The slider-pinning problem

Audrey Lee *† Ileana Streinu ‡ Louis Theran §

Abstract

A **Laman mechanism** is a flexible planar bar-and-joint framework with $m \leq 2n-3$ edges and exactly $k = 2n-m$ degrees of freedom. The **slider-pinning problem** is to eliminate all the degrees of freedom of a Laman mechanism, in an optimal fashion, by individually fixing x or y coordinates of vertices. We describe two easy to implement $O(n^2)$ time algorithms.

1 Introduction

A **bar-and-joint framework** in the plane is a structure made of fixed-length **bars** connected by universal **joints**. In other words, the framework is allowed to move in any way that preserves the lengths of the bars. A framework is **rigid** if all of these motions are trivial rigid motions (translations or rotations).

A **Laman graph** is a graph with n vertices and $2n-3$ edges with the property that any subgraph on $n' \geq 2$ vertices has at most $2n' - 3$ edges. A graph with the hereditary count condition but fewer than $2n-3$ edges is called a **Laman mechanism**. A **Laman framework** is an embedded Laman graph or mechanism: each of its vertices is mapped to some point in the plane.

Laman's theorem [5] states that Laman graphs are exactly those that, when embedded on a *generic* point set, correspond to minimally rigid frameworks, i.e., they become flexible if any edge is removed. More generally, Laman mechanisms with $2n-3-f$ edges have f non-trivial degrees of freedom as generic frameworks.

Finding the minimum number of **thumbtacks** that would immobilize (pin down) a Laman mechanism is a well-studied problem. Each thumbtack fixes the position of a vertex (both x- and y-coordinates). We call this the **thumbtack-pinning problem**. Lovász [10, 9] gave an algorithm based on Khachian's ellipsoid method and polymatroid matching to compute the minimum number of thumbtacks needed to pin a framework with given numeric coordinates. Recently, Fekete [3] gave a combinatorial solution based on the rigidity matroid's rank function, which leads to a polynomial time algorithm, based on bipartite matching, for computing the set of thumbtacks.

The slider-pinning problem. In this paper, we consider a simpler model for pinning, in which we constrain a vertex to move on a fixed line, or fix the x- or y-coordinates of a vertex separately, similar to the way a mechanical slider joint acts. The problem is to identify a set of k **sliders** (vertices and choices of coordinate to fix) that pin down a framework with k degrees of freedom, including the trivial ones ($k = f+3$).

We call the problem of adding sliders to a Laman mechanism so that it is minimally pinned down the **slider-pinning problem**. The input is a Laman mechanism, and the output is a set of sliders that pin it down. In this abstract, we consider only a subset of problems and work with axis-parallel sliders. In the full paper, we develop the entire theory of pinning (infinitesimal and combinatorial) and give algorithms for arbitrary slider-joints and various other generalizations.

Counting degrees of freedom in a Laman mechanism shows that any slider pinning gives a 2-approximation for the thumbtack-pinning problem. Beyond this observation, we do not consider the thumbtack-pinning problem here.

Contributions. We describe two algorithms for the slider-pinning problem. Both algorithms run in $O(n^2)$ time and are very easy to implement. They are based

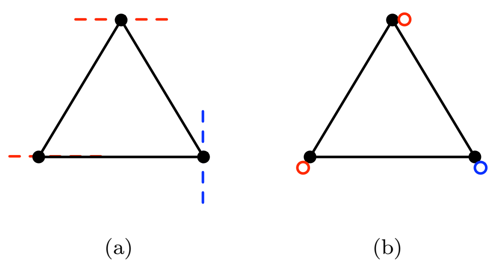

Figure 1: Bar-slider frameworks and graphs: (a) a bar-slider framework; (b) the same framework given combinatorially as a graph with edges and colored loops.

*All authors have been supported by the NSF grants CCF-0430990 and CCR-0310661 of Ileana Streinu. Audrey Lee also supported by an NSF Graduate Research Fellowship.

