

Fast Algorithm for 2D Rigidity Percolation

Nina Javerzat^{*,‡}

Université Grenoble Alpes, CNRS, LIPhy, 38000 Grenoble, France

Daniele Notarmuzi^{†,‡}

Institut für Theoretische Physik, TU Wien, Wiedner Hauptstraße 8-10, A-1040 Wien, Austria



(Received 2 July 2025; accepted 11 June 2026; published 10 August 2026)

Rigidity percolation is a crucial framework for describing rigidity transitions in amorphous systems. We present an efficient algorithm to study central-force rigidity percolation in two dimensions. This algorithm combines the pebble game algorithm, the Newman-Ziff approach to connectivity percolation, as well as novel rigorous results in rigidity theory, to exactly identify rigid clusters over the full bond concentration range, in a time that scales as $N^{1.02}$ for a system of N nodes. We perform extensive numerical simulations with systems larger than 500 million nodes, far beyond the previous limitations. We obtain precise estimates for the critical exponents $\nu = 1.1694(8)$ and $D_f = 1.8423(7)$ and locate the critical threshold at $p_c = 0.660\,274\,1(4)$. Besides opening the way to further accurate numerical studies of rigidity percolation, our work provides rigorous theoretical insights on specific cluster merging mechanisms that distinguish it from the standard connectivity percolation problem.

DOI: [10.1103/sv88-j2wj](https://doi.org/10.1103/sv88-j2wj)

Subject Areas: Complex Systems,
Condensed Matter Physics, Soft Matter

I. INTRODUCTION

Percolation is one of the most versatile conceptual frameworks in statistical physics and describes a wealth of natural phenomena, from polymer gelation to the propagation of diseases. In standard—connectivity—percolation (CP), the central objects are connectivity clusters, namely, sets of mutually connected nodes. As more and more bonds are added to the network, clusters grow and coalesce, and a system-spanning (macroscopic) cluster eventually emerges at a critical value of the bond concentration.

While CP theory provides an extensive framework to characterize the connectivity properties of networks [1,2], rigidity percolation (RP) addresses instead their *mechanical stability*: Is the network able to transmit stresses and sustain external loads? It was realized early that this question can be formulated as a constraint-counting problem [3–5]; the rigidity of a network depends only on its topology and is largely independent of the precise physical nature of bonds. In central-force networks, on which we focus in this work, nodes possess d degrees of freedom in d dimensions, while

bonds introduce constraints by fixing the distances between pairs of nodes.

Rigid clusters are sets of mutually rigid bonds, that is, subnetworks in which all degrees of freedom are fixed by the constraints except for the $d(d+1)/2$ global ones. Floppy clusters, instead, have unfixed degrees of freedom that allow for deformation along soft (floppy) modes. While at low bond concentration the system is typically macroscopically floppy (i.e., composed of small, nonrigidly connected rigid clusters), increasing the concentration above a critical threshold leads to the emergence of a macroscopic rigid cluster (see, e.g., Ref. [6]). Rigidity percolation has attracted much attention in soft matter, as a simple and versatile framework to model liquid to amorphous solid transitions in systems such as colloidal gels [7–9], living tissues [10–13], fiber networks [14,15], or granular packings [16–19]. Notably, it has recently been revealed how the RP critical exponents control the viscoelastic properties of colloidal gels in their solid phase [20,21]. Moreover, recent works gather evidence for a larger role of rigid clusters in the physics of amorphous systems, as controlling, e.g., the behavior at the brittle-ductile transition [22] or the large stress fluctuations at the shear thickening transition [23].

Inquiring about mutual rigidity, rather than mutual connectivity, makes RP much more difficult than the usual CP problem. A distinguishing feature of RP is its non-locality: As we make precise in Sec. IV, the activation of a single bond can trigger the rigidification of arbitrarily large spatial regions, by coalescence of numerous rigid clusters into a single one. Likewise, cutting a single bond can lead

^{*}Contact author: nina.javerzat@univ-grenoble-alpes.fr

[†]Contact author: daniele.notarmuzi@tuwien.ac.at

[‡]These authors contributed equally to this work.

to cascades of rigidity loss [24]. As a matter of fact, RP remains largely less understood than CP. Rigorous results are rather scarce: The existence of a critical bond concentration strictly above the CP threshold has been proven on the 2D triangular lattice [25]; the uniqueness of the infinite rigid cluster has been proven in Ref. [26]. Exact results have been obtained on nongeneric networks such as hierarchical and Erdős-Rényi graphs [27,28] (see also Refs. [29,30] for connections with satisfiability problems).

In most cases however, especially those relevant to soft matter applications, the characterization of the RP universality classes relies essentially on numerical simulations, either to determine the values of critical exponents [6,17] or to explore fine aspects of the universality class such as conformal invariance [31,32].

Yet, identifying efficiently rigid clusters in large systems is not an easy task. Numerical studies were made possible thanks to the pebble game algorithm [6,33] (later generalized to frictional systems [16,34]), a combinatorial implementation of Laman theorem for graph rigidity, that we briefly describe in Sec. II. Using the pebble game algorithm, the time to identify rigid clusters at p_c^{RP} scales with the system size N as $N^{1.15}$ [33]. This enabled estimating the values of the critical exponents for central-force RP, $\nu^{\text{RP}} = 1.21 \pm 0.06$, $\beta^{\text{RP}} = 0.175 \pm 0.02$, and $D_f^{\text{RP}} = 1.86 \pm 0.02$, respectively, the correlation length and order parameter exponents and the fractal dimension of the spanning rigid cluster, with a sufficient precision to claim the existence of a new universality class, distinct from random CP [6]. To our knowledge, there has been no attempt to determine the central-force RP exponents with better precision since then, the barrier being the harsh computational cost of analyzing systems larger than approximately 10^6 nodes [for comparison, accurate studies of the CP critical point involve systems with $O(10^8)$ nodes [35]].

Classifying rigidity transitions as belonging or not to the (uncorrelated [36]) central-force RP universality class is crucial to achieve a better understanding of the variety of liquid-to-amorphous solid transitions [7,10,17,37–40] and to assess the robustness of the central-force RP universality class. Providing accurate numerical standards for the critical exponents, and, more generally, understanding the specificities of the RP problem as compared to the well-studied CP one, are, therefore, major goals.

In this paper, we introduce a new algorithm for RP. It is based on the powerful Newman-Ziff (NZ) approach to CP [41], where bonds are activated one at a time, and the state of the system, namely, the identification of clusters, is fully known at each step.

Studying percolation *à la* Newman-Ziff allows one notably to measure the quantities of interest (probability to percolate, size of the largest component) at each bond activation, namely, for each value of the bond filling fraction (fraction of active bonds), thereby covering the

complete phase diagram in a single run. To obtain a NZ-like algorithm for RP, we first shed new light on the specific cluster merging mechanisms at play, by providing rigorous results that predict the consequences of each new bond activation. Exploiting these results, we construct an algorithm that scales with the system size as $N^{1.02}$, i.e., that covers the entire phase diagram in almost linear time. Our algorithm pushes the size limitation to $N \gtrsim 10^8$, allowing for precise studies of the RP universality class, starting with accurate determination of the main critical exponents.

The paper is organized as follows. In Sec. II, we briefly describe the standard pebble game as defined by Jacobs and Hendrickson [6,33] and explain in more detail how to perform pebble searches that are key to the implementation of our own algorithm. In Sec. III, we recall the Newman-Ziff algorithm for CP. In Sec. IV, we present the three theorems that allow us to design a NZ-like algorithm for RP, namely, that allow us, with minimal computational cost, to deduce the new state of the system after each bond activation. The proofs of those theorems, as well as a brief review of 2D rigidity theory and additional new results, are presented in Supplemental Material [42], Sec. I.

We give an overview of the strategy behind our algorithm in Sec. V, while the implementation is detailed in Sec. VI. In Sec. VII, we adapt the (CP) Machta algorithm [44] to detect the wrapping of rigid clusters. Finally, we discuss the performance of our complete algorithm in Sec. VIII. We conclude with several perspectives in Sec. IX.

II. THE JACOBS AND HENDRICKSON APPROACH

The central challenge in studying RP is the identification of rigid clusters, namely, of the maximal sets of mutually rigid bonds. In 2D, an exact criterion is given by Laman theorem on graph rigidity [5] (cf. Supplemental Material Sec. I [42] for more details). Jacobs and Hendrickson (JH) [33] introduced a clever algorithm to implement efficiently Laman’s criterion, the pebble game: The degrees of freedom of the system are represented as “pebbles” assigned to the nodes. Pebbles are removed each time an independent bond (constraint) is added to the system. Note that, in this formulation, three pebbles can always be gathered across any bond, since the system always has three global degrees of freedom in 2D. Additional pebbles correspond to floppy modes.

A. The pebble game

The JH algorithm is divided into two stages, filling and clustering. All bonds are initially inactive and each node has two pebbles attached to it, representing its two degrees of freedom. In the filling stage, bonds are independently activated with probability p . Each new activated bond is classified as an independent or redundant constraint using *pebble searches* (see the next section): A new bond uv represents an independent constraint if u and v were not

mutually rigid, namely, if four pebbles can be gathered across uv . If the two nodes were already mutually rigid, only three pebbles can be gathered, and the new bond is redundant.

Once all the bonds are activated and classified, rigid clusters are identified. Each rigid cluster is constructed by starting from a given unlabeled bond b and testing the mutual rigidity of the other (unlabeled) bonds of the system with b , by performing pebble searches. Each bond that is found to be mutually rigid with respect to b is given the same label as b . This is conveniently implemented as a breadth-first search that stops when only floppy, or already labeled bonds, can be found.

By construction, one run of the JH algorithm corresponds to a single value of the bond concentration p . The overall cost of the JH algorithm is determined by the number of pebble searches to be performed in a run—which depends on N —as well as the cost of each search—which also depends on N . JH reported a scaling of the algorithm that depends on p , scaling as $N^{1.15}$ near the critical point $p \approx p_c$, and linearly near full concentration. In practice, this means that studying the phase diagram with the JH algorithm requires one to run it for (as many as possible) different values of p , with a total computing time that depends on the number of values considered, most of which being typically concentrated near the critical point with cost $N^{1.15}$ each.

Our algorithm solves these two limitations at once. First, the whole phase diagram is obtained in a single run. This comes from adopting a Newman-Ziff approach, which we describe in the next section. Second, we make such an approach efficient by exploiting rigidity theory to make a strategic use of pebbles searches, keeping their number low.

