Enumerating Planar Minimally Rigid Graphs

David Avis[1], Naoki Katoh[2] ⋆, Makoto Ohsaki[2],
Ileana Streinu[3] ⋆⋆, and Shin-ichi Tanigawa[2]

[1] School of Computer Science, McGill University, Canada. avis@cs.macgill.ca
[2] Department of Architecture and Architectural Engineering, Kyoto University, Kyoto 615-8450 Japan, ohsaki,naoki,is.tanigawa@archi.kyoto-u.ac.jp.
[3] Dept. of Comp. Science, Smith College,Northampton, MA 01063, USA, streinu@cs.smith.edu.

Abstract. We present an algorithm for enumerating without repetitions all the planar (non-crossing) minimally rigid (Laman) graphs embedded on a given generic set of n points. Our algorithm is based on the Reverse search paradigm of Avis and Fukuda. It generates each output graph in $O(n^4)$ time and $O(n)$ space, or, with a slightly different implementation, in $O(n^3)$ time and $O(n^2)$ space.
In particular, we obtain that the set of all planar Laman graphs on a given point set is connected by flips which remove an edge and then restore the Laman property with the addition of a non-crossing edge.

1 Introduction

Let $G = (V, E)$ be a graph with $n = |V|$ vertices and $m = |E|$ edges. G is a *Laman graph* if $m = 2n - 3$ and every subset of $n' \leq n$ vertices spans at most $2n' - 3$ edges. An embedding $G(P)$ of the graph G on a set of points $P = \{p_1, \cdots, p_n\} \subset R^2$ is a mapping of the vertices V to points in the Euclidian plane $i \mapsto p_i \in P$. The edges $ij \in E$ are mapped to straight line segments p_ip_j. An embedding $G(P)$ is *planar* or *non-crossing* if no pair of segments p_ip_j and p_kp_l corresponding to non-adjacent edges $ij, kl \in E, i, j \notin \{k, l\}$ have a point in common.

Laman graphs embedded on *generic* point sets enjoy the special property of being *minimally rigid* [14, 10], when viewed as bar-and-joint frameworks with fixed edge-lengths, which motivates the tremendous interest in their properties.

In this paper we give an algorithm for enumerating all the *planar Laman graphs* $G(P)$ embedded on a *given* **generic** *point set* P.

Novelty. To the best of our knowledge, this is the first algorithm proposed for enumerating (without repetitions, in polynomial time and without using additional space) all the planar Laman graphs. We achieve $O(n^4)$ time per graph in $O(n)$ space (or, with a slightly different implementation, in $O(n^3)$ time and $O(n^2)$ space) by using Reverse Search.

Historical Perspective. The Reverse Search enumeration technique of Avis and Fukuda [2, 3] has been successfully applied to a variety of combinatorial and geometric enumeration problems. The necessary ingredients to use the method are an implicitly described connected graph on the objects to be generated, and an implicitly defined spanning tree in this graph. In this paper we supply these ingredients for the problem of generating Laman graphs.

⋆ Supported by JSPS Grant-in-Aid for Scientific Research on priority areas of New Horizons in Computing.
⋆⋆ Supported by NSF grant CCF-0430990 and NSF-DARPA CARGO CCR-0310661.

Relevant to the historical context of our work are the results of Bereg [6, 7] using Reverse Search combined with data-specific lexicographic orderings to enumerate triangulations and pointed pseudo-triangulations of a given point set. We notice in passing that there exist several other algorithms for enumerating (pseudo-)triangulations [9, 8, 1], but they are based on different techniques.

Also relevant is the pebble game algorithm of Jacobs and Hendrickson [11] for 2-dimensional rigidity, see also [5]. Our complexity analysis relies on the recent results, due to Lee, Streinu and Theran [15, 16], regarding the detailed data-structure complexity of finding and maintaining rigid components during the pebble game algorithm. Indeed, the time-space trade-off of our algorithm is inherited from [16]: without the data structure details, the overall complexity would be $O(n^3)$ steps and linear space for the storage of the current output graph.

Search and flips in pointed pseudo-triangulations To better put our problem (and solution) in context and relate it to previous work, we briefly discuss now the difference between generating arbitrary planar Laman graphs, as opposed to just pointed pseudo-triangulations.

A *pointed pseudo-triangulation* is a special case of a planar Laman graph on a given point set [20], where every vertex in the embedding is incident to an angle larger than π.

Pseudo-triangulations are connected via simple *flips*, in which the removal of any non-convex-hull edge leads to the choice of a *unique* other edge that can replace it, in order to restore the pointed pseudo-triangulation property.

The flip graph of all pointed pseudo-triangulations is a connected subgraph of the graph of all the Laman graphs. In fact, it is the one-skeleton of a polytope [18], and the reverse search technique can be directly applied to it. Bereg's efficient algorithm makes use of specific properties of pointed pseudo-triangulations which do not extend to arbitrary planar Laman graphs. In particular, remove-add flips are *not unique*, relative to the removed edge, in the case of planar Laman graphs. Moreover, it is not even known *a priori* whether the set of all the planar Laman graphs of p is connected via these flips.

Motivating application. Although applications are not the main focus of our paper, we describe now briefly how this problem came to our attention via the work of the third author. Graph theoretical approaches are widely used in *structural mechanics* [12], where the edges and vertices in the graph represent the bars and rotation-free joints of a structure called a *truss*. It is well-known [4] that the stiffest truss under static loads is statically determinate, a concept directly related to the previously defined Laman-graph property. Ohsaki and Nishiwaki [17] presented a method for generating multi-stable flexible bar-joint system, and found that the optimal structure is statically determinate. Since the optimal topology is found by removing the unnecessary nodes and members from the highly connected initial structure, the computational cost of this procedure can be reduced if the candidate set of Laman graphs are first enumerated. In a related direction, Kawamoto et al. [13] presented a method based on the enumeration of *planar* graphs to find an optimal mechanism. Although this is a problem of high practical interest in structural mechanics, none of these papers give a general approach for the systematic enumeration of rigidity-constrained structures - which is the topic of our paper.

Organization. We define flips in planar Laman graphs and review the Reverse Search technique in Section 2. The search-tree structure underlying the algorithm is defined in Section 3. In Section 4, we prove the correctness of the search-tree definition, and thus of the Reverse Search-based enumeration algorithm. The data structure specific details of the implementation and the complexity analysis are done in Appendix A. In the Conclusions, we mention two open problems which may help reduce the complexity of our algorithm.

2 Preliminaries

Let $G(P)$ be a planar embedded Laman graph on the generic point set P. The planar subdivision induced by the embedding will be denoted as $G = (V, E, F)$, with V, E and F, respectively, being the vertices, edges and faces.

Since all throughout the paper we consider only generic point sets, purely combinatorial graph-theoretic properties correspond to intuitive, rigidity-theoretic properties such as *rigid*, *flexible*, *degree of freedom*, etc. We will use this physically inspired terminology, as well as *body*, *joint* and *mechanism*, instead of a more technical graph-theoretic language.

A *mechanism* is a flexible framework obtained by removing one or more edges from a generic Laman framework. Its *number of degrees of freedom* or *dof*s, is the number of removed edges. We will encounter mostly *one-degree-of-freedom (1dof) mechanisms*, which arise from a Laman graph by the removal of one edge.

