

Graph Rigidity on Localization in WSNs

Buddhadeb Sau

Advanced Computing and Microelectronics Unit
Jadavpur University, Kolkata, India
bsau@math.jdvu.ac.in

QIP Short Term Course
Recent Trends in Networks and Distributed Computing
IITG, February 10-14, 2015

Organization of the paper

- 1 Network localization Problem in WSN**
- 2 Techniques for network localization**
- 3 Rigidity and localizability**

Basic localization model in a homogeneous WSN

We consider the basic localization model:

- Field of interest is **convex**.
- Sensors are **identical**.
- **Sensing range** $\rightarrow s$. **Communication range** $\rightarrow r$.
- **No sensing and communication barrier** present in the WSN field.
- Sensors are in **general position**.
- Distances are measured **accurately**.
- For (s_i, s_j) ,
if $\text{dist}(s_i, s_j) \leq r$, the value of $\text{dist}(s_i, s_j)$ is **known**.
If $\text{dist}(s_i, s_j) > r$, the value of $\text{dist}(s_i, s_j)$ is **unknown**.

Basic localization model in a homogeneous WSN

Graph Model of WSN

Let WSN contain n sensors s_1, s_2, \dots, s_n . We construct an **undirected edge-weighted** graph $G = (V, E, w)$ as follows:

- **Each vertex** in V represents a sensor.
- $(s_i, s_j) \in E$ iff the **distance** between s_i and s_j is **known**.
- $\forall (s_i, s_j) \in E, \text{dist}(s_i, s_j) = w(s_i, s_j)$.
- $\text{dist}(s_i, s_j) > r, \forall (s_i, s_j) \notin E$.

In \mathbb{R}^2 , this graph model is a **Unit Disk Graph** (UDG).

Basic localization model in a homogeneous WSN

Graph Model of WSN

Let WSN contain n sensors s_1, s_2, \dots, s_n . We construct an **undirected edge-weighted** graph $G = (V, E, w)$ as follows:

- Each vertex in V represents a sensor.
- $(s_i, s_j) \in E$ iff the **distance** between s_i and s_j is **known**.
- $\forall (s_i, s_j) \in E, \text{dist}(s_i, s_j) = w(s_i, s_j)$.
- $\text{dist}(s_i, s_j) > r, \forall (s_i, s_j) \notin E$.

In \mathbb{R}^2 , this graph model is a **Unit Disk Graph** (UDG).

Problem

Given an **edge-weighted undirected graph** $G = (V, E, w)$, our objective is to find **possible position assignments** to the nodes of the graph G in \mathbb{R}^2 under the above model.

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Graph realization problem

Given a graph $G = (V, E, w)$. Does there exist a realization of G in \mathbb{R}^m ?

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Graph realization problem

Given a graph $G = (V, E, w)$. Does there exist a realization of G in \mathbb{R}^m ?

- Saxe [10] has shown that the graph realization problem is **NP-complete in one dimension** and **NP-hard in higher dimensions**.

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Graph realization problem

Given a graph $G = (V, E, w)$. Does there exist a realization of G in \mathbb{R}^m ?

- Saxe [10] has shown that the graph realization problem is ***NP-complete in one dimension*** and ***NP-hard in higher dimensions***.
- Breu and Kirkpatrick [2] has shown that **unit disk graph realization problem** is also ***NP-hard***.

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Is there any realization of the WSN at all?

Graph realization

Definition

A **realization** of $G = (V, E, w)$ in \mathbb{R}^m (Euclidean space of dimension m) is an assignment of coordinates (x_1, \dots, x_m) to the vertices so that weight of an edge represents the Euclidean distance between the vertices incident on the edge.

Is there any realization of the WSN at all?

- The graph underlying the proposed WSN model has **at least one realization**, since the distance information is coming from an actual deployment of sensors

Techniques for location estimation of network nodes

Multilateration technique:

- An unknown node can **estimate** its location with **information from its neighbors**.
- Some of the neighbors (beacons) know their positions by GPS or some other means.
- A node becomes a beacon after knowing its location in the network.