†Department of Computer Science, UMass, Amherst. *Email:* alee@cs.umass.edu

‡Computer Science Department, Smith College. *Email:* streinu@cs.smith.edu

§Department of Computer Science, UMass, Amherst. *Email:* theran@cs.umass.edu

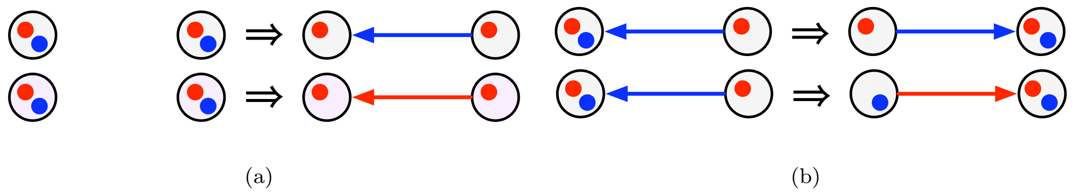

Figure 2: Moves in the $(2, 3)$-pebble game with colors: (a) add edge; (b) pebble slide.

on **pebble games** (in particular, the original algorithm for 2D rigidity of Hendrickson and Jacobs [4] and its extensions [1, 6, 12, 7]).

In this paper, we will use the pebble games for: (1) Finding the **rigid components** (maximal rigid subgraphs) of Laman mechanisms [4, 1, 6, 7] and (2) Finding 3T2 decompositions (colorings) of Laman graphs and mechanisms [12]. All of these algorithms run in $O(n^2)$ time and require $O(n^2)$ space.

Related work. Pebble games are a family of simple construction rules for sparse graphs, which generalize the Laman counts. They are used as subroutines for algorithms that decide rigidity, sparsity and related decompositions (into rigid components, into edge-disjoint spanning trees, etc.). For details, history and references, see our previous papers [6, 12]. In a nutshell, the basic 2D pebble game begins with an empty directed graph; 2 pebbles are placed on each vertex. Edges are then considered one at a time and moves are played on the pebble game graph. Valid moves insert and direct an edge exactly when the underlying graph remains $(2, 3)$-sparse after the addition. In Section 2, we describe the pebble game with colors [12], which is the main subroutine of our first pinning algorithm.

In a companion paper [8], we prove the following Laman-type theorem for **bar-joint-and-slider** (shortly, **bar-slider**) frameworks. The combinatorial model for bar-slider frameworks consists in graphs with simple **edges** (having two distinct endpoints) and **loops** (having one endpoint). Edges in the graph represent the bars, and loops represent sliders at a vertex. Additionally, loops may be colored red or blue to indicate which axis the associated slider is parallel to. Multiple copies of edges are not allowed, but there may be up to two distinct loops (one red, one blue) on each vertex. Figure 1 shows an example of a bar-slider framework and its associated graph.

Proposition 1 (Bar-slider framework rigidity). *Let G be a graph with $2n - k$ edges and k loops colored red or blue. G is realizable as a pinned bar-and-slider framework if and only if there is a 2-coloring of the edges of G so that: (1) Each color forms a forest, and no subset of ≥ 2 vertices induces two edge-disjoint spanning trees; (2) Each monochromatic tree contains exactly one loop of its color.*

The edge coloring in the statement of Prop. 1 is a generalization of the so-called 3T2 decomposition [2], shown to be equivalent to the Laman counting condition. Together the coloring of the edges and loops certify that G satisfies the hereditary condition that no set of n' vertices spans more than $2n'$ loops and edges [13]. The graphs corresponding to pinned bar-and-slider frameworks have interesting matroidal properties which we do not go into here; these, along with some additional algorithmic consequences, are discussed in [8].

Servatius et al. [11] present a counting condition for thumbtack pinning that coincides with Prop. 1 in the case where each vertex has either zero or two sliders. They are not concerned with slider-pins or algorithmic questions, which are our focus here.

2 The pebble game with colors

We briefly describe now our pebble game with colors [12] before using it as a subroutine for Algorithm 1. It is an adaptation of our pebble games for sparse graphs in [6] for finding **sparsity-certifying decompositions**, which are partitions of the edges and loops of a graph into k color classes certifying that the graph is (k, ℓ)-sparse. Here we are concerned with $k = 2$ and $\ell = 3$, and, for these parameters, the desired coloring is the 3T2 decomposition described above. In the rest of this section, we describe the $(2, 3)$-pebble game with colors in terms of its initial configuration and the allowed moves. The pebble game is an algorithmic set of rules described as a *game* with a single player, who tries to insert as many edges as possible onto an initially empty *board*.