In particular, a mechanism with k dofs has exactly $2n - 3 - k$ edges, and each subset of n' vertices spans at most $2n' - 3$ edges. A subset of some n' vertices spanning *exactly* $2n' - 3$ edges is called a *rigid block*. A *maximal block* is called a *rigid component* or a *body*. The edge set of a mechanism is partitioned into bodies, with some bodies possibly containing just one edge each. Two bodies are either disjoint (vertex-wise), or may share one vertex. A *joint* is a vertex shared by at least two bodies. These properties, as well as efficient algorithms for computing rigid components in Laman mechanisms, can be found in [11, 5, 15, 16].

The Laman graphs on n vertices form the set of *bases* of the *generic rigidity matroid* in dimension 2, see [10]. The ground set of the matroid is the set of edges E. The bases have all the same size $2n - 3$. Bases may be related via the *basis exchange* operation, which we will call a *flip* between two Laman graphs. Two Laman graphs G_1 and G_2 are connected by a flip if their edge sets agree on $2n - 4$ positions. The flip is given by the pair of edges (e_1, e_2) not common to the two bases, $e_1 \in G_1 \setminus G_2$, $e_2 \in G_2 \setminus G_1$. Using flips, we can define a graph whose nodes are *all* the Laman graphs on n vertices, and whose edges correspond to flips.

Remove-Add flips. When we remove an edge from a Laman graph G, we obtain a *1dof mechanism* $G' \subset G$ whose edge set can be partitioned into *rigid components*. If we add now to G' any edge of $K_n \setminus G$ whose endpoints do not belong to the same rigid component, we obtain a new Laman graph. Therefore the set of all *remove-add flips* from a Laman graph G can be systematically generated as follows: consider the set of all the edges $e_1 \in G$. For each one, compute the rigid components of the corresponding 1dof mechanism $G_{-e_1} = G \setminus \{e_1\}$. For each edge $e_2 \in K_n \setminus G_{-e_1}$ whose endpoints do not belong to the same rigid component, its addition leads to a new Laman graph $G_{-e_1,+e_2}$.

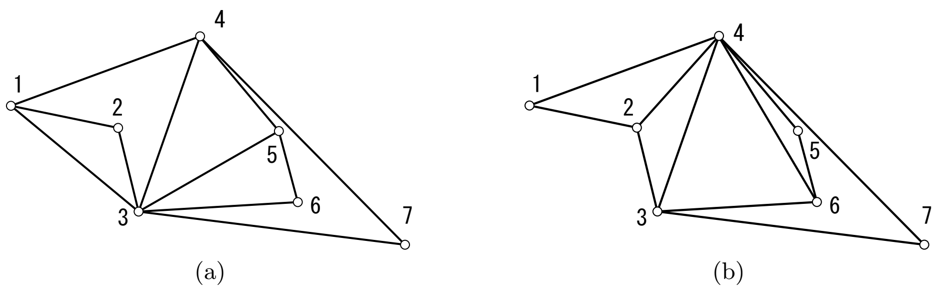

Fig. 1. (a)The root planar Laman graph on a set of 7 points. (b)A non-root Laman framework.

It is well-known that the graph whose nodes are the bases of a matroid connected via flips, is connected. But a priori, the subset of *planar Laman graphs* may not necessarily be. We will prove this later in Section 4.

Reverse Search. This technique is a memory efficient method for visiting all the nodes of a connected graph that can be defined implicitly by an adjacency oracle. It can be used whenever a spanning tree of the graph can be defined implicitly by a *parent* function. This function is defined for each vertex of the graph except a prespecified *root*. Iterating the parent function leads to a path to the root from any other vertex in the graph. The set of such paths defines a spanning tree, known at the *search tree*.

3 The Search Tree

In this section we define the main structure required by Reverse Search, a *search tree* on the set of all the planar Laman graphs of a given point set. We choose a certain Laman framework to be the root. Then we define a *parent* for every non-root Laman graph. To show that the parent function defines a search tree we associate an *index* to every Laman graph, such that the parent function always returns a Laman graph with smaller index. This gives a forest structure. To prove that it is connected and thus a search tree, we will show in Section 4 that the parent of every non-root node indeed exists.

Root. We choose the root of the search tree to be a *greedy pseudo-triangulation* corresponding to a fixed direction. The simplest way to define it is relative to the horizontal direction (x-axis).

We first sort the points by x-coordinate and label them as $\{1, 2, \cdots, n\}$ in this order. Then we construct a *Henneberg I* pointed pseudo-triangulation as follows. Start with the edge 12 and continue for $n-2$ steps. At each step, the next vertex (in x-sorted order) is added (vertex $i+2$ at step i, $i = 1, \cdots, n-2$), together with the two *tangents* from point p_i to the (convex hull of) the framework constructed so far. Fig. 1(a) illustrates the root, and Fig. 1(b) gives an example of a non-root framework.

Index. Given a non-root planar Laman framework G, we define its *index* as a pair *index*$(G) = (c, d)$, where $c = c(G) \in \{2, \cdots, n\}$ and $d = d(G) \in \{1, \cdots, n\}$ are, respectively, the label of the *critical vertex* of G and the *critical degree* of the critical vertex, defined below.

The *critical vertex* (with respect to the root) is the largest label of a vertex whose incident edges differ from the corresponding set of incident edges in the root framework. For instance, the critical vertex for the non-root graph in Fig. 1 is 6. Since all the vertices with labels at least $c+1$ have the same edges as their counterparts in the root graph, it follows easily that the subgraph of G spanned by the vertices $\{1, \cdots, c\}$ is Laman. The *critical edge set* is the set of mismatched edges incident to the critical vertex. An edge is *mismatched* if it doesn't exist in the *root.* The *critical degree* is the number of mismatched edges incident to the critical vertex. Notice that the critical degree is always at least 1, or else the vertex is not critical. For example, the non-root graph in Fig. 1(b) has index $(6, 1)$: the critical vertex is 6, and it is incident to three edges, two of which exist in the root and one doesn't.

We use the index as a measure of how far a node (Laman framework) is from the root, whose index is, by definition, $(1, 0)$.

Parent Rule. The *parent* of a node (Laman framework) is defined in terms of its critical vertex via a certain *Remove-add flip.* The removed edge which does not exist in the *root* will be incident to the critical vertex, and the added edge will be chosen so that it will decrease the index of the critical vertex.

In general there will be several choices of such flips. To uniquely define the parent, we will use a lexicographic ordering of these flips and choose the smallest one. The efficiency of the *Parent* function depends on the lexicographic ordering. We will discuss it in Appendix.

The correctness of the *Parent* definition follows from our Main Theorems:

Theorem 1. *Every non-root planar Laman graph has a parent whose index is smaller than that of the current node.*

The proof of the above theorem relies on properties of planar Laman graphs described in the next section. Based on Theorem 1, we will propose a reverse search algorithm by following the standard techniques devloped by [2, 3]. It is well known [2, 3] that the reverse search can be developed if we can define two basic operations, i.e., *parent* function and *adjacency* function. Given a non-root planar Laman graph G, the *parent* function returns another planar Laman graph which is parent of G. Given a planar Laman graph G, the adjacency function returns a planar but possibly non-Laman graph G' such that G' is obtained by a single remove-add operation.

The detailed description of the parent and adjacency function will be given in the appendix and the complexity analysis of these functions will also be proved in the appendix.

Theorem 2. *The set of all planar Laman graphs on a given point set can be reported in $O(n^3)$ time per planar Laman graph using $O(n^2)$ space and in $O(n^4)$ time using $O(n)$ space.*

4 Remove-Add flips in planar minimally rigid graphs

This section contains the proof of Theorem 1: every non-root planar Laman graph has a parent. The proof follows from a sequence of Lemmas.

