The Fourth China-Japan-Korea Joint Symposium on Optimization of Structural and Mechanical Systems
Kunming, Nov. 6-9, 2006, China

Enumeration of Optimal Pin-Jointed Bistable Compliant Mechanisms

Naoki Katoh[1]*, Makoto Ohsaki[1] , Takuya Kinoshita[1], Shin-ichi Tanigawa[1], David Avis[2], Ileana Streinu[3]

[1]*Dept. of Architecture and Architectural Engineering, Kyoto University, Kyotodaigaku-Katsura, Nishikyo, Kyoto 615-8540, Japan*
**E-mail: naoki@archi.kyoto-u.ac.jp*
[2]*School of Computer Science, McGill University, Canada*
[3]*Dept. of Computer Science, Smith College, Northampton, MA 01063, USA*

Abstract

Recently, a new type of mechanism called compliant mechanism has been developed and applied mainly in the field of micro-mechanics. A compliant mechanism has flexible parts to stabilize the structure, which is contrary to the conventional unstable mechanism. Although a compliant mechanism is usually modeled as a continuum with elastic joints, it is possible to generate the similar mechanism by a bar-joint system. Ohsaki and Nishiwaki (Struct. Multidisc. Optim., Vol. 30, pp. 327-334, 2005) presented a method for generating flexible multistable bar-joint mechanisms using nonlinear programming approach. However, due to high nonlinearity of the problem, the nonlinear programming problem should be solved many times starting from different initial solutions to obtain several types of mechanisms. Since the compliant bar-joint mechanism is usually statically determinate, the optimization problem can be solved easily if the design space is limited to statically determinate structures.

In this paper, we present an algorithm for enumerating without repetitions all the non-crossing generically minimally rigid bar-joint frameworks, which are regarded as statically determinate trusses in structural engineering. Bistable mechanisms utilizing snapthrough behavior are generated from the statically determinate trusses. In the numerical examples, many bistable compliant mechanisms are generated to show the effectiveness of the proposed method.

Keywords: *Structural Optimization, Bistable Compliant Mechanism, Minimally Rigid Frameworks, Minimally Rigid Graph*

1 Introduction

Contrary to unstable conventional bar-joint mechanisms, a *compliant mechanism* utilizes elastic deformation of structural parts to realize the mechanism for producing large output displacement in different direction from the input displacement. Although compliant mechanism is usually designed as continuum, it is also possible to use flexibility of members of a bar-joint structure to realize shape transformation of structure. Conventional link mechanism is unstable, and achieves stability at undeformed and deformed states by applying additional forces or constraints. However, *a bistable compliant mechanism* can keep stability at undeformed and deformed states through its own stiffness, because it has two stable states without any additional constraint.

Ohsaki and Nishiwaki [7] presented an optimization algorithm for generating multistable compliant mechanisms by utilizing snapthrough behavior, where the conventional ground structure approach is used and the design variables of the optimization problem are the cross-sectional areas of members and the nodal coordinates of a bar-joint system (truss). Design conditions are given on the magnitude of output displacement, and the stability before and after deformation. The total structural volume is minimized as the objective function. However, due to high nonlinearity of the problem, the nonlinear programming problem should be solved many times starting from different initial solutions to obtain several types of mechanisms.

Since the compliant bar-joint mechanism is usually *statically determinate*, the optimization problem can be solved easily if the design space is limited to statically determinate structure. To obtain several types of mechanisms for given members and nodes that can exist, many candidates of statically determinate structures should be given as initial solutions for the optimization problem. Therefore in this paper we propose an approach based on the enumeration of statically determinate structures to find many types of compliant mechanisms.

The methodologies in graph theory and computational geometry can be effectively used for generating the candidate structures. Graph theoretical approaches are widely used in structural mechanics, where the bars and joints of the structure represent the edges and vertices in the graph, respectively. In this study, we present an algorithm for enumerating without repetitions all the *planar embedded minimally rigid graphs*, which are called *non-crossing generically minimally rigid bar-joint frameworks* or simply called *non-crossing Laman frameworks*. A Laman framework is regarded as a statically determinate truss in structural engineering.