Method: The game is played on a directed graph G (which initially is empty); one red pebble and one blue pebble (for a total of 2) are placed on each vertex. Input edges (from an initially given graph) are subsequently considered in an arbitrary order. For each edge, the algorithm tries to collect 4 pebbles on its endpoints using **pebble-slide** moves (described below) only if monochromatic cycles are not created. If the pebbles cannot be collected, the edge is discarded. Otherwise, the edge is inserted into G, using the **add-edge**

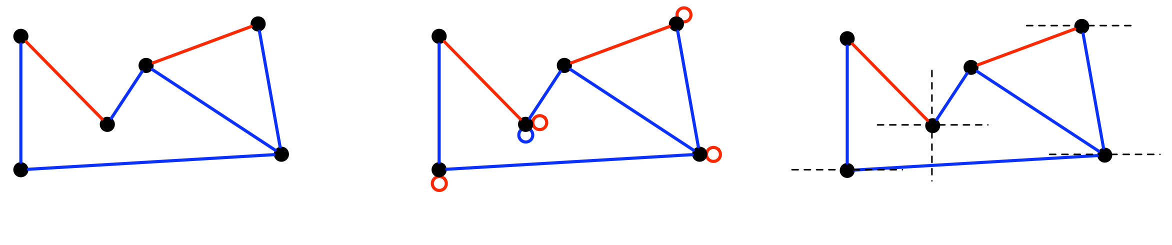

Figure 3: First pinning algorithm: (a) a Laman mechanism with a 3T2 coloring; (b) the mechanism from (a) with appropriately colored loops; (c) the sliders corresponding to the loops in (b).

move (also described below). At any point during the game, each edge of G has exactly one pebble on it and is interpreted as having the color of its pebble.

Pebble-slide move: Let vw be an edge in G, and let w have a pebble on it. Replace vw with wv, move the pebble from vw to v, and move the pebble from w on to the reversed edge wv.

Add-edge move: Let v and w be vertices with two pebbles (one red and one blue) on each. Insert the directed edge vw in G, covering (and thus coloring) it with one of the pebbles on v. Subsequently, this pebble is removed from v.

The moves are shown in Figure 2. The key lemma of [12] is that, if a pebble can be moved without creating a monochromatic cycle, the corresponding **pebble-slide** moves can be found and performed in $O(n)$ time.

3 Algorithms for slider-pinning

We present now our slider-pinning algorithms and sketch the proofs of correctness and running times.

Slider-pinning with a 3T2 decomposition. Our first algorithm is based on Prop. 1.

Algorithm 1. Pinning with a 3T2 decomposition.
Input: *A Laman mechanism G with $2n - k$ edges.*
Output: *A set of k sliders (with colors) that pin G.*
Method: *Use the $(2,3)$-pebble game with colors to verify that G is Laman and find a 3T2 decomposition of G. For each tree in the decomposition, add exactly one loop of the same color somewhere in it. These loops are the sliders to return.*

Figure 3 illustrates Algorithm 1. The mechanism has five monochromatic trees (three are single-vertex red trees), corresponding to its five degrees of freedom (three are trivial rigid motions). Thus, we use five sliders to pin it down. The 3T2 decomposition guides the choice of vertical or horizontal orientation, by identifying orientations with colors.

Correctness follows immediately from Prop. 1.

Running time: Adding the loops takes $O(n)$ time. In [12], we proved that the pebble game requires $O(n^2)$ time to find a 3T2 decomposition, so the total time is $O(n^2)$.

Slider-pinning with rigid components. Our other algorithm for the slider-pinning problem is based on the **rigid components** in a Laman mechanism. Recall that these are maximal rigid subgraphs of a Laman mechanism; see [6] for an investigation of their properties.

Algorithm 2. Pinning with components.
Input: *A Laman mechanism G with $2n - k$ edges.*
Output: *A set of k sliders that pin G.*
Method: *Use the pebble game components algorithm from [6] to find the rigid components of G. Pick a component as the base, and pin it by pinning down an edge in the base with 3 sliders on the endpoints.*

Until there is only one rigid component, repeat the following:

- *Pick a component that shares a vertex i with the base.*
- *Use a slider to pin either coordinate of any vertex $j \neq i$ in that component.*
- *Add the edge jk for any vertex $k \neq i$ of the base to G, and use the component detection algorithm from [6] to enlarge the base to be the component containing jk.*

Finally, return the sliders added.

Figure 4 illustrates one iteration of the component-based pinning algorithm. Initially, the red base is pinned. Adding an arbitrary slider to any vertex pins the neighboring component and fixes the distance between vertices in the base and the neighbor. We model this as a new edge in the graph, which then lets us recompute components and enlarge the base.

