A Henneberg-based Algorithm for Generating Tree-Decomposable Minimally Rigid Graphs

Marta R. Hidalgo[a,1], Robert Joan-Arinyo[b,2]

[a] *Systems Genomics Lab, Centro de Investigación Príncipe Felipe, València, Spain*
[b] *GIE, Universitat Politècnica de Catalunya, Barcelona, Catalonia*

Abstract

In this work we describe an algorithm to generate tree-decomposable minimally rigid graphs on a given set of vertices V. The main idea is based on the well-known fact that all minimally rigid graphs, also known as Laman graphs, can be generated via Henneberg sequences. Given that not each minimally rigid graph is tree-decomposable, we identify a set of conditions on the way Henneberg steps are applied so that the resulting graph is tree-decomposable. We show that the worst case running time of the algorithm is $O(|V|^3)$.

Keywords:
Minimally rigid graphs, Laman graphs, Henneberg sequences, Geometric constraint solving, geometric constraint graphs, tree-decomposition.

1. Introduction

We address the problem of generating tree-decomposable minimally rigid graphs, also known as Laman graphs, by applying sequences of Henneberg constructions on a given set of vertices.

This kind of graphs are of interest in graph-based geometric constraint solving and its applications in many different fields, such as computer-aided design, molecular modelling, tolerance analysis and theorem proving. In graph-based geometric constraint solving technology, the problem is defined as a rough sketch

[1] *Email Address:* mhidalgo@cipf.es
[2] *Email Address:* robert@cs.upc.edu

of an object made out of simple geometric elements. Then the user selects the intended exact shape by annotating the sketch with constraints. The resulting annotated sketch is captured as a graph where vertices are geometric elements and edges are the constraints. Finally, a geometric constraint solver checks whether the set of geometric constraints coherently defines the object and, if so, determines the position of the geometric elements.

The success of the geometric constraint solver depends to a great extent on the combinatorial properties of the graph. If the graph is minimally rigid, the geometric constraint problem defines a rigid object and, consequently, the solution to the constraint problem has finitely many solution instances, [1]. If the graph is tree-decomposable, tools developed in graph-based geometric constraint solving can be applied to solve the constraint problem at hand, [2, 3, 4, 5, 6].

It is well known that the set of graphs generated by Henneberg sequences and the set of minimally rigid graphs is the same set, [7, 8, 9, 10]. However, not every minimally rigid graph is tree-decomposable [11, 12]. Hence, the idea that guided this work was to find out conditions on the application of Henneberg construction steps so that the resulting graph is minimally rigid and tree-decomposable.

In this paper we present a theory that characterizes tree-decomposable minimally rigid graphs by an inductive construction of Henneberg steps. Then we describe an algorithm for generating tree-decomposable minimally rigid graphs of a given order based on this theory.

The rest of the paper is organized as follows. In Section 2 we recall known theoretical results that we will use later on. In Section 3 we provide new theoretical results that characterize a class of Henneberg sequences which generates tree-decomposable minimally rigid graphs. Section 4 is devoted to describing the algorithm that actually builds this kind of graphs on a set of given vertices V. The algorithm implements the results of the previous section. We show that the algorithm's worst case running time is $O(|V|^3)$. We provide some conclusions in Section 5. Finally, proofs for the theorems in the manuscript are developed in Section 6.

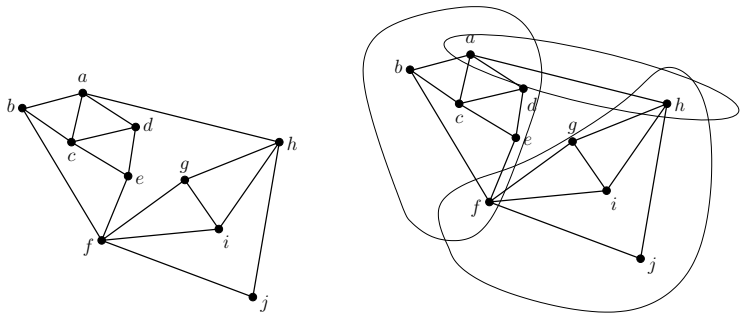


Figure 1: a) Graph. b) Graph tree decomposition step induced by the triple $\{a, h, f\}$.

2. Preliminaries

In this section we describe tools that will be used later on. First we define the concept of graph tree-decomposability and recall Henneberg constructions. Then we formalize sequences of Henneberg constructions as rewrite systems. Finally we recall a characterization of minimally rigid graphs.

2.1. Tree-decomposable Graphs

Consider the graph $G = (V, E)$ and let $G_1 = (V_1, E_1)$, $G_2 = (V_2, E_2)$ and $G_3 = (V_3, E_3)$ be three subgraphs of G such that

$$V = V_1 \cup V_2 \cup V_3, \qquad E = E_1 \cup E_2 \cup E_3$$

and the set of vertices pairwise share one vertex

$$V_1 \cap V_2 = \{a\}, \quad V_2 \cap V_3 = \{b\}, \quad V_3 \cap V_1 = \{c\}$$

We say that $\{G_1, G_2, G_3\}$ is a ternary decomposition of G induced by the vertices $\{a, b, c\}$. Figure 1a shows a graph and Figure 1b shows a ternary decomposition induced by the vertices $\{a, h, f\}$. In what follows we will refer to this set of vertices as *triple of hinges* or just as *triple*.

Definition 2.1. Let $G = (V, E)$ be a graph. A ternary tree T is a *tree-decomposition* of G if

1. G is the root of T,
2. Each node $G' \subseteq G$ of T is the father of exactly three nodes, say $\{G'_1, G'_2, G'_3\}$, which is a ternary decomposition of G', and
3. Each leaf node is the graph $(\{a, b\}, \{(a, b)\})$, that is, an edge (a, b) of $E(G)$.

A graph for which there is a tree-decomposition is called *tree-decomposable*.

In general, a tree-decomposition of a graph is not unique. Figure 2 shows two different tree-decompositions for the graph given in Figure 1a. For the sake of clarity, tree-decompositions only show the set of vertices within each node. The label on each tree edge is the triple of hinges that induces the ternary decomposition.

2.2. Henneberg Constructions

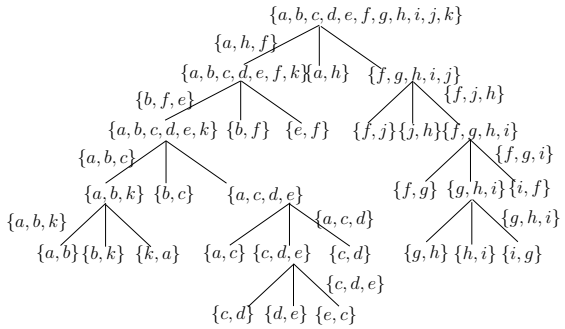
In this section we recall Henneberg constructions. For an in-depth study on this subject see [13, 14, 9]. Henneberg constructions include two different construction steps defined as follows, [13].

1. *Henneberg I (vertex addition).* Let $G = (V, E)$ be a graph with two distinct vertices $v_1, v_2 \in V(G)$ and let $G^* = (V^*, E^*)$ be the graph obtained by attaching to G a new vertex v with edges (v, v_1) and (v, v_2). Then G^* is the graph derived from G by a Henneberg I step. See Figure 3.
2. *Henneberg II (edge replacing).* Let $G = (V, E)$ be a graph with an edge $e = (v_1, v_2) \in E(G)$ and a third vertex $v_3 \in V(G)$. The graph $G^* = (V^*, E^*)$ obtained from G by deleting the edge (v_1, v_2) and inserting a new vertex v plus three edges $(v, v_1), (v, v_2)$ and (v, v_3) is the graph derived from G by a Henneberg II step. See Figure 4.

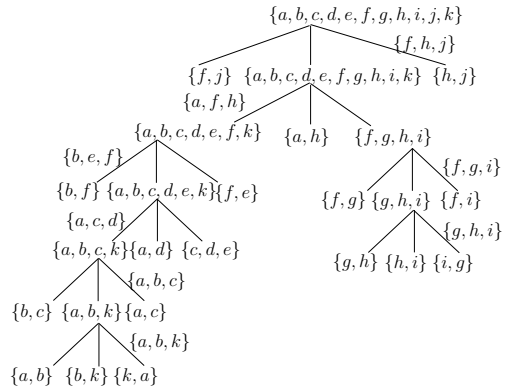
In what follows Henneberg I and Henneberg II steps will be denoted as H1S and H2S respectively.