In a graph G, a subgraph induced by a vertex set $V' \subset V$ is a *cut* if its removal disconnects the graph.

Lemma 1. *A rigid component cannot be a cut in a 1dof mechanism.*

Proof. Let $G = (V, E)$ be a 1dof mechanism, and $G' = (V', E')$ be one if its rigid components. Assume, for the sake of a contradiction, that G' is a cut. Then the complement of G', the subgraph $G/(V \setminus V')$ induced on $V \setminus V'$, can be partitioned into two disjoint subsets V_1 and V_2 with no edges between them. Let with $n = |V|$, $n' = |V'|$, $n_i = |V_i|, i = 1, 2$, with $n = n' + n_1 + n_2$. Since G' is a body, $|E'| = 2n' - 3$. Consider the two subgraphs induced on $V' \cup V_i, i = 1, 2$. It is not possible that both of them have at least 1dof, because otherwise the whole graph G would have more dofs, as it would span less than $2(n' + n_1) - 4 + 2(n' + n_2) - 4 - (2n' - 3) = 2(n' + n1 + n2) - 5$ edges. Therefore, at least one has zero dofs, contradicting the maximality of G' as a component. □

For planar mechanisms, we can say more. The following statements are better understood if we forget for a moment the geometry of the embedding of the planar Laman graph *in the Euclidean plane*, and think of G as a spherical (topological) embedding.

Proposition 1. *In a topologically embedded planar Laman graph, each face cycle is simple and subdivides the sphere into two disk-like regions.*

Proof. This is a direct consequence of the fact that a Laman graph is always 2-connected. □

Lemma 1 shows that a body cannot disconnect the graph, therefore for each cycle in a body, only one (but not both) of the two (spherical) disk-like regions induced by it may contain vertices not in the body. Since a face is empty, if its boundary cycle is part of a body, we will say that the face *belongs* to the body. Notice that a body may thus contain the outer face. We obtain:

Lemma 2. *In a planar 1dof Laman mechanism, the union of all the faces belonging to the same body form a topological disk.*

Proof. Assume that the subset $V' \subset V$ spans a component $G' = (V', E')$ of G. By Lemma 1, the removal of G' cannot disconnect G, therefore there exists a cycle C in G containing the entire G' in one of its two induced disk-like spherical regions. Since G' is planar and at least 2-connected, and it inherits a face structure from G, it follows that the cycle C falls in one of these induced faces of G'. In fact, the whole complement of G' in G must fall inside this face of G', since otherwise G' would be a cut of G. Therefore, all the other faces of G' are empty of other vertices of G and thus the union of faces has no holes (it is topologically a disk). □

We stated this topological property on the sphere, not in the plane. If the removed edge does not belong to the outer face of a planar embedding of G, then the complement of G' may fall inside an interior face and the union of the (embedded) interior faces of G' may look like an annulus. See Fig. 2. Therefore notice that, for the uniformity of the argument, when the outer face of a planar Laman mechanism belongs to a rigid component, we defined the rigid body as the union of all the *empty* faces (thus including the outer face).

Lemma 2 implies that the faces of the planar framework can be divided into two categories: *rigid*, if they belong to a rigid body and *flexible*, otherwise. In fact, we will

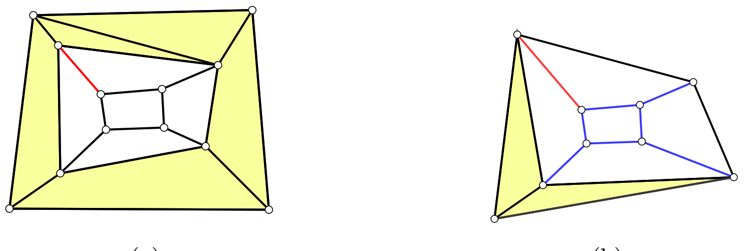
(a) (b)

Fig. 2. Removing the red edge results in a 1dof mechanism. (a) The removed edge is interior. The highlighted body is not a topological disk (the other bodies are just bars). In (b), the outer body is made out of all the black bars, and each of the blue bars is a separate body.

not be concerned with separate rigid faces, only with the rigid bodies (which are disk-like unions of their faces) and the flexible faces. A body *bounds* a flexible face, if they share at least one edge. Since Laman graphs are 2-connected, all the rigid bodies are made out of simple faces, and therefore the boundary of a body is also a simple cycle. For flexible faces, this may not be true.

The next Lemmas give some useful properties of flexible faces. We split the analysis into two cases: when the mechanism is 2-connected, and when it is only 1-connected. Indeed, it is well-known that a Laman graph is at least 2-connected, so the removal of one edge produces a graph that is at lest 1-connected.

Lemma 3. *All the flexible faces of a 2-connected 1dof planar Laman mechanism have at least four bounding bodies.*

Proof. Two-connectivity implies that each face is a simple cycle. This cycle either lies inside a body or is bounded by some bodies. Two bodies cannot have more than one vertex in common: if they have two, their union is a larger Laman subgraph, which contradicts the maximality of the body. A face cannot be bounded by two bodies only, because then those bodies would have two vertices in common and their union would then be over-constrained. A face also cannot be bounded by only three bodies, because in that case the union of the three bodies would be a larger rigid body, and the face would be a face of that larger body, not a flexible face. So each flexible face is bounded by at least four bodies. □

Lemma 4. *If the 1dof planar Laman mechanism G is not 2-connected, then all but one face are as in Lemma 3. The unique exception is a* special *flexible face incident to exactly two bodies, and whose bounding cycle is not simple.*

Proof. Let G be a 2-connected but not 3-connected Laman graph, and remove an edge to obtain a mechanism. Since a Laman graph is 2-connected, a 1dof mechanism is at least 1-connected. It is not possible that the endpoints of the removed edge belong to the same body in the mechanism, because otherwise Laman's property would have been violated on the union of the body and the edge. If the mechanism is not 2-connected, then exactly one endpoint of the removed edge belongs to a pair of vertices that would have disconnected G. □

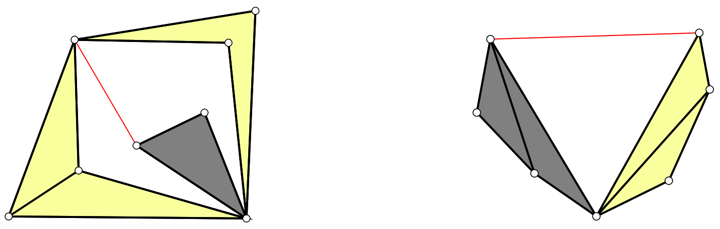
(a) (b)

Fig. 3. The special type of flexible face which is incident to two bodies only. (a) Unbounded and (b) Bounded face. Notice that the original graph is 2- but not 3-connected, and the removed edge (thin red) is incident to the cut pair.

The situations described in Lemma 4 are illustrated in Fig. 3.

From these properties of planar Laman graphs and 1dof mechanisms, we now sketch the proof for our main theorem. In the proof we work with a coarser planar structure, possibly ignoring some vertices, edges and faces in a 1dof mechanism. This coarser structure is given by the collection of bodies, joints connecting them and flexible faces.