B. Pebble searches

Pebbles searches are used to check for mutual rigidity. These searches take place on the pebble graph, distinct from the physical lattice. Every time an independent bond of the system is activated, a directed edge is added to the pebble graph and a pebble is removed from one of the end nodes, accounting from the removal of one degree of freedom. The edge is initially directed away from the node that loses a pebble. For later use, we distinguish two types of pebble searches, whose dynamics is illustrated in Fig. 1.

1. Type I searches

The purpose of pebble searches of type I is to actually move the pebbles across the pebble graph, in order to keep track of the repartition of degrees of freedom. To gather a pebble at node j , a breadth-first search (BFS) [45] starting at j is performed over the pebble graph, with the aim of finding a path that leads from node j to a node that has a pebble [Fig. 1(a)]. If a pebble is found, the direction of the pebble edges along the path is reversed and the pebble is

moved from the node where it has been found to node j [Fig. 1(b)].

2. Type II searches

The purpose of pebble searches of type II is to test mutual rigidity without actually moving the pebbles across the pebble graph. To check mutual rigidity of bonds b_1 and b_2 , three pebbles (that are always available) are first gathered at b_1 using type I searches. These pebbles are temporarily frozen at their location; i.e., they are made unavailable to the rest of the system, while the nodes of b_1 are marked as rigid.

Then, a search is performed for a fourth pebble that could be gathered at the nodes of b_2 : If such a pebble exists, bonds b_1 and b_2 are mutually floppy; otherwise, they are mutually rigid.

Note that, for b_2 to be rigid with respect to b_1 , both its nodes must be rigid. If one node is rigid and the other is floppy, the bond is overall floppy. This implies that, in contrast to *CP*, nodes in *RP* can belong to more than a rigid cluster. Each bond, however, belongs to a single rigid cluster. Nodes belonging to more than a rigid cluster are called pivots [33].

The search proceeds as in the previous section, but the direction of pebble edges is never changed by type II searches, as pebbles are not actually moved. Moreover, if the search is successful (a pebble turns out to be available), all nodes along the path are marked as floppy [Fig. 1(e)]. Indeed, the found pebble could be gathered at any of them. On the contrary, after a failed search, all the visited nodes are marked rigid [Fig. 1(f)]: No pebble can be gathered at any of those visited nodes.

This marking of the nodes reduces the cost of subsequent searches that can be terminated whenever a node, already marked as floppy during a previous search, is encountered. Moreover, edges of the pebble graph that start at nodes marked rigid during previous searches are not walked, as they cannot lead to any pebble nor to a floppy node [33].

3. Triangulation

Note that both types of searches can, in principle, result in a pebble not being found.

When a pebble search fails, all the pebble edges traversed during the search are triangulated over the bond that has three pebbles [Fig. 1(c)]. The triangulation procedure is essential to keep pebble searches short [33].

III. THE NEWMAN-ZIFF ALGORITHM FOR CP

In this section, we recall briefly the NZ approach to *CP* [41,46]. NZ simulated percolation as a dynamical process, where bonds are activated one by one, and the state of the system is updated at each step, following simple rules. The algorithm allows one, therefore, to generate instances of the system at every filling fraction and does so

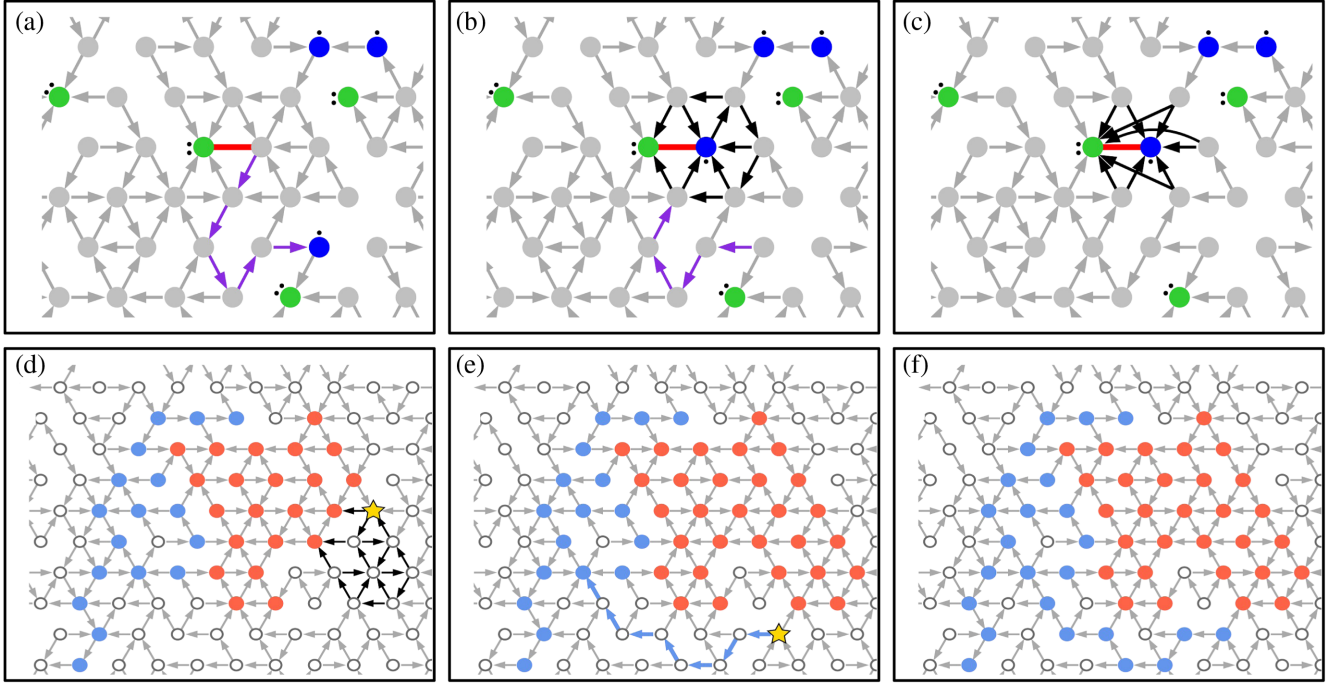


FIG. 1. Dynamics of pebble searches on the pebble graph. Top: searches of type I. Bottom: searches of type II. To ease visualization, bonds are not triangulated in the bottom panels. (a) Green nodes have two pebbles, blue nodes have one, and gray have zero. Pebbles are depicted as black dots. The red undirected bond is the newly activated bond. A pebble search of type I starts at the gray end node of the new bond. The purple path leads to a pebble. (b) The path identified in (a) is reversed, the node where it started has now one pebble, and a second search is started. The search traverses the black edges without finding a pebble. (c) The edges traversed during the previous search are triangulated over the base made by the new bond. (d) Red (blue) nodes have been marked rigid (floppy) during previous searches, while open circles represent unmarked nodes. A pebble search of type II starts at the yellow star. The search traverses the black edges without finding a floppy node. Note that edges originating from rigid nodes are not walked. (e) The nodes visited during the search in (d) are marked rigid. The visited edges are triangulated (not shown). A new search starts at the yellow star and, following the blue path, stops when it hits a floppy node. (f) The nodes traversed by the path identified in (e) are marked floppy.

in linear time. Measurements are then convoluted with the binomial distribution to be transformed in functions of the bond concentration p [41]. The key idea in the NZ algorithm for CP , which we adapt to RP , is to describe each cluster as a directed tree: All nodes belonging to a given connectivity cluster point to the same root in the corresponding tree. Each cluster is identified by its root, whose entry in the tree gives (minus) the size of the cluster.

When a new bond uv is activated, trees (clusters) must be updated accordingly. These updates are simple in CP , as only two events can happen. If u and v already belong to the same cluster (they point to the same root), the corresponding tree is unchanged. If u and v belong to two distinct clusters, C_u and C_v (they point to different roots), the new bond coalesces them into $C_u \cup C_v \cup \{uv\}$. Coalescence is performed by redirecting the root of the smallest cluster toward the root of the largest one.

The roots of u and v are identified by the so-called find operation, i.e., by traversing the tree from u (v) to the corresponding root. The find operation is coupled with the so-called path compression: Once the root is found, the tree is compressed by redirecting all the traversed edges toward

the identified root. This mechanism allows the find operation to have constant cost even on large trees [41].

IV. THEOREMS FOR SINGLE BOND ACTIVATION

In this paper, we adapt the NZ strategy described in the previous section to rigidity percolation. Namely, starting with an empty lattice and progressively activating the bonds, we are able to deduce the new state of the system after each new bond activation. While this is relatively simple in CP , due to simple cluster events (cf. Sec. III), the situation is more involved in RP , where the consequence of single bond activation is *a priori* not clear: Is the new bond independent or redundant? Are new rigid clusters created or existing ones coalesced? If so, which ones and how? In this section, we provide theoretical results in 2D rigidity theory that allow us to identify the only three possible cluster events resulting from single bond activation, that we name pivoting, overconstraining, and rigidification. The following results allow us to furthermore determine, without making use of pebble searches, (i) if an added bond is independent or redundant and (ii) what type of cluster event is triggered by a given bond activation. In addition, defining the

pivotal class of a node as the number of distinct rigid clusters the node belongs to [47], we also determine (iii) how the pivotal classes of nodes change.

In Sec. V, we explain how to combine these results with optimal pebble searches to obtain our final algorithm. We start by giving basic principles of 2D rigidity and refer to Sec. I in Supplemental Material [42] for further details.

Proposition 1 (transitivity). Denote $\mathcal{R}_1$, $\mathcal{R}_2$, and $\mathcal{R}_3$ three rigid bodies.

- (1) If $\mathcal{R}_1$ is rigid with respect to $\mathcal{R}_2$ and $\mathcal{R}_2$ is rigid with respect to $\mathcal{R}_3$, then $\mathcal{R}_1$ is rigid with respect to $\mathcal{R}_3$.
- (2) If $\mathcal{R}_1$ is rigid with respect to $\mathcal{R}_2$ and $\mathcal{R}_2$ is floppy with respect to $\mathcal{R}_3$, then $\mathcal{R}_1$ is floppy with respect to $\mathcal{R}_3$.

From now on, $\mathcal{R}$ denotes a rigid cluster, $\mathcal{C}$ is a connectivity cluster, and $\mathcal{C}_u$ is the connectivity cluster that contains node u .

Proposition 2 (“two-pivots rule”) [48]. In two dimensions, if two rigid clusters $\mathcal{R}_1$ and $\mathcal{R}_2$ share at least two pivots, then $\mathcal{R}_1 \cup \mathcal{R}_2$ is a rigid cluster.

In the following, uv denotes a new activated bond. The proofs of the following statements are given in Sec. I B in Supplemental Material [42]. Additional results on rigidity, which to our knowledge had not been proven before, are given in Sec. I A in Supplemental Material [42].