2.3. Henneberg Sequences as Rewrite Systems

Henneberg constructions are inductive constructions of sequences of graphs such that each graph in a sequence is obtained from the previous one by applying

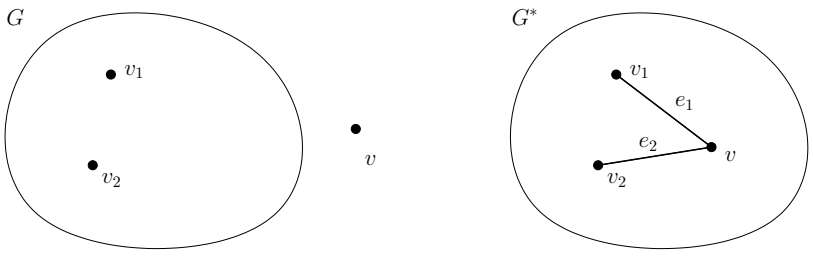


(a)



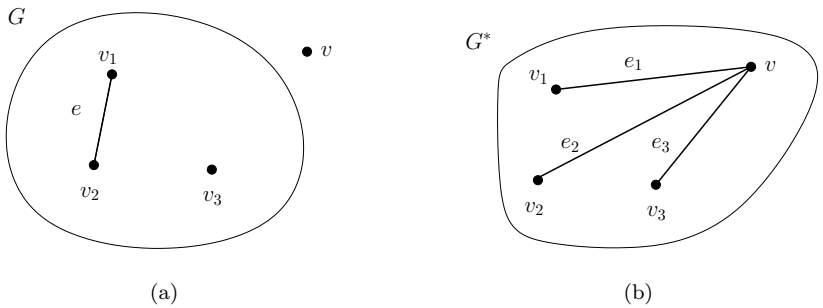
(b)

Figure 2: Two different tree-decompositions for the graph shown in Figure 1a. Labels in tree edges are the ternary decomposition triples.



(a) (b)

Figure 3: a) Graph G and new vertex v. b) Graph G^* derived from gaph G by application of a Henneberg I step.



(a) (b)

Figure 4: a) Graph G and new vertex v. b) Graph G^* derived from graph G by the application of a Henneberg II step.

one of the steps defined in the previous section. This idea can be formalized by associating a rewrite system with each Henneberg construction as follows.

Let $V = \{v_1, v_2, \ldots, v_n\}$ be a set of vertices, let a, b, c be three different vertices in V and E_0 be the set of edges $\{(a, b), (b, c), (c, a)\}$. Let $\mathbf{G}$ be the set of Henneberg graphs that can be built on V. Consider the graph $G_0 = (V, E_0)$ and define the set $\rightarrow_\rho = \{\rightarrow_1, \rightarrow_2\}$ where $\rightarrow_1$ and $\rightarrow_2$ denote respectively H1S and H2S steps. Then the pair $(\mathbf{G}, \rightarrow_\rho)$ is a rewrite system with starting term $\mathbf{G}_0 = (V, E_0)$ and reduction rules set $\rightarrow_\rho$, [15]. Notice that the graph G_0 is the K_3 graph induced by the vertices a, b, c in V.

Definition 2.2. A *derivation* in $(\mathbf{G}, \rightarrow_\rho)$ is any sequence

$$G_0 \rightarrow_\rho G_1 \rightarrow_\rho G_2 \rightarrow_\rho \ldots \rightarrow_\rho G_k$$

of applications of reduction rules in $\rightarrow_\rho$.

In general a derivation in $(\mathbf{G}, \rightarrow_\rho)$ will be written as $G_0 \rightarrow_\rho^* G^*$. When needed, we shall refer to it as the *Henneberg derivation* of G^*.

2.4. Minimally Rigid Graphs

Minimally rigid graphs are the fundamental objects in 2-dimensional Rigidity Theory. They are known as Laman, isostatic or generically minimally rigid graphs and combinatorially capture the property that a graph, embedded on a

generic set of points in the plane, is infinitesimally rigid. Concerning rigidity see, for example, [8, 16] and references therein. Minimally rigid graphs are characterized in [17] in four equivalent different ways. Here we are interested in the following two:

Theorem 2.1. *A minimally rigid graph $G = (V, E)$ with $|V| \geq 2$ can be characterized in any of the following two equivalent ways:*

1. *(Laman's theorem) If the number of vertices and edges are respectively $|V| = n$ and and $m = |E|$ then G is minimally rigid if $m = 2n - 3$ and every subset of $k \geq 2$ vertices spans at most $2k - 3$ edges.*
2. *(Henneberg's theorem) There is a Henneberg construction for G.*

Figure 5 shows the construction of an example graph G with six vertices demostrating the application of Henneberg steps. Vertices v_1, v_2, v_3 belong to the current graph and v is the new vertex to be added to the graph by the construction step. H1S steps are applied on vertices labeled v_1, v_2 and v. H2S steps are applied on vertices labeled v_1, v_2, v_3 and v. Notice that the first graph as well as each graph resulting from a Henneberg step fulfills the Laman conditions in Theorem 2.1.

3. Henneberg Constructions and Tree-Decomposability

Two different families of graphs generated by Henneberg derivations are of special interest, [18, 19]. One family, denoted $\mathbf{H}_1$, includes those graphs derived by $K_3 \rightarrow_1^* G^*$. The other family, denoted $\mathbf{H}$, includes those graphs derived by $K_3 \rightarrow_\rho^* G^*$. As shown in Section 2, the set of graphs $\mathbf{H}$ and the set of minimally rigid graphs are the same set.

In general, minimally rigid graphs are not tree-decomposable. However we are specifically interested on those graphs which are both minimally rigid and tree-decomposable. In order to achieve this goal, we identify the conditions under which each reduction in $\rightarrow_\rho$ preserves graph tree-decomposability and, then, resulting Henneberg sequences generate graphs within the desired class.

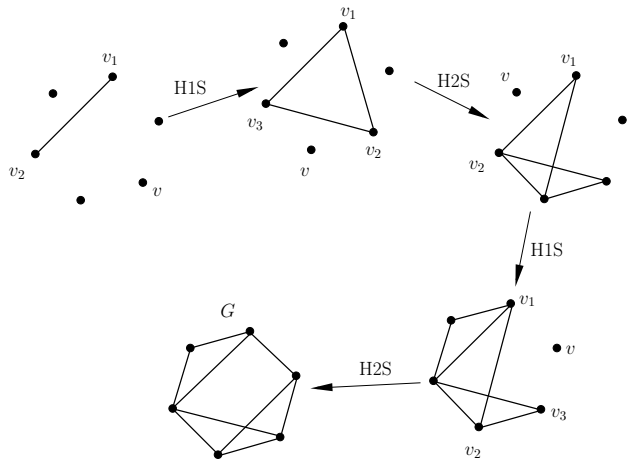


Figure 5: Graph G built by a sequence of Henneberg steps. Vertices v_1, v_2, v_3 belong to the current graph. Vertex v is the vertex to be added to the graph. H1S constructions are applied on vertices $\{v_1, v_2, v\}$. H2S constructions are applied on vertices $\{v_1, v_2, v_3, v\}$.

3.1. H1S and Tree-Decomposability

It is easy to see that the H1S reduction preserves tree-decomposability and, therefore, the following result holds.

Theorem 3.1. *Let G and G^* be two graphs such that $G \rightarrow_1 G^*$. Then G^* is tree-decomposable if and only if G is tree-decomposable.*

Proofs for results in this section have been included in Section 6. They can also be found in [20]. Figure 6 shows a derivation, $K_3 \rightarrow_1^* G^*$, including only H1S steps that yields a tree-decomposable graph. Notice that given that K_3 is trivially tree-decomposable, reversing the Henneberg sequence yields a tree-decomposition.

3.2. H2S and Tree-Decomposability

Henneberg sequences which include H2S steps create minimally rigid graphs. However, these graphs are not necessarily tree-decomposable. Figure 7 shows a Henneberg sequence for the Desargues graph, [18], where the H2S step removes edge (a, d) and adds the new vertex f plus edges $(f, a), (f, d)$ and (f, e). The resulting graph is minimally rigid but is not tree-decomposable.