Our enumeration algorithm is based on the *reverse search* paradigm of Avis and Fukuda [2,3], which 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 particular, we obtain that the set of all non-crossing Laman frameworks 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. 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 non-crossing generically minimally rigid frameworks.

In the numerical examples, many bistable compliant mechanisms are generated to show the effectiveness of the proposed method. It is shown that large deformation can be realized by snapthrough behavior of local triangular elements of several types even for a simple structure with ten nodes including supports.

2 Preliminaries

2.1 Non-crossing Laman frameworks

Let $G=(V,E)$ be a graph with $n=|V|$ vertices and $m=|E|$ edges. G is a *minimally rigid graph* (also called *Laman graph*) if and only if $m=2n-3$ and every subset of $n'>1$ vertices spans at most $2n'-3$ edges. An *embedding* $G(\mathbf{p})$ of the graph G on a set of points $\mathbf{p}=\{p_1,\cdots,p_n\}\subset\mathbb{R}^2$ is a mapping of the vertices $V=\{1,\cdots,n\}$ to points in the Euclidian plane $i\mapsto p_i$. The edges $ij\in E$ are mapped to straight line segments p_ip_j. An embedding $G(\mathbf{p})$ is *non-crossing* if no pair of segments p_ip_j and p_kp_l corresponding to non-adjacent edges $ij,kl\in E$ have a point in common.

Laman graphs embedded on generic point sets are called *Laman frameworks* and have the special property of being *minimally rigid* [5,6], when viewed as bar-joint frameworks with fixed edge-lengths, which motivates the tremendous interest in their properties. It is well-known that Laman frameworks are regarded as statically determinate trusses in structural engineering [8].

Besides the above definition, Laman graphs can be characterized in a variety ways. In particular, Laman graph on *n* vertices has an inductive construction, called *Henneberg construction* [8,9]. Start from an edge for $n=2$. At each step, add a new vertex in one of the following two ways:

Henneberg I: add a new vertex and connect it to two old vertices via two new edges.

Henneberg II: remove an old edge, add new vertex, and connect it to two endpoints of the removed edge and to some other vertex.

The Laman frameworks on a generic point set form the set of *bases* of the *generic rigidity matroid* on the complete graph K_n, see [9]. The bases have all the same size $2n-3$. Bases may be related via the *base exchange* operation, which we will call a *flip* between two Laman frameworks. Two Laman frameworks L_1 and L_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, $e_1=L_1/L_2$, $e_2=L_2/L_1$. Using flips, we can define a graph whose nodes are all the Laman frameworks on *n* vertices, and whose edges correspond to flips. 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 non-crossing Laman frameworks may not necessarily be.

2.2 Reverse search

This technique is a memory efficient method for visiting all the nodes of a implicitly defined connected graph whose nodes are objects to be enumerated and are connected to the other nodes by some local search operation (flip). 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 as 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 non-crossing Laman frameworks of a given point set. We choose a certain Laman framework to be the root. Then we define a *parent function* for every non-root Laman framework. To show that the parent function defines a search tree we associate an *index* to every Laman framework such that the parent function always returns a Laman framework with smaller index. This gives a forest structure. We omit the proof that it forms a search tree (see [4] for the proof).

Let us define the root. We choose the root of the search tree to be a *greedy pseudo-triangulation* corresponding to a fixed direction. For this, we first sort the points by *x*-coordinates and label them as $\{1,\ldots,n\}$ in this order. Then we construct a **Henneberg I** pointed pseudo-triangulation as follows. Start with the edge 12 and continue followings for $n-2$ steps. At each step, the next vertex (in *x*-sorted order) is added (vertex $i+2$ at step i, $i=1,\ldots,n-2$), together with the two *tangents* from point p_{i+2} 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.