Theorem 1 (pivoting). Assume $u \in \mathcal{C}_u$ and $v \in \mathcal{C}_v$ with $\mathcal{C}_u \neq \mathcal{C}_v$. Then,

- (1) the bond uv is independent;
- (2) the activation of uv creates a new rigid cluster composed of the bond uv only;
- (3) the pivotal class of u and v increases by 1. The pivotal class of any other node is unchanged.

Theorem 2 (overconstraining). Assume that there exists a rigid cluster $\mathcal{R}$ such that $u, v \in \mathcal{R}$ before the activation of uv . Then,

- (1) uv is a redundant bond;
- (2) upon activation of uv , $\mathcal{R}$ becomes $\mathcal{R} \cup \{uv\}$. All other rigid clusters remain identical;
- (3) the pivotal classes of all nodes are unchanged.

Theorem 3 (rigidification). Assume $\mathcal{C}_u = \mathcal{C}_v = \mathcal{C}$ and that, before the activation of uv , there is no rigid cluster that contains both u and v . Then,

- (1) uv is an independent bond;
- (2) the activation of uv triggers a rigidification process resulting in the coalescence of $k + 1$ rigid clusters $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k$ and $\mathcal{R}_{k+1} = \{uv\}$ into a single rigid cluster $\mathcal{R}^+ = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \dots \cup \mathcal{R}_k \cup \{uv\}$. The clusters $\mathcal{R}_1, \mathcal{R}_2, \dots, \mathcal{R}_k$ all belong to $\mathcal{C}$, and each of them is connected to at least two other ones via at least two distinct pivots;
- (3) the pivotal class of these pivots decreases by at least one.

V. OVERVIEW OF THE ALGORITHM

In this section, we present an algorithm that exploits the three theorems in Sec. IV. How we efficiently implement it in practice is explained in the next section.

A. Basic settings

The system is initially empty; i.e., there are no active bonds or nodes: Each node has pivotal class 0 and carries two pebbles. The pebble graph is empty.

For the hypotheses of the theorems to be verifiable, the connectivity state of the network must be known at each new bond activation. We, therefore, implement the NZ algorithm for CP as done in Ref. [41]. The NZ algorithm scales linearly with system size, so that, with a very small additional time and memory cost, our algorithm produces the entire phase diagram of the CP transition as well.

To monitor the rigidity state of the system, we adopt the same strategy as NZ [46] and represent rigid clusters as trees. For the sake of clarity, we indicate the trees that describe connectivity and rigid clusters as $\mathcal{T}_C$ and $\mathcal{T}_R$, respectively.

As nodes might belong to multiple rigid clusters, while bonds cannot, each node in a tree $\mathcal{T}_R$ represents a bond in the corresponding rigid cluster $\mathcal{R}$. Traversing a tree $\mathcal{T}_R$ leads to a root bond that uniquely identifies the rigid cluster. Every time a root is identified by a find operation, the path compression is also performed in order to keep a constant cost for each of these operations (cf. Sec. III).

B. Stages of the algorithm

As for the NZ algorithm for CP , our goal is to deduce the new state of the system after each new bond activation, namely, to update the rigid clusters trees from the knowledge of the system state prior to the bond activation. To this aim, we proceed as follows.

- (1) Select a random bond uv to activate.
- (2) Identify $\mathcal{C}_u$ and $\mathcal{C}_v$ by finding their roots.
- (3) If $\mathcal{C}_u \neq \mathcal{C}_v$, merge them, perform a pivoting step (cf. Sec. V C), and go back to step 1.
- (4) If instead $\mathcal{C}_u = \mathcal{C}_v$, find all roots of the rigid clusters to which u and v belong [49].
- (5) If a rigid cluster $\mathcal{R}$ containing both u and v does not already exist, i.e., if a common root has not been identified by the previous step, perform a rigidification step (cf. Sec. V D) and go back to step 1.
- (6) Otherwise, perform an overconstraining step (cf. Sec. V E) and go back to step 1.

The algorithm ends when all the bonds have been activated, hence covering the entire filling fraction range.

The occurrence of pivoting, rigidification, or overconstraining events typically depends on the bond concentration, as we show in Fig. 2. We now describe steps 3, 5, and 6 in more detail.

C. Pivoting

At low concentration, and, in particular, below the CP transition, the activation of new bonds typically leads to pivoting, as shown in Fig. 2.

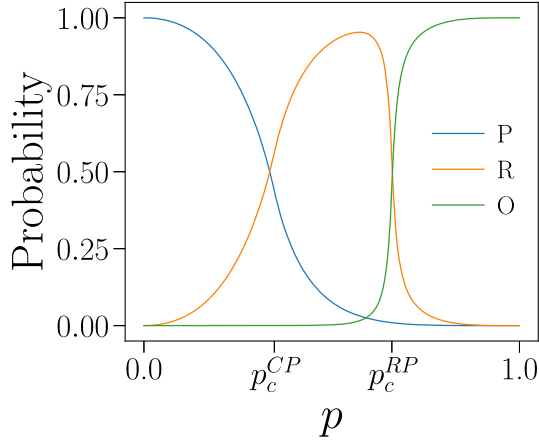


FIG. 2. Stages of the rigidity transition. The blue, orange, and green curves represent the probability, as a function of the bond concentration, that the activation of a new bond results, respectively, in a pivoting event, a rigidification event, or an over-constraining event. We indicate the critical bond concentrations $p_c^{CP} = 2 \sin(\pi/18)$ [50] and $p_c^{RP} \approx 0.6602$ [6,33] of the CP and RP transition, respectively. Curves are shown for $L = 2^{10}$ and are observed not to depend on L .

A pivoting step requires one (i) to increase the pivotal class of both u and v , (ii) to initialize a new $\mathcal{T}_R$ tree with size one and root uv , and (iii) to update the pebble graph by covering uv with a pebble. Note that, by Theorem 1, the bond is already known to be independent, and having one pebble to cover it is, therefore, sufficient. Hence, if at least one node between u and v has a pebble, we simply use it without searching for it. If both have zero pebbles, a pebble search of type I is performed. Note that this search never fails. As the sizes of $\mathcal{C}_u$ and $\mathcal{C}_v$ are known, the search is performed across the smallest of the two to make it faster.

A pivoting step has constant cost if the pebble search is not performed; otherwise, the cost is that of the only search performed. Overall, the cost of a pivoting step is quite low; see Supplemental Material Sec. II [42].

D. Rigidification

Above the CP transition, the probability of having a macroscopic connectivity cluster increases sharply, so that the probability of activating a bond within the same connectivity cluster is large. However, while $p < p_{RP}$, rigid clusters are small (connectivity clusters have numerous floppy modes), leading to a large probability of a rigidification event (see Fig. 2). Namely, the new independent bond will connect a pair of nodes belonging to the same connectivity cluster but different rigid clusters, and its activation will lead to the coalescence of a variable number of distinct rigid clusters into a single one. The construction of the new rigid cluster is at the core of our algorithm.

The most obvious strategy to construct this rigid cluster is that of the JH algorithm, i.e., starting with all nodes

unmarked except u and v that are marked rigid, use a breadth-first search over the neighbors of nodes marked rigid. In this way, the rigid cluster is constructed by identifying a surrounding shell of nodes marked floppy. However, the implementation of this approach results in an algorithm that we observe to be quadratic (see Supplemental Material Sec. 3 A [42]).

The key of our approach is first to realize that it is sufficient to test mutual rigidity of at most one bond per existing rigid cluster, with respect to the newly activated bond. We conveniently choose these bonds as the roots of the rigid clusters. Indeed, by transitivity (Proposition 1), if the root of a rigid cluster $\mathcal{R}$ is rigid (floppy) with respect to uv , then all the bonds in $\mathcal{R}$ are also rigid (floppy) with respect to uv and do not need to be checked.

To build the rigid cluster, one could, therefore, test mutual rigidity between uv and all the roots of the rigid clusters belonging to the connectivity cluster $\mathcal{C}$.

While testing the roots in an arbitrary order does not result in satisfactory scaling (see Supplemental Material Sec. 3 B [42]), the approach becomes highly efficient when the roots are tested by following paths in a *pivot network*, as we now explain.

To construct the rigid cluster resulting from rigidification, we start a breadth-first search from uv over roots of rigid clusters that could potentially coalesce into the forming one. To identify them, we introduce the pivot network $\mathcal{P}$, a graph whose nodes represent the rigid clusters existing in the system. If two rigid clusters share a pivot node in the physical system, they share an edge in the pivot network. Note that, from the “two-pivots rule” (Proposition 2), two distinct rigid clusters can share at most one pivot. In practice, the nodes of the pivot network correspond to the root bonds of the physical system that uniquely identify each rigid cluster. The goal of the rigidification step is to identify the k coalescing clusters of Theorem 3, in order to construct $\mathcal{R}^+ = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \dots \cup \mathcal{R}_k \cup \{uv\}$. Initially, $\mathcal{R}^+$ is composed of the new activated bond uv only, and both u and v are pivots of it. The BFS over the pivot network is performed by identifying the roots connected to $\mathcal{R}^+$ in the pivot network and by checking the mutual rigidity of each of these roots with respect to uv . For each root r found to be rigid with respect to uv , the corresponding rigid cluster is coalesced into $\mathcal{R}^+$, i.e., $\mathcal{R}^+ \rightarrow \mathcal{R}^+ \cup \mathcal{R}_r$. The pivots of each coalesced rigid cluster $\mathcal{R}_r$ become pivots of $\mathcal{R}^+$. The BFS then proceeds by further checking the new roots connected to $\mathcal{R}^+$ through these pivots. As shown in Fig. 3, this iterative procedure builds the rigid cluster resulting from rigidification by walking the paths in the pivot network that connect all the rigid clusters coalescing into $\mathcal{R}^+$. The BFS stops when a surrounding shell of floppy clusters has been identified.

Practically, the cost of the algorithm is dominated by the cost of the rigidification steps (whose cost generally depends on the bond concentration; see the next section).