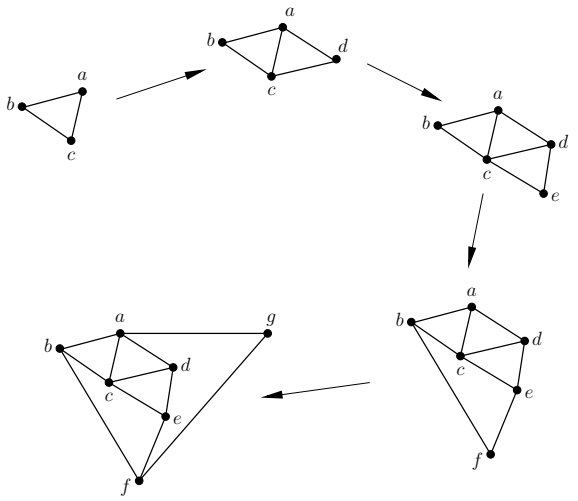


Figure 6: A Hennberg sequence of H1S steps guarantees that the graph generated is tree-decomposable.

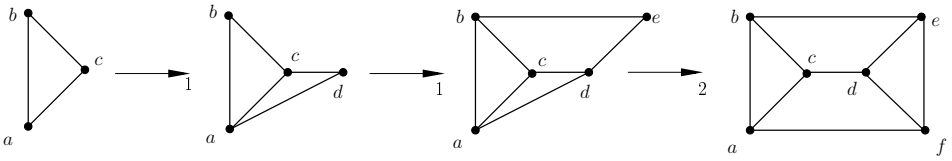


Figure 7: A Henneberg sequence with H2S steps that creates the Desargues graph which is minimally rigid but is not tree-decomposable.

We start establishing the conditions under which an H2S step preserves graph tree-decomposability.

Theorem 3.2. *Let $G = (V, E)$ be a graph in the $\mathbf{H}_1$ family and G^* be the graph such that $G \to_2 G^*$, where the H2S step involves the edge $(v_1, v_2) \in E(G)$ and vertex $v_3 \in V(G)$. If $\{v_1, v_2, v_3\}$ is a triple of hinges for some reduction in the derivation $K_3 \to_1^* G$, then $G^* \in \mathbf{H}_1$.*

Therefore, graphs created by derivations $K_3 \to_1^* G$ are tree-decomposable whenever reductions $\to_2$ are applied under the conditions in Theorem 3.2. This is a limited way of building tree-decomposable minimally rigid graphs. The main result will broaden the scope of this result.

The statement of the main result requires the concept of *lowest common ancestor* of a set of leaf nodes in a tree-decomposition. First we recall from graph theory the concept of *lowest common ancestor* of two vertices in a tree, [21], then we adapt this concept to our needs.

Let $\mathbf{T}$ be a rooted tree. A vertex $u \in \mathbf{T}$ is an *ancestor* of a vertex $v \in \mathbf{T}$ if the path from the root of $\mathbf{T}$ to v goes through u. A vertex $w \in \mathbf{T}$ is a *common ancestor* of u and v if it is an ancestor of both u and v. The *lowest common ancestor* of vertices $u, v \in \mathbf{T}$ is the common ancestor of vertices u, v for which the path from the root is maximal.

Now, with each tree-decomposable graph $G = (V, E)$ we associate a rooted tree $\mathbf{T}$ corresponding to the tree-decomposition of G. Notice that each vertex in $\mathbf{T}$ is a subgraph of G and that the root, $\mathbf{T}_0$, is the given graph G.

Finally, we define the *lowest common ancestor of vertices* $u, v, w, \ldots \in V(G)$ as the lowest common ancestor of the set of leaf nodes in $\mathbf{T}$ which include vertices $u, v, w, \ldots \in V(G)$. In what follows we shall denote the lowest common ancestor of vertices in the tree-decomposition $\mathbf{T}$ as $LCA(u, v, w, \ldots)$. We do not allow a vertex to be a descendant of itself.

We are now ready to state our main result concerning the H2S step. Refer to Figure 8.

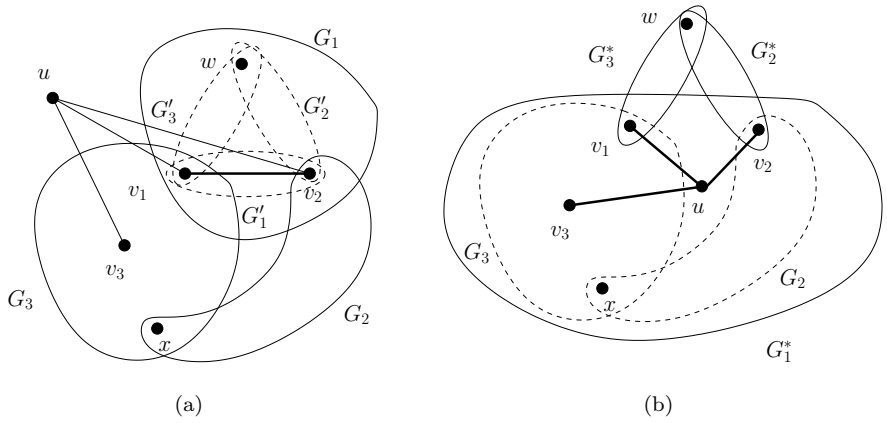


Figure 8: a) Given graph G. The triple $\{v_1, v_2, x\}$ induces the decomposition G_1, G_2, G_3 and $\{v_1, v_2, w\}$ is a triple for G_1. b) Graph G^* created by the reduction $G \rightarrow_{2,\{v_1,v_2,v_3,u\}} G^*$. G^*_1, G^*_2, G^*_3 is a decomposition induced in G^* by the triple $\{v_1, v_2, w\}$.

Theorem 3.3. *Let $G = (V, E)$ be a tree-decomposable minimally rigid graph with $\mathbf{T}$ the associated tree-decomposition. Let G^* be the grah created by the derivation $K_3 \rightarrow_\rho G' \rightarrow_\rho G \rightarrow_2 G^*$ where $\rightarrow_2$ is applied on edge $(v_1, v_2) \in E(G') \subset E(G)$ and vertex $v_3 \in V(G') \subset V(G)$. Let $\hat{G}_1, \hat{G}_2, \hat{G}_3$ be a decomposition of $LCA(v_1, v_2, v_3)$ with $(v_1, v_2) \in E(\hat{G}_i)$ for some $1 \leq i \leq 3$. Then G^* is tree-decomposable if one of the following two holds:*

1. $\hat{G}_i = (\{v_1, v_2\}, \{(v_1, v_2)\})$.
2. *There are two vertices $x, w \in V(G)$ such that $\{v_1, v_2, x\}$ is a triple for G and there is a vertex $w \in V(\hat{G}_i)$ such that $\{v_1, v_2, w\}$ is a triple for $\hat{G}_i$.*

To further illustrate Theorem 3.3 consider the Henneberg sequence in Figure 7 that creates the Desargues graph. Figure 9 shows a tree decomposition, $\mathbf{T}$, for the graph created after the second H1S reduction. Then the H2S reduction is applied on edge (a, d), vertex e and the new vertex f. Therefore $LCA(a, d, e) = \{a, b, c, d, e\}$ is the root of $\mathbf{T}$ in this case. Notice that $\{b, d, e\}$ is the only triple for the current graph and that there is no vertex $w \neq e$ in $LCA(a, d, e)$ such that $\{b, d, w\}$ is a triple of hinges for $LCA(a, d, e)$. Thus the final graph is not tree-decomposable.

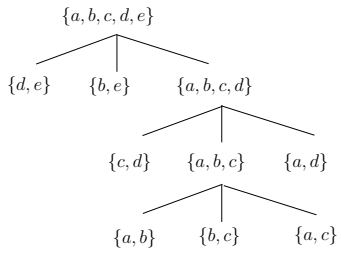


Figure 9: Tree-decomposition for the graph created in Figure 9 after the second H1S reduction, $\rightarrow_1$, is applied.