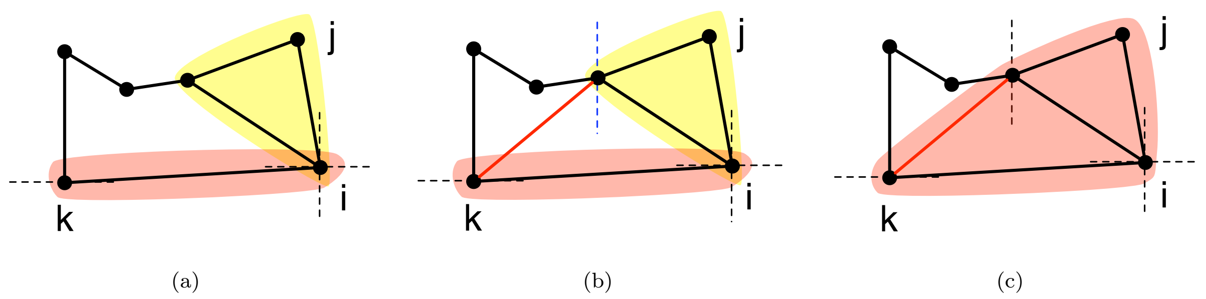

Figure 4: Second pinning algorithm: (a) the pinned base (red) with the neighbor component (yellow); (b) adding a slider (blue) and modeling it as a new edge (red); (c) expanding the base (red) to be the component containing the new edge.

Correctness: The idea of the proof is that the only motion available to a neighboring component touching the pinned base is rotation around the vertex where they overlap, so constraining a vertex to a line prevents the neighbor from moving. This with Laman's theorem implies that the base is always pinned as the algorithm runs.

Running time: The algorithm runs for $k-3$ phases, each of which takes time $O(n)$ by the analysis in [6, 7]. Then the total time is dominated by the $O(n^2)$ time required to initially find the rigid components of G.

Comparison with Algorithm 1: Algorithm 1 is based on our self-contained development of bar-slider pinning rigidity in [8]. Because of its close connection with Prop. 1, Algorithm 1 can generate all possible sets of sliders that pin its input, and thus may lead to an efficient algorithm for enumerating these.

The sets of sliders produced by Algorithm 2 are in correspondence with sets of independent edges that complete a Laman mechanism to a Laman graph. Efficiency in Algorithm 2 comes from component structure theorems from [6], which allow the edge to be added to be found in constant time. However, this procedure may not find *all* such sets of independent edges, implying that some ways of pinning cannot be found by Algorithm 2.

References

[1] A. R. Berg and T. Jordán. Algorithms for graph rigidity and scene analysis. In G. D. Battista and U. Zwick, editors, *ESA*, volume 2832 of *Lecture Notes in Computer Science*. Algorithms - ESA 2003, 11th Annual European Symposium, Budapest,Hungary, Springer, 2003.

[2] H. Crapo. On the generic rigidity of plane frameworks. Technical Report 1278, Institut de recherche d'informatique et d'automatique, 1988.

[3] Z. Fekete. Source location with rigidity and tree packing requirements. TR 2005-04, Egerváry Research Group, Eötvös University, Budapest, 2005.

[4] D. J. Jacobs and B. Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *Journal of Computational Physics*, 137:346 – 365, November 1997.

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

[6] A. Lee and I. Streinu. Pebble game algorithms and sparse graphs. *Discrete Mathematics*, 2007. http://arxiv.org/abs/math.CO/0702129.

[7] A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proceeding of the Canadian Conference of Computational Geometry*. Windsor, Ontario, 2005.

[8] A. Lee, I. Streinu, and L. Theran. Pinning laman mechanisms with sliders. Manuscript, 2007.

[9] L. Lovász. The matroid matching problem. *Algebraic methods in graph theory*, 25:495–517, 1978.

[10] L. Lovász. Matroid matching and some applications. *Journal of Combinatorial Theory, Series (B)*, 28:208–236, 1980.

[11] B. Servatius, O. Shai, and W. Whiteley. Assurance for assur graphs by rigidity circuits. Manuscript, 2006.

[12] I. Streinu and L. Theran. Sparsity-certifying graph decompositions. Submitted to Graphs and Combinatorics, http://arxiv.org/abs/0704.0002, 2007.

[13] W. Whiteley. The union of matroids and the rigidity of frameworks. *SIAM Journal Discrete Mathematics*, 1(2):237–255, May 1988.