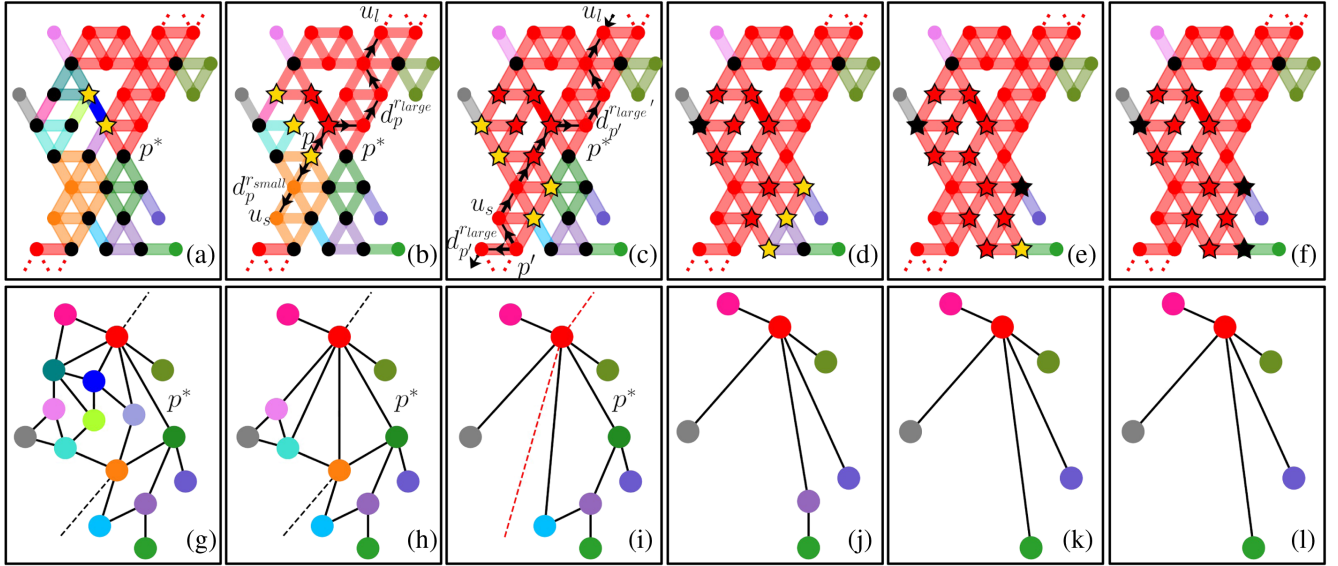


FIG. 3. Example of a rigidification process as implemented by our algorithm. The different steps are illustrated both on the triangular lattice (top row) and on the corresponding pivot network (bottom row). Each color indicates a different rigid cluster. Top row: pivots are indicated with black dots. At each step, enqueued pivots are highlighted as stars. (a) The blue bond is the newly activated one: its two end nodes, marked with yellow stars, are initially enqueued in the queue of pivots. Each rigid cluster is represented by a different color. Nodes with the same color of their bonds belong to that rigid cluster only; black nodes are pivots. (b) The system state after having checked the rigid clusters led to by the pivots enqueued in (a). Red stars indicate pivots that have been processed. The arrows illustrate the Machta displacements $d_p^{r_{\text{large}}}$ and $d_p^{r_{\text{small}}}$, from pivot p to the large and small roots. (c) The system state after having checked the rigid clusters led to by the pivots enqueued in (b). The displacement $d_{u_s}^{r_{\text{large}}}$ has been updated according to Eq. (1). The arrows show the displacements $d_{p'}^{r_{\text{large}}}$ and $d_{p'}^{r'_{\text{large}}}$ given by Eq. (2), whose difference allows one to detect that wrapping has occurred along the vertical axis. Note that the pivot p^* is never enqueued but that the green cluster is nonetheless reached via another pivot (yellow star); cf. discussion at the end of Sec. VI. (d) The system state after having checked the rigid clusters led to by the pivots enqueued in (c). Black stars indicate processed pivots leading to floppy clusters. (e) The system state after having checked the rigid clusters led to by the pivots enqueued in (d). (f) The final lattice configuration. Bottom row: the pivot network corresponding to each panel of the top row. Nodes correspond to rigid clusters; edges represent pivots. Note that pivots with pivotal class three form triangles in this network; we indicate again the nonenqueued pivot p^* . Note that, as rigid clusters coalesce, nodes disappear and the network is rearranged. Also, note that the bond representing pivot p^* disappears from (i) to (l). If it had larger pivotal class, it would still be an edge in the pivot network.

E. Overconstraining

At high bond concentration, and, in particular, right above the RP transition, the existence of a macroscopic rigid cluster leads to a sharp increase of the probability of overconstraining events, namely, of connecting two nodes belonging to the same rigid cluster (see Fig. 2).

An overconstraining step requires one only to update the $\mathcal{T}_R$ of the rigid cluster to which both u and v belong by directing uv to its root. This operation clearly has constant cost.

VI. IMPLEMENTATION OF THE ALGORITHM

We provide an explicit C++ implementation of the algorithm presented in the previous section for RP on the triangular lattice [51]. The generalization to any two-dimensional graph, including off-lattice settings relevant for particle systems, is straightforward.

A. Data structures

The system is a triangular lattice of linear size L , where $i = 0, \dots, N - 1$ integers are used to label nodes and $b = 0, \dots, M - 1$ integers are used to label bonds, with $N = L^2$ and $M = 3N$. $\mathcal{T}_C$ and $\mathcal{T}_R$ are two arrays of length N and M , respectively. $\mathcal{T}_C$ and $\mathcal{T}_R$ are treated using the non-recursive implementation of path compression in Ref. [46].

The system is represented by a standard adjacency list; the pebble graph is a $2N$ array where entries $2i$ and $2i + 1$ are the pebble neighbors of node i . Negative values correspond to inactive pebble edges.

We further define several other arrays of length N , the use of which is detailed in Supplemental Material Sec. IV [42], including notably an array π , which stores the pivotal class of each node.

The pivot network $\mathcal{P}$ is implemented as an array of sets with M entries. C++ sets are balanced binary search trees that guarantee that no duplicates exist, and, for a set with n

elements, they have $\mathcal{O}(\log n)$ cost of insertion, deletion, and search [52].

Entries corresponding to root bonds contain the pivots of the corresponding rigid cluster. All other entries are empty sets. If the activation of uv triggers a pivoting step, both u and v are inserted in the entry of bond uv if their pivotal class becomes larger than one. Since in this case u and v become pivots of the other rigid cluster(s) they belong to, they must be inserted in the corresponding entries of $\mathcal{P}$ as well. We identify these rigid clusters by applying find operations on the active bonds adjacent to uv . The rigidification step is the more involved one, and we describe it in the next section. The pivot network is not affected by overconstraining.

B. Rigidification step

The pivot network is used to efficiently perform the BFS over the rigid clusters. We start the BFS with $\mathcal{R} = uv$ and iterate over its pivots, which are stored in the corresponding entry of $\mathcal{P}$. Every time two rigid clusters coalesce, $\mathcal{P}$ must be updated accordingly.

The first part of the rigidification step requires one to update the pebble graph. To this aim, we (i) gather four pebbles at the new bond by performing pebble searches of type I. Note that we know the bond to be independent, so these searches never fail (implying that we never fail pebble searches of type I); (ii) use one of the pebbles to cover the edge and update the pebble graph; (iii) initialize a new $\mathcal{T}_R$ tree with size one and root uv , just like in a pivoting step; (iv) increase the pivotal class of u and v and update $\mathcal{P}$ accordingly. Finally, (v) freeze the remaining three pebbles at their location.

Then, we proceed with the BFS, which starts by inserting u and v in a queue of pivots. Every time a pivot is inserted in this queue, we label it as enqueued using a dedicated array (see Supplemental Material Sec. IV [42]). All nodes are initially unmarked, except for u and v that are marked rigid. The rigidification step proceeds as follows.

- (1) If the queue of pivots is empty, terminate the rigidification step. Otherwise, pick a pivot p and remove it from the queue.
- (2) Iterate over the bonds of p . For each bond, find its root ij . Once all the roots are processed as detailed below, go back to step 1 and process the next pivot.
- (3) Consider the mark on node i .
 - (i) If i is marked rigid $\rightarrow$ go to step 4.
 - (ii) If i is marked floppy $\rightarrow$ go back to step 2.
 - (iii) If i is unmarked, perform a pebble search of type II to mark it and then operate according to cases (i) and (ii).
- (4) Consider the mark on node j and operate like in step 3.
 - (i) If j is marked rigid $\rightarrow$ go to step 5.
 - (ii) If j is marked floppy $\rightarrow$ go back to step 2.

- (iii) If j is unmarked, perform a pebble search of type II to mark it and then operate according to cases (i) and (ii).

Note that the next steps are executed only if both i and j have been marked as rigid. This means, by transitivity, that the whole rigid cluster rooted by the bond ij , which we denote $\mathcal{R}_{ij}$, is mutually rigid with respect to the new bond and must be coalesced into the forming cluster $\mathcal{R}^+$. To this end, we first identify the largest among $\mathcal{R}^+$ and $\mathcal{R}_{ij}$. Let us assume, without loss of generality, that $\mathcal{R}_{ij} \equiv \mathcal{R}_{\text{large}}$ is larger than $\mathcal{R}^+ \equiv \mathcal{R}_{\text{small}}$. We perform the following steps.

- (5) Update the trees $\mathcal{T}_R$: Redirect the root r_{small} of $\mathcal{R}_{\text{small}}$ to the root r_{large} of $\mathcal{R}_{\text{large}}$ and update the entry of r_{large} in $\mathcal{T}_R$ to keep track of the updated cluster size. This step is the analog of a standard coalescence step in the NZ algorithm for CP (see Sec. III).
- (6) Remove p from the set of pivots of $\mathcal{R}_{\text{small}}$, stored in $\mathcal{P}_{r_{\text{small}}}$ (after coalescence, p is not anymore a pivot between $\mathcal{R}_{\text{large}}$ and $\mathcal{R}_{\text{small}}$) and decrease its pivotal class by one, $\pi_p \rightarrow \pi_p - 1$. If this results in $\pi_p = 1$ (p belongs only to one cluster and is not a pivot anymore), remove p from $\mathcal{P}_{r_{\text{large}}}$ as well.
- (7) Iterate over the remaining pivots p' of $\mathcal{R}_{\text{small}}$. If p' is already in $\mathcal{P}_{r_{\text{large}}}$, $\pi_{p'} \rightarrow \pi_{p'} - 1$. If $\pi_{p'}$ becomes 1, remove p' from $\mathcal{P}_{r_{\text{large}}}$. If p' is not yet in $\mathcal{P}_{r_{\text{large}}}$, insert it and go to step 8. In any case, remove p' from $\mathcal{P}_{r_{\text{small}}}$.
- (8) If p' is not labeled as enqueued, insert it in the queue of pivots and label it as enqueued. This is absolutely crucial to prevent the insertion of the same pivot in the queue more than once.