Proof. **of Theorem 1**. Let p_c be the critical vertex of a given planar Laman graph G. Then the subgraph induced by $\{p_1, p_2, \ldots, p_c\}$ is still a planar Laman graph (denoted by G'). From the definition of the critical vertex, p_c is always on the outer face in G' and there is at least one non-root edge incident to p_c. Then we delete the non-root edge p_cp_i from G' in the *Parent* function. It suffices to prove that there always exists an edge in G' satisfying: (i) after inserting the edge to $G' \setminus \{p_cp_i\}$, we obtain a planar Laman graph, and (ii) the edge is disjoint from p_c or a root edge incident with p_c. Adding such an edge creates another planar Laman graph for $\{p_1, \ldots, p_c\}$ and the number of mismatched edges incident with p_c decreases by one. Thus adding $p_{c+1}, p_{c+2}, \ldots, p_n$ together with two hull edges produces a planar Laman graph with a smaller index.

The proof has two cases depending on whether p_cp_i is on the outer face or not.

Case 1: p_cp_i is not on the outer boundary. In this case, 1dof mechanism $G' \setminus \{p_cp_i\}$ satisfies the properties proven in the previous lemmas: all the components are topological disks with the possible exception of the component containing, spherically, the outer face, which may be an annulus in the Euclidean embedding. Then, the case 1 also has two cases: the mechanism has a special face, an annulus-like body or not.

(1-a) The 1dof mechanism $G' \setminus \{p_cp_i\}$ does not have an annulus-like body. In this case all flexible faces are incident to at least four bodies and four distinct joints (see Fig.4(a)). We call a pair of joints *incident* if they belong to the same body. Consider the geodesic paths between any pair of non-incident joints, and call them *geodesic diagonals* of the faces. There are at least two such geodesic diagonals in each face, since each face has at least four joints. A geodesic diagonal may lie entirely on the boundaries of the bodies incident to the face, in which case it must also go through the joints between those bodies, or may have at least one segment lying inside the face. Moreover, such a segment may go between vertices belonging to the same body, or may go between two different bodies. In this last case, we say that the segment is *free*. We claim that there exists at least one free segment, on at least one geodesic diagonal path in all the flexible faces of the mechanism (see Fig.4(b)).

Suppose that there is more than one flexible face in $G' \setminus \{p_c p_i\}$. Since each of them has at least one free segment, there always exist at least two free segements one of which is not incident to p_c. Then, we consider the case where there is only one flexible face in $G' \setminus \{p_c p_i\}$. Note that all the joints are on the outer face and the flexible face. If all four joints are the convex hull vertices, then the interior face contains two distinct *free* segments, and one of which is not incident to p_c. Otherwise, suppose the interior face contains only one free segment $p_c p_i$. Then, there exists one free segment in the outer face which is certainly not incident to p_c.

(1-b) The 1dof mechanism $G' \setminus \{p_c p_i\}$ has an annulus-like body. Then its inner boundary is a simple polygon P. Hence P must consist of at least three *vertices* including p_c. Let p_a and p_b be two of these vertices different from p_c. Since the vertex p_i does not belong to this annulus-like body, there is another disk-like body lying inside P to which p_i belongs.

When $G' \setminus \{p_c p_i\}$ is 1-connected. Since two bodies cannot have more than one vertex (joint) in common, at least one vertex of p_a and p_b does not belong to the body to which p_i belongs, (see Fig.5(a)). Let p_a be such a vertex. (The vertex p_b may be a joint.) Consider the two paths inside the flexible face between pairs of vertices (p_c, p_i) and (p_a, p_i) respectively: they must contain at least two distinct line segments lying entirely inside the face (diagonals). These two paths connect vertices lying on two distinct bodies of the mechanism (one is the annulus-like body), so each of them can be added as a bar to create a Laman graph. Furthermore, the diagonal between p_i and p_a is not incident to p_c.

When $G' \setminus \{p_c p_i\}$ is 2-connected. Then all the flexible faces have at least four bounding bodies from Lemma 3. Therefore, there are at least two flexible faces inside P (see Fig.5(b)). Then, as we remarked in the case (1-a), each of them has at least one free segment, and there exist at least two free segments one of which is not incident to p_c.

Case2: $p_c p_i$ is on the outer boundary. At least one of the upper and lower hull edges incident to p_c is missing in G' because $p_c p_i$ is not on the convex hull of $\{p_1, \ldots, p_c\}$. Without loss of generality, we assume the upper hull edge is missing, and let p_c^{up} be the endpoint of upper hull edge other than p_c.

Suppose that p_c and p_c^{up} belong to different bodies. Adding $p_c p_c^{up}$ to $G' \setminus \{p_c p_i\}$ creates another planar Laman graph which satisfies the desired property that the added edge is a root edge. Then, we consider the case where p_c and p_c^{up} belong to the same body. We add an auxiliary vertex $\bar{p}$ incident to p_c and p_c^{up} such that $G' \cup \{\bar{p}p_c, \bar{p}p_c^{up}\}$ is a planar Laman graph. Such graph can always be constructed by a *Henneberg I* move. Let $\bar{G} = G' \cup \{\bar{p}p_c, \bar{p}p_c^{up}\}$. Consider the 1dof mechanism $\bar{G} \setminus \{p_c p_i\}$. It has an annulus-like body and its inner boundary is a simple polygon P. This must consist of at least four vertices including $\bar{p}, p_c$ and p_c^{up}. In this case, there is at least one *free* segment not incident to p_c and $\bar{p}$, in the same way as in the case (1-b). Deleting $\bar{p}$ and adding this free edge produces a planar Laman graph for G'. Note that the added edge is not incident with p_c.

This completes the proof of Theorem 1: the Search Tree for Reverse Search is well defined. □

Remark. Notice that the *Flip* operation is not always valid when the removed edge is on the convex hull. Fig. 3(b) illustrates a Laman graph where the removal of the top thin

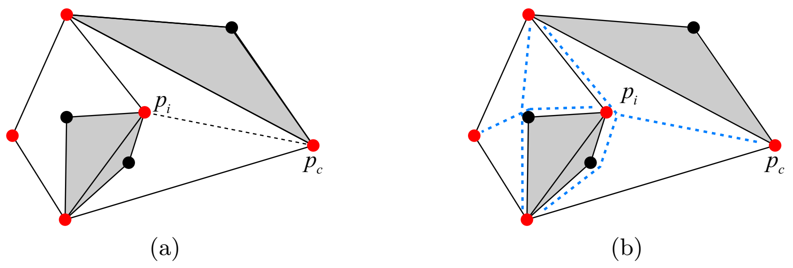

Fig. 4. (a)Removing $p_c p_i$ results in a 1dof mechanism. The shaded faces represent rigid bodies and the other faces are flexible faces. The red vertices represent joints. (b)The blue dotted lines show the geodesic diagonals (geodesic paths between any pair of non-incident joints). There is at least one free segment in each flexible face.

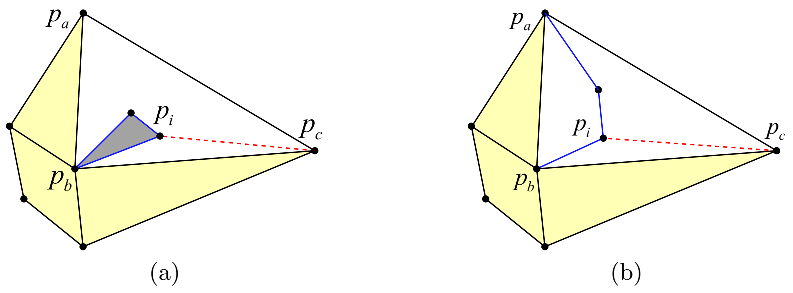

Fig. 5. Removing $p_c p_i$ results in a 1dof mechanism. The vertices p_c, p_a and p_b belong to annulus-like body. (The annulus-like body is made out of all black bars.) The resulting mechanism is (a)1-connected and (b)2-connected.

convex hull edge leads to a mechanism with no other option for an edge to be placed back. This is the reason why we do not remove them arbitrarily, and why we must take care of such special cases separately in the detailed case analysis and in the implementation.