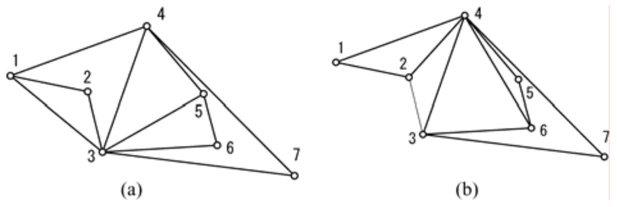


Figure 1. (a)The root framework. (b)A non-root Laman framework.

Next let us define the index for each non-crossing Laman frameworks. Given a non-crossing Laman framework *L*, we define its *index* as a pair of integers $\text{index}(L)=(c,d)$, where $c=c(L)\in\{2,\ldots,n\}$ and $d=d(L)\in\{1,\ldots,n-2\}$ are, respectively, the label of the *critical vertex* and the *critical degree* of the critical vertex, defined below. An edge in *L* is called *non-root* if it doesn't exist in the *root*. The *critical vertex* is the largest label of a vertex which has non-root edges. The *critical degree* is the number of non-root edges incident to the critical vertex. For example, the non-root framework in Fig.1(b) has index (6,1). We use the index as a measure of how far a node (Laman framework) is from the root, whose index is defined to be (1,0).

The *parent function* for each Laman framework is defined in term 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 its index. The efficiency of the parent function depends on the lexicographic ordering on the edges. The correctness of the parent function which will be described in the next section follows from our Main Theorem:

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

Based on Theorem 1, we will propose the following efficient algorithm for the enumeration:

Theorem 2: *The set of all non-crossing Laman frameworks on a given point set can be reported in* $O(n^3)$ *time per non-crossing Laman framework using* $O(n^2)$ *space.*

The precise proofs of the above theorems are omitted in this proceeding version (see [4] for the proof) .

4 Algorithm for Enumerating Non-crossing Laman Frameworks

In this section we give a more detailed version of the algorithm for enumerating non-crossing Laman frameworks. We start by introducing some notations. For each vertex p_i, *an upper-hull edge* (*lower-hull edge*) of p_i is defined as the upper (lower) convex hull edge of $\{p_1,\ldots,p_{i-1}\}$ 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 $p_i^{up}p_i$ and $p_i^{low}p_i$ are both root edges. When we refer to an edge p_ip_j, the label of p_i is assumed to be smaller than that of p_j. We define a lexicographical ordering on the set of all possible edges in such a way that: (i) an edge p_ip_j precedes an edge p_kp_l whenever $j<l$, and (ii) when $j=l$

holds, $p_j^{low} p_j$ precedes all the other edges incident to p_j, and $p_j^{up} p_j$ is ordered next to $p_j^{low} p_j$. If neither $p_i p_j$ nor $p_k p_j$ is a root edge, $p_i p_j$ precedes $p_k p_j$ when $i < k$. We use the notations $p_i p_j < p_k p_l$ when $p_i p_j$ precedes $p_k p_l$, and $p_i p_j = p_k p_l$ when they coincide. For an edge set *A*, we use the notation **max** to denote the lexicographically largest edge in *A*.

4.1 Parent function

Let $\mathbf{L}$ be the set of *all* the non-crossing Laman frameworks on a given generic point set, and let L_{root} be the root on a search tree. We define the following *parent function* $f_{\mathrm{parent}} : \mathbf{L} / \{L_{\mathrm{root}}\} \to \mathbf{L}$ based on the correctness of Theorem 1:

Definition 1. (Parent function) *For* $L' = \mathbf{L} \setminus \{L_{\mathrm{root}}\}$, *let* $p_{c'}$ *be the critical vertex in* L'. $L = L' \setminus \{e_1'\} \cup \{e_2'\}$ *is the parent of* L', *where* $e_1' = \max\{e \mid e \in L' \setminus L_{\mathrm{root}}\}$, *and* $e_2' = \max\{e \in K_n \mid e \le p_{c'}^{up} p_{c'}, \mathit{and}\ L' \setminus \{e_1'\} \cup \{e\} \in \mathbf{L}\}$.