This process ends only when the queue is empty, i.e., when all the pivots have been processed. This protocol implements the rigidification strategy described in Sec. V D and in Fig. 3: Processing all pivots means that we attempt to build all possible paths in the pivot network that, starting from a rigid cluster that coalesces into $\mathcal{R}^+$, connect it to other existing rigid clusters. The queue is empty when the only remaining paths lead to floppy clusters.

Note that, as long as the rigidification process is not over, $\mathcal{R}_{\text{small}}$ and $\mathcal{R}_{\text{large}}$ may share several pivots, whose pivotal classes get updated at step 7. As can be seen in Fig. 3 [the green and red clusters in Fig. 3(c)], such pivots arise from previous coalescence events and do not contradict Proposition 2.

Moreover, note that we enqueue only pivots of $\mathcal{R}_{\text{small}}$ (step 7); namely, the BFS progresses always from the pivots of the smallest cluster without enqueueing the ones of the largest cluster. This implies that, at the end of the process, the pivots of exactly one of the $\mathcal{R}_1, \dots, \mathcal{R}_{k+1}$, say, $\bar{\mathcal{R}}$, have not been enqueued. To see this, consider a coalescence between $\mathcal{R}_1$ and $\mathcal{R}_2$ and then a coalescence between $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{R}_3$. After the first coalescence, the nonenqueued pivots belong to one rigid cluster only (say, $\mathcal{R}_1$). After the second event, if $\mathcal{R}_{\text{large}} = \mathcal{R}_3$, then the pivots of $\mathcal{R}_1$ gets

enqueued and the ones of $\mathcal{R}_3$ are the only ones not enqueued, while in the opposite case the only nonenqueued ones remain those of $\mathcal{R}_1$. The same reasoning applies to all the successive coalescences. Ignoring the pivots of $\bar{\mathcal{R}}$ still leads to the correct construction of $\mathcal{R}^+$. Namely, none of the $\mathcal{R}_1, \dots, \mathcal{R}_{k+1}$ is “forgotten” during rigidification. Indeed, by Theorem 3, any rigid cluster that shares a pivot with $\bar{\mathcal{R}}$ and that coalesces into $\mathcal{R}^+$ shares at least one other pivot with at least another coalescing cluster, from which it will be eventually reached by the BFS. In other words, clusters that share a pivot with $\bar{\mathcal{R}}$ but with none of the other coalescing ones are surely floppy. This is illustrated in Fig. 3, where the pivot of the large (red) cluster labeled p^* is never enqueued. The green rigid cluster connected to the large one via p^* is, nonetheless, reached from another pivot and coalesced.

Not enqueueing all pivots is crucial to make the algorithm fast. Indeed, when a large rigid cluster exists, a very large number of pivots that lead to floppy clusters would be enqueued, making the iteration of step 1 exceedingly long. This is especially true above the RP transition, where the macroscopic rigid cluster has $\mathcal{O}(N)$ pivots: Enqueueing them would result in a quadratic algorithm.

VII. DETECTING WRAPPING

A. The Machta algorithm in CP

In systems with doubly periodic boundary conditions, the criterion for percolation is the wrapping of a cluster around one, the other, or both periodic boundaries. This corresponds to the existence of a noncontractible loop inside a cluster. A clever way to detect wrapping in *CP* is the Machta algorithm [44], that we recall briefly before generalizing it to the more involved RP problem (we refer to Refs. [44,46] for a more detailed explanation of the Machta algorithm). The key difference between a contractible loop (nonwrapping) and a noncontractible one (wrapping) is the algebraic distance traveled as one goes around the loop. The essence of Machta algorithm is to exploit this fact, by keeping track of the algebraic displacements from each node to a given reference node inside its cluster, conveniently taken as the root. When a new activated bond connects two nodes of a same cluster, the difference of their respective displacements to the root is computed: Wrapping has occurred if the difference of displacements is of the order of the system’s length.

In the original Machta algorithm for *CP*, two variables are assigned to each node that give the x and y displacements along a path inside the cluster from that node to its parent in the tree. Machta displacements are updated in sync with the tree, whenever roots get redirected and when path compression is performed. Wrapping is checked only when connecting two nodes that are found to belong to the same cluster, as this is the only cluster event that might lead to wrapping.

B. A Machta algorithm for RP

In RP, a first complication comes from the fact that nodes may belong to more than one rigid cluster. For each node, we therefore keep track of the displacements associated to each rigid cluster it belongs to. Namely, if node u belongs to k rigid clusters, so that it is shared by bonds $b_1, b_2, \dots, b_k$, whose respective parents in the trees are $a_1 = (u_1, v_1), a_2 = (u_2, v_2), \dots, a_k = (u_k, v_k)$, we compute the Machta displacements $d_u^{a_i} \equiv \{\delta_x(u \rightarrow u_i), \delta_y(u \rightarrow u_i)\}$, for $i = 1, \dots, k$. We conventionally set the reference point as the first node (u_i) of the parent bond. Two arrays of length N are used to store the displacements along the two dimensions of the lattice. In our C++ implementation, each entry in the array contains a map. So, for example, to store the horizontal displacements of u , we use the array dx such that $dx[u]$ is a map and the displacement $\delta_x(u \rightarrow u_i)$ is stored as the value with key a_i , i.e., $dx[u][a_i]$.

The second notable difference with *CP* is the type of cluster events that may lead to wrapping. In RP, wrapping may occur during both overconstraining and rigidification. Below, we detail the update of displacements and wrapping detection, when activating the new bond $b = (u, v)$.

Pivoting—Bond b is a rigid cluster of size one (cf. Sec. IV) and the root of its own tree. The displacement of its first node is, therefore, set as $d_u^b = \{0, 0\}$ (reference node), while $d_v^b = \{\delta_x(v \rightarrow u)\}$.

Overconstraining—This case is akin to *CP*: Adding a new bond to an existing rigid cluster may “close it” around the periodic boundary(ies). Therefore, when activating $b = (u, v)$ leads to overconstraining, we simply compare the total displacements to the root for u and v .

Rigidification—Each time a rigid cluster is coalesced into the forming one (as described in Sec. VIB), the root of the smallest cluster, $r_{\text{small}} = (u_s, v_s)$, is redirected to the root of the largest, $r_{\text{large}} = (u_l, v_l)$. Denoting p the pivot node between $\mathcal{R}_{\text{small}}$ and $\mathcal{R}_{\text{large}}$, we update the displacements as

$$\begin{aligned} d_{u_s}^{r_{\text{large}}} &\rightarrow d_{u_s}'^{r_{\text{large}}} = -d_p^{r_{\text{small}}} + d_p^{r_{\text{large}}}, \\ d_{v_s}^{r_{\text{large}}} &\rightarrow d_{v_s}'^{r_{\text{large}}} = d_{v_s}^{r_{\text{small}}} + d_{u_s}^{r_{\text{large}}}. \end{aligned} \quad (1)$$

An example update of $d_{u_s}^{r_{\text{large}}}$ is shown in Figs. 3(b) and 3(c). After coalescing $\mathcal{R}_{\text{small}}$, some of its pivots are added to the pivots of $\mathcal{R}_{\text{large}}$ (step 7 in Sec. VIB). We update their displacements accordingly:

$$\forall p' \in \mathcal{P}_{r_{\text{small}}}, \quad d_{p'}^{r_{\text{large}}} = d_{p'}^{r_{\text{small}}} + d_{u_s}^{r_{\text{large}}}. \quad (2)$$

To check if the coalescence step has created a noncontractible loop, we compute, for each node p' that is a pivot between small and large, the difference $d_{p'}^{r_{\text{large}}} - d_{p'}^{r_{\text{large}}}$. Namely, while in *CP* and overconstraining the closure

of the loop is checked at the new bond, during rigidification it is more convenient to check it at pivots. This difference is the algebraic distance along the closed loop. It is zero if the rigid cluster does not (yet) wrap and nonzero if wrapping has occurred. An example of such wrapping detection is given in Fig. 3(c). Note that the nodes of the small root may be pivots, too. Therefore, we also check wrapping by computing $d_{u_s}^{\prime\text{large}} - d_{u_s}^{\text{large}}$ and $d_{v_s}^{\prime\text{large}} - d_{v_s}^{\text{large}}$.

VIII. RESULTS

A. Performance of the algorithm

Figure 4 shows the performance of our C++ implementation of the algorithm presented in the previous sections, including the implementation of wrapping detection, on the triangular lattice. Quantities that depend on the filling fraction are convoluted as in Ref. [41] and shown as a function of p . Averages are made over 10^5 for the smallest sizes and over about 600 for the largest one. The performance of our implementation of the algorithm, quantified by the average time to complete a simulation, is observed to scale as $N^{1.02}$ [Fig. 4(a)]. A striking feature of the algorithm is the extremely low number of pebble searches needed to update the system's state after the activation of a bond, shown as a function of the bond concentration p [Fig. 4(b)]. This function peaks near p_c^{RP} with a maximum that is of the order of 10 even at large system sizes.

The average time of each bond activation as a function of the bond concentration is shown in Fig. 4(c). Despite this function also showing a peak near the rigidity transition, the fundamental observation here is that the time per bond is not proportional to the number of pebble searches. Therefore, contrary to rigidification protocols based on the JH approach (cf. Supplemental Material Sec. 3 A [42]), the total execution time cannot be deduced from the

number of searches. For example, for $p_c^{\text{CP}} \lesssim p < p_c^{\text{RP}}$ the number of searches grows much more rapidly than the time per bond, which remains low. A deeper investigation is required to understand the scaling of the algorithm.

The computational complexity is quantified by the average number of operations performed during one execution of the algorithm for a given system size. Noting that $M = 3N$ bonds need to be activated, it is easy to understand that the algorithm is at best linear time. Considering, however, that the bond concentration p affects the number of pebble searches per bond activation [Fig. 4(b)] and the number of pivots considered in a rigidification step (Supplemental Material Sec. V [42]) in a strongly nonlinear way, the analytical quantification of the asymptotic computational complexity is beyond the scope of the present work. In the following, we therefore rely on the measured run-time of our implementation over a finite range of N to characterize the computational cost of our algorithm. In particular, the rate of growth of the computational complexity cannot be larger than that of the run-time, regardless of the efficiency of the implementation.

In Fig. 4(a), we measured the total runtime of our implementation and further quantified the number of operations performed during the most demanding stages of the algorithm.

