Recognizing Planar Laman Graphs

Jonathan Rollin
Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany
jonathan.rollin@fernuni-hagen.de

Lena Schlipf
Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany
lena.schlipf@fernuni-hagen.de

André Schulz
Theoretische Informatik, FernUniversität in Hagen, Hagen, Germany
andre.schulz@fernuni-hagen.de

Abstract

Laman graphs are the minimally rigid graphs in the plane. We present two algorithms for recognizing planar Laman graphs. A simple algorithm with running time $O(n^{3/2})$ and a more complicated algorithm with running time $O(n \log^3 n)$ based on involved planar network flow algorithms. Both improve upon the previously fastest algorithm for general graphs by Gabow and Westermann [*Algorithmica*, 7(5-6):465–497, 1992] with running time $O(n\sqrt{n \log n})$.

To solve this problem we introduce two algorithms (with the running times stated above) that check whether for a directed planar graph G, disjoint sets $S, T \subseteq V(G)$, and a fixed k the following connectivity condition holds: for each vertex $s \in S$ there are k directed paths from s to T pairwise having only vertex s in common. This variant of connectivity seems interesting on its own.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms

Keywords and phrases planar graphs, Laman graphs, network flow, connectivity

Digital Object Identifier 10.4230/LIPIcs.ESA.2019.79

Related Version A preliminary version of this article appeared at EuroCG 2019 (available from `http://www.eurocg2019.uu.nl/`).

1 Introduction

Let $G = (V, E)$ be a graph with n vertices. The graph G is called a *Laman graph* if it has $2n - 3$ edges and every subset $V' \subseteq V$, with $|V'| \geq 2$, induces a subgraph with no more than $2|V'| - 3$ edges. A *bar-joint framework* is a physical structure made from fixed-length bars that are linked by universal joints (allowing 360° rotations) at their endpoints. A bar-joint framework is *flexible* if it has a continuous motion other than a global rotation or translation. A nonflexible framework is called *rigid*. Moreover it is called *minimally rigid*, if it is rigid, but it becomes flexible after removing any bar. For detailed definitions we refer to Jordán [21]. Interestingly, in 2d a bar-joint framework (in a generic configuration) is minimally rigid, if and only if its underlying graph is a Laman graph. In other words, almost every embedding of a Laman graph will be rigid. For nongeneric configurations Laman graphs can yield a flexible framework though (see Figure 1).

Various characterizations of Laman graphs are known [17, 25, 26]. The class of *plane* Laman graphs provides even more structure [9, 16, 23]. Of particular interest for our result is the following geometric characterization: A geometric graph is a *pointed pseudotriangulation* (PPT) if each inner face contains exactly three angles less than π, called *small*, and every vertex is incident to an angle larger than π, called *big* [31]. Streinu [33] proved that the underlying graph of a pointed pseudotriangulation is a Laman graph. Moreover, Haas et al. [16] showed that every planar Laman graph has an embedding as a pointed pseudotriangulation. Pointed

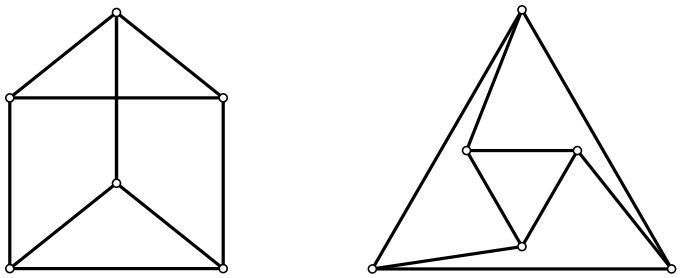


Figure 1 A flexible drawing of a Laman graph (a), and a rigid drawing of the same graph (b).

pseudotriangulations have many applications as a geometric data structure. Maybe most prominently, they are used as a tool to show that every polygonal linkage can unfold in 2d without intersections [6, 33]. More applications come from problems in ray shooting, motion planning and polygon guarding. We refer the interested reader to the survey of Rote, Santos, and Streinu [31].

Haas et al. [16] also give an algorithm that computes a PPT realization for a Laman graph in time $O(n^{3/2})$. We could turn their algorithm into a recognition algorithm by checking whether the derived realization is a PPT. If the original graph is not a Laman graph, then the construction either fails at some intermediate step or some parts of the drawing collapse. To check whether it collapsed requires computations with exponentially large numbers. So it depends on the model of computation if the overall algorithm runs in $O(n^{3/2})$ time. We present combinatorial algorithms for recognizing planar Laman avoiding these subtleties.

Checking the Laman condition for general graphs can be done in polynomial time. The fastest (but very complicated) algorithm is due to Gabow and Westermann [13] and needs $O(n\sqrt{n \log n})$ time. Their algorithm is based on a characterization by means of matroid sums (see also [7] for a time analysis specifically for Laman graphs). There is also a very easy so-called pebble-game algorithm that runs in $O(n^2)$ time [3, 20] (a brief survey of further algorithmic aspects is given in the introduction of [26]).

1.1 Our contribution

Our recognition algorithms for planar Laman graphs rely on the theory of combinatorial pseudotriangulations developed by Haas et al. [16] which will be described in detail in Section 4. The crucial step in the algorithm is to check the following connectivity condition for a certain auxiliary directed planar graph. Consider a positive integer k, a directed graph G, and disjoint sets $S, T \subseteq V(G)$. We call S *k-connected to T* if for each vertex $s \in S$ there are k directed paths from s to T pairwise having only vertex s in common.