5 Conclusions

We presented an algorithm for enumerating all the planar minimally rigid graphs embedded on a point set. While our main focus in this paper is to show that this can be done in polynomial time, we expect that the complexity of our algorithm may be improved via a combination of more sophisticated data structures and further insights into the specific properties of *planar* Laman graphs (as opposed to just Laman graphs). In particular, this depends on two *open problems*: how to beat $O(n^2)$ space for $O(n^2)$-time rigid components, and how to take advantage of *planarity* for computing the rigid components in better than $O(n^2)$ time (which can be done in $O(n)$ time for pseudo-triangulations).

References

1. O. Aichholzer, G. Rote, B. Speckmann, and I. Streinu. The zig-zag path of a pseudo-triangulation. In *Proc. 8th International Workshop on Algorithms and Data Structures (WADS)*, Lecture Notes in Computer Science 2748, pages 377–388, Ottawa, Canada, 2003. Springer Verlag.
2. D. Avis and K. Fukuda. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. *Discrete and Computational Geometry*, 8:295–313, 1992.
3. D. Avis and K. Fukuda. Reverse search for enumeration. *Discrete Applied Mathematics*, 65(1-3):21–46, March 1996.
4. M. P. Bendsøe and O. Sigmund. *Topology Optimization: Theory, Methods and Applications*. Springer, 2003.
5. A. Berg and T. Jordán. Algorithms for graph rigidity and scene analysis. In G. D. Battista and U. Zwick, editors, *Proc. 11th Annual European Symposium on Algorithms (ESA)*, volume 2832 of *Lecture Notes in Computer Science*, pages 78–89. Springer, 2003.
6. S. Bespamyatnikh. An efficient algorithm for enumeration of triangulations. *Comput. Geom. Theory Appl.*, 23(3):271–279, 2002.
7. S. Bespamyatnikh. Enumerating pseudo-triangulations in the plane. *Comput. Geom. Theory Appl.*, 30(3):207–222, 2005.
8. H. Brönnimann, L. Kettner, M. Pocchiola, and J. Snoeyink. Enumerating and counting pseudo-triangulations with the greedy flip algorithm. In *Proc. ALENEX*, Vancouver, Canada, 2005.
9. A. Dumitrescu, B. Gärtner, S. Pedroni, and E. Welzl. Enumerating triangulation paths. *Computational Geometry: Theory and Applications*, 20(1-2):3–12, 2001. A preliminary version in Proceedings of the Twelfth Canadian Conference on Computational Geometry, 2000 (CCCG'00), 233-238.
10. J. Graver, B. Servatius, and H. Servatius. *Combinatorial Rigidity.* Graduate Studies in Mathematics vol. 2. American Mathematical Society, 1993.
11. 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.
12. A. Kaveh. *Structural Mechanics: Graph and Matrix Methods.* Research Studies Press, Somerset, UK,, 3rd edition, 2004.
13. A. Kawamoto, M. Bendsøe, and O. Sigmund. Planar articulated mechanism design by graph theoretical enumeration. *Struct Multidisc Optim*, 27:295–299, 2004.
14. G. Laman. On graphs and rigidity of plane skeletal structures. *Journal of Engineering Mathematics*, 4:331–340, 1970.
15. A. Lee and I. Streinu. Pebble game algorihms and sparse graphs. In *Proc. EUROCOMB*, Berlin, September 2005.
16. A. Lee, I. Streinu, and L. Theran. Finding and maintaining rigid components. In *Proc. Canad. Conf. Comp. Geom.*, Windsor, Canada, August 2005.
17. M. Ohsaki and S. Nishiwaki. Shape design of pin-jointed multi-stable compliant mechanisms using snapthrough behavior. *Struct. Multidisc. Optim.*, 2005. published on-line.
18. G. Rote, F. Santos, and I. Streinu. Expansive motions and the polytope of pointed pseudo-triangulations. In J. P. Boris Aronov, Saugata Basu and M. Sharir, editors, *Discrete and Computational Geometry - The Goodman-Pollack Festschrift*, Algorithms and Combinatorics, pages 699–736. Springer Verlag, Berlin, 2003.
19. I. Streinu. A combinatorial approach to planar non-colliding robot arm motion planning. In *IEEE Symposium on Foundations of Computer Science*, pages 443–453, 2000.
20. I. Streinu. Pseudo-triangulations, rigidity and motion planning. *Discrete and Computational Geometry*, 34:587–635, December 2005. A preliminary version appeared in [19].

Appendix

A Algorithm Analysis

In this Appendix we give a more detailed version of the algorithm for enumerating planar Laman graphs and analyse its running time. We start by introducing some notation. For two vertices p_i and p_j we write $p_i < p_j$ when the label of p_i is smaller than that of p_j, and $p_i = p_j$ when they coincide. For each vertex $p_i \in V$, an *upper-hull edge* (*lower-hull edge*) of p_i is defined as the upper (lower) convex hull edge of $\{p_1, \ldots, p_i\}$ incident to p_i, and let p_i^{up} (p_i^{low}) denote the other endpoint of the upper-hull (lower-hull) edge of p_i. Note that $p_i p_i^{up}$ and $p_i p_i^{low}$ are both root edges. For a given graph G, let F_{p_i} be a set of vertices incident to p_i and whose labels are smaller than that of p_i. Let p_c denote the critical vertex of G.

Edge ordering. When we refer to an edge $p_i p_j$, the label of p_i is assumed to be larger than that of p_j. We define an edge ordering on the set of all possible edges in such a way that: (i) an edge $p_i p_j$ precedes an edge $p_k p_l$ whenever $i < k$, and (ii) when $i = k$ holds, $p_i p_i^{low}$ has the smallest label among the edges incident to p_i, and $p_i p_i^{up}$ has the label next to that of $p_i p_i^{low}$. If neither $p_i p_j$ nor $p_i p_l$ is a root edge, $p_i p_j$ precedes $p_i p_l$ when $j < l$. For example, Fig.1(b) have the ordering $\langle p_2p_1, p_3p_2, p_4p_3, p_4p_1, p_4p_2, p_5p_4, p_6p_3, p_6p_5, p_6p_4, p_7p_3, p_7p_4 \rangle$.

We use the notations $p_i p_j < p_k p_l$ when $p_i p_j$ proceeds $p_k p_l$, and $p_i p_j = p_k p_l$ when they coincide.

A.1 Complexity of the Parent operation.

Let $\mathbb{G}$ be the set of *all* the planar Laman graphs on our point set p, and let G_{root} be the root on a search tree on $\mathbb{G}$. The *Parent* function $f_{parent} : \mathbb{G} \setminus \{G_{root}\} \to \mathbb{G}$ is defined in such a way that, for $G \in \mathbb{G} \setminus \{G_{root}\}$ returns $G \setminus \{p_a p_b\} \cup \{p_{a'} p_{b'}\}$. We now give the details of how $p_a p_b$ and $p'_a p'_b$ are determined to identify the parent node uniquely.

The edge $p_a p_b$ to be removed is defined to be the one such that the following three conditions hold:

PR(1) p_a is a critical vertex of G,
PR(2) $p_a p_b$ is not an edge in the root, and
PR(3) $p_a p_b$ has the maximal label among the edges in $F_{p_i} \setminus \{p_a^{up}, p_a^{low}\}$.