The observation is that there are mainly three operations that determine how expensive a rigidification step is. The first operation is the preliminary search of four pebbles using type I searches. The second operation is the repeated application of type II searches to mark nodes. The last expensive operation is the iteration of pivots p' , at step 7 of the algorithm. Note that the number of pivots p in step 1 is at most equal to the number of pivots p' , as only a fraction of the p' gets enqueued and, hence, iterated over in step 1

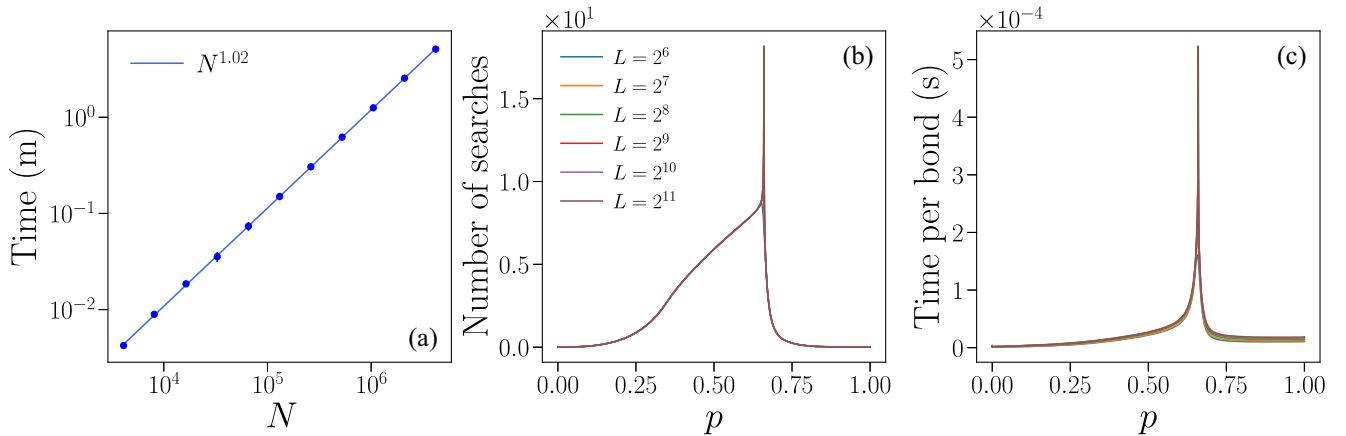


FIG. 4. Performance of our implementation of the algorithm. (a) The average time (in minutes) to complete a simulation as a function of the system size. The blue solid line is the best fit to the log-log of the data and shows the scaling $N^{1.02}$. Error bars are smaller than the markers' size. (b) Average number of pebble searches (of any type) needed at each bond activation, as a function of the bond concentration p , for different system sizes $N = L^2$. (c) Average time (in seconds) of each bond activation as a function of p , for the same system sizes as in (b).

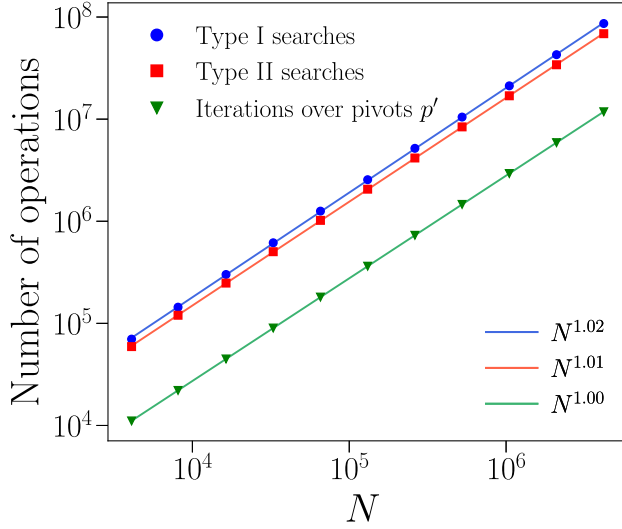


FIG. 5. Average number of nodes visited during type I searches (blue circles, scaling as $N^{1.02}$), average number of nodes visited during type II searches (red squares, scaling as $N^{1.01}$), and average number of pivots p' over which we iterate during step 7 of the algorithm (green triangles, scaling as $N^{1.00}$). Data are shown as a function of the system size.

(with the exception of u and v , which represent, however, an N -independent cost). To quantify how expensive pebble searches are, we count the total number of nodes visited during each search, and we further count the number of pivots p' . Results are shown in Fig. 5.

Data show that the main cost of the algorithm is due to pebble searches of type I, despite that they are less numerous than pebble searches of type II (which, however, are only slightly less expensive than the former).

The number of pivots p' represents instead a minor contribution to the cost of the algorithm. Importantly, pebbles searches of type I are estimated to have a cost scaling as $N^{1.02}$, i.e., the same scaling inferred for the performance of our implementation. In summary, the algorithm is at best linear time and at most scales as the performance of our implementation does, i.e., as $N^{1.02}$. This means that, to any practical extent, the phase diagram of the RP transition can be computed in a time that is linear in the system size (see Supplemental Material Sec. V [42]).

That the number of nodes visited during type I searches scales like the performance suggests, moreover, that the observed performance represents well the computational complexity and that its (extremely small) superlinearity is an intrinsic feature of the algorithm, i.e., that it is implementation independent.

B. Estimation of the RP critical exponents and critical threshold

We apply our algorithm to the determination of the main critical exponents—the correlation length exponent ν , the fractal dimension D_f , the susceptibility exponent γ , and the

order parameter exponent β . We also determine the critical point p_c .

We perform simulations for system sizes ranging from $L = 2^6 = 64$ to $L = 2^{14.5} = 23170$ and choose the numbers of samples so as to activate at least 7×10^{11} bonds per value of L (see Supplemental Material Sec. VI [42]).

For each run of the algorithm, and for both the connectivity and rigid clusters, we store the number of bonds m_x and m_{xy} at which wrapping happens for the first time, respectively, along the horizontal and along both horizontal and vertical directions; the size S_{xy} of the xy -wrapping cluster right at wrapping; and the maximum value $\chi_{\max}$ of the susceptibility.

We then perform scaling analysis to determine D_f , ν , γ/ν , and p_c , for both the RP and CP transitions, the latter being used to benchmark our scaling analysis.

The values of all exponents are reported in Table I, giving a comparison of the CP and RP universality classes. These values are validated by the excellent collapse of the rescaled scaling functions (order parameter, susceptibility, and wrapping probability) shown in Fig. 7. Overall, our results strengthen the long-standing hypothesis [6] that 2D CP and central-force RP belong to different universality classes.

1. Fractal dimension D_f

The fractal dimension D_f is estimated from the (rescaled) size of the wrapping cluster $S_{xy}/(3L^2)$, expected to scale as $S_{xy} \sim L^{D_f-2}$ [1]. For CP, a simple power-law fit gives an estimate in excellent agreement with the exact value $91/48$, reported in Fig. 6(a). For RP, deviations to scaling are present at small sizes, so that we choose to fit S_{xy}^{RP} as a power law, but only for sufficiently large L . The number of data points removed is chosen as the one maximizing the goodness of fit and indicated in Fig. 6. Note that we also tried to fit the data of S_{xy} (and similarly for Δ , $\chi_{\max}$, and p_c) to the form $\mathcal{O}(L) = AL^y(1 + B/L^w)$, which incorporates corrections to scaling. For all observables, such fits lead values of the exponent y consistent with those obtained by removing the smallest sizes from a power-law fit.

TABLE I. CP and RP universality classes and critical thresholds. The value of β is computed using hyperscaling as $\beta = \nu(2 - D_f)$.

	CP	RP
D_f	$91/48 \approx 1.895\,833$	1.8423(7)
ν	$4/3 \approx 1.333\,333$	1.1694(8)
γ	$43/18 \approx 2.388\,889$	1.928(3)
β	$5/36 \approx 0.138\,889$	0.1844(8)
p_c	$2 \sin(\pi/18) \approx 0.347\,296$	0.660\,274\,1(4)

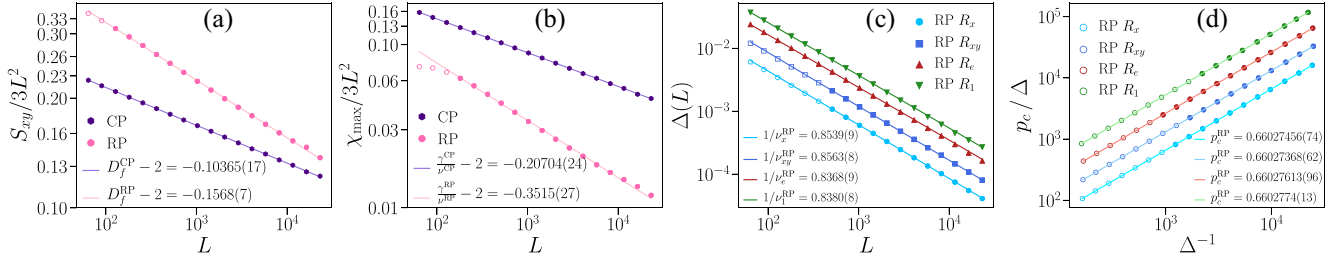


FIG. 6. Scaling analysis for RP. (a),(b) Size $S_{xy}/(3L^2)$ of the wrapping cluster, and maximum $\chi_{\max}$ of the susceptibility, both rescaled by the total number of bonds, as a function of L , shown for CP and RP . The lines are power-law fits of the data marked with filled symbols. (c) Widths $\Delta(L)$ of the transition region for each wrapping type x , xy , e , and 1 , as functions of the system size L . The slopes of the solid lines are reported in the legend and are obtained from power-law fits of the data marked with filled symbols. For the sake of readability, data from R_{xy} , R_e , and R_1 have been shifted by a factor of 2, 4, and 6, respectively. (d) Finite-size critical thresholds p_c/Δ , for each wrapping type, as functions of Δ^{-1} . The slopes of the solid lines, reported in the legend, give the corresponding values of p_c^{RP} . Data have been shifted and shown in log-log scale to ease readability.

2. Susceptibility exponent γ

The maximum of the susceptibility is known to scale as $\chi_{\max} \sim L^{\gamma/\nu}$ [1], from which we estimate the exponent γ/ν . As for the size of the wrapping cluster S_{xy} (cf. Sec. VIII B 1), the data for CP are perfectly fitted by a power law, while RP shows significant deviations to scaling at small sizes; see Fig. 6(b). We adopt the same strategy as in Sec. VIII B 1 and fit only data with sufficiently large $L > L_{\min}$, the value of $L_{\min}$ being chosen as maximizing the goodness of fit. The susceptibility exponent γ^{RP} is then computed using the value of ν^{RP} obtained independently in Sec. VIII B 3 and reported in Table I.

