like RBNs, even though they are "normal," regular networks with loops.

As noted in Sec. VI B, we have not observed any stressed regions in nonpercolating networks in our relaxation runs. In

the much larger series of pebble game runs mentioned above, there were 1331 stressed regions in the 9 242 577 nonpercolating networks of 500 sites each, or about one region per 3.5×10^6 sites. All nonpercolating networks analyzed with the relaxation algorithm combined contain only 906 000 sites, so the absence of stressed regions in our relaxation runs is not surprising. In fact, remember that the smallest stressed regions are of size ten, and we have seen only two rigid clusters of size ten and larger (both turned out to be unstressed).

A similar simulated annealing procedure can be used to find possible double-banana-type networks. For this, a rigid cluster is chosen and fixed to serve as the first banana; two sites belonging to it are also fixed as hinge ends. A set of sites is then allowed to move around on the lattice, as in the previous case, but with the restriction that no sites can ever coincide with any of the sites belonging to the first banana, except for two of them that should always coincide with the assigned hinge ends. This should be tried for all possible first bananas and, for each first banana, for every possible pair of hinge ends. The result of this procedure is that in fcc networks, the smallest double-banana-type graph with an implied hinge that will violate the floppy mode count when the pebble game is run consists of two eight-site bananas, as shown in Fig. 10. We have already mentioned in Sec. VI B that these are very rare—probably less than one in 10^9 sites. On the other hand, the smallest two-cluster graph with a hinge that cannot influence the correctness of the floppy mode count, but can influence the correctness of rigid cluster decomposition, consists of an eight-site cluster and a triangle. These should be more frequent, although we have not seen them in our relaxation simulations, even though we have seen 26 eight-site clusters; there were a few hinges of this type involving the 10- and 11-site clusters (each size seen once in our relaxation simulations). Still, the presence of such a hinge only means missing a rigid triangle at most.

So far, we have mostly concentrated on the floppy phase. We now look at the small clusters in the rigid phase, again for obvious reasons concentrating on fcc networks. In the rigid phase, the percolating cluster takes up most of the network. Small clusters larger than triangles are rather rare. In all 1288 percolating networks, only 14 four-site clusters were observed, and larger clusters were never seen. Part of the reason is obviously the fact that only a small part of each network is not in the percolating cluster. Perhaps more importantly, for a small rigid cluster to exist, not only should there be enough bonds locally for the configuration to be rigid (which was enough in the floppy phase), but also this configuration should be only sparsely connected to the rest of the network, or else it will be part of the percolating cluster instead. So the conditions of existence for larger nonpercolating clusters are rather stringent and become progressively more stringent as the size increases, since it becomes harder and harder to ensure that the region is sufficiently disconnected from the rest of the network.

For example, the smallest graph of the double-banana type involving the percolating cluster is the combination of the percolating cluster and a six-site cluster (an octahedron), with the hinge between two opposite vertices of the octahedron. Octahedra are rare in the floppy phase, but not totally

TABLE II. Frequencies per present site of different cluster sizes in site-diluted fcc networks, obtained using the relaxation algorithm in 1000 realizations on 500-site lattices with 235 present sites.

Cluster size n	Observed no. of clusters	Observed frequency
3	159147	6.8×10^{-1}
4	20881	8.9×10^{-2}
5	1	4×10^{-6}
6	23	9.8×10^{-5}
7	202	8.6×10^{-4}
8	375	1.6×10^{-3}
9	273	1.2×10^{-3}
10	197	8.4×10^{-4}
11	161	6.9×10^{-4}

unseen (there is roughly one per 10 000 sites, according to Table I). But this should not be a reason for concern: octahedra, especially those sharing an implied hinge with the percolating cluster, are much more rare in the rigid phase, for reasons explained above. As mentioned, we have not seen them in our relaxation simulations. In a much larger pebble game run, with 3×10^7 networks of 1372 sites each, and about 20% of networks percolating, there were 21 octahedra in percolating networks, or about one per 4×10^8 sites. This count does not include the double-banana situations (if any), since they are not detected by the pebble game. But it is expected to include most of the rest of the octahedra: it may miss those sharing an uncovered explicit hinge with the percolating cluster, but these are rather rare, since not so many constraints are redundant and thus uncovered close to the transition; any other possibilities are extremely rare. Since there is no reason to believe that octahedra attached specifically at two opposite vertices to the percolating cluster (and thus forming a double-banana graph) constitute a majority of all octahedra, the frequency of double-banana graphs of this kind is certainly less than 10^{-8} per site.