Techniques for location estimation of network nodes

Atomic multilateration

Let (x, y) be the position of a node v with N beacon neighbors v_1, v_2, \dots, v_N . Let s be the ultrasound signal propagation speed s and t_i be the time to propagate the signal from v_i to v . Let

$$f_i(x, y, s) = s t_i - \sqrt{(x_i - x)^2 + (y_i - y)^2}$$

- For **noisy distances**, if an **adequate number of beacons** are available, a **Maximum Likelihood Estimate** of (x, y) can be computed from a system of equations obtained by taking the **minimum mean square estimate** of $f_i(x, y, s)$.
- For **noise-less distances**, **three or more beacons** result in unique position of v .

Techniques for location estimation of network nodes

Iterative multilateration

- In a **cluster based network**, the positions of the **cluster heads are known**.
- The cluster heads are assumed to have **knowledge about the complete network**.
- **Computations** in clusters run in a **distributed manner**.
- Each cluster head uses **atomic multilateration** to estimate the positions of non-localized nodes as much as possible.
- It can start with the maximum possible number of beacons to estimate the remaining unknown nodes.

Techniques for location estimation of network nodes

Optimization techniques in localization:

- If the **average error** in positioning is taken as the **performance metric**, the localization problem becomes a **geometric optimization problem** (Doherty et al. [3]).
- It finds the points $x_1, x_2, \dots, x_n \in \mathbb{R}^m$ which optimize the cost function

$$Z_m = \min \sum_{i < j} (\|x_i - x_j\| - \delta_{ij})^2$$

where δ_{ij} are given for all $i < j = 1, 2, \dots, n$ and $D = (\delta_{ij})_{n \times n}$ is the corresponding distance matrix.

- Depending on **different types of distance information**, different **mathematical optimization tools** are used to estimate the locations.

Techniques for location estimation of network nodes

Optimization techniques in localization:

Multidimensional scaling (MDS)

MDS generally uses **eigendecomposition** (spectral decomposition) of the matrix to solve this optimization problem.

- In real applications of WSNs, **inter node distance** estimates are available only for the node **pairs within the communication range**.
- The **complete distance matrix** may not be available. In such cases, **MDS-MAP** finds the **all-pair-shortest path distances** and forms the distance matrix with these **rough distance estimates**.
- With this distance matrix, MDS-MAP estimates the positions of the nodes which gives the **best fit**.

Techniques for location estimation of network nodes

Optimization techniques in localization:

Quadratic Programming Formulation (QPP)

For **noisy distances**, if given δ_{ij} is an **upper bound** of the distance then the localization problem in WSN may be **formulated as a QPP** with the above cost function Z_m subject to the **distance constraints**:

$$\|x_i - x_j\| \leq \delta_{ij}, \text{ if the distance } \delta_{ij} \text{ is known.}$$

$$\|x_i - x_j\| > r, \text{ if the distance } \delta_{ij} \text{ is unknown.}$$

This QPP is non-convex for **non-convex constraints**:

- **non-adjacency conditions**: $\|x_i - x_j\| > r$ and
- $\|x_i - x_j\| = \delta_{ij}$, if δ_{ij} is the **exaxct distance** between the pair.

Techniques for location estimation of network nodes

Optimization techniques in localization:

Quadratic Programming Formulation (QPP)

- Doherty et al. [3] **ignored non-adjacency conditions** to make it convex.
- Positions of sensors are estimated by using **Semidefinite Program**, in polynomial time, optimizing the total distance error in location estimation.
- Biswas et al. [1] used *relaxation technique* to reduce the error in position estimation.

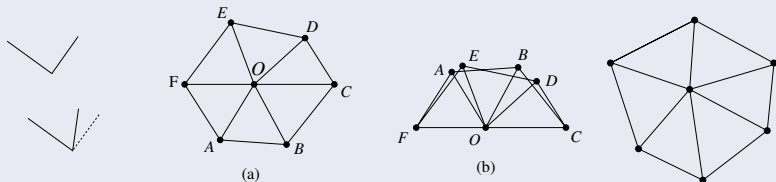
Graph rigidity

- A realization may be visualized as a frame constructed by a **finite set of rods** joined at their **end points**.
- A **perturbation** on the frame gives **different realizations**.