Note that the edge e_1' to be deleted is incident to $p_{c'}$ from the definition of the critical vertex in the previous section. We can show that the parent function can be performed in $O(n^2)$ time using $O(n^2)$ space (see [4]).

4.2 Finding a next object

Given $L \in \mathbf{L}$. Let $List_L$ and $List_{Kn}$ be the list of edges of *L* and K_n ordered lexicographically. Let $|L|$ and $|K_n|$ be the number of elements of *L* and K_n, and let $List_L(i)$ and $List_{Kn}(i)$ be the *i*-th elements of $List_L$ and $List_{Kn}$, respectively. Then, we define the local search operation for each *L*, called *adjacency* function, such that,

$$Adj(L, i, j) = \begin{cases} L \setminus \{e_1\} \cup \{e_2\} & \mathit{if}\ L \setminus \{e_1\} \cup \{e_2\} \in \mathbf{L}, \\ \mathit{null} & \mathit{otherwise}, \end{cases}$$

where $e_1 = List_L(i)$ and $e_2 = List_{Kn}(j)$.

Based on the algorithm in [2,3], we describe our algorithm in Fig.2. An example of the search tree of non-crossing Laman frameworks on five points is given in Fig.3.

```
Algorithm Reverse Search
1: $L_{root}$:= the root of the search tree;
2: $L := L_{root}; i, j := 0; Output(L)$;
3: repeat
4:   while $i \le |L|$ do
5:     $i := i + 1$;
6:     while $j \le |K_n|$ do
7:       $j := j + 1$;
8:       if $Adj(L, i, j) \ne null$ and $f_{parent}(Adj(L, i, j)) = L$ then
9:         $L := Adj(L, i, j)$; $i, j := 0$;
10:        $Output(L)$;
11:        go to line 4;
12:      end if
13:    end while
14:  end while
15:  if $L \ne L_{root}$ then
16:    $L' := L$; $L := f_{parent}(L)$;
17:    determine integers pair $(i, j)$ such that $Adj(L, i, j) = L'$;
18:    $i := i - 1$;
19:  end if
20: until $L = L_{root}, i = |L|, j = |K_n|$
```

Figure 2. The algorithm for enumerating non-crossing Laman frameworks.

5 Finding Bistable Compliant Mechanisms

Various types of compliant mechanisms are found from the set of eight free nodes and two supports as shown in Fig. 4, where *W* = 0.2 m and *H* = 0.1 m. An input force is applied at node B in the negative *y*-direction to produce an output displacement of 0.05 m at node A in the *y*-direction. Young's moduls is 2.0 GPa and the rotated engineering strain is used for the definition of strain-displacement relation. The equilibrium path is traced by the displacement increment method.

Optimization is carried out by IDESIGN Ver. 3.5 [1], where the sequential quadratic methods is used. The lower bound A^{L} for the cross-sectional area A_i of the *i*-th member is 1.0×10^{-6} m^2, and the members with $A_i = A^{\mathrm{L}}$ after optimization are removed. The details of the problem formulation and the optimization approach are described in [7].

Using the proposed enumeration method, 7793 different initial topologies have been generated. The total number of non-crossing statically determinate structure is rather small, because the nodes are located at a regular grid and many solutions are rejected due to existence of overlapping bars.

By carrying out optimization from 7793 initial solutions, we found 1148 different types of mechanisms. Figs. 5-7 show the three typical solutions, where (a) is the initial topology, (b) is the relation between the input displacement and the input displacement at node B, (c) is the optimal topology, (d) is the deformed shape of the optimal solution, and (e) is the optimal topology consisting of the flexible members only. It is seen from Figs. 5-7(b) that a limit point of the load that leads to snapthrough is reached as the input force is increased. It can be confirmed from Figs. 5-7(c-e) that a triangle unit in each type degenerate to a set of three collinear members after deformation to produce the snapthrough behavior.