Now consider the Henneberg sequences in Figure 10. In both sequences, the reduction H2S is applied on edge (a, d) and the new vertex is f. In the sequence at the top, the third vertex in $V(G)$ chosen to apply the construction is c and $LCA(a, d, c) = \{a, b, c, d\}$. Notice that $\{a, d, c\}$ is a triple of hinges for $LCA(a, d, c)$. In the sequence at the bottom, the third vertex considered is b. Again $LCA(a, d, b) = \{a, b, c, d\}$ and $\{a, d, c\}$ is a triple of hinges for $LCA(a, d, b)$. Consequently both sequences create tree-decomposable graphs.

Corollary 3.1. *Let $G = (V, E)$ be a tree-decomposable graph and (v_1, v_2) be an arbitrary edge in $E(G)$. Then there is always a vertex $v_3 \in V(G)$ different from v_1 and v_2 such that the reduction $G \rightarrow_2 G^*$ involving the edge (v_1, v_2) and vertex v_3 creates a tree-decomposable graph G^*.*

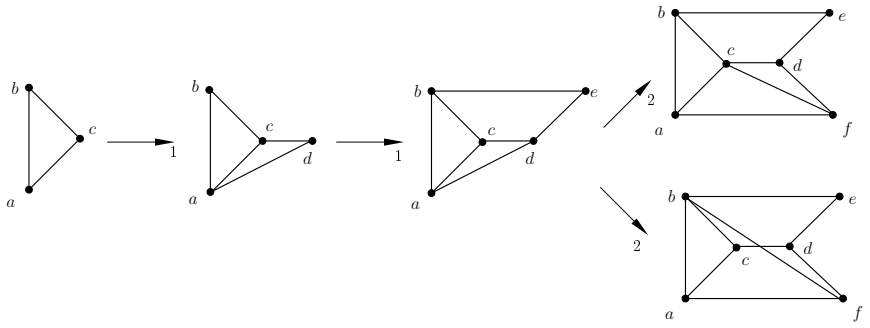


Figure 10: Two Henneberg sequences in H2S that create tree-decomposable minimally rigid graphs.

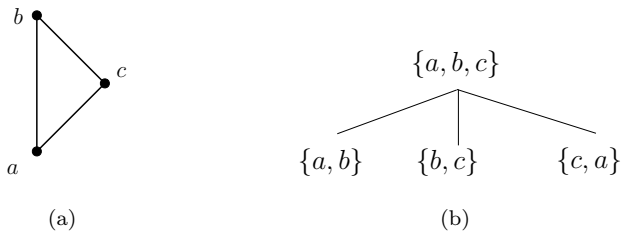


Figure 11: a) Triangle. b) Triangle tree-decomposition.

This result means that any Henneberg derivation $K_3 \rightarrow_\rho^* G$ where G is minimally rigid and tree-decomposable can be extended to build a tree-decomposable minimally rigid graph with an arbitrary order.

4. Algorithm

In this section we describe an algorithm to build tree-decomposable minimally rigid graphs of arbitrary order using Henneberg sequences. To speed up the search for candidates of vertices to be included in the graph, the algorithm dynamically manages a tree-decomposition of the current graph.

4.1. Creating Tree-decomposable Minimally Rigid Graphs

The algorithm to build tree-decomposable minimally rigid graphs computes the required graph $G = (V, E)$ as the Henneberg sequence $K_3 \rightarrow_\rho^* G$. The H1S step is implemented following Theorem 3.1. The H2S step is implemented following Corollary 3.1. Graphs output by the algorithm belong to the **H** family. The procedure is described in Algorithm 1. There we assume that the input data is the set of vertices on which a tree-decomposable minimally rigid graph must be built.

4.2. Updating the Tree Decomposition

We consider tree-decomposable graphs created by Henneberg sequences with the triangle K_3 as the starting term, that is, $K_3 \rightarrow_{\mathbf{H}}^* G$. Notice that the tree decomposition of K_3, shown in Figure 11, is trivial.

Algorithm 1 Building a tree-decomposable minimally rigid graph on a given set of vertices

▷ INPUT

▷ V : given set of vertices

▷ OUTPUT

▷ G : a tree-decomposable minimally rigid graph, G = (V,E)

procedure generate_Graph(V)

G(V', E) = $K_3(V', E_3)$ randomly generated with $V' \subseteq V$

V = V - V'

T = Tree-decomposition of G(V',E)

while V $\neq \emptyset$ **do**

hs = Randomly select a step type among H1S and H2S

if hs == H1S **then**

Randomly select two vertices v1, v2, from V'

Randomly extract a vertex v from V ▷ (v is no longer in V)

V'= V' $\cup$ {v}

E = E $\cup$ {(v, v1), (v, v2)}

T = updateT_H1S(T)

else

Randomly select an edge e from E to be removed

G = predecessor of e in T

G2, G3 = sons of G and brothers of e

Randomly select a vertex v from V(G2) $\cup$ V(G3) with v not in e

V'= V' $\cup$ {v}

E = (E - {(v1, v2)}) $\cup$ {(v, v1), (v, v2), (v, v3)}

T = updateT_H2S(T)

return G = (V', E)

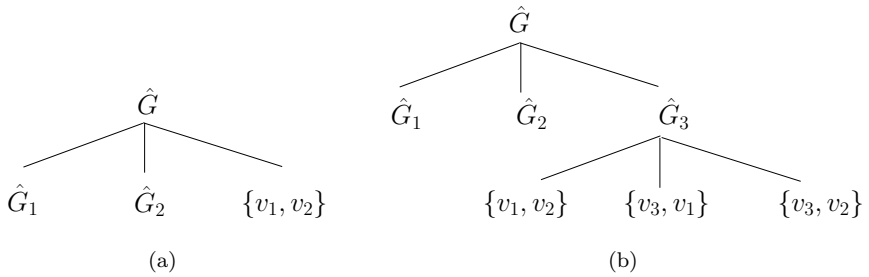


Figure 12: An H1S step applied on vertices v_1, v_2 and the new vertex v_3. Edge (v_1, v_2) is a leaf node of $\mathbf{T}$. a) Tree-decompositon before applying the H1S step. b) Updated tree-decomposition.

As the Henneberg sequence evolves creating new and larger graphs, when an H2S step is applied, the tree-decomposition of the current graph plays a central role to identify candidate vertices on which the reduction generates a new tree-decomposable graph. Thus it is convenient to keep the tree decomposition properly updated. Next we consider how to update the tree decomposition depending on whether the last construction applied is either H1S or H2S, that is, procedures `updateT_H1S()` and `updateT_H2S()` in Algorithm 1.

4.2.1. Procedure updateT_H1S

Assume that a H1S step is applied, $G \rightarrow_1 G^*$, over the vertices v_1, v_2 and that the new vertex is v_3. We distinguish two different situations. In the first one, vertices v_1, v_2 bound an edge which is a leaf node in $\mathbf{T}$. That is, in the tree-decomposition there is a node $\hat{G}$ decomposed into $\hat{G}_1, \hat{G}_2$ and $(\{v_1, v_2\}, \{(v_1, v_2)\})$ as depicted in Figure 12a. $\mathbf{T}$ is updated in two steps as follows. Refer to Figure 12b.

1. Replace the leaf node graph $(\{v_1, v_2\}, \{(v_1, v_2)\})$ with the tree rooted at the graph $\hat{G}_3 = (\{v_1, v_2, v_3\}, \{(v_1, v_2), (v_3, v_1), (v_3, v_2)\})$ which is decomposed as $\hat{G}'_1 = (\{v_1, v_2\}, \{(v_1, v_2)\})$, $\hat{G}'_2 = (\{v_3, v_1\}, \{(v_3, v_1)\})$, and $\hat{G}'_3 = (\{v_3, v_2\}, \{(v_3, v_2)\})$.
2. Propagate vertex v_3 and edges $(v_3, v_1), (v_3, v_2)$ through $\mathbf{T}$ up to the root.

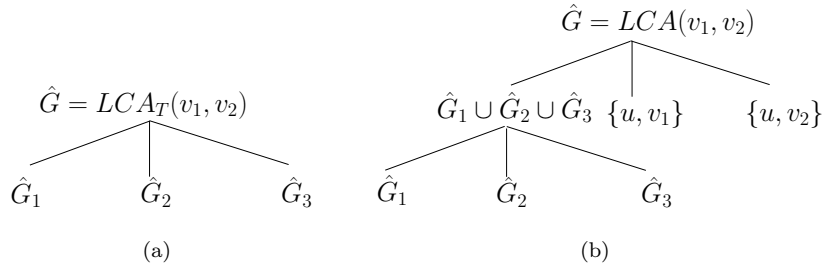


Figure 13: An H1S step applied on vertices v_1, v_2 and the new vertex v_3. Vertices v_1 and v_2 belong to different branches of the subtree of $\mathbf{T}$ rooted at $LCA(v_1, v_2)$. a) Tree-decomposition before applying the H1S step. b) Updated tree-decomposition.