The realizations obtained by **rotating**, **flipping** or **shifting** the whole structure, do not really count as different.

Definition

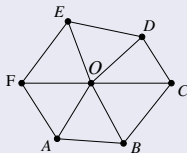
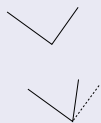
Two realizations preserving the distances among all pairs of nodes, irrespective of whether the pair is in E or not, are called **congruent**.



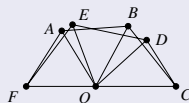
Graph rigidity

Definition

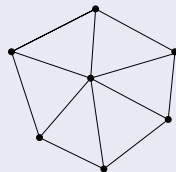
A realization of an edge-weighted graph $G = (V, E, w)$ is called **rigid**, if it has no continuous deformation which generates another realization of G .



(a)



(b)



Graph rigidity

Definition

A realization of an edge-weighted graph $G = (V, E, w)$ is called **rigid**, if the it has no continuous deformation which generates another realization of G .

Theorem (Laman [8])

*An edge-weighted graph $G = (V, E, w)$, $|V| = n$, is **generically rigid** in \mathbb{R}^2 if and only if there is a subset $E' \subseteq E$ consisting of $2n - 3$ edges such that, for any nonempty subset $E'' \subseteq E'$, $|E''| \leq n'$ where n' is the number of vertices of G which are incident with edges in E'' .*

Graph rigidity

Definition

A realization of an edge-weighted graph $G = (V, E, w)$ is called **rigid**, if it has no continuous deformation which generates another realization of G .

Theorem (Laman [8])

*An edge-weighted graph $G = (V, E, w)$, $|V| = n$, is **generically rigid** in \mathbb{R}^2 if and only if there is a subset $E' \subseteq E$ consisting of $2n - 3$ edges such that, for any nonempty subset $E'' \subseteq E'$, $|E''| \leq n'$ where n' is the number of vertices of G which are incident with edges in E'' .*

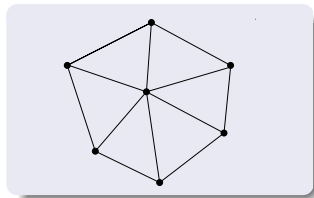
Generic rigidity testing

In \mathbb{R}^2 , based on variants of Laman's theorem, Hendrickson [5] has proposed a fast $|V|^2$ algorithm for generic rigidity testing.

Graph rigidity

Definition

$G = (V, E, w)$ is **globally rigid**, if the **distance** between every pair of nodes remains preserved in **every realization**.



Localizable (uniquely realizable) graphs

Consider a network deployed in \mathbb{R}^2 with nodes in **general position**.

Localizable (uniquely realizable) graphs

Consider a network deployed in \mathbb{R}^2 with nodes in **general position**.

Theorem ([4])

*The graph underlying such a network is **uniquely realizable** if and only if it has at least **three anchors** (nodes with known positions) and the network is globally rigid.*

Localizable (uniquely realizable) graphs

Consider a network deployed in \mathbb{R}^2 with nodes in **general position**.

Theorem ([4])

*The graph underlying such a network is **uniquely realizable** if and only if it has at least **three anchors** (nodes with known positions) and the network is globally rigid.*

Theorem (Jackson and Jordán [7])

*An edge-weighted graph $G = (V, E, w)$, $|V| = n \geq 4$, is **generically globally rigid** in \mathbb{R}^2 if and only if it is 3-connected and redundantly rigid in \mathbb{R}^2 .*

Localizable (uniquely realizable) graphs

Consider a network deployed in \mathbb{R}^2 with nodes in **general position**.