A naive approach to check this condition can be implemented in $O(|S|m)$ time and is described in Section 1.2 . We present two algorithms checking this condition for planar directed graphs that are faster for large sets S. In Section 2 we present a simple algorithm that checks this connectivity condition for planar directed graphs provided that each vertex is either in S or T. This gives the following theorem.

▶ **Theorem 1.** *For each fixed $k \geq 1$ there is an algorithm deciding for a directed planar graph G and a partition $V(G) = S \cup T$ whether S is k-connected to T in $O(n^{3/2})$ time.*

We show how to combine this algorithm with simple algorithms of Haas et al. [16] to check the Laman property for a plane graph in time $O(n^{3/2})$ in Section 4.

If all vertices in T are incident to a common face then we obtain a much faster algorithm for the connectivity check. We then even could drop the condition that each vertex is either in S or T. It is based on latest planar network flow algorithms and presented in Section 3.

▶ **Theorem 2.** *For each fixed $k \geq 1$ there is an algorithm deciding for a directed planar graph G and disjoint sets $S, T \subseteq V(G)$ whether S is k-connected to T in $O(n \log^3 n)$ time, provided that all vertices in T are incident to a common face in G.*

It turns out that this condition is satisfied when checking for the Laman property. This leads to a fast algorithm for recognizing planar Laman graphs which we describe in Section 4.

▶ **Theorem 3.** *There is an algorithm deciding for each planar graph whether it is a Laman graph in $O(n \log^3 n)$ time.*

1.2 Related work on connectivity in directed graphs

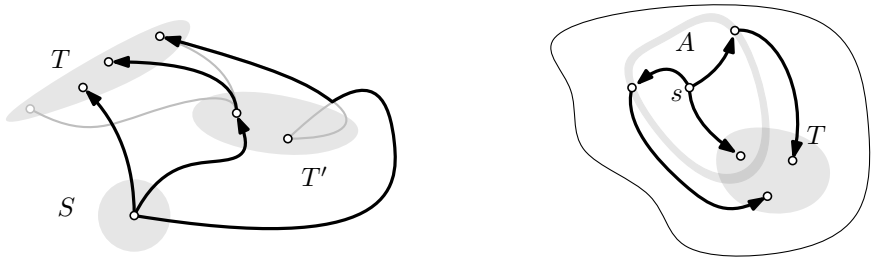
Here we discuss how the algorithms from Theorems 1 and 2 relate to known algorithms. We briefly recall the basic definitions and results on connectivity of directed graphs first. Note that we consider only unweighted graphs. For two vertices s and t in a directed graph let $\kappa(s,t)$ and $\lambda(s,t)$ denote the largest number of internally vertex disjoint respectively edge disjoint directed paths from s to t. It is well known that $\lambda(s,t)$ is equal to the value of a maximum s-t flow and hence to the size of a minimum s-t cut. A directed graph is strongly connected if for each vertex there is a directed path to each other vertex. A directed graph with at least k vertices is called k-vertex connected (k-edge connected) if the removal of any $k-1$ vertices (edges) yields a strongly connected directed graph. Let $\kappa(G)$ ($\lambda(G)$) denote the largest k such that G is k-vertex connected (k-edge connected). By means of Menger's theorem we have that $\kappa(G) = \min_{s \neq t} \kappa(s,t)$ and $\lambda(G) = \min_{s \neq t} \lambda(s,t)$ [2].

For general directed graphs $\kappa(s,t)$ and $\lambda(s,t)$ can be computed with time in $O(nm)$ [30]. Checking whether $\kappa(s,t) \geq k$ or whether $\lambda(s,t) \geq k$ for some fixed k can be done by k steps of the Ford–Fulkerson algorithm and hence in $O(m)$ time (see [2, p. 205]). In both cases the computation of vertex connectivity can be easily reduced to the edge connectivity [10]. Further the values of $\kappa(G)$ and $\lambda(G)$ can be computed with algorithms due to Gabow with time in $O(m(n + \min(\kappa(G)^{5/2}, \kappa(G)n^{3/4})))$ [12] and time in $O(\lambda(G)m \log(n^2/m))$ [11] respectively. Checking whether $\kappa(G) \geq 2$ and whether $\lambda(G) \geq 2$ can be done with time in $O(n+m)$ [14, 19]. For larger fixed k the fastest algorithms checking $\kappa(G) \geq k$ and $\lambda(G) \geq k$ are based on Gabows algorithms above with times in $O(m(n + \min(k^{5/2}, kn^{3/4})))$ and $O(km \log(n^2/m))$ respectively.

For planar directed graphs $\kappa(s,t)$ and $\lambda(s,t)$ can be computed in linear time [8, 22] (and $O(n \log n)$ in the weighted case). Further $\lambda(G)$ can be calculated with time in $O(n \log \log n)$ [28]. We are not aware of algorithms computing $\kappa(G)$ specifically for planar directed graphs.

A naive approach to check whether a set S is k-connected to T (the notion considered in this paper) works as follows. First introduce a new vertex t and all arcs xt, $x \in T$, and then check whether $\kappa(s,t) \geq k$ for each $s \in S$. This leads to an $O(|S|m)$ time algorithm for general directed graphs and constant k. For planar directed graphs our algorithm from Theorem 1 is faster than this approach in case $|S| \in \omega(\sqrt{n})$ and $V(G) = S \cup T$. For planar directed graphs where the vertices in T can be embedded incident to a common face our algorithm from Theorem 2 is faster already for $|S| \in \omega(\log^3 n)$.