In the second situation vertices v_1, v_2 belong to different branches of the subtree of $\mathbf{T}$ rooted at $LCA(v_1, v_2)$, that is, they do not bound an edge in a tree-decomposition leaf. The update procedure now takes five steps. See Figure 13.

1. Identify $\hat{G} = LCA(v_1, v_2)$ and let $\hat{G}_1$, $\hat{G}_2$, $\hat{G}_3$ be the decomposition of $\hat{G}$ with, say, $v_1 \in V(\hat{G}_i)$ and $v_2 \in V(\hat{G}_j)$ and $i \neq j$.
2. Replace $\hat{G}_1$ with a tree rooted at $\hat{G}_1 \cup \hat{G}_2 \cup \hat{G}_3$ whose decomposition is $\hat{G}_1$, $\hat{G}_2$, $\hat{G}_3$.
3. Replace $\hat{G}_2$ with $(\{v_3, v_1\}, \{(v_3, v_1)\})$.
4. Replace $\hat{G}_3$ with $(\{v_3, v_2\}, \{(v_3, v_2)\})$.
5. Propagate vertex v_3 and edges $(v_3, v_1), (v_3, v_2)$ through $\mathbf{T}$ up to the root.

4.2.2. Procedure updateT_H2S

Now assume that the reduction to be applied is an H2S, $G \rightarrow_2 G^*$, where the edge and vertex involed are (v_1, v_2) and v_3 respectively. Let u be the new vertex. Notice that edge (v_1, v_2) and vertex v_3 necessarily belong to different branches in the current tree-decomposition $\mathbf{T}$. The algorithm has the following steps:

1. Identify $\hat{G} = LCA(v_1, v_2, v_3)$ and let $\hat{G}_1$, $\hat{G}_2$, $\hat{G}_3$ be the decomposition of $\hat{G}$ with, say, $(v_1, v_2) \in E(\hat{G}_1)$ and $v_3 \in V(\hat{G}_2)$.

2. Replace $\hat{G}_1$ with $(\{u, v_i\}, \{(u, v_i)\})$ where v_i is either v_1 or v_2.
3. Replace $\hat{G}_2$ with the new graph $\hat{G}'_2$ such that $V(\hat{G}'_2) = V(\hat{G}_2) \cup \{u, v_j\}$ and $E(\hat{G}'_2) = (E(\hat{G}_1) \cup E(\hat{G}_2)) - \{(v_1, v_2)\}) \cup \{(u, v_j), (u, v_3)\}$, where v_j is either v_1 or v_2 and $v_j \neq v_i$ with v_i the vertex chosen in step 2.
4. Propagate vertex u and edges $(u, v_1), (u, v_2), (u, v_3)$ through $\mathbf{T}$ up to the root.

4.3. Complexity

We analize the complexity of our algorithm as the worst case running time by applying the usual unit-cost operations, unbounded memory random access machine computational model which have unit cost for read and write access to all of its memory cells, [22]. The input size for the measures will be the number of nodes $|V|$ on which the output graph should be built.

Let $G' = (V', E')$ be the current tree-decomposable minimally rigid graph built and $\mathbf{T}$ the associted tree-decomposition. Assuming that nodes in $\mathbf{T}$ store a pointer to its predecessor, all the computations in the algorithm run in constant time except those that update the current tree-decomposition $\mathbf{T}$ according to the Henneberg step applied.

The starting graph for the Henneberg sequence is K_3. Let $\mathbf{T}$ be the associated tree-decomposition. The number of nodes in $\mathbf{T}$ is $n = 4$, the number of edges is $m = 3$ and the tree height is $h = 1$, as shown in Figure 11. Notice that each leaf in $\mathbf{T}$ stores one graph with just one edge and its bounding vertices.

Consider now the running time of propagating vertices and edges in $\mathbf{T}$. Notice that in both H1S and H2S steps, propagating entails visiting the ancestors of nodes starting in the LCA of the vertices involved in the Henneberg step up to the root of $\mathbf{T}$. Therefore the number of vertices to be visited is given by the current tree-decomposition height. A tree-decomposition with the tallest height for a given number of nodes is illustrated in Figure 14. Given that each tree level includes four edges except the first one that includes three, the tree-decomposition height is $h \leq |E'|$. But $E' \leq 2|V'| - 3$ because the current

graph $G' = (V', E')$ is minimally rigid, thus the tree-decomposition height is $h \leq 2|V'| - 3$. Consequently propagation takes at most $O(|V'|)$ time.

Consider now the time needed to identify the LCA in the current tree-decomposition $\mathbf{T}$. First, let us see how the number of nodes in $\mathbf{T}$ changes depending on the Henneberg step type applied. Let n be the number of nodes in $\mathbf{T}$ and apply one H1S step on vertices v_1, v_2 and a new vertex v_3. Consider the case where (v_1, v_2) is an edge in $G' = (V', E')$. The update results in a new tree-decomposition $\mathbf{T}$ where the number of nodes is $n + 2$ (Figure 12). Similarly, when vertices v_1 and v_2 do not bound an edge and thus belong to different branches of the subtree of $\mathbf{T}$ rooted at $LCA(v_1, v_2)$, the number of nodes in the updated tree-decomposition is $n+4$ (Figure 13). Finally, the application of one H2S step does not change the number of nodes in $\mathbf{T}$.

After applying m Henneberg steps, and considering the largest increment in the number of tree-decomposition nodes, we have that the number of nodes is

$$n \leq \sum_{i=1}^{m} 4i = 4\frac{m(1+m)}{2}$$

Recall that each Henneberg step consumes one vertex from the input set of vertices V. Thus $m \leq |V|$ and $n \leq 2|V|(1 + |V|)$. That is, $n = O(|V|^2)$.

According to [23], to identify the $LCA(v_1, v_2)$ in a tree takes constant time in the number of nodes of the tree, after a linear preprocessing of the input tree. In our algorithm, $\mathbf{T}$ changes as the generated graph evolves, thus we need to apply the preprocess after each tree-decomposition update. Therefore the work done while identifying the $LCA(v_1, v_2)$ is $O(|V|^2)$.

The main loop in the algorithm visits each vertex in $V - V(K_3)$ just once. Thus completing the loop takes $O(|V|)$ time. The fact that the number of vertices in the current graph, $|V'|$, is at most the number of total nodes, $|V|$, completes the proof that the algorithm's worst case running time is $O(|V|^3)$.

5. Conclusion

We have described an algorithm to build tree-decomposable minimally rigid graphs on a given set of vertices, say V. The algorithm is based on Henneberg

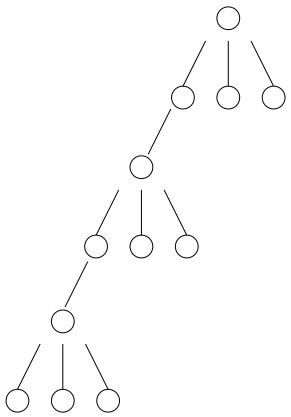


Figure 14: Tree-decomposition of a minimally rigid graph with a maximum height.

constructions where the H2S is applied under conditions that guarantee to preserve graph tree-decomposability. We have formalized these conditions and have shown that they are sound. The algorithm worst case running time has been shown to be $O(|V|^3)$.

As described, the input to the algorithm is a set of vertices on which the tree-decomposable minimally rigid graph is built and the starting term for the Henneberg sequence is the graph K_3. No upper limit on the number of input vertices is imposed.

The algorithm can be easily adapted to deal with other initial conditions. For example, if the starting graph is an egde and the vertices that bound it, say $G = (\{a, b\}, \{(a, b)\})$, we just need to start the Henneberg sequence with an H1S step.