3. Correlation length exponent ν

The measurement of m_x and m_{xy} allows one to compute the wrapping probabilities R_x (along the horizontal axis), R_{xy} (along both axes), R_e (along either axis), and R_1 (along one axis); see Ref. [46] and Fig. S6 [42]. For each wrapping probability, we compute the width of the transition region $\Delta(L)$, as explained in Supplemental Material Sec. 6 A [42]. For each wrapping type, $\Delta(L)$ is expected to scale as $\Delta(L) \sim L^{-1/\nu}$ [1].

For CP , simple power-law fits of $\Delta(L)$ yield exponents in excellent agreement with the exact value $\nu^{CP} = 4/3$, for all four wrapping types. For RP , small deviations to scaling are visible in the residual plots, especially for R_x and R_{xy} , while they appear suppressed in R_e and R_1 . We adopt the same strategy as in Sec. VIII B 1 and fit the data for $\Delta(L)$ as power laws, discarding the smallest sizes as indicated in Fig. 6(c). The corresponding values of $1/\nu^{RP}$ are reported in Fig. 6(c). Note that not all four wrapping probabilities are independent [41]. We find that the estimates from R_x and R_{xy} lead to the best collapse for all four wrapping probabilities and, therefore, compute the final value of ν^{RP} , reported in Table I, as the average of ν_x^{RP} and ν_{xy}^{RP} .

4. Critical threshold p_c

The wrapping probabilities R_i allow one to further compute, for each wrapping type, the size-dependent transition thresholds $p_c(L)$, namely, the bond concentration at which the emergence of a wrapping cluster is expected on a finite lattice of size L (cf. Supplemental Material Sec. 6 A [42]). In general, we expect $p_c(L) - p_c(L \rightarrow \infty) \propto L^{-1/\nu} \propto \Delta(L)$. For RP , pronounced deviations to scaling are visible at small sizes, namely, $p_c^{RP}(\Delta) = p_c^{RP} + A\Delta^{RP}(1 + B\Delta^w)$ for some positive exponent w . We consider the quantity p_c^{RP}/Δ^{RP} and fit

$$y(x) \equiv \frac{p_c^{RP}(\Delta^{RP})}{\Delta^{RP}} = p_c x + A \quad (3)$$

for $L > L_{\min}$ sufficiently large, namely, when $x = 1/\Delta$ is sufficiently large that the correction term $\propto x^{-w}$ becomes negligible. For each wrapping type, the value of $L_{\min}$ is chosen to maximize the goodness of fit. All four wrapping types yield excellent fits, as shown in Fig. 6(d). Our final estimate of p_c^{RP} reported in Table I is the average of the estimates obtained from R_x and R_{xy} that have the lowest uncertainties.

IX. CONCLUSION

In this work, we have presented novel theoretical results on two-dimensional central-force rigidity percolation and have exploited them to derive an exact algorithm that computes the whole phase diagram of the model in a time that is almost linear in the system size. A C++ implementation of the algorithm is provided in Ref. [51]. We have assessed the complexity intrinsic to the algorithm on the triangular lattice, where each node has exactly six neighbors. The rigidity transition, however, is of interest also on different topologies [27,28]. Our algorithm can be straightforwardly implemented on any system where rigidity is ruled by the Laman theorem.

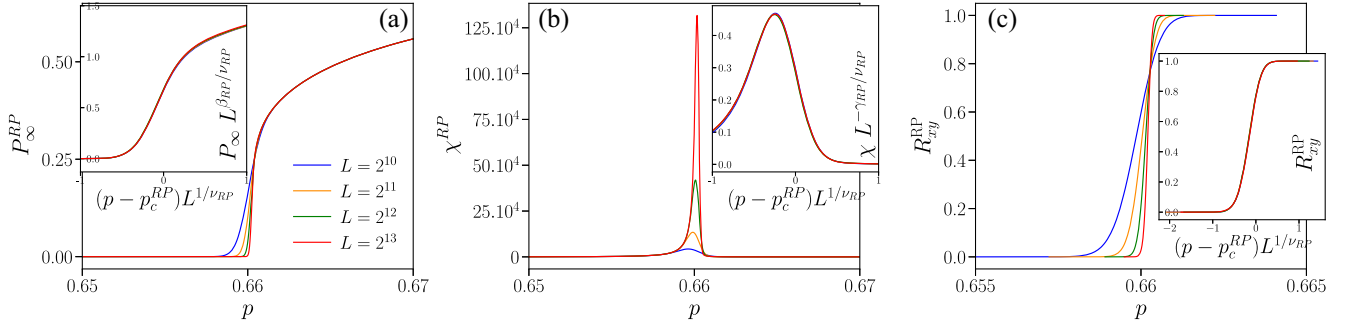


FIG. 7. Scaling functions for RP (main plots) and their collapse (insets) when rescaled using the values of the critical threshold and exponents in Table I. (a) Order parameter P_∞ . (b) Susceptibility χ . (c) Wrapping probability R_{xy} .

In particular, it can be readily used to analyze the rigidity of off-lattice configurations obtained from simulations of particle systems. On networks with a broad degree distribution, where the average degree might grow significantly with N [53], the repeated application of the find operation would introduce a supplementary, N -dependent cost. In any other network as, e.g., Erdős-Rényi graphs, we expect our algorithm to perform well. The actual performance might depend on fine details of the system such as the structure of the pebble graph (pebble searches might be more or less expensive than on the triangular lattice) and on the geometry of rigid clusters (number of pivots they have).

Our algorithm pushes the size limitation encountered in previous works [6,7,31,32], allowing us to characterize accurately the universality class: We have performed scaling analysis on systems up to 536848900 nodes and provide new estimates of the main critical exponents ν , β , D_f , and γ , as well as the critical threshold p_c .

The consistency of these values is validated by the excellent collapse of the rescaled scaling functions, shown in Fig. 7. The ability to simulate very large systems opens the possibility of characterizing further the RP universality class, beyond critical exponents: For instance, it was recently suggested that the RP transition be conformal invariant [31,32], which calls for the determination of the so-called structure constants, key ingredients of the underlying conformal field theory (see, e.g., Ref. [35] for the case of CP).

On the theoretical side, our results provide new material to the long-standing question of the relationship between CP and RP [17,24,54], by showing explicitly that different cluster merging mechanisms are at play in the two problems. RP features notably nonlocal “rigidification” events that lead to the coalescence of several (rigid) clusters, a phenomenon absent in CP . In contrast with the standard pebble game, we construct the system in a dynamic way, making it straightforward to study accurately the frequency and spatial extension of rigidification events [55]. Moreover, given that our algorithm enables one to track both wrapping and dynamical aspects, it would be insightful to quantify the impact of rigidification

cascades on wrapping events. This could shed light on the importance of such nonlocal events in the rigidification of amorphous media. It would also be very interesting to generalize this approach to study rigidity loss [57], in particular, large-scale fluidization events whereby the removal of a single bond makes a macroscopic rigid cluster collapse into smaller clusters connected by nonrigid hinges.

Although we focused here on networks with central forces, it is important to examine if some of our theoretical conclusions generalize to systems with noncentral interactions. This could allow one to combine generalized pebbles games [16,34] with the NZ approach we presented to develop a faster algorithm for RP with, for instance, frictional interactions. Furthermore, our exact results may be exploited to inform nonexact, neural network-based algorithms for rigidity transitions [58].

Finally, the exact determination of rigid clusters in three (or more) dimensions remains a crucial bottleneck [21,23]. Very recent results [59] have opened the possibility of designing exact algorithms for rigidity analysis in any dimension, and we can hope to make these highly efficient by incorporating the insights of the present work.

Note added. While finalizing the second version of this work—which completes our original study by determining the critical exponents—we became aware of the preprint [60], which also reports new estimates for the RP critical exponents. To the best of our knowledge (the code not yet being publicly available), the algorithm described in Ref. [60] incorporates several strategies that we introduced in the first version of our work, although the system sizes considered in their analysis do not exceed $L = 8192$, and the measured runtime is reported to scale as $N^{1.15}$. Interestingly, the reported values of the critical threshold and the correlation length exponent appear in good agreement with our estimates, although the analysis relies on a different set of “dynamic” observables and even a different definition of the critical point, as wrapping is not implemented in the algorithm [60]. The largest discrepancy is observed for the value of the fractal dimension,

$d_f = 1.850(2)$, while our estimate from the size of the wrapping cluster is $D_f = 1.8423(7)$. This variation may be explained by the different type of events chosen to define the rigid cluster whose scaling is governed by the fractal dimension.

ACKNOWLEDGMENTS

The authors are grateful to Lorenzo Rovigatti, Silvano Ferrari, Emanuela Bianchi, and Claudio Castellano for insightful discussions and valuable comments on the manuscript. We especially thank Hugo Vanneuville and Vincent Beffara for their precious help in finalizing the proofs given in Supplemental Material [42]. Finally, we thank the organizers of the 2024 Network Dynamics workshop in Les Houches, where this project started. This research was funded in whole by the Austrian Science Fund (FWF) under Project No. Y-1163-N27. D.N. acknowledges computation time at the Vienna Scientific Cluster (VSC). N.J. has been supported by a grant from MIAI@Grenoble Alpes (ANR-19-P3IA-0003). Part of the computations presented in this paper were performed using the GRICAD infrastructure, which is supported by Grenoble research communities.

DATA AVAILABILITY

The original code is available [51], together with a minimal PYTHON notebook that enables one to measure and plot the performance estimators, as well as the scaling functions described in this work.