Further questions related to connectivity of (planar) directed graphs include the all-pairs reachability [15, 18], all pairs minimum cuts [4].



(a) (b)

Figure 2 If S is k-connected to $T \cup T'$ and T' is k-connected to T, then S is k-connected to T (a). This statement applied where $T' = A$ is a separator (b).

2 Checking directed k-connectivity

In this section we prove the statement of Theorem 1. We first give some structural results. The following statement is similar to Menger's theorem. We provide a proof for completeness.

▶ **Lemma 4.** *Let $k \geq 0$, G be a directed (not necessarily planar) graph, and $S, T \subseteq V(G)$ be disjoint with $|T| \geq k$. Then S is k-connected to T if and only if for each $s \in S$ and $A \subseteq V(G) \setminus \{s\}$ with $|A| = k-1$ there is a directed path from s to T not using the vertices in A.*

Proof. Clearly, if S is k-connected to T then for each $s \in S$ and $A \subseteq V(G) \setminus \{s\}$ with $|A| = k-1$ there is a directed path from s to T not using the vertices in A. We deduce the reverse statement from Menger's theorem. Obtain a directed graph G' by adding a new vertex z and edges tz for each $t \in T$. Then S is k-connected to T in G if and only if for each $s \in S$ there are k internally vertex disjoint directed paths joining s and z in G'. Suppose that for each $s \in S$ and $A \subseteq V(G) \setminus \{s\}$ with $|A| = k-1$ there is a directed path from s to T in G not using the vertices in A. Then, under the same assumptions, there is a directed path from s to z in G' not using the vertices in A. So by Menger's theorem there are k internally vertex disjoint directed paths joining s and z in G'. In particular S is k-connected to T in G. ◀

The following observation is crucial for using separators recursively. An illustration is given in Figure 2.

▶ **Lemma 5.** *Let G be a directed graph and let S, T, $T' \subseteq V(G)$ be disjoint. If S is k-connected to $T \cup T'$ and T' is k-connected to T, then S is k-connected to T.*

Proof. Let $s \in S$ and fix a set $A \subseteq V(G) \setminus \{s\}$ of size $k-1$. We shall find a directed path from s to T not using vertices from A. There is a directed path from s to some vertex $u \in T \cup T'$ not using vertices from A since S is k-connected to $T \cup T'$. If $u \in T$ we are done. If $u \in T'$, then there is a directed path from u to T not using vertices from A since T' is k-connected to T. In each case there is a directed path from s to T not using vertices from A. So S is k-connected to T by Lemma 4. ◀

For a single vertex we can decide in linear time whether it is k-connected to T by means of Ford–Fulkerson's algorithm (similar to the naive approach from Section 1.2). An st-connector in a directed graph G is a set of edge disjoint directed paths from vertex s to vertex t. Given an st-connector $\mathcal{P}$ the residual graph $G_{\mathcal{P}}$ is obtained by reversing all arcs of paths in $\mathcal{P}$. It is well known that $\mathcal{P}$ is a largest st-connector in G if and only if there is no directed path from s to t in $G_{\mathcal{P}}$ [2].

▶ **Lemma 6.** *For each $k \in \mathbb{N}$ there is an algorithm deciding for any directed (not necessarily planar) graph G, $s \in V(G)$, and $T \subseteq V(G)$ whether s is k-connected to T in $O(k(|V(G)| + |E(G)|))$ time.*

Proof. We apply the following standard modification due to Ford and Fulkerson [10]. Split each $v \in V(G)$ into two vertices v_i and v_o and replace each arc uv by an arc $u_o v_i$. Further add all arcs $v_i v_o$, $v \in V(G)$. Finally add a new vertex z and the arcs $t_o z$, $t \in T$. Let G' denote the resulting directed graph. Then s is k-connected to T in G if and only if there is an $s_o z$-connector of size k in G'.

The algorithm iteratively builds an $s_o z$-connector $\mathcal{P}$ in G' in at most k steps. In each step a directed path P from s_o to z is searched in the residual graph $G'_{\mathcal{P}}$. If no such path exists, then the set $\mathcal{P}$ is a largest $s_o z$-connector in G'. Otherwise, in G' the symmetric difference of the set of arcs from paths in $\mathcal{P}$ and the set of arcs from P (which might not be a path in G') forms an $s_o z$-connector of size $|\mathcal{P}| + 1$ in G'.

Clearly the algorithm decides whether there is an $s_o z$-connector of size k in G' and hence whether s is k-connected to T in G. The construction of G' in the beginning and in each step the construction of the residual graph as well as the search can be implemented in $O(n + m)$ time. Since there are at most k steps the overall time is $O(k(n + m))$. ◀

We call $A \subseteq V(G)$ a *separator* if removing A splits G into two (not necessarily connected) subgraphs G_1 and G_2, such that $|V(G_1)|, |V(G_2)| \leq \frac{2}{3}|V(G)|$. For every planar graph G a separator with size in $O(\sqrt{n})$ can be found in linear time [27].

Proof of Theorem 1. The algorithm works recursively as follows. Let A denote a separator of G of size $O(\sqrt{n})$. Use Lemma 6 to check for each $a \in A \cap S$ whether a is k-connected to T in G. Let G_1 and G_2 denote the two subgraphs of G separated by A (each including A). For $i = 1, 2$ let $S_i = (S \cap V(G_i)) \setminus A$ and let $T_i = (T \cap V(G_i)) \cup A$. Apply the algorithm recursively to check whether S_i is k-connected to T_i in G_i, for $i = 1, 2$.