Our approach can be applied to solve the *completion* problem, [24]. Here the input is a tree-decomposable graph, say $G_0 = (V, E_0)$, with $|E_0| < 2|V| - 3$. The goal is to define a set of additional edges E over V such that the graph $G = (V, E_0 \cup E)$ is tree-decomposable and minimally rigid. Now the starting graph in our approach K_3 should be replaced with the subgraph G' induced in G_0 by E_0 and the starting set of vertices would be $V - V(G')$.

In the current implementation, the algorithm randomly selects the H1S or

H2S step type to be applied. An avenue to explore is to study different strategies to select the type of the next construction step to be applied. In geometric constraint solving, strategies could be based on either technological rules convenient for the specific design at hand or the kind of geometric elements on which the new constraint should be defined. Similarly, vertices and edges already included in the graph under construction that are involved in the next construction step are selected at random among all the candidates. Strategies to select vertices and edges, when more than one candidate can be found, would be of great interest to explore the space of tree-decomposable minimally rigid graphs induced by the given set of vertices. In geometric constraint solving these strategies could be defined, for example, taking into account the nature of the geometric elements associated to the constraint graph vertices.

Successful approaches to solve these issues would be of help, for example, in the development of techniques to explore different ways for building rigid frameworks from smaller ones in engineering.

6. Proof of Theorems

In this section we develop proofs for the claims in the manuscript. Here Henneberg sequences will be considered as rewrite systems. Recall from Section 3 that $\rightarrow_1$ and $\rightarrow_2$ denote respectively the reductions corresponding to the H1S and the H2S constructions and that $\rightarrow_\rho$ denotes de set $\{\rightarrow_1, \rightarrow_2\}$. An arbitrary sequence of reductions in $\rightarrow_\rho$ is denoted by $\rightarrow_\rho^*$. When the sequence includes just one of the two reductions we will denoted it as either $\rightarrow_1^*$ or $\rightarrow_2^*$.

We will be interested in establishing a difference between reductions depending on the geometric elements involved. An H1S reduction which adds the new vertex u and two edges, (u, v_1) and (u, v_2), will be denoted as $\xrightarrow{v_1,v_2,u}_1$. An H2S reduction which adds the new vertex u, removes edge (v_1, v_2) and adds three new edges (u, v_1), (u, v_2) and (u, v_3) will be denoted as $\xrightarrow{v_1,v_2,v_3,u}_2$.

Theorem 3.1. Let G and G^* be two graphs such that $G \rightarrow_1 G^*$. Then G^* is tree-decomposable if and only if G is tree-decomposable.

Proof. Assume that G is a tree-decomposable graph and consider the reduction $G \xrightarrow{v_1,v_2,u}_1 G^*$. Then

$$G^* = (V(G) \cup \{u\},\ E(G) \cup \{(v_1, u), (v_2, u)\})$$

and the graphs G, $G_1 = (\{v_1, u\}, \{(v_1, u)\})$ and $G_2 = (\{v_2, u\}, \{(v_2, u)\})$ define a tree-decomposition for G^* with G tree-decomposable. Assume now that G^* is tree-decomposable. For the same reason, G is tree-decomposable. □

Theorem 3.2. Let $G = (V, E)$ be a graph in the $\mathbf{H}_1$ family and G^* be the graph such that $G \to_2 G^*$, where the H2S step involves the edge $(v_1, v_2) \in E(G)$ and vertex $u \in V(G)$. If $\{v_1, v_2, u\}$ is a triple of hinges for some reduction in the derivation $K_3 \to_1^* G$, then $G^* \in \mathbf{H}_1$.

Proof. Assume that $G \in \mathbf{H}_1$, then the derivation $K_3 \to_1^* G$ is a Henneberg sequence for G. If we assume that $\{v_1, v_2, u\}$ is a triple on which a H1S step has been applied, the derivation for G can be rewritten in general as

$$K_3 \to_1^* G' \xrightarrow{v_1,v_2,u}_1 G'' \to_1^* G \tag{1}$$

By definition, reduction $\to_1$ does not remove graph edges and always connects a new vertex to the bounds of a single edge in $E(G')$ with two new edges. Thus after applying reductions $\xrightarrow{v_1,v_2,u}_1$ and $\to_1^*$, edges (v_1, v_2), (u, v_1) and (u, v_2) are in $E(G)$ and clearly v_1, v_2, u are in $V(G)$. Consider the derivation

$$K_3 \to_1^* G' \xrightarrow{v_1,v_2,u}_1 G'' \to_1^* G \xrightarrow{v_1,u,v_2,w}_2 G^* \tag{2}$$

built by extending the derivation (1) with the H2S reduction $\xrightarrow{v_1,u,v_2,w}_2$. Then

$$V(G^*) = V(G) \cup \{w\}$$

with $v_1, v_2, u \in V(G)$ and

$$E(G^*) = (E(G) - \{(u, v1)\}) \cup \{(w, v_1), (w, u), (w, v_2)\} \tag{3}$$

Reductions in derivation (1) belong to the H1S class and vertices $v_1, v_2, u \in V(G)$ and edges $(v_1, v_2), (u, v_1), (u, v_2) \in E(G)$. Hence the derivation

$$K_3 \rightarrow_1^* G' \xrightarrow{v_1, v_2, u} {}_1\, G'' \rightarrow_1^* G \xrightarrow{v_2, u, w} {}_1\, G^{*'} \tag{4}$$

is well defined. Then

$$V(G^{*'}) = V(G) \cup \{w\}$$

with $v_1, v_2, u \in V(G)$ and

$$E(G^{*'}) = E(G) \cup \{(w, v_2), (w, u)\} \tag{5}$$

Let E' denote the set of edges added to $E(G)$ by $G'' \rightarrow_1^* G$ in the derivation (1). Then

$$E(G) = E(G') \cup \{(u, v_1), (u, v_2)\} \cup E'$$

Replacing $E(G)$ in equation (3) we have (See Figure 15 Top),

$$\begin{aligned} E(G^*) &= (E(G') \cup E' \cup \{(u, v_2)\}) \cup \{(w, v_1), (w, u), (w, v_2)\} \\ &= E(G') \cup E' \cup \{(u, v_2), (w, v_1), (w, u), (w, v_2)\} \end{aligned}$$

Replacing $E(G)$ in equation (5) we have (See Figure 15 Bottom),

$$\begin{aligned} E(G^{*'}) &= (E(G') \cup E' \cup \{(u, v_1), (u, v2)\}) \cup \{(w, v_2), (w, u)\} \\ &= E(G') \cup E' \cup \{(u, v_1), (u, v_2), (w, v_2), (w, u)\} \end{aligned}$$

A proper relabeling of vertices u and w shows that $E(G^*) = E(G^{*'})$. This along with the fact that $V(G^*) = V(G^{*'})$ lead to $G^* = G^{*'}$. Thus graph G^* belongs to $\mathbf{H}_1$ because derivation (4) is in H1S. □

In the proof of the main theorem we shall make use of the following two lemmas.

Lemma A1. Let $G = (V, E)$ be a tree-decomposable Laman graph for which G_1, G_2, G_3 is a decomposition induced by the triple $\{v_1, v_2, x\}$ and such that $G_i = (\{v_1, v_2\}, \{(v_1, v_2)\})$ for some $1 \leq i \leq 3$. Then the graph G^* created by the reduction $G \rightarrow_2 G^*$ involving the edge $(v_1, v_2) \in E(G)$ and vertex $v_3 \in V(G)$ is tree-decomposable.

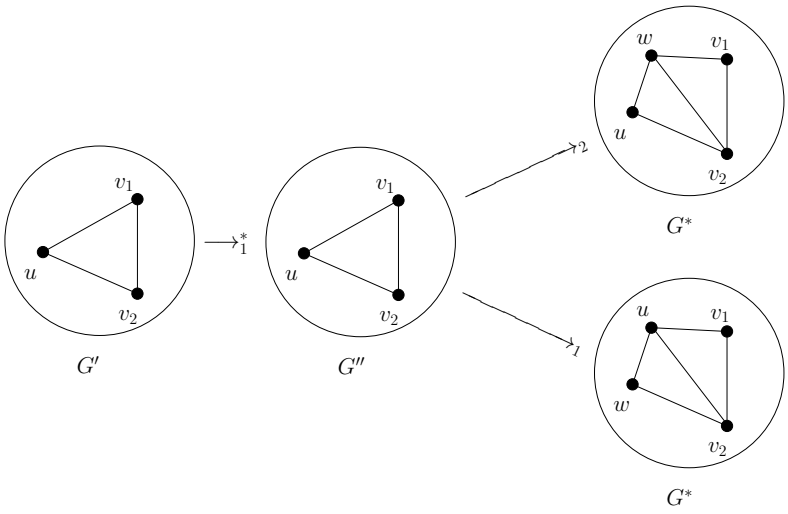


Figure 15: Derivation $\longrightarrow_1^*$ preserves edges. Top) Reduction $\xrightarrow{v_1,u,v_2,w}{}_2$ removes egde (v_1, u) and adds edges (w, v_1), (w, u) and (w, v_2). Bottom) Reduction $\xrightarrow{v_2,u,w}{}_1$ preserves edges plus adds edges (w, v_2) and (w, u).