The edge $p_{a'} p_{b'}$ to be inserted is defined to be the one such that the following three conditions hold:

PI(1) the graph obtained after performing the flip $(p_a p_b, p_{a'} p_{b'})$ is a planar Laman graph,
PI(2) it has a smaller index than that of G, and
PI(3) $p_{a'} p_{b'}$ has the maximal label among the edges satisfying PI(1) and PI(2).

From PR(1) and PI(2), $p_{a'} \leq p_c$ holds. Recall that $p_c p_c^{low}$ has the smallest label among the edges incident to p_c, and $p_c p_c^{up}$ has the label next to that of $p_c p_c^{low}$. Supposing that $p_c p_c^{up} < p_{a'} p_{b'}$ holds, the index does not decrease after performing the flip $(p_a p_b, p_{a'} p_{b'})$. Then we have $p_{a'} p_{b'} \leq p_c p_c^{up}$. From this observation, we now describe how an edge satisfying PI(1), PI(2) and PI(3) is found:

(a) if the graph obtained after performing the flip $(p_a p_b, p_c p_c^{up})$ is a planar Laman graph, then $p_{a'} p_{b'}$ is $p_c p_c^{up}$,

(b) else if the graph obtained after performing the flip $(p_ap_b, p_cp_c^{low})$ is a planar Laman graph, then $p_{a'}p_{b'}$ is $p_cp_c^{low}$,
(c) else $p_{a'} < p_c$ and $p_{a'}p_{b'}$ has the largest label among the edges satisfying PI(1).

Removing an edge. From the definition PR(1), one endpoint p_a of a removing edge coincides with p_c. The other endpoint of removing edge is a vertex of $F_{p_a} \setminus \{p_a^{up}, p_a^{low}\}$ which has the maximum label among them.

Inserting an edge. There might be $O(n^2)$ candidates for possible edge insertions satisfying the definition PI(1). One simple method to find an edge to be inserted is to test a new edge one by one in the decreasing order of labels until we get planar Laman graph. If we check planarity (in $O(n)$ time) and rigidity (in $O(n^2)$ time by using the pebble game algorithm of Hendrickson and Jacobs [11]), then the parent function requires $O(n^4)$ time in the worst case. However, the parent function can be computed in $O(n^2)$ time with $O(n^2)$ space or $O(n^3)$ time with $O(n)$ space respectively using the data structure for maintaining *rigid components* from Lee, Streinu and Theran [15, 16]. This data structure supports a *pair-find* query which determines whether two vertices are spanned by a common *component* in $O(1)$ time with $O(n^2)$ space and in $O(n)$ time with $O(n)$ space. Therefore, after removing one edge p_ap_b, we will calculate the rigid components of $G \setminus \{p_ap_b\}$ in $O(n^2)$ time. Then we maintain computing the data structure of [15, 16] so that the rigidity query can be answered for each insertion in $O(1)$ time afterwards with $O(n^2)$ space, or in $O(n)$ time with $O(n)$ space. For the planarity test, we will calculate the visibility graph. In this case, the visibility information is stored in an $n \times n$ matrix to answer the query in $O(1)$ time whether two points are visible to each other. Using the above data structures, we conclude that the complexity of the parent function computation is $O(n^2)$ time in $O(n^2)$ space or alternatively $O(n^3)$ time using $O(n)$ space.

A.2 Complexity of the *adjacency* operation.

Suppose we are given a planar Laman graph G. First, we define remove-add flip such that, for $e_1 \in G$ and $e_2 \in K_n$,

$$\mathit{Flip}(G, (e_1, e_2)) := G \setminus \{e_1\} \cup \{e_2\}.$$

If $\mathit{Flip}(G, (e_1, e_2))$ returns a planar Laman graph, we call a flip (e_1, e_2) *flippable.* Let L_G and L_{K_n} be the list of edges of G and K_n ordered lexicographically, let $\delta(G)$ and $\delta(K_n)$ be the number of elements of L_G and L_{K_n} and let $L_G(i)$ and $L_{K_n}(i)$ be the i-th elements of L_G and L_{K_n}, respectively. Then, the *adjacency* function is defined by

$$\mathit{Adj}(G, i, j) := \begin{cases} G \setminus \{e_1\} \cup \{e_2\} & \text{if } (e_1, e_2) \text{ is flippable,} \\ \mathit{null} & \text{otherwise,} \end{cases}$$

where $e_1 = L_G(i)$ and $e_2 = L_{K_n}(j)$.

Based on the algorithm in [2, 3], we describe our algorithm as follows:

Algorithm *Reverse Search*
1. G_{root} :=the *root* basis of the search tree

```
2.  $G := G_{root}$; $i, j := 0$;
3.  **repeat**
4.      **while** $i \leq \delta(L_G)$
5.          $i := i + 1$;
6.          **while** $j \leq \delta(L_{K_G})$
7.              $j := j + 1$;
8.              **if** $f_{parent}(Adj(G, i, j)) = G$ **then**
9.                  $G := Adj(G, i, j)$; $i, j := 0$;
10.                 $Output(G)$
11.             **endif**
12.         **endwhile**
13.     **endwhile**
14.     **if** $G \neq G_{root}$ **then**
15.         $G' := G$; $G := f_{parent}(G)$;
16.         determine integers pair $(i, j)$ such that $Adj(G, i, j) = G'$
17.         $i := i - 1$
18.     **endif**
19. **until** $G = G_{root}$, $i = \delta(L_G)$ and $j = \delta(L_{K_G})$
```

The algorithm has $\delta(L_G) \cdot \delta(L_{K_G})$ iterations, which is $O(n^3)$, and in each iteration we must check $f_{parent}(Adj(G, i, j)) = G$. Since the pebble game takes $O(n^2)$ time, each execution of f_{parent} and Adj operations take $O(n^2)$. Thus, this algorithm requires $O(n^5)$ time. We will reduce the running time to $O(n^3)$ by further characterising the eligible integer pairs (i, j).

For an integer i that represents a removing edge, we will show that

- $Adj(G, i, j)$ can be calculated in $O(1)$ time for each integer j by preprocessing the data structure in $O(n^2)$ time with $O(n^2)$ space and
- an integer j satisfying $f_{parent}(Adj(G, i, j)) = G$ can be found in $O(n^2)$ time if such an integer exists. We also show there exists at most one such an integer.

Thus, we achieve $O(n^3)$ time algorithm since we take $O(n^2)$ time for each integer i.

Let us consider the first term. We calculate rigid components of $G \setminus L_G(i)$ using the pebble game as the *Parent* function. As we mentioned in the previous section, the data structure for maintaining rigid components from [15, 16] supports a *pair-find* query by which rigidity query can be answered for each j in $O(1)$ time with $O(n^2)$ space. Such data structure can be computed in $O(n^2)$ time.