Recall that $V(G) = S \cup T$. The algorithm indeed checks whether S is k-connected to T in G since either it finds some vertex in $A \cap S$ that is not k-connected to T in G, or it is sufficient to check whether $S \setminus A$ is k-connected to $A \cup T$ by Lemma 5. Then it is sufficient to check G_1 and G_2 separately. See Figure 2(b) for an illustration

The separator can be found in linear time. Then the algorithm from Lemma 6 is called $O(\sqrt{n})$ times for each vertex in the separator, each call with linear time (since k is constant and $|E(G)|$ is linear in n). So the total time for each step of the recursion and hence for the whole algorithm is $O(n^{3/2})$. ◀

3 A faster algorithm for checking directed k-connectivity

In this section we prove Theorem 2. First we use a construction similar to one of Kaplan and Nussbaum [22] transforming the problem into a question about network flow. Consider a directed graph H and distinct $s, t \in V(H)$. Kaplan and Nussbaum construct a directed planar graph $H_{s,t}$ obtained from H by replacing each $v \in V(H) \setminus \{s, t\}$ by a cycle C_v with vertices $v_1, \ldots, v_d$, where d is the degree of v in H, such that arcs incident to v are replaced by arcs of (flow) capacity 1 not sharing endpoints while keeping their orientation and the cyclic order around v. The edges of the cycle C_v are oriented in both directions and receive (flow) capacity 1/2. See Figure 3.

▶ **Lemma 7** ([22]). *There are k internally vertex disjoint s-t-paths in H if and only if in $H_{s,t}$ the maximum s-t-flow has value at least k.*

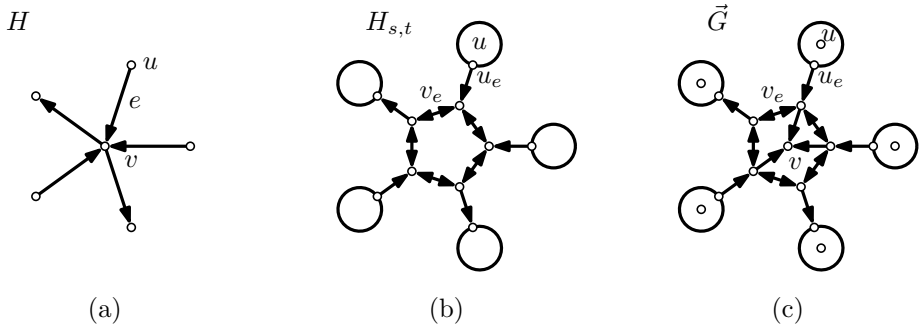


Figure 3 A directed planar graph H (a), the modification $H_{s,t}$ of Kaplan and Nussbaum (b), and the modification $\vec{G}$ used in the algorithm for Theorem 2 (c). The modification of Kaplan and Nussbaum does not include the original vertices inside of the new cycles. We need this since all the original vertices except a fixed source are targets in our algorithm.

Since we need to calculate s-t-flows for several distinct targets t at the same time we need a modification independent of t.

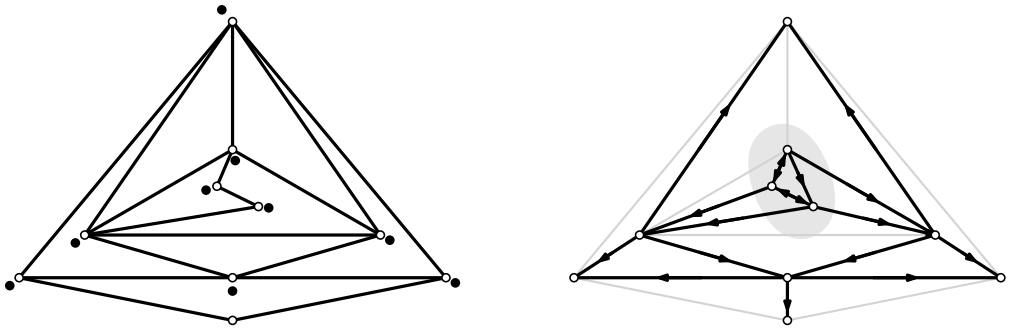
▶ **Lemma 8.** *Consider $k \geq 1$, a directed planar graph G and disjoint sets S, $T \subseteq V(G)$ where all vertices of T are incident to a common face. Then a directed planar graph $\vec{G}$ with $|V(\vec{G})| \leq 7|V(G)|$ can be computed in linear time together with some $s \in V(\vec{G})$ and $T' \subseteq V(\vec{G})$ such that S is k-connected to T in G if and only if in $\vec{G}$ for each $t \in T'$ the value of a maximum s-t-flow is at least k.*

Proof. We consider an embedding of G where all vertices of T are incident to the outer face. Let H denote the directed planar graph obtained by reversing the direction of each arc in G and by adding a new vertex s in the outer face of G connected by arcs sv to all vertices $v \in T$. Further let $T' = S$. Then S is k-connected to T in G if and only if in H there are k internally vertex disjoint s-t-paths for each $t \in T'$ (note that we reversed the direction of all the edges).

We obtain a directed planar graph $\vec{G}$ by splitting each vertex in $V(H) \setminus \{s\}$ into a cycle as follows. Replace each arc e in H, directed from $u \neq s$ to v, by arcs $u_e v_e$ and $v_e v$, where u_e, v_e are new vertices (and distinct for all e). Replace each arc e in H, directed from s to v, by arcs sv_e, and $v_e v$, where v_e is a new vertex. Further for each vertex v in H connect the new vertices v_e by a cycle C_v in the cyclic order of the arcs e around v, where the edges are directed in both directions. Finally a (flow) capacity function is defined where all arcs on cycles C_v receive capacity $1/2$ and all other arcs capacity 1. See Figure 3.