Proof. Refer to Figure 16. Without loss of generality, assume that the given graph G is decomposed into G_1, G_2, G_3 with $G_1 = (\{v_1, v_2\}, \{(v_1, v_2)\})$, $v_3, v_1 \in V(G_3)$ and $v_2 \in V(G_2)$.

The graph G^* created by the reduction $G \rightarrow_{2,\{v_1,v_2,v_3,u\}} G^*$ is such that $V(G^*) = V(G) \cup \{u\}$ and $E(G^*) = (E(G) - \{(v_1, v_2)\}) \cup \{(u, v_1), (u, v_2), (u, v_3)\}$. Then G^* can be decomposed into three subgraphs, $G_1^* = (\{u, v_2\}, \{(u, v_2)\})$, $G_2^* = G_2$ and, $G_3^* = (V(G_3) \cup \{u\}, E(G_3) \cup \{(u, v_1), (u, v_3)\})$. Subgraph G_1^* is a leaf node in a tree-decomposition. Subgraph G_2^* is clearly tree-decomposable. The triple $\{u, v_1, v_3\}$ decomposes the graph G_3^* into G_3, $(\{u, v_1\}, \{(u, v_1)\})$ and $(\{u, v_3\}, \{(u, v_3)\})$. The fact that, by hypothesis, G_3 is tree-decomposable completes the proof. □

Lemma A2. Let $G = (V, E)$ be a tree-decomposable Laman graph for which G_1, G_2, G_3 is a decomposition induced by the triple $\{v_1, v_2, x\}$ and such that edge $(v_1, v_2) \in E(G_i)$ for some $1 \leq i \leq 3$. If there is a vertex $w \in V(G_i)$ such that the triple $\{v_1, v_2, w\}$ decomposes G_i, then the graph G^* created by

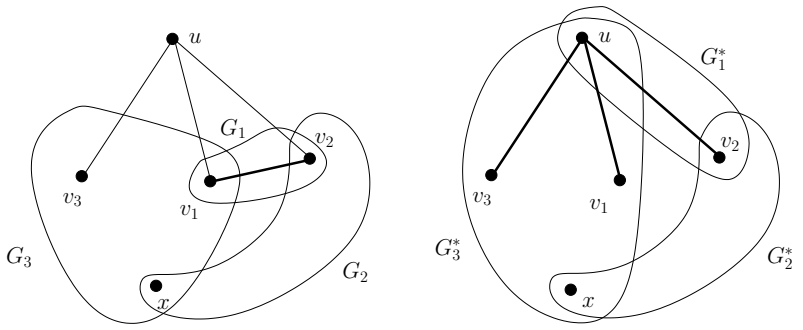


Figure 16: a) The given graph $G = G_1 \cup G_2 \cup G_3$ with $G_1 = (\{v_1, v_2\}, \{(v_1, v_2)\})$. b) Decomposition for the graph G^* created by reduction $G \xrightarrow{v_1,v_2,v_3,u}_2 G^*$.

the reduction $G \to_2 G^*$ involving edge $(v_1, v_2) \in E(G)$ and a third vertex $v_3 \in V(G)$ is tree-decomposable.

Proof. Let G_1, G_2, G_3 be the decomposition induced by the triple $\{v_1, v_2, x\} \in V(G)$ in the tree-decomposable graph G, as depicted in Figure 8a. Without loss of generality, assume that egde $(v_1, v_2) \in E(G_1)$ and that $V(G_1) \cap V(G_2) = \{v_2\}$, $V(G_2) \cap V(G_3) = \{x\}$ and $V(G_3) \cap V(G_1) = \{v_1\}$. By hypothesis G is tree-decomposable, hence G_1 is also tree-decomposable. In particular, assume that there is a vertex $w \in V(G_1)$ such that the triple $\{v_1, v_2, w\}$ decomposes G_1 into G_1', G_2', G_3' with, say, $G_1' = (\{v_1, v_2\}, \{(v_1, v_2)\})$. See Figure 8a. Now consider the reduction $G \to_{2,\{v_1,v_2,v_3,u\}} G^*$ and, in G^*, define the graphs

$$G_1^* = (V(G_2) \cup V(G_3) \cup \{u\}, E(G_2) \cup E(G_3) \cup \{(u, v_1), (u, v_2), (u, v_3)\})$$

$$G_2^* = G_2', \qquad G_3^* = G_3'$$

Clearly, G_1^*, G_2^*, G_3^* is a decomposition induced in G^* by the triple $\{v_1, v_2, w\}$. G_2^* and G_3^* are tree-decomposable because G_2' and G_3' are tree-decomposable. Consider the graph G_1^* as the graph created by an H2S step on edge (v_1, v_2) and vertex v_3 on the graph $G_1 \cup G_2$. See Figure 8b. Apply Lemma A1 to show that G_1^* is tree-decomposable. Thus G^* is tree-decomposable. □

Elements and concepts occuring in the statement of our main result are depicted in Figure 8.

Theorem 3.3. Let $G = (V, E)$ be a tree-decomposable minimally rigid graph with $\mathbf{T}$ the associated tree-decomposition. Let G^* be the grah created by the derivation $K_3 \rightarrow_\rho G' \rightarrow_\rho G \rightarrow_2 G^*$ where $\rightarrow_2$ is applied on edge $(v_1, v_2) \in E(G') \subset E(G)$ and vertex $v_3 \in V(G') \subset V(G)$. Let $\hat{G}_1, \hat{G}_2, \hat{G}_3$ be a decomposition of $LCA(v_1, v_2, v_3)$ with $(v_1, v_2) \in E(\hat{G}_i)$ for some $1 \leq i \leq 3$. Then G^* is tree-decomposable if one of the following two holds:

1. $\hat{G}_i = (\{v_1, v_2\}, \{(v_1, v_2)\})$.
2. There are two vertices $x, w \in V(G)$ such that $\{v_1, v_2, x\}$ is a triple for G and there is a vertex $w \in V(\hat{G}_i)$ such that $\{v_1, v_2, w\}$ is a triple for $\hat{G}_i$.

Proof. Consider a tree-decomposable graph $G = (V, E)$ with $\mathbf{T}$ as the associated tree-decomposition and denote the graph $LCA(v_1, v_2, v_3) \subset G$ as $\hat{G}$. $\hat{G}$ is tree-decomposable because so is G. Let $\hat{G}_1$, $\hat{G}_2$, $\hat{G}_3$ denote this decomposition.

First, without loss of generality, assume that $\hat{G}_1 = (\{v_1, v_2\}, \{(v_1, v_2)\})$ and that $v_3 \in V(\hat{G}_3)$, $\hat{G}_1 \cap \hat{G}_2 = \{v_2\}$, $\hat{G}_2 \cap \hat{G}_3 = \{x\}$ and $\hat{G}_3 \cap \hat{G}_1 = \{v_1\}$.

Reduction $\xrightarrow{v_1, v_2, v_3, u}_2$ in the derivation $K_3 \rightarrow^*_{\mathbf{H}} G' \rightarrow^*_{\mathbf{H}} G \rightarrow_2 G^*$ only affects edges and vertices in $\hat{G}$. Hence the reduction $\hat{G} \xrightarrow{v_1, v_2, v_3, u}_2 \hat{G}^*$ is well defined. Moreover, by Lemma A1, the graph $\hat{G}^*$ is tree-decomposable by the triple $\{u, v_2, x\}$.