In the case of fcc site dilution, on the other hand, the number of medium-sized clusters is much higher. Table II shows the frequencies for the same cluster sizes as in Table I obtained using the relaxation algorithm in 1000 realizations on 500-site fcc lattices, all with the fraction of present sites $p=0.47$ (235 sites), slightly below the transition (see Fig. 17). The frequency is given per *present* site; unlike in Table I, both percolating and nonpercolating networks were used (averages over nonpercolating networks only would likely be even higher). While the frequencies of six-site clusters are about the same in the bond and site cases, eight-site clusters are about 50 times more frequent in the site case, and the ratio reaches about 300 by $n=10$. Even though analytical estimates of the type done for bond dilution are much harder to make in the site case, since the assumption about the cluster frequency decreasing rapidly with the size is no longer valid, qualitatively it is clear what makes the difference between the two cases. Assume that a certain set of n sites can be mutually rigid in principle in the fcc network. In the site dilution case, for this set to actually be rigid, these n sites

just need to be present, since all connections between them are guaranteed automatically—the probability of this is p^n. But in the case of bond dilution, bonds can be present or absent independently. At least $3n-6$ of them need to be present, and for $n<10$, this is also the maximum total number of bonds that a cluster of size n can have. Then the probability of this is p^{3n-6}. Given that the transition in the site and bond cases occurs at about the same value of $p \approx 1/2$, it is clear that clusters of the same size are more frequent in the site case. These arguments are clearly very simplistic: for the cluster to be of exactly size n, not only should the n sites be mutually rigid, but no other sites should be rigid with respect to them all (taking this into account is especially important in the site case); also the arguments need to be modified for $n \geq 10$, when the maximum possible number of bonds is not $3n-6$. But the main idea remains valid.

[1] R. J. Baxter, *Exactly Solved Models in Statistical Mechanics* (Academic Press, New York, 1982).

[2] M. F. Thorpe, J. Non-Cryst. Solids **57**, 355 (1983).

[3] M. F. Thorpe, D. J. Jacobs, N. V. Chubynsky, and A. J. Rader, in *Rigidity Theory and Applications*, edited by M. F. Thorpe and P. M. Duxbury (Kluwer Academic/Plenum, New York, 1999), p. 239.

[4] M. F. Thorpe, D. J. Jacobs, M. V. Chubynsky, and J. C. Phillips, J. Non-Cryst. Solids **266-269**, 859 (2000).

[5] M. V. Chubynsky and M. F. Thorpe, Curr. Opin. Solid State Mater. Sci. **5**, 525 (2001).

[6] D. J. Jacobs, A. J. Rader, L. A. Kuhn, and M. F. Thorpe, Proteins **44**, 150 (2001).

[7] A. J. Rader, B. M. Hespenheide, L. A. Kuhn, and M. F. Thorpe, Proc. Natl. Acad. Sci. U.S.A. **99**, 3540 (2002).

[8] M. Latva-Kokko, J. Mäkinen, and J. Timonen, Phys. Rev. E **63**, 046113 (2001).

[9] M. Latva-Kokko and J. Timonen, Phys. Rev. E **64**, 066117 (2001).

[10] M. Alava and K. Niskanen, Rep. Prog. Phys. **69**, 669 (2006).

[11] D. A. Head, A. J. Levine, and F. C. MacKintosh, Phys. Rev. Lett. **91**, 108102 (2003).

[12] B. Servatius and H. Servatius, in *Rigidity Theory and Applications* (Ref. [3]), p. 1.

[13] J. Graver, *Counting on Frameworks: Mathematics to Aid the Design of Rigid Structures* (Mathematical Association of America, Washington, DC, 2001).

[14] W. Whiteley, Phys. Biol. **2**, S116 (2005).

[15] J. C. Maxwell, Philos. Mag. **27**, 294 (1864).

[16] G. Laman, J. Eng. Math. **4**, 331 (1970).

[17] T.-S. Tay and W. Whiteley, Structural Topology **9**, 31 (1984).

[18] W. Whiteley, in *Rigidity Theory and Applications* (Ref. [3]), p. 21.