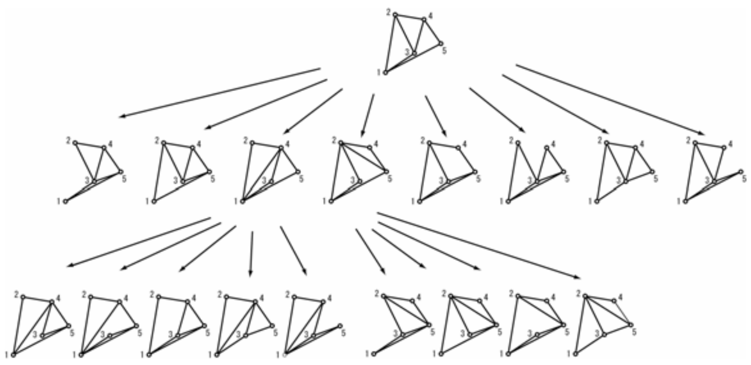


Figure 3. An example of a search tree of non-crossing Laman frameworks on five points. Top figure represents the root.

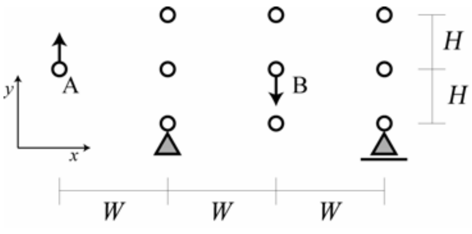
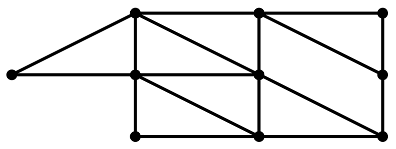
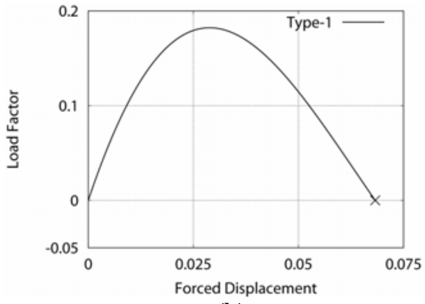
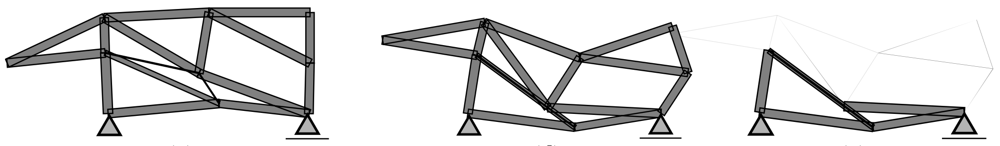


Figure 4. Initial set of nodes and supports.



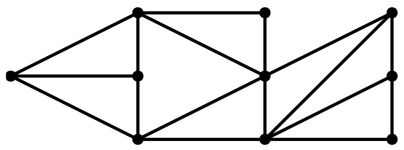


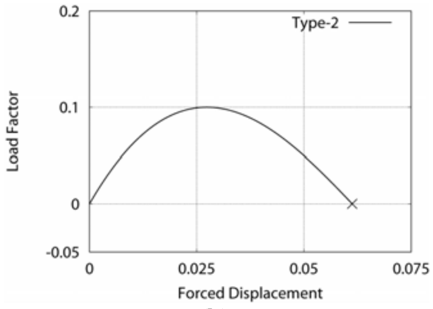
(a) (b)



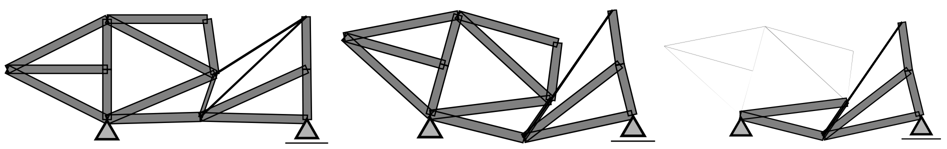
(c) (d) (e)