Let $\hat{\mathbf{T}}^*$ be the tree-decomposition associated to the graph $\hat{G}^*$ and let $\mathbf{T}^*$ be the tree-decomposition resulting from replacing in $\mathbf{T}$ the tree rooted at node $LCA(v_1, v_2, v_3)$ with $\hat{\mathbf{T}}^*$, as illustrated in Figure 17. Clearly the resulting tree is a tree decomposition for G^*. Therefore G^* is tree-decomposable.

Now, assume that $v_3 \in V(\hat{G}_3)$, $\hat{G}_1 \cap \hat{G}_2 = \{v_2\}$, $\hat{G}_2 \cap \hat{G}_3 = \{x\}$, $\hat{G}_3 \cap \hat{G}_1 = \{v_1\}$, edge (v_1, v_2) is in $E(\hat{G}_1)$ and there is a vertex $w \in V(\hat{G}_1)$ such that $\{v_1, v_2, w\}$ is a triple for $\hat{G}_1$. Lemma A2 along with the rationale above show that G^* is tree-decomposable. □

Corollary 3.1. Let $G = (V, E)$ be a tree-decomposable graph and (v_1, v_2) be

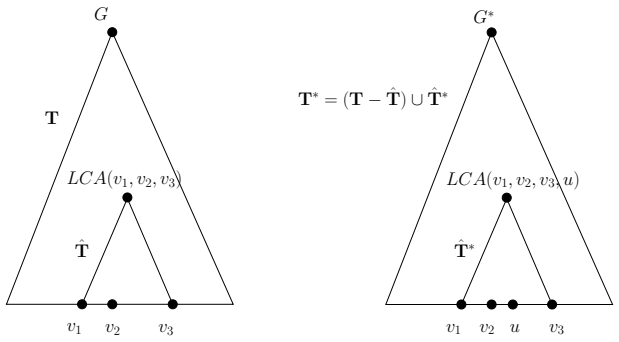


Figure 17: Illustration for Theorem 3.3. a) Tree-decomposition $\mathbf{T}$ for the given graph G and tree-decomposition $\hat{\mathbf{T}}$ for the subgraph rooted at $LCA(v_1, v_2, v_3)$. b) Tree-decomposition $\mathbf{T}^*$ resulting after replacing the tree-decomposition $\hat{\mathbf{T}}$ in $\mathbf{T}$ with $\hat{\mathbf{T}}^*$. $\mathbf{T}^*$ is a tree-decomposition for the graph G^*.

an arbitrary edge in $E(G)$. Then there is always a vertex $u \in V(G)$ different from v_1 and v_2 such that the derivation $G \rightarrow_2 G^*$ involving the edge (v_1, v_2) and vertex u creates a tree-decomposable graph G^*.

Proof. Let $G = (V, E)$ be the given graph and $\mathbf{T}$ the associated tree decomposition. Let $\hat{G}$ be the node in $\mathbf{T}$ such that it is decomposed into $\hat{G}_1$, $\hat{G}_2$ and $(\{v_1, v_2\}, \{(v_1, v_2)\})$. Apply Theorem 3.3 to the reduction $G \rightarrow_{2,v_1,v_2,u,w} G^*$ with $u \in \{V(\hat{G}_1) \cup V(\hat{G})_2\}$ and $u \notin \{v_1, v_2\}$. □

Acknowledgments

The material in this paper is based upon work supported by the Spanish *Ministerio de Economia y Competitividad* and by FEDER (EU) funds under Grant No. TIN2014-52211-C2-1-R. The authors would like to thank the anonymous reviewers for their valuable comments and suggestions to improve the quality of this paper.

References

[1] C. Hoffmann, R. Joan-Arinyo, A brief on constraint solving, Computer-Aided Design and Applications 2 (5) (2005) 655–663.

[2] C. Hoffmann, A. Lomonosov, M. Sitharam, Decompostion Plans for Geometric Constraint Problems, Part II: New Algorithms, Journal of Symbolic Computation 31 (2001) 409–427.

[3] C. Hoffmann, A. Lomonosov, M. Sitharam, Decompostion Plans for Geometric Constraint Systems, Part I: Performance Measurements for CAD, Journal of Symbolic Computation 31 (2001) 367–408.

[4] C. Jerman, G. Trombettoni, B. Neveu, P. Mathis, Decomposition of geometric constraint systems: A survey, International Journal of Computational Geometry and Applications 16 (5-6) (2006) 379–414.

[5] R. Joan-Arinyo, A. Soto, A correct rule-based geometric constraint solver, Computer & Graphics 21 (5) (1997) 599–609.

[6] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, J. Vilaplana, On the domain of constructive geometric constraint solving techniques, in: R. Duricovic, S. Czanner (Eds.), Spring Conference on Computer Graphics, IEEE Computer Society, Los Alamitos, CA, Budmerice, Slovakia, 2001, pp. 49–54.

[7] H. Crapo, W. Whiteley, Spacess of stresses and projected polyhedra I: The basic pattern, Structural Topology (20) (1993) 55–77.

[8] J. Graver, B. Servatius, H. Servatius, Combinatorial Rigidity, Vol. 2 of Graduate Studies in Mathematics, American Mathematical Society, 1993.

[9] T.-S. Tay, W. Whiteley, Generating isostatic frameworks, Structural Topology (11) (1985) 21–69.

[10] W. Whiteley, Some matroids from discrete applied geometry, in: B. S. J. Bonnin, J. Oxley (Ed.), Contemporary Mathematics, Vol. 197, American Mathematical Society, 1996, Ch. Matroid Theory, pp. 171–311.

[11] I. Fudos, C. Hoffmann, A graph-constructive approach to solving systems of geometric constraints, ACM Transactions on Graphics 16 (2) (1997) 179–216.

[12] J. Owen, S. Power, The nonsolvability by radicals of generic 3-connected planar graphs, Transactions of AMS 5 (359) (2006) 2269–2303.

[13] L. Henneberg, Die Graphische Statik der Starren Systeme, Leipzig, 1911, Johnson Reprint 1968.

[14] B. Servatius, H. Servatius, Rigidity, global rigidity, and graph decomposition, European Journal of Combinatorics 31 (4) (2010) 1121–1135.

[15] J. Klop, Term rewriting systems, in: S. Abramsky, D. Gabbay, T. Maibaum (Eds.), Background: Computational Structures, Vol. 2 of Handbook of Logic in Computer Science, Clarendon Press, 1992, pp. 1–117.

[16] R. Haas, D. Orden, G. Rote, F. Santos, B. Servatius, H. Servatius, D. Souvaine, I. Streinu, W. Whiteley, Planar minimally rigid graphs and pseudo-triangulations, Computational Geometry 31 (2005) 31–61, doi:10.1016/j.comgeo.2004.07.003.

[17] W. Whiteley, Rigidity and scene analysis, in: J. Goodman, J. O'Rourke (Eds.), Handbook for discrete and computational geometry, CRC Press LLC, 1998, pp. 893–916.

[18] C. Borcea, I. Streinu, The number of embeddings of minimally rigid graphs, Discrete & Computational Geometry 31 (2004) 287–303.

[19] A. Frank, L. Szegö, An extension of a theorem of Henneberg and Laman, Tech. Rep. TR-2001-05, Egérvary Research Group on Combinatorial Optimization, www.cs.elte.hu/egres (February 2001).

[20] M. R. Hidalgo, Geometric constraint solving in a dynamic geometry framework, Ph.D. thesis, Universitat Politècnica de Catalunya (2013).

[21] A. Aho, J. Hopcroft, L. Ullmann, On finding lowest common ancestors in trees, in: STOC'73 Proc. 5th annual ACM Symposium on Theory of Computing, ACM New York, 1973, pp. 253–265, doi:10.1145/800125.804056.

[22] J. Savage, Models Of Computation: Exploring the Power of Computing, 2008, creative Commons, http://cs.brown.edu/ jes/book/home.html.

[23] M. Bender, M. Farach-Colton, The LCA probelm revisited, in: 12th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA'01), 2001, pp. 845–853.

[24] R. Joan-Arinyo, A. Soto-Riera, S. Vila-Marta, J. Vilaplana, Transforming an underconstrained geometric constraint problem into a wellconstrained one, in: G. Elber, V.Shapiro (Eds.), Eight Symposium on Solid Modeling and Applications, ACM Press, Seattle (WA) USA, 2003, pp. 33–44.