-
- [1] D. Stauffer and A. Aharony, *Introduction to Percolation Theory* (Taylor & Francis, London, 1992).
 - [2] H. Duminil-Copin, *Sixty years of percolation*, in *Proceedings of the International Congress of Mathematicians (ICM 2018)* (World Scientific, Singapore, 2019), pp. 2829–2856.
 - [3] J. C. Maxwell F. R. S., *L. on the calculation of the equilibrium and stiffness of frames*, *London, Edinburgh, Dublin Philos. Mag. J. Sci.* **27**, 294 (1864).
 - [4] H. Pollaczek-Geiringer, *Über die gliederung ebener fachwerke*, *Z. Angew. Math. Mech.* **7**, 58 (1927).
 - [5] G. Laman, *On graphs and rigidity of plane skeletal structures*, *J. Eng. Math.* **4**, 331 (1970).
 - [6] D. J. Jacobs and M. F. Thorpe, *Generic rigidity percolation: The pebble game*, *Phys. Rev. Lett.* **75**, 4051 (1995).
 - [7] S. Zhang, L. Zhang, M. Bouzid, D. Z. Rocklin, E. Del Gado, and X. Mao, *Correlated rigidity percolation and colloidal gels*, *Phys. Rev. Lett.* **123**, 058001 (2019).
 - [8] H. Tsurusawa, M. Leocmach, J. Russo, and H. Tanaka, *Direct link between mechanical stability in gels and percolation of isostatic particles*, *Sci. Adv.* **5**, eaav6090 (2019).
 - [9] S. M. Fenton, P. Padmanabhan, B. K. Ryu, T. T. D. Nguyen, R. N. Zia, and M. E. Helgeson, *Minimal conditions for solidification and thermal processing of colloidal gels*, *Proc. Natl. Acad. Sci. U.S.A.* **120**, e2215922120 (2023).
 - [10] N. I. Petridou, B. Corominas-Murtra, C.-P. Heisenberg, and E. Hannezo, *Rigidity percolation uncovers a structural basis for embryonic tissue phase transitions*, *Cell* **184**, 1914 (2021).
 - [11] E. Hannezo and C.-P. Heisenberg, *Rigidity transitions in development and disease*, *Trends Cell Biol.* **32**, 433 (2022).
 - [12] P.-F. Lenne and V. Trivedi, *Sculpting tissues by phase transitions*, *Nat. Commun.* **13**, 664 (2022).
 - [13] M. L. Manning, *Rigidity in mechanical biological networks*, *Curr. Biol.* **34**, R1024 (2024).
 - [14] C. P. Broedersz, X. Mao, T. C. Lubensky, and F. C. MacKintosh, *Criticality and isostaticity in fibre networks*, *Nat. Phys.* **7**, 983 (2011).
 - [15] F. Bolton and D. Weaire, *Rigidity loss transition in a disordered 2D froth*, *Phys. Rev. Lett.* **65**, 3449 (1990).
 - [16] S. Henkes, D. A. Quint, Y. Fily, and J. M. Schwarz, *Rigid cluster decomposition reveals criticality in frictional jamming*, *Phys. Rev. Lett.* **116**, 028301 (2016).
 - [17] K. Liu, S. Henkes, and J. M. Schwarz, *Frictional rigidity percolation: A new universality class and its superuniversal connections through minimal rigidity proliferation*, *Phys. Rev. X* **9**, 021006 (2019).
 - [18] W. G. Ellenbroek, V. F. Hagh, A. Kumar, M. F. Thorpe, and M. van Hecke, *Rigidity loss in disordered systems: Three scenarios*, *Phys. Rev. Lett.* **114**, 135501 (2015).
 - [19] E. Berthier, J. E. Kollmer, S. E. Henkes, K. Liu, J. M. Schwarz, and K. E. Daniels, *Rigidity percolation control of the brittle-ductile transition in disordered networks*, *Phys. Rev. Mater.* **3**, 075602 (2019).
 - [20] M. Bantawa, B. Keshavarz, M. Geri, M. Bouzid, T. Divoux, G. H. McKinley, and E. Del Gado, *The hidden hierarchical nature of soft particulate gels*, *Nat. Phys.* **19**, 1178 (2023).
 - [21] D. Richard and M. Bouzid, *How rigidity percolation and bending stiffness shape colloidal gel elasticity*, *arXiv:2504.19568*.
 - [22] E. Berthier, J. E. Kollmer, S. E. Henkes, K. Liu, J. M. Schwarz, and K. E. Daniels, *Rigidity percolation control of the brittle-ductile transition in disordered networks*, *Phys. Rev. Mater.* **3**, 075602 (2019).
 - [23] A. Goyal, N. S. Martys, and E. Del Gado, *Flow induced rigidity percolation in shear thickening suspensions*, *J. Rheol.* **68**, 219 (2024).
 - [24] C. Moukarzel and P. M. Duxbury, *Comparison of rigidity and connectivity percolation in two dimensions*, *Phys. Rev. E* **59**, 2614 (1999).
 - [25] A. E. Holroyd, *Existence and uniqueness of infinite components in generic rigidity percolation*, *Ann. Appl. Probab.* **8**, 944 (1998).
 - [26] O. Häggström, *Uniqueness of infinite rigid components in percolation models: The case of nonplanar lattices*, *Probab. Theory Relat. Fields* **127**, 513 (2003).
 - [27] J. Barré, *Hierarchical models of rigidity percolation*, *Phys. Rev. E* **80**, 061108 (2009).
 - [28] T. Jordán and S.-i. Tanigawa, *Rigidity of random subgraphs and eigenvalues of stiffness matrices*, *SIAM J. Discrete Math.* **36**, 2367 (2022).
 - [29] J. Barré, A. R. Bishop, T. Lookman, and A. Saxena, *The cavity method for the rigidity transition*, *J. Stat. Phys.* **118**, 1057 (2005).
 - [30] O. Rivoire and J. Barré, *Exactly solvable models of adaptive networks*, *Phys. Rev. Lett.* **97**, 148701 (2006).

- [31] N. Javerzat and M. Bouzid, *Evidences of conformal invariance in 2D rigidity percolation*, *Phys. Rev. Lett.* **130**, 268201 (2023).
- [32] N. Javerzat, *Schramm-Loewner evolution in 2D rigidity percolation*, *Phys. Rev. Lett.* **132**, 018201 (2024).
- [33] D. J. Jacobs and B. Hendrickson, *An algorithm for two-dimensional rigidity percolation: The pebble game*, *J. Comput. Phys.* **137**, 346 (1997).
- [34] D. Lester and R. Li, *The frictional pebble game: An algorithm for rigidity percolation in saturated frictional assemblies*, *J. Comput. Phys.* **369**, 225 (2018).
- [35] R. M. Ziff, J. J. H. Simmons, and P. Kleban, *Factorization of correlations in two-dimensional percolation on the plane and torus*, *J. Phys. A* **44**, 065002 (2011).
- [36] In our work, as well as earlier work [6], bonds are activated in a random, uncorrelated way. Recent work has examined the effect of local correlations [7], but to our knowledge the effect of long-range correlations in the bond activation on the RP universality class has not been addressed. Our algorithm can be generalized to incorporate such correlations, and we plan this for future work.
- [37] H. Dashti, A. A. Saberi, S. Rahbari, and J. Kurths, *Emergence of rigidity percolation in flowing granular systems*, *Sci. Adv.* **9**, eadh5586 (2023).
- [38] G. Lois, J. Blawdziewicz, and C. S. O'Hern, *Jamming transition and new percolation universality classes in particulate systems with attraction*, *Phys. Rev. Lett.* **100**, 028001 (2008).
- [39] M. Latva-Kokko, J. Mäkinen, and J. Timonen, *Rigidity transition in two-dimensional random fiber networks*, *Phys. Rev. E* **63**, 046113 (2001).
- [40] M. Latva-Kokko and J. Timonen, *Rigidity of random networks of stiff fibers in the low-density limit*, *Phys. Rev. E* **64**, 066117 (2001).
- [41] M. E. J. Newman and R. M. Ziff, *Efficient Monte Carlo algorithm and high-precision results for percolation*, *Phys. Rev. Lett.* **85**, 4104 (2000).
- [42] See Supplemental Material at <http://link.aps.org/supplemental/10.1103/sv88-j2wj> for proofs of results in rigidity theory; details on the performance of the algorithm; and details on the estimation of the critical exponents and threshold. Supplemental Material includes Ref. [43].
- [43] H. Gluck, *Almost all simply connected closed surfaces are rigid*, in *Geometric Topology*, edited by L. C. Glaser and T. B. Rushing (Springer, Berlin, Heidelberg, 1975), pp. 225–239.
- [44] J. Machta, Y. S. Choi, A. Lucke, T. Schweizer, and L. M. Chayes, *Invaded cluster algorithm for Potts models*, *Phys. Rev. E* **54**, 1332 (1996).
- [45] Depth-first search can be used as well. JH used BFS, and we do the same in our work.
- [46] M. E. J. Newman and R. M. Ziff, *Fast Monte Carlo algorithm for site or bond percolation*, *Phys. Rev. E* **64**, 016706 (2001).
- [47] Inactive nodes have $\pi = 0$ by definition.
- [48] J. Graver, B. Servatius, and H. Servatius, *Combinatorial Rigidity*, Graduate Studies in Mathematics (American Mathematical Society, Providence, 1993).
- [49] Note that $C_u = C_v$ happens only if both u and v are active; hence, each of them belongs to at least one rigid cluster.
- [50] M. F. Sykes and J. W. Essam, *Exact critical percolation probabilities for site and bond problems in two dimensions*, *J. Math. Phys. (N.Y.)* **5**, 1117 (1964).
- [51] D. Notarmuzi and N. Javerzat, [10.5281/zenodo.15584520, https://github.com/NinaJaverzat/NZRP/tree/main](https://github.com/NinaJaverzat/NZRP/tree/main) (2025), simulation code, v1.
- [52] We preferred sets to unordered sets, as we observed the former to perform better than the latter. Indeed, the complexity of basic operations on sets is guaranteed to be $\mathcal{O}(\log n)$. In contrast, while unordered sets offer $\mathcal{O}(1)$ complexity, their performance can degrade to $\mathcal{O}(n)$ due to rehashing of the underlying hash table.
- [53] M. Boguñá, C. Castellano, and R. Pastor-Satorras, *Langevin approach for the dynamics of the contact process on annealed scale-free networks*, *Phys. Rev. E* **79**, 036110 (2009).
- [54] De Gennes and P.G., *On a relation between percolation theory and the elasticity of gels*, *J. Phys. Lett.* **37**, 1 (1976).
- [55] While concluding the first version of this work, we became aware of the preprint [56] that proposes a similar dynamic approach to RP. The algorithm has not been published but is reported to perform slightly worse than the standard pebble game.
- [56] M. Lu, Y.-F. Song, M. Li, and Y. Deng, *Self-similar gap dynamics in percolation and rigidity percolation*, *arXiv:2411.04748*.
- [57] W. G. Ellenbroek, V. F. Hagh, A. Kumar, M. F. Thorpe, and M. van Hecke, *Rigidity loss in disordered systems: Three scenarios*, *Phys. Rev. Lett.* **114**, 135501 (2015).
- [58] D. A. Head, *Predicting rigidity and connectivity percolation in disordered particulate networks using graph neural networks*, *Phys. Rev. E* **111**, 045411 (2025).
- [59] K. Kim and J. M. Schwarz, *Rigidity condition for gluing two bar-joint rigid graphs embedded in $\mathbb{R}^d$* , *arXiv:2410.00317*.
- [60] M. Lu, Y. Song, Q. Shi, M. Li, and Y. Deng, *Dynamic Monte Carlo study of rigidity percolation using an event-based ensemble approach*, *Phys. Rev. E* **113**, 064141 (2026).