Theorem (Jackson and Jordán [7])

*An edge-weighted graph $G = (V, E, w)$, $|V| = n \geq 4$, is **generically globally rigid** in \mathbb{R}^2 if and only if it is 3-connected and redundantly rigid in \mathbb{R}^2 .*

Localizability testing in a central machine

In view of this theorem, **localizability testing can efficiently be done** as:

- Testing of 3-connectivity can be done in polynomial time with standard techniques [6, 9].
- Redundant rigidity testing can efficiently be done by the algorithm proposed by Hendrickson [5].

Localizable (uniquely realizable) graphs

Localizability testing distributedly

- Efficient distributed localizability testing is an **open problem** for arbitrary networks.

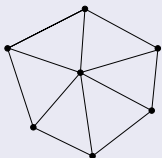
Localizable networks: distributedly recognizable by existing techniques

Trilateration graph:

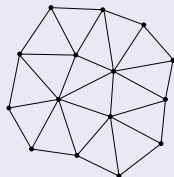
A graph is a **trilateration graph** if it has a **trilateration ordering** $\pi = \{u_1, u_2, \dots, u_n\}$ of nodes, where u_1, u_2, u_3 form a K_3 and each u_i ($i \geq 3$) has at least three neighbors before u_i in π .

Wheel extension graph:

G is a wheel extension if it has an ordering $\pi = \{u_1, u_2, \dots, u_n\}$, where u_1, u_2, u_3 form a K_3 and each u_i lies in a wheel graph (a subgraph of G) containing at least three nodes before u_i in π .

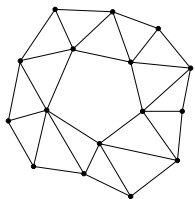


(a) Wheel graph

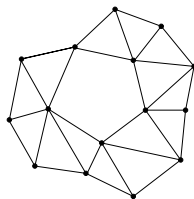


(b) Wheel extension

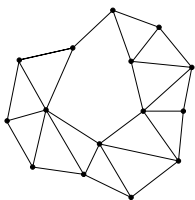
Examples of some other classes of localizable networks



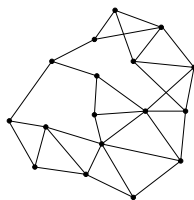
(a) Triangle cycle



(b) Triangle circuit



(c) Triangle bridge



(d) Triangle net

Outline

- 1 Network localization Problem in WSN
- 2 Techniques for network localization
- 3 Rigidity and localizability**

Shift, flip and rotation operations in \mathbb{R}^2

- A WSN may have **multiple realizations**.
- A realization of a given network may be flipped, rotated and/or translated (shifting origin), (**like any rigid body**), with respect to the coordinate system to get another realization. These are structurally identical, i.e., congruent.
- We are interested only in realizations which are **structurally different**.
- From here onwards **flip**, **rotation** and **shift** in a realization of a WSN graph, we mean a **part of the realization** is flipped, rotated or shifted, giving us a new realization while rest of the realization remains fixed.

Shift, flip and rotation operations in \mathbb{R}^2

Definition

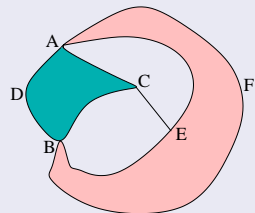
- If two globally rigid graphs share exactly one vertex, then one of them may be rotated, around the common vertex, keeping the other fixed. Such a vertex will be called a **joint**.
- If two globally rigid graphs share exactly two vertices, rotation about these vertices is no longer possible, but one of the graphs may be flipped, about the line joining the common vertices, keeping the other fixed. This pair of vertices is called a **flip**.

Lemma

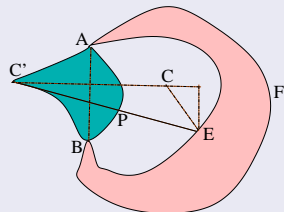
If two globally rigid bodies B_1 and B_2 , in a sensor realization of a graph $G = (V, E, w)$ share three or more common vertices, $B_1 \cup B_2$ (alongwith all edges between them) forms a rigid body.

Shift, flip and rotation operations in \mathbb{R}^2 **Lemma**

If two globally rigid subgraphs, B_1 and B_2 , of a graph G share two common vertices and there is an edge connecting a vertex in B_1 to another vertex in B_2 , $B_1 \cup B_2$ is globally rigid.