[19] D. J. Jacobs and M. F. Thorpe, Phys. Rev. Lett. **75**, 4051 (1995).

[20] D. J. Jacobs and B. Hendrickson, J. Comput. Phys. **137**, 346 (1997).

[21] Software for use in analyzing the flexibility of networks, including various pebble games, is available through http://flexweb.asu.edu/

[22] D. J. Jacobs, J. Phys. A **31**, 6653 (1998).

[23] D. J. Jacobs, L. A. Kuhn, and M. F. Thorpe, in *Rigidity Theory and Applications* (Ref. [3]), p. 357.

[24] D. J. Jacobs (private communication).

[25] D. J. Jacobs and M. F. Thorpe, Phys. Rev. E **53**, 3682 (1996).

[26] B. M. Hespenheide, D. J. Jacobs, and M. F. Thorpe, J. Phys.: Condens. Matter **16**, S5055 (2004).

[27] S. Feng and P. N. Sen, Phys. Rev. Lett. **52**, 216 (1984).

[28] S. Feng, M. F. Thorpe, and E. Garboczi, Phys. Rev. B **31**, 276 (1985).

[29] M. F. Thorpe and M. V. Chubynsky, in *Properties and Applications of Amorphous Materials*, edited by M. F. Thorpe and L. Tichý, NATO Science Series, II. Mathematics, Physics and Chemistry Vol. 9 (Kluwer Academic, Dordrecht, 2001), p. 61.

[30] M. Micoulaut, Europhys. Lett. **58**, 830 (2002).

[31] M. Micoulaut and J. C. Phillips, Phys. Rev. B **67**, 104204 (2003).

[32] J. Barré, A. R. Bishop, T. Lookman, and A. Saxena, Phys. Rev. Lett. **94**, 208701 (2005).

[33] M. V. Chubynsky, M.-A. Brière, and N. Mousseau, Phys. Rev. E **74**, 016116 (2006).

[34] C. Moukarzel, P. M. Duxbury, and P. L. Leath, Phys. Rev. E **55**, 5800 (1997).

[35] P. M. Duxbury, D. J. Jacobs, M. F. Thorpe, and C. Moukarzel, Phys. Rev. E **59**, 2084 (1999).

[36] M. Chubynsky and M. F. Thorpe, in *Physics and Applications of Disordered Materials*, edited by M. Popescu (INOE, Bucharest, Romania, 2002), p. 229.

[37] M. V. Chubynsky, Ph.D. thesis, Michigan State University, 2003, URL http://phy.asu.edu/mfthorpe/Nikita_thesis.pdf

[38] M. F. Thorpe and E. J. Garboczi, Phys. Rev. B **35**, 8579 (1987).

[39] S. Arbabi and M. Sahimi, Phys. Rev. B **47**, 695 (1993).

[40] B. Jackson and T. Jordán, Adv. Appl. Math. **35**, 355 (2005).

[41] S. Pellegrino, Int. J. Solids Struct. **30**, 3025 (1993).

[42] W. H. Press, S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery, *Numerical Recipes in Fortran 77*, 2nd ed. (Cambridge University Press, Cambridge, U.K., 1992), Sec. 10.6.

[43] G. Golub and C. Van Loan, *Matrix Computations*, 3rd ed. (Johns Hopkins University Press, Baltimore, 1996).

[44] S. Wells, S. Menor, B. Hespenheide, and M. F. Thorpe, Phys. Biol. **2**, S127 (2005).

[45] F. Y. Wu, Rev. Mod. Phys. **54**, 235 (1982).

[46] C. Moukarzel and P. M. Duxbury, Phys. Rev. Lett. **75**, 4055 (1995).

[47] D. A. Head, F. C. MacKintosh, and A. J. Levine, Phys. Rev. E **68**, 025101(R) (2003).

[48] A. Sartbaeva, S. A. Wells, M. M. J. Treacy, and M. F. Thorpe, Nat. Mater. **5**, 962 (2006).

[49] D. Stauffer and A. Aharony, *Introduction to Percolation Theory*, 2nd ed. (Taylor & Francis, London, 1992), Chap. 2.

[50] S. Kirkpatrick, C. D. Gelatt, Jr., and M. P. Vecchi, Science **220**, 671 (1983).