This construction corresponds to the graph $H_{s,t}$ constructed by Kaplan and Nussbaum, except that there is not a specific target t and, additionally the original vertices from H are kept inside of the cycles together with their incoming arcs. In particular $\vec{G}$ is planar and $V(G) \subseteq V(H) \subseteq V(\vec{G})$. Consider some $t \in T' = S$. Note that each s-t-flow in $\vec{G}$ does not use vertices from $V(H) \setminus \{s, t\}$, since these vertices do not have outgoing arcs in $\vec{G}$. Hence for each $t \in T'$ any s-t-flow in $\vec{G}$ corresponds to an s-t-flow in $H_{s,t}$ (by contracting t and C_t to a single vertex). By Lemma 7 there are k internally vertex disjoint s-t-paths in H if and only if in $\vec{G}$ the value of a maximum s-t-flow is at least k.

Clearly $\vec{G}$ can be constructed in linear time and $|V(\vec{G})| \leq |V(G)| + 2|E(G)| + 1 \leq 7|V(G)|$. This shows that $\vec{G}$ together with s and the set $T' = S$ satisfies the desired conditions. ◀



(a) (b)

Figure 4 A CPPT with big angles are marked by solid circles that is not stretchable (a) and the derived directed graph $\vec{G}$ where the highlighted vertices do not have 3 disjoint paths to the outer face (b).

Proof of Theorem 2. First construct a planar directed graph $\vec{G}$ with some $s \in V(\vec{G})$ and $T' \subseteq V(\vec{G})$ with the algorithm from Lemma 8. Then apply an algorithm due to Łącki et al. [24] to $\vec{G}$, which computes for the source s the maximum flow value to each other vertex in $\vec{G}$ in $O(n \log^3 n)$ time. By Lemma 8 S is k-connected to T in G if and only if in $\vec{G}$ for each $t \in T'$ the value of a maximum s-t-flow is at least k. ◀

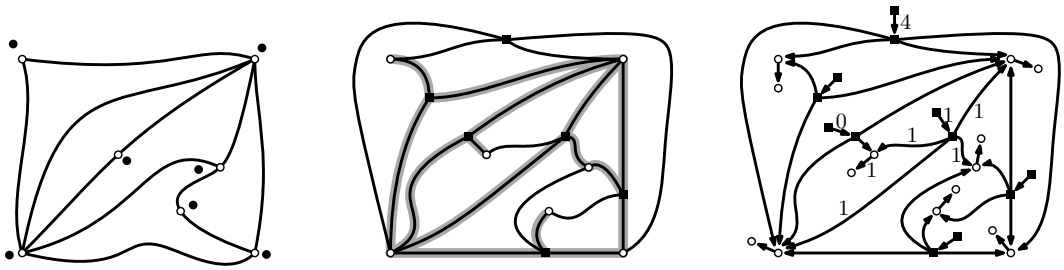
4 Recognizing Planar Laman Graphs

Since any Laman graph is 2-connected and 2-connectivity can be checked in linear time, see, e.g., [32], we consider only 2-connected graphs in this section. Further we assume that the graph is given with some plane embedding. As mentioned in the introduction our recognition algorithms are based on the theory due to Haas et al. [16] which they developed in order to show that exactly the planar Laman graphs can be realized as pointed pseudotriangulations. They transfer the concept of PPTs to graphs given with a plane (combinatorial) embedding. A *combinatorial pointed pseudotriangulation* (CPPT) is a plane graph with $2n-3$ edges and with an assignment of the labels "small"/"big" to the angles satisfying the properties of a PPT. That is, only big angles are incident to the outer face, each inner face contains exactly three small angles and each vertex is incident to exactly one big angle. A CPPT is called *stretchable* if there is a PPT realization (that is, a drawing) of its underlying graph respecting the given labels of the angles and the embedding. Not every CPPT can be stretched to a PPT, see Figure 4(a) for an example. But Haas et al. show that every plane Laman graph admits a CPPT assignment and each CPPT whose underlying graph is Laman is stretchable. This shows that every Laman graph has a realization as a pointed pseudotriangulation. Note that the nonstretchable CPPT in Figure 4(a) contains K_4 as a subgraph which has $6 > 2 \cdot 4 - 3$ edges and hence is not a Laman graph.

This already outlines how our algorithms check whether a given plane graph G is Laman:
Step 1: Check whether G admits a CPPT assignment and compute one if it exists.
Step 2: Check whether the CPPT from Step 1 is stretchable.

The first step can be reduced to searching for a (generalized) matching in the following planar bipartite graph. The vertex-face incidence graph of a plane graph $G = (V, E)$ with face set F is a planar bipartite graph $H = (V \cup F, E')$ where $(v, f) \in E'$ if and only if $v \in V$

is incident to $f \in F$ in G. Note that for each face each vertex appears at most once along its boundary since we assume that G is 2-connected. A plane graph G admits a CPPT assignment if and only if its vertex-face incidence graph has a subgraph A where each interior face of G has degree 3 in A, the outer face of G has degree 0 in A, and each vertex of G has degree in A equal to its degree in G minus 1. This means that the edges of A correspond to small angles in the CPPT assignment. See Figure 5 for an illustration. Haas et al. provide a simple algorithm with running time $O(n^{3/2})$ checking whether such a subgraph A exists, that is, the algorithm computes a CPPT assignment for a given plane graph if any exists. Note that this algorithm does not compute a drawing and hence does not face the numerical issued mentioned in the introduction.