(a)

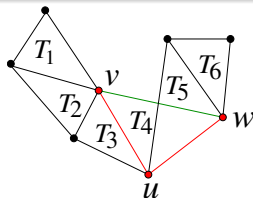


(b)

Triangle chain

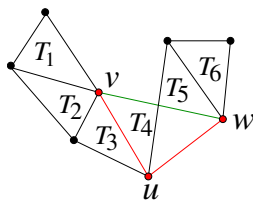
Definition

- A *triangle stream* is a sequence of distinct triangles $\mathcal{T} = (T_1, T_2, \dots, T_m)$ such that for every i , $2 \leq i \leq m - 1$, T_i shares two distinct edges with T_{i-1} and T_{i+1} . $G(\mathcal{T})$ is the union of the T_i s in \mathcal{T} .
- A node u of a triangle T_i is termed a *pendant* of T_i , if the edge opposite to u in T_i is shared by another triangle in \mathcal{T} . $T_4 = \{u, v, w\}$ has two pendants v and w .



Triangle chain

Triangle chain



Triangle chain

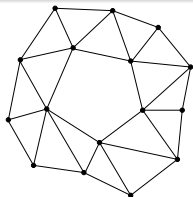
Definition

- This shared edge is called an *inner side* of T_i . Each T_i has at least one edge which is not shared by another triangle in \mathcal{T} . Such an edge is called an *outer side* of T_i . In figure, uw and uv are inner sides, and vw is an outer side.
- If T_1 and T_m have unique and distinct pendants, then $G(\mathcal{T})$ is called a *triangle chain*.
- A *triangle chain* involves only flips; hence it is *rigid*.

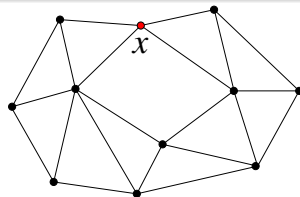
Triangle cycle, triangle circuit, triangle bridge

Definition

- If T_1 and T_m share an edge other than those shared with T_2 and T_{m-1} , then $G(\mathcal{T})$ is called a **triangle cycle**. In a triangle cycle, each triangle has exactly two inner and one outer sides. A wheel graph is a triangle cycle.
- If $G(\mathcal{T})$ is not a triangle cycle and T_1 and T_m have a unique pendant in common, then $G(\mathcal{T})$ is called a **triangle circuit**. The common pendant is called a **circuit knot** (e.g., x).



Triangle cycle

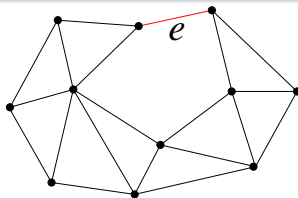


Triangle circuit

Triangle cycle, triangle circuit, triangle bridge

Definition

- Let T_1 and T_m have unique and distinct pendants. If the pendants are connected by an edge e , then $G(\mathcal{T}) \cup \{e\}$ is called a *triangle bridge*.
- The edge e is called the *bridging edge*.
- The *length of a triangle stream* \mathcal{T} is the number of triangles in it and is denoted by $l(\mathcal{T})$.

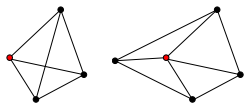


Triangle bridge

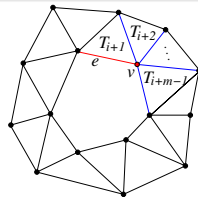
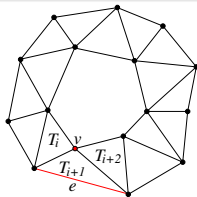
Triangle cycle, triangle circuit, triangle bridge

Lemma

Every *triangle cycle* has a *spanning wheel* or *triangle circuit* (a wheel or triangle circuit which is a spanning subgraph of the triangle cycle).