Let us consider the second term. It is described by the following lemma:

Lemma 5. *Let $p_a p_b = L_G(i)$ and $p_{a'} p_{b'} = L_{K_n}(j)$, and let (c, d) be the index of G. Then, $f_{parent}(Adj(G, i, j)) = G$ holds if and only if the edge pair $(p_a p_b, p_{a'} p_{b'})$ satisfies the following three conditions (A),(B) and (C).*

(A) *$p_{a'} p_{b'}$ is not the root edge.*

(B) **(B-1)** *if $p_a < p_c$, then $p_{a'} p_{b'} > p_c p_m$, where p_m is a vertex of $F_{p_c} \setminus \{p_c^{up}, p_c^{low}\}$ which has the maximum label among them,*

(B-2) *else if $p_a = p_c$ holds,*

(B-2-1) *if p_ap_b is neither an upper-hull edge nor a lower-hull edge of p_c, then $p_{a'}p_{b'} \geq p_{c+1}p_1$,*

(B-2-2) *else (p_ap_b is an upper-hull or lower-hull edge of p_c), then $p_{a'}p_{b'} > p_cp_m$,*

(B-3) *else if $p_a > p_c$ holds, $p_{a'}p_{b'} \geq p_ap_1$.*

(C) *$p_{a'}p_{b'}$ is an edge which has the minimum label among the edges $\{p_kp_l \mid p_ap_b < p_kp_l$, and $G \setminus \{p_ap_b\} \cup \{p_kp_l\}$ is planar Laman graph$\}$.*

We give some observations of these conditions since they are rather complicated. The proof is given afterwards.

We remind the reader that the f_{parent} is defined with the edge removal conditions **PR(1)**,**(2)** and **(3)** and the edge insertion conditions **PI(1)**,**(2)** and **(3)**. From the definition, the *Adj* guarantees condition PI(1). Condition PR(2) is guaranteed by (A). The condition (B) implies that G' has a greater index than that of G and $p_{a'}p_{b'}$ has the maximum label among the edges F_{p_c}, which ensures conditions PR(1),PR(3) and PI(2). Finally, the condition (C) implies that $p_{a'}p_{b'}$ has the minimum label among the edges $\{p_kp_l \mid p_kp_l > p_ap_b, G \setminus \{p_ap_b\} \cup \{p_kp_l\}$ is planar Laman graph$\}$ which ensures the condition PI(3).

We now describe explicitly these conditions. The second condition (B) is further divided into three subcases depending on the inserted edge p_ap_b. From the definition of the *Parent* function, the inserted edge always has a smaller index than that of the removed edge. Therefore, the inserted edge must have a greater index than that of the removed edge after *Adj* operation is performed. Then, we have $p_ap_b \leq p_{a'}p_{b'}$.

(B-1) When $p_a < p_c$ holds, then also $p_{a'} \geq p_c$ must hold, because otherwise the index does not change. If $p_{a'} = p_c$ holds, $p_{a'}p_{b'}$ must have the maximum label among the edges F_{p_c} in the order in which each edge $p_{a'}p_{b'}$ is removed when the f_{parent} is performed. Then we have $p_{a'}p_{b'} > p_cp_m$, where p_m is a vertex of $F_{p_c} \setminus \{p_c^{up}, p_c^{low}\}$, which has the maximum label among them (see example Fig.6).

(B-2-1) When $p_a = p_c$ holds and p_ap_b is neither an upper-hull edge nor a lower-hull edge of p_c, then the number of mismatched edges of the critical vertex is $d - 1$ after removing p_ap_b. Suppose that $p_{a'} = p_c$. Then the critical vertex does not change and the critical degree becomes d after the insertion of $p_{a'}p_{b'}$, and thus the index does not change after performing the flip. Therefore $p_{a'} > p_c$ must hold (see example Fig.7).

(B-2-2) When $p_a = p_c$ holds and p_ap_b is an upper-hull edge or a lower-hull edge of p_c, the number of mismatched edges of the critical vertex becomes $d + 1$ after removing an upper-hull or lower-hull edge. Thus, from the definition of the critical degree d, the index increases to $d + 1$ even after the edge $p_{a'}p_{b'}$ with $p_{a'} = p_c$ is inserted. For the same reason as (B-1), we obtain that $p_{a'}p_{b'} > p_cp_m$ (see Fig.8).

(B-3) When $p_a > p_c$ holds, p_ap_b is always a root edge. Then the index of G increases after removing p_ap_b. Note that the subgraph of G induced by $\{p_1, \ldots, p_c\}$ is a planar Laman graph and the subgraph of G induced by $\{p_1, \ldots, p_c, \ldots, p_{a-1}\}$ can be constructed by a *Henneberg I* construction. Therefore the subgraph of G induced by $\{p_1, p_2, \ldots, p_c, \ldots, p_{a-1}\}$ is a planar Laman graph, and we cannot insert an edge $p_{a'}p_{b'}$ with $p_{a'} < p_a$. Thus we have $p_{a'}p_{b'} \geq p_ap_1$ (see Fig.9).

To summarize the above observations, we have the condition(B).

Now we will describe the third condition (C). Suppose there exists an edge p_ip_j satisfying $p_ap_b < p_ip_j < p_{a'}p_{b'}$ and $G \setminus \{p_ap_b\} \cup \{p_ip_j\}$ is a planar Laman graph. The inserted edge cannot be $p_a p_b$ when $f_{parent}(Adj(G,i,j))$ is performed. Thus, $f_{parent}(Adj(G,i,j)) \neq G$(see example in Fig.10).

Putting together these observations, we obtain the proof of lemma 5 as follows:

Proof. The above observations implies necessity the conditions.

Suppose that the condition (A)(B) and (C) hold. From the conditions (A) and (B), the index of $G' = Adj(G,i,j))$ always increases. This implies that $p_{a'}p_{b'}$ is chosen as the removed edge returned by the *Parent* function when applied to G'.

From the edge insertion condition (C), there is no edge $p_i p_j$ with $p_a p_b < p_i p_j < p_{a'}p_{b'}$ such that $G' \setminus \{p_{a'}p_{b'}\} \cup \{p_i p_j\}$ is a planar Laman graph. Then $p_a p_b$ has the maximum label among the edges $\{p_k p_l \in K_n \mid G' \setminus \{p_{a'}p_{b'}\} \cup \{p_k p_l\}$ is a planar Laman graph$\}$. Then the inserted edge returned by the *Parent* function when applied to G' is $p_a p_b$. □

This completes the proof of Theorem 2.

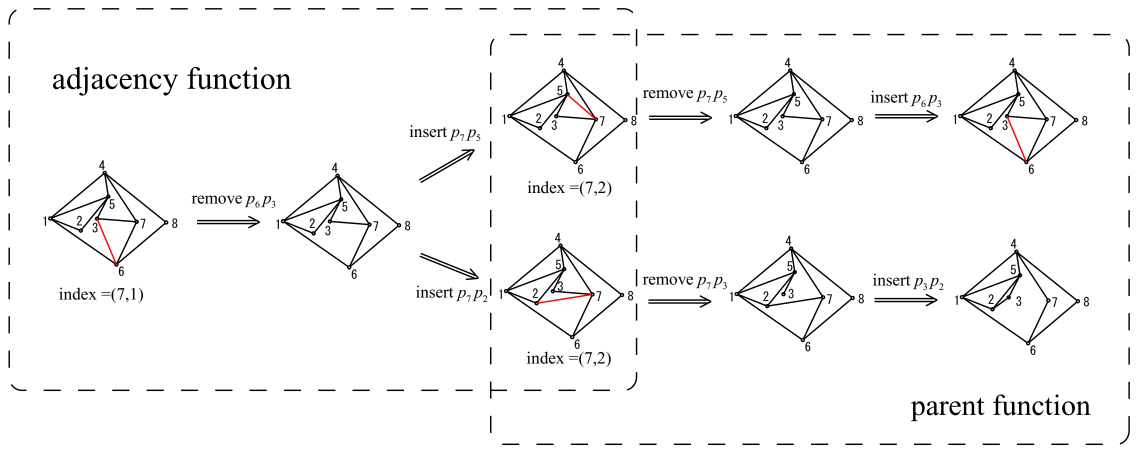

Fig. 6. Example of the case B-1. Consider removing p_6p_3. p_6 is less than p_7 which is the critical vertex of G. The flip (p_6p_3, p_7p_2) (in the lower diagram) is not eligible because p_7p_2 does not have the maximal label among the non-root edges incident to p_7 after performing the flip. Then the removed edge returned by the parent function when applied to G' is p_7p_3, not p_7p_2. On the other hand, the flip (p_6p_3, p_7p_5) is eligible.