(a) (b) (c)

Figure 5 A plane graph G with a CPPT assignment with big angles marked with a solid circle (a), the vertex-face incident graph of G where vertices corresponding to faces in G are marked with a solid square and the subgraph A corresponding to the CPPT is highlighted in gray (b), the flow network H' constructed from G in the proof of Lemma 9 (some capacities are omitted for better readability) (c).

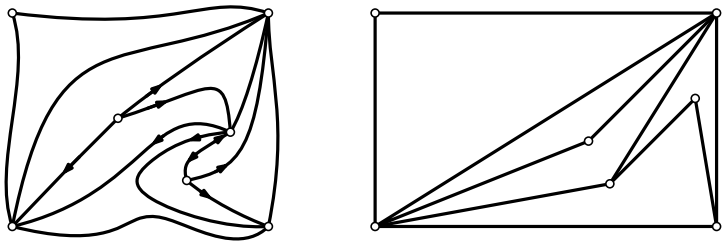
In order to implement the first step with running time in $O(n \log^3 n)$ we use an algorithm by Borradaile et al. [5] computing a maximum flow between multiple sources and sinks.

▶ **Lemma 9.** *There is an algorithm computing a CPPT assignment for any planar graph G if any exists in $O(n \log^3 n)$ time.*

Proof. Consider the planar directed graph H' whose vertex set consists of two vertices u_1 and u_2 for each $u \in V(G)$ and two vertices f_1, f_2 for each face of G. For each $u \in V(G)$ add an arc u_1u_2 with capacity 1, for each interior face f of G add an arc from f_1f_2 with capacity equal to the number of vertices incident to f minus 3, and for the outer face f of G add an arc f_1f_2 with capacity equal to the number of vertices incident to f. Finally for each vertex u and face f add an arc f_2u_1 with capacity 1 if u is incident to f. Then there is a 1-1-correspondence between CPPT assignments of G and integer flows in H' with sources in $\{f_1 \mid f \text{ face in } G\}$, sinks in $\{u_2 \mid u \in V(G)\}$, and value $|V(G)|$. Note that the size of H' is linear in the size of G and can be constructed in linear time. In particular the algorithm of Borradaile et al. [5] (which computes an integer flow) can be used to compute a maximum flow in $O(n \log^3 n)$ time. If this flow has value $|V(G)|$ we computed a CPPT assignment if any exists, otherwise there is no CPPT assignment. ◀

For the second step, that is, checking whether a given CPPT is stretchable, we use the following characterization of stretchability by Haas et al. Recall that a set $S \subseteq V(G)$ is 3-connected to a set $T \subseteq V(G) \setminus S$ if for each vertex $s \in S$ there are three vertex disjoint directed paths from u to distinct vertices in T. Figures 4 and 6 give illustrations of the following result.

▶ **Lemma 10** ([16]). *For a CPPT G a directed plane graph $\vec{G}$, with $V(\vec{G}) = V(G)$, can be computed in linear time such that G is stretchable if and only if the set of interior vertices of $\vec{G}$ is 3-connected to the set of boundary vertices of $\vec{G}$.*

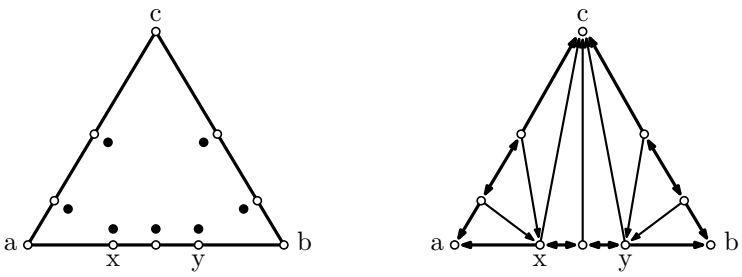


(a) (b)

Figure 6 (a) The derived directed graph $\vec{G}$ from the CPPT in Figure 5(c). (b) The stretched pointed pseudotriangulation.

In fact any directed graph $\vec{G}$ satisfying the following conditions is suitable in the previous lemma (see [16, Lemma 16] and note that G is a CPPT): (i) $\vec{G}$ contains the underlying graph of G and has a planar embedding that respects the embedding of G, (ii) the vertices on the outer face have no outgoing edges, and (iii) every interior vertex v has three outgoing edges where two of them are the extreme edges in G (these are the edges incident to the big angle) and one is an edge to a vertex in the boundary of the face of G containing the big angle at v.

We give a computation of such a graph $\vec{G}$ similar to the one of Hass et al. [16] for completeness. First for each vertex v the two extreme edges incident to its big angle are oriented from v to the respective neighbors (it might happen that some edges are oriented in both directions). Next, we triangulate the underlying graph of G as follows. Consider an inner face of G. If the face contains no big angles, we are done. Otherwise let a, b, and c be the vertices corresponding to the three small angles and let X_{ab}, X_{bc}, and X_{ac} denote the vertices between a and b, between b and c, and between a and c, respectively. Assume that $X_{a,b} \neq \emptyset$. Let x and y denote the vertices in $X_{a,b}$ adjacent to a respectively b (it might happen that $x = y$). Then we add directed diagonals from each vertex in $X_{a,b}$ to c, from each vertex in $X_{a,c}$ to x, and from each vertex in $X_{b,c}$ to y. See Figure 7 for an illustration. We do this for all faces and the obtained triangulation satisfies the conditions (i) – (iii) stated above. Finally we remove all undirected edges. This construction can be implemented in linear time by traversing the boundary of each face once.