Figure 5. Optimal topology (Type-1)



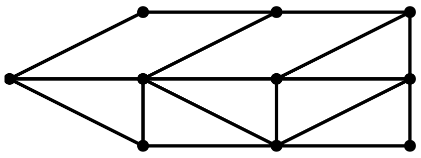


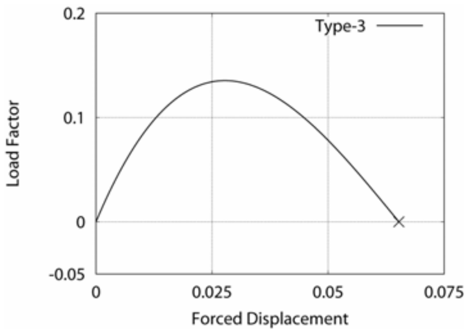
(a) (b)



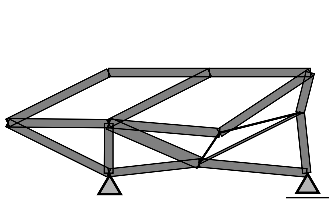
(c) (d) (e)

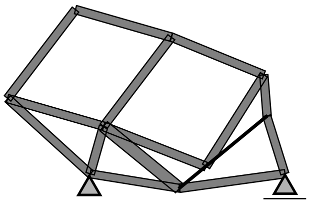
Figure 6. Optimal topology (Type-2)


(a)



(b)


(c)


(d)

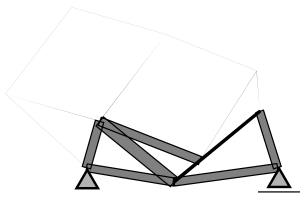

(e)

Figure 7. Optimal topology (Type-3)

6 Conclusions

An enumeration algorithm of non-crossing Laman frameworks, which is equivalent to a statically determinate bar-joint system in engineering field, has been developed based on the *reverse search* paradigm of Avis and Fukuda [2,3]. Many bistable compliant mechanisms can be generated from the initial solutions obtained by the proposed enumeration algorithm. It has been shown in the numerical examples that large deformation and bistability can be realized by snapthrough behavior of local triangular elements.

References

1. Arora J. S. and Tseng C. H., IDESIGN User's Manual, Ver. 3.5., Technical Report, Optimal Design Laboratory, The University of Iowa, 1987
2. Avis D. and Fukuda K. A pivoting algorithm for convex hulls and vertex enumeration of arrangements and polyhedra. Discrete and Computational Geometry, 1992, 8:295-313
3. Avis D. and Fukuda K. Reverse search for enumeration. Discrete and Computational Geometry, 1996, 65(1-3):21-46
4. Avis D., Katoh N., Ohsaki M, Streinu I. and Tanigawa S., Enumerating non-crossing minimally rigid frameworks In Proc. of COCOON 2006, LNCS 4112, pages 205-215, Springer, 2006
5. Graver J., Servatius B. and Servatius H. Combinatorial Rigidity. Graduate Studies in Mathematics vol.2. American Mathematical Society, 1993
6. Laman G. On graphs and rigidity of plane skeletal structures. J. Eng. Mathematics, 1970, 4:331-340
7. Ohsaki M. and Nishiwaki S. Shape design of pin-jointed multi-stable compliant mechanisms using snapthrough behavior. Struct. Multidisc. Optim., 2005, 30:327-334
8. Tay T. S. and Whiteley W. Generating isostatic frameworks. Structural Topology, 1985, 11:21-69
9. Whiteley W. Some matroids from discrete applied geometry. In Matroid Theory, Contemporary Mathematics, vol.197, American Mathematical Society, 1997