Triangle cycles
with three and
four triangles

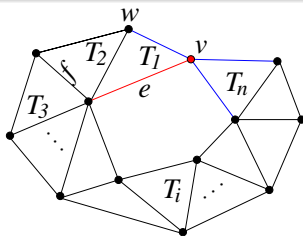


Triangle cycles without spanning wheel

Triangle cycle, triangle circuit, triangle bridge

Lemma

Every *triangle circuit* has a *spanning triangle bridge* (a triangle bridge which is a spanning subgraph of the triangle circuit).



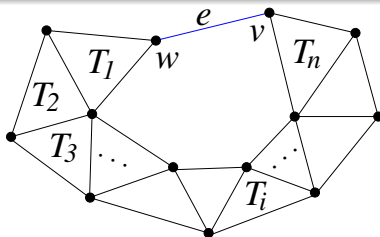
Triangle circuit \mathcal{T} gives a spanning triangle bridge $\mathcal{T} - e$

Triangle cycle, triangle circuit, triangle bridge

In \mathbb{R}^2 , if a rigid realization admits no flip ambiguity, then it is globally rigid. \Rightarrow

Lemma

Triangle cycle, circuit and bridge are generically globally rigid.



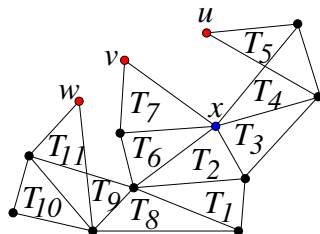
A generic configuration of a triangle bridge $G(\mathcal{T})$

Triangle tree

Definition

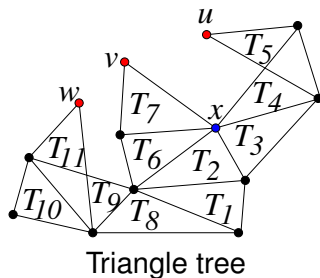
Consider a sequence $\mathcal{T} = (T_1, T_2, \dots, T_m)$ of triangles while each T_i (for $i = 2, 3, \dots, m$) shares exactly one edge with exactly one T_j , $1 \leq j < i$.

- The node opposite to this sharing edge is called a *pendant* of T_i in \mathcal{T} (e.g., x is a pendant of T_2 ; T_1 has no pendant).
- The graph $G(\mathcal{T})$ is called a *triangle tree*.



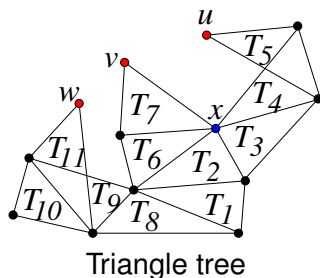
Triangle tree

Triangle tree



- For $2 \leq i \leq m$, each T_i has exactly one pendant in \mathcal{T} . If T_i shares no edge with no $T_j, j > i$, T_i is called a *leaf triangle*.
- A leaf triangle shares exactly one edge with other triangles in \mathcal{T} . It has a unique pendant, called a *leaf knot*. T_5, T_7 and T_{11} are leaf triangles and u, v and w are leaf knots.

Triangle tree



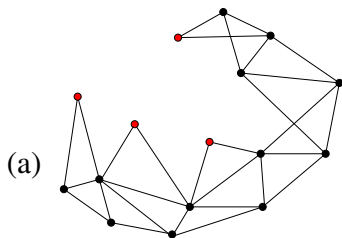
- $G(\mathcal{T})$ contains no triangle cycle. Otherwise, there always exists a T_j which shares two edges with some triangles before T_j in \mathcal{T} .
- By construction, any realization of a triangle tree is rigid.

Extended node

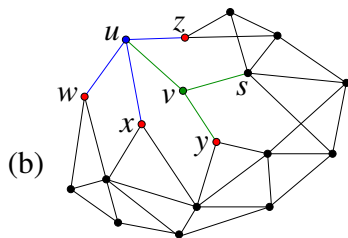
Definition

Let $G(\mathcal{T})$ be a triangle tree. A node v , outside $G(\mathcal{T})$, is called an *extended node* of $G(\mathcal{T})$, if v is adjacent (corresponding connecting edge is called an *extending edge*) to at least three nodes, each being a pendant or an extended node in $G(\mathcal{T})$

u and v are two extended nodes of $G(\mathcal{T})$. uw , ux and uz are the extending edges of u ; vu , vs and vy are those of v .