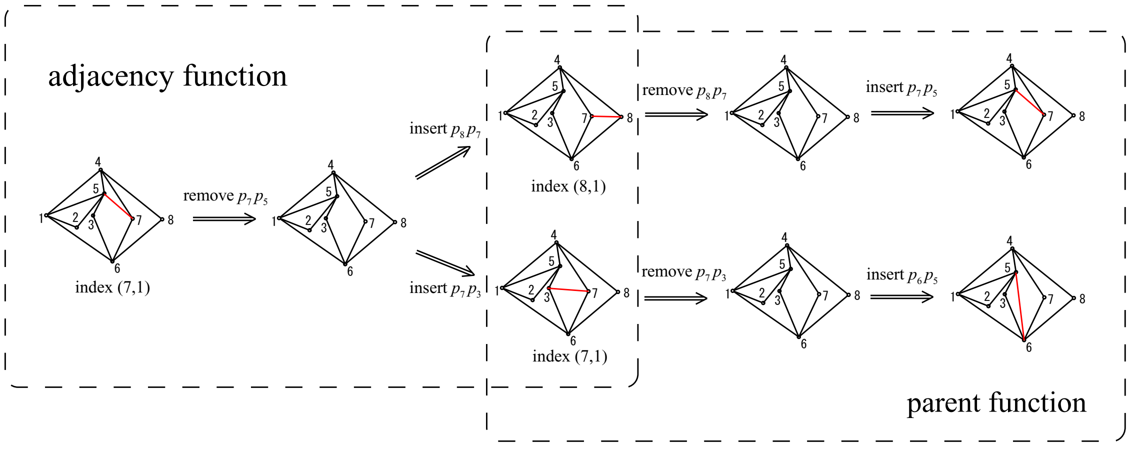

Fig. 7. Example of the case B-2-1. Consider removing p_7p_5. After inserting p_7p_3, the index of a child node does not change. Thus, the flip (p_7p_5, p_7p_3) is not eligible. On the other hand, the flip (p_7p_5, p_8p_7) increases the index because one endpoint of the inserted edge, namely p_8, has a greater label than that of the critical vertex of G.

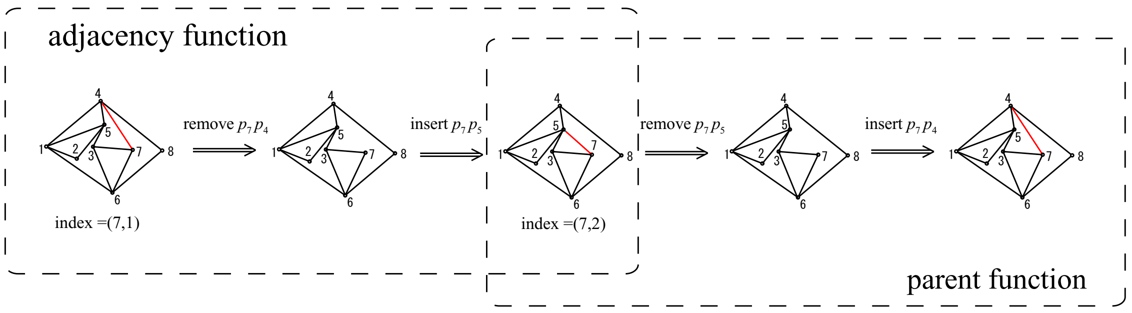

Fig. 8. Example of the case B-2-2. Consider removing p_7p_4, which is a root edge incident to the critical vertex p_7. The index increases even if one endpoint of the inserted edge is p_7.

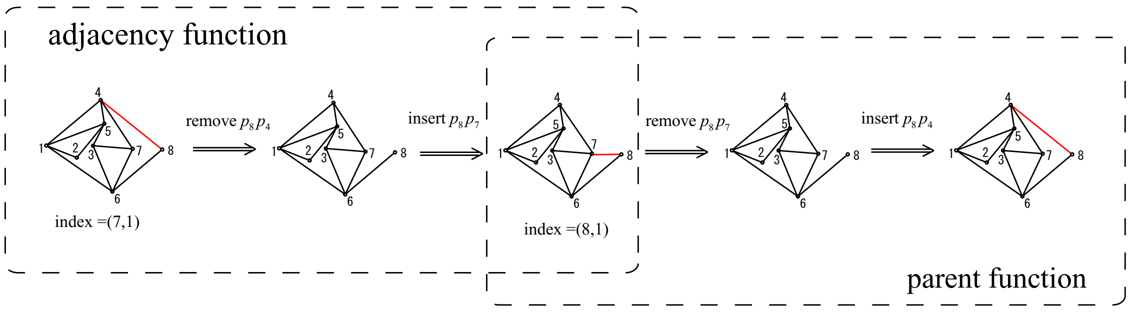

Fig. 9. Example of the case B-3. Consider removing p_8p_4, where p_8 is greater than the critical vertex p_7. The subgraph of G induced by $\{p_1, p_2, \ldots, p_7\}$ is a planar Laman graph. Then we can insert only p_8p_7.

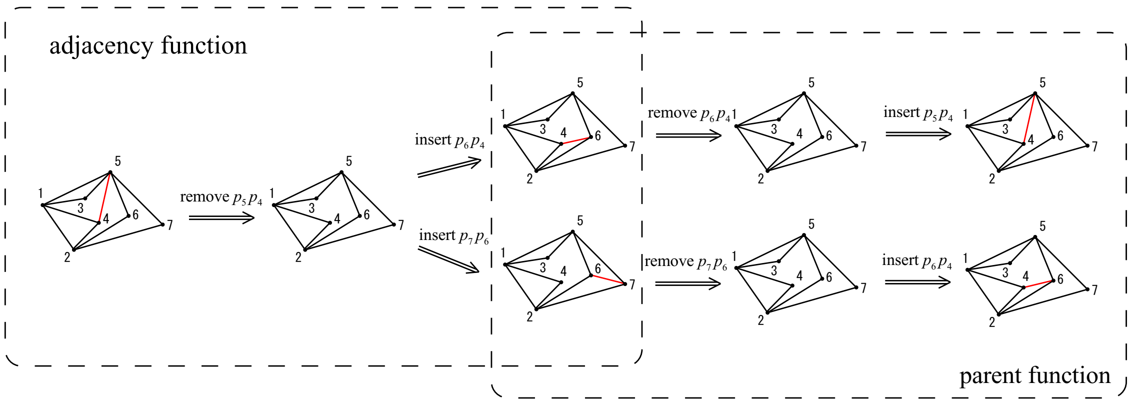

Fig. 10. Example of the condition (C). The removed edge in the adjacency function and the inserted edge in the parent function coincide in the upper diagram, and do not coincide in the lower diagram. This follows from the fact that there is an edge p_6p_4 with $p_5p_4 < p_6p_4 < p_7p_6$ satisfying that $G \setminus \{p_5p_4\} \cup \{p_6p_4\}$ is a planar Laman graph. Only the upper flip (p_5p_4, p_6p_4) is eligible.