(a) (b)

Figure 7 (a) An inner face in a CPPT with big angles marked with a solid circle. (b) The directed diagonals added to that face to obtain a directed graph satisfying the assumption of Lemma 10.

Now we are ready to describe how to check whether a given plane graph G is Laman. Note that a graph that admits a CPPT assignment has exactly $2n-e$ edges [16]. The simpler algorithm uses the algorithms of Haas et al. [16] to find a CPPT assignment and the directed plane graph $\vec{G}$ from Lemma 10, and then the algorithm from Theorem 1 to decide whether the set of interior vertices of $\vec{G}$ is 3-connected to the set of boundary vertices. Note that each vertex in $\vec{G}$ is either an inner vertex or on the boundary. This decides whether G is a plane Laman graph by Theorem 1 and Lemma 10 and has running time $O(n^{3/2})$. The faster algorithm works as follows.

Proof of Theorem 3. First, search for a CPPT assignment of G using the algorithm from Lemma 9. If there is no such assignment then G is not Laman. Otherwise G is Laman if and only if the CPPT computed in the first step is stretchable. To check this we compute the auxiliary directed plane graph $\vec{G}$ from Lemma 10 such that the CPPT is stretchable if and only if the set of interior vertices of $\vec{G}$ is 3-connected to the set of boundary vertices of $\vec{G}$. Note that the boundary vertices of $\vec{G}$ are clearly incident to a common face. Hence we can check the connectivity property using the algorithm from Theorem 2. This algorithm decides whether G is a plane Laman graph by Theorem 2 and Lemma 10 and has running time $O(n \log^3 n)$. ◀

5 Conclusions and further directions

An obvious direction for future research is to search for faster or simpler algorithms recognizing (planar) Laman graphs.

Our algorithms do not provide any certificate for their correctness. This could be a Henneberg sequence [17] or a decomposition into two acyclic subgraphs [7]. We do not know how to compute either of these faster than using the algorithm of Gabow and Westermann [13].

Further our strategy heavily depends on planarity. Thus it seems unlikely that we can extend our approach to nonplanar Laman graphs. Instead it seems interesting to extend our strategies to more general pseudotriangulations. Orden et al. [29] show a characterization of general pseudotriangulations that extends the one for PPTs described in Section 4. Again there is a notion of combinatorial pseudotriangulations (CPT) which are stretchable if and only if certain combinatorial conditions hold. Besides a connectivity condition similar to the one for CPPTs there is an additional condition on sizes of subgraphs. In particular, a CPT might not be stretchable although the underlying graph is realizable as a pseudotriangulation. We do not know how to identify the stretchable CPTs.

Instead of asking if a graph has a representation as a (pointed) pseudotriangulation one can also ask for other representations such as straight-line triangle representations [1]. For this problem no polynomial time algorithm is known, although the problem shares many similarities with the Laman graph recognition problem.

Finally it is of independent interest to see if the connectivity results for directed graphs can be extended. In the last part of the introduction we describe a naive approach checking for a general directed graph whether a set S is k-connected to a set T with time in $O(|S|m)$.

When each vertex is either in S or in T we can adapt the ideas presented in Section 2 for classes of graphs with small separator. Then the running time becomes linear for graphs with separators of constant size and stays in $O(n^{3/2})$ as long as there are separators of size $O(\sqrt{n})$. We do not know how to improve upon the naive approach for general directed graphs when some vertices are not in $S \cup T$.

References

1 Nieke Aerts and Stefan Felsner. Straight line triangle representations. *Discrete Comput. Geom.*, 57(2):257–280, 2017.

2 Jørgen Bang-Jensen and Gregory Gutin. *Digraphs*. Springer Monographs in Mathematics. Springer, London, second edition, 2009.

3 Alex R. Berg and Tibor Jordán. Algorithms for graph rigidity and scene analysis. In *Algorithms—ESA 2003*, volume 2832 of *Lecture Notes in Comput. Sci.*, pages 78–89. Springer, Berlin, 2003.

4 Glencora Borradaile, David Eppstein, Amir Nayyeri, and Christian Wulff-Nilsen. All-pairs minimum cuts in near-linear time for surface-embedded graphs. In *32nd International Symposium on Computational Geometry (SoCG'16)*, volume 51 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 22:1–22:16, Dagstuhl, Germany, 2016. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

5 Glencora Borradaile, Philip N. Klein, Shay Mozes, Yahav Nussbaum, and Christian Wulff-Nilsen. Multiple-source multiple-sink maximum flow in directed planar graphs in near-linear time. *SIAM J. Comput.*, 46(4):1280–1303, 2017.

6 Robert Connelly, Erik D. Demaine, and Günter Rote. Straightening Polygonal Arcs and Convexifying Polygonal Cycles. *Discrete Comput. Geom.*, 30(2):205–239, 2003.

7 Ovidiu Daescu and Anastasia Kurdia. Towards an optimal algorithm for recognizing Laman graphs. *J. Graph Algorithms Appl.*, 13(2):219–232, 2009.

8 David Eisenstat and Philip N. Klein. Linear-time algorithms for max flow and multiple-source shortest paths in unit-weight planar graphs. In *45th ACM Symposium on Theory of Computing (STOC'13)*, pages 735–744. ACM, New York, 2013.