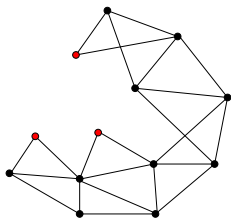
(a) Triangle tree

(b) u and v are Extended nodes

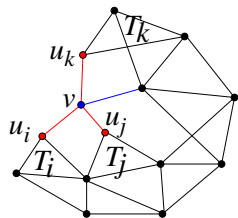
Triangle notch

Definition

A graph G is called a *triangle notch*, if it can be generated from a triangle tree $G'(\mathcal{T})$, where G' is proper subgraph of G , by adding only one extended node v where all the leaf knots of $G'(\mathcal{T})$ are adjacent to v . v is called the *apex* of G .



(a) Triangle tree $G'(\mathcal{T})$



(b) Triangle notch G with apex v

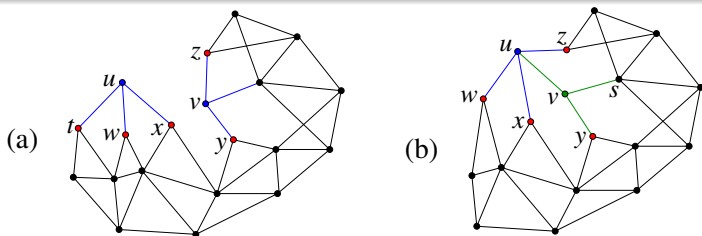
Lemma

A triangle notch is generically globally rigid.

Triangle notch

Lemma

Let G be a graph obtained from a triangle tree $G'(T)$ by adding extended nodes, where G' is a proper subgraph of G . Any extended node along with all pendants and extended nodes adjacent to it lie in a generically globally rigid subgraph.



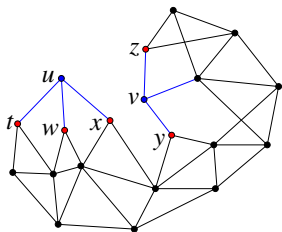
u and v are extended nodes in two scenarios: (a) u , v are adjacent to pendants only, (b) u , v are adjacent to both pendant and extended nodes

Triangle net

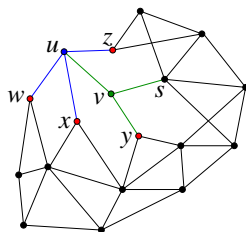
Definition

A *triangle net* is a graph G generated from a triangle tree $G'(\mathcal{T})$ by adding one or more extended nodes such that

- 1 G contains no triangle cycle, circuit or bridge; and
- 2 there exists an extended node u such that every leaf knot of $G'(\mathcal{T})$ is connected to u by a path (called *extending path*) containing only extending edges.

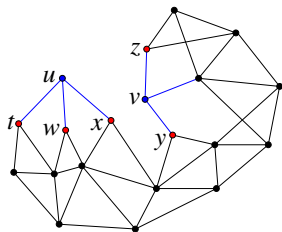


(a) Not a triangle net

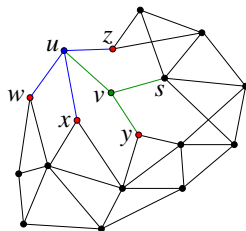


(b) A triangle net

Triangle net



(a) Not a triangle net



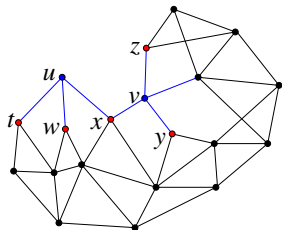
(b) A triangle net

- The last extended node added to generate the triangle net is called an *apex* of the triangle net.
- Triangle notch is a special case of triangle net.