9 Zsolt Fekete, Tibor Jordán, and Walter Whiteley. An inductive construction for plane Laman graphs via vertex splitting. In *Algorithms—ESA 2004*, volume 3221 of *Lecture Notes in Comput. Sci.*, pages 299–310. Springer, Berlin, 2004.

10 Lester R. Ford, Jr. and Delbert R. Fulkerson. *Flows in networks*. Princeton University Press, Princeton, N.J., 1962.

11 Harold N. Gabow. A matroid approach to finding edge connectivity and packing arborescences. *J. Comput. System Sci.*, 50(2):259–273, 1995.

12 Harold N. Gabow. Using expander graphs to find vertex connectivity. *J. ACM*, 53(5):800–844, 2006.

13 Harold N. Gabow and Herbert H. Westermann. Forests, frames, and games: algorithms for matroid sums and applications. *Algorithmica*, 7(5-6):465–497, 1992.

14 Loukas Georgiadis. Testing 2-vertex connectivity and computing pairs of vertex-disjoint s-t paths in digraphs. In *37th International Colloquium on Automata, Languages and Programming (ICALP'10)*, volume 6198 of *Lecture Notes in Comput. Sci.*, pages 738–749. Springer, Berlin, 2010.

15 Loukas Georgiadis, Daniel Graf, Giuseppe F. Italiano, Nikos Parotsidis, and Przemysław Uznański. All-pairs 2-reachability in $O(n^\omega \log n)$ time. In *44th International Colloquium on Automata, Languages, and Programming*, volume 80 of *LIPIcs. Leibniz Int. Proc. Inform.*, pages 74:1–74:14, Dagstuhl, Germany, 2017. Schloss Dagstuhl–Leibniz-Zentrum für Informatik.

16 Ruth Haas, David Orden, Günter Rote, Francisco Santos, Brigitte Servatius, Herman Servatius, Diane Souvaine, Ileana Streinu, and Walter Whiteley. Planar minimally rigid graphs and pseudo-triangulations. *Comput. Geom.*, 31(1-2):31–61, 2005.

17 Lebrecht Henneberg. Die Graphische Statik der Starren Körper. In Felix Klein and Conrad Müller, editors, *Encyklopädie der Mathematischen Wissenschaften mit Einschluss ihrer Anwendungen: Vierter Band: Mechanik*, pages 345–434. Vieweg+Teubner Verlag, Wiesbaden, 1908.

18 Jacob Holm, Eva Rotenberg, and Mikkel Thorup. Planar reachability in linear space and constant time. In *56th Annual IEEE Symposium on Foundations of Computer Science (FOCS'15)*, pages 370–389. IEEE Computer Society, Los Alamitos, CA, 2015.

19 Giuseppe F. Italiano, Luigi Laura, and Federico Santaroni. Finding strong bridges and strong articulation points in linear time. *Theoret. Comput. Sci.*, 447:74–84, 2012.

20 Donald J. Jacobs and Bruce Hendrickson. An algorithm for two-dimensional rigidity percolation: the pebble game. *J. Comput. Phys.*, 137(2):346–365, 1997.

21 Tibor Jordán. Combinatorial rigidity: graphs and matroids in the theory of rigid frameworks. In *Discrete geometric analysis*, volume 34 of *MSJ Mem.*, pages 33–112. Math. Soc. Japan, Tokyo, 2016.

22 Haim Kaplan and Yahav Nussbaum. Maximum flow in directed planar graphs with vertex capacities. *Algorithmica*, 61(1):174–189, 2011.

23 Stephen Kobourov, Torsten Ueckerdt, and Kevin Verbeek. Combinatorial and geometric properties of planar Laman graphs. In *24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'12)*, pages 1668–1678. SIAM, Philadelphia, PA, 2012.

24 Jakub Łącki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single Source – All Sinks Max Flows in Planar Digraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science (FOCS'12)*, pages 599–608. IEEE Computer Society, Washington, DC, 2012.

25 Gerard Laman. On graphs and rigidity of plane skeletal structures. *J. Engrg. Math.*, 4:331–340, 1970.

26 Audrey Lee and Ileana Streinu. Pebble game algorithms and sparse graphs. *Discrete Math.*, 308(8):1425–1437, 2008.

27 Richard J. Lipton and Robert E. Tarjan. A separator theorem for planar graphs. *SIAM J. Appl. Math.*, 36(2):177–189, 1979.

28 Shay Mozes, Kirill Nikolaev, Yahav Nussbaum, and Oren Weimann. Minimum cut of directed planar graphs in $O(n \log \log n)$ time. In *29th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'18)*, pages 477–494. SIAM, Philadelphia, PA, 2018.

29 David Orden, Francisco Santos, Brigitte Servatius, and Herman Servatius. Combinatorial pseudo-triangulations. *Discrete Math.*, 307(3-5):554–566, 2007.

30 James B. Orlin. Max flows in $O(nm)$ time, or better. In *24th ACM Symposium on Theory of Computing (STOC'13)*, pages 765–774. ACM, New York, 2013.

31 Günter Rote, Francisco Santos, and Ileana Streinu. Pseudo-triangulations—a survey. In *Surveys on discrete and computational geometry*, volume 453 of *Contemp. Math.*, pages 343–410. Amer. Math. Soc., Providence, RI, 2008.

32 Jens M. Schmidt. A simple test on 2-vertex- and 2-edge-connectivity. *Inform. Process. Lett.*, 113(7):241–244, 2013.

33 Ileana Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete Comput. Geom.*, 34(4):587–635, 2005.