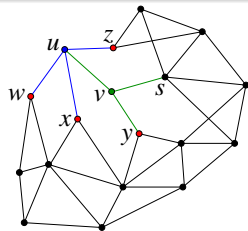
Triangle net

Lemma

A triangle net is generically globally rigid.



(a)



(b)

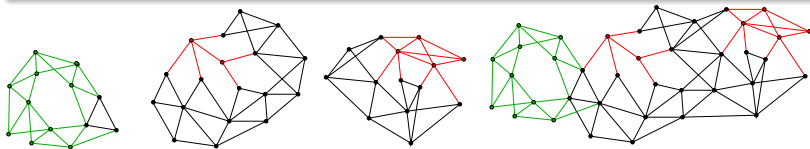
Triangle nets with extended nodes u and v where (a) u , v are adjacent to pendants only, (b) u , v are adjacent to both pendant and extended nodes

Triangle bar

Definition

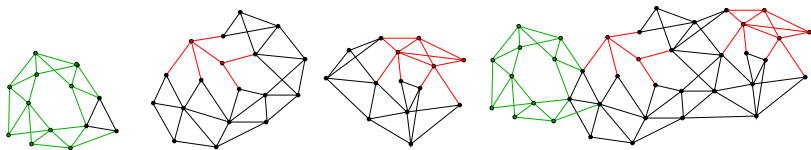
A *triangle bar* is a graph G that satisfies one of the followings:

- 1 G can be obtained from a triangle cycle, triangle circuit, triangle bridge or triangle net by adding zero or more edges, but no extra node;
- 2 $G = B_i \cup B_j$ where B_i and B_j are triangle bars which share at least three nodes; or
- 3 $G = B_i \cup \{v\}$ where B_i is a triangle bar and v is a node not in B_i , and adjacent to at least three nodes of B_i .



Examples of triangle bar: The first figure is a triangle cycle. Next two are triangle nets.

Triangle bar



Examples of triangle bar: The first figure is a triangle cycle. Next two are triangle nets.




Note:

- Triangle cycle, triangle circuit, triangle bridge and triangle net are also triangle bars.
- These triangle bars will be referred as *elementary bars*.

Theorem

Trilateration graph and wheel extension are triangle bars.

Thank you !

-  P. Biswas, K.-C. Toh, and Y. Ye.
A distributed sdp approach for large-scale noisy anchor-free graph realization with applications to molecular conformation.
SIAM J. Sci. Comput., 30(3):1251–1277, 2008.
-  H. Brey and D.G. Kirkpatrick.
Unit disk graph recognition is np-hard.
Computational Geometry, 9(1-2):3–24, 1998.
-  L. Doherty, K.S.J. Pister, and L. El Ghaoui.
Convex position estimation in wireless sensor networks.
In *Proc. of Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM 2001)*, volume 3, pages 1655–1663, Anchorage, Alaska, USA, April 2001. IEEE.
-  T. Eren, D. Goldenberg, W. Whitley, Y.R. Yang, A.S. Morse, B.D.O. Anderson, and P.N. Belhumeur.

Rigidity, computation, and randomization in network localization.

In *Proc. IEEE INFOCOM 2004*, volume 4, pages 2673–2684, 2004.



B. Hendrickson.

Conditions for unique graph realizations.

SIAM J. Comput., 21:65–84, 1992.



J.E. Hopcroft and R. Tarjan.

Dividing a graph into triconnected components.

SIAM J. Computing, 3:135–158, 1973.



B. Jackson and T. Jordán.

Connected rigidity matroids and unique realizations of graphs.

Journal of Combinatorial Theory Series B, 94(1):1–29, 2005.



G. Laman.

On graphs and rigidity of plane skeletal structures.

Journal of Engineering Mathematics, 4(4), December 1970.



G.L. Miller and V Ramachandran.

A new graph triconnectivity algorithm and its parallelization.

Combinatorica, 12:53–76, 1992.



J.B. Saxe.

Embeddability of weighted graphs in k -space is strongly np-hard.

In *Proc. 17th. Allerton Conference in Communications, Control and Computing*, page 480–489, 1979.