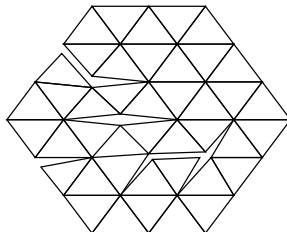
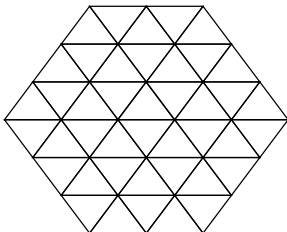


Support-Graph Preconditioners for Trusses

Samuel I. Daitch Daniel A. Spielman

Department of Computer Science
Yale University

SIAM Workshop on Combinatorial Scientific Computing
February 19, 2006



Our Result

For stiffness matrix $A_{\mathcal{T}}$ of a truss structure \mathcal{T} with angles, element lengths, and weights in constant-bounded ranges, our algorithm solves systems in $A_{\mathcal{T}}$ within relative error ϵ , in time

$$O(n^{5/4}(\log^2 n \log \log n)^{3/4} \log \frac{1}{\epsilon})$$

Our Result

For stiffness matrix $A_{\mathcal{T}}$ of a truss structure \mathcal{T} with angles, element lengths, and weights in constant-bounded ranges, our algorithm solves systems in $A_{\mathcal{T}}$ within relative error ϵ , in time

$$\tilde{O}(n^{5/4} \log \frac{1}{\epsilon})$$

- 1 Overview of Concepts
 - Support Theory
 - Trusses
 - Fretsaw Extensions
- 2 Path Lemma
- 3 Open Questions

Brief History of Support Theory

- **Vaidya**: preconditioners for diagonally-dominant matrices

Brief History of Support Theory

- **Vaidya**: preconditioners for diagonally-dominant matrices
- **Gremban Miller**
Bern Boman Chen Gilbert Hendrickson Nguyen Toledo:
further developed tools for support theory

Brief History of Support Theory

- **Vaidya**: preconditioners for diagonally-dominant matrices
- **Gremban Miller**
Bern Boman Chen Gilbert Hendrickson Nguyen Toledo: further developed tools for support theory
- **Spielman Teng '04**: nearly linear-time solver for diag.-dom.

Brief History of Support Theory

- **Vaidya**: preconditioners for diagonally-dominant matrices
- **Gremban Miller**
Bern Boman Chen Gilbert Hendrickson Nguyen Toledo:
further developed tools for support theory
- **Spielman Teng '04**: nearly linear-time solver for diag.-dom.
- **Boman Hendrickson Vavasis '04**:
nearly linear-time for elliptic finite element systems

Brief History of Support Theory

- **Vaidya**: preconditioners for diagonally-dominant matrices
- **Gremban Miller**
Bern Boman Chen Gilbert Hendrickson Nguyen Toledo: further developed tools for support theory
- **Spielman Teng '04**: nearly linear-time solver for diag.-dom.
- **Boman Hendrickson Vavasis '04**: nearly linear-time for elliptic finite element systems
- **Shklarski Toledo '06**: fretsaw extensions

1 Overview of Concepts

- Support Theory
- **Trusses**
- Fretsaw Extensions

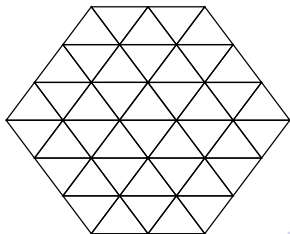
2 Path Lemma

3 Open Questions

Definition of a Truss

2-dimensional truss $\mathcal{T} = \langle n, \{\mathbf{v}_i\}_{i=1}^n, E, \gamma \rangle$

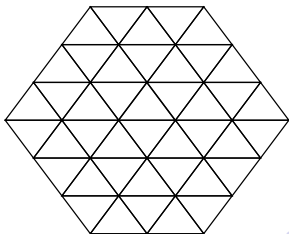
- planar graph with vertices $[n] = \{1, \dots, n\}$ and edges E
- vertex i located at $\mathbf{v}_i \in \mathbb{R}^2$



Definition of a Truss

2-dimensional truss $\mathcal{T} = \langle n, \{\mathbf{v}_i\}_{i=1}^n, E, \gamma \rangle$

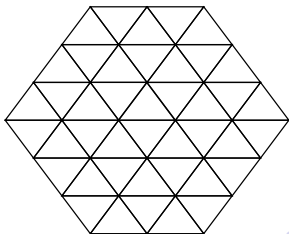
- planar graph with vertices $[n] = \{1, \dots, n\}$ and edges E
- vertex i located at $\mathbf{v}_i \in \mathbb{R}^2$
- edge $e = (i, j)$ represents bar, or **truss element**, from \mathbf{v}_i to \mathbf{v}_j
- weight $\gamma(e) > 0$ determined by bar's material and cross-section



Definition of a Truss

2-dimensional truss $\mathcal{T} = \langle n, \{\mathbf{v}_i\}_{i=1}^n, E, \gamma \rangle$

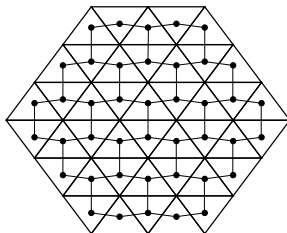
- planar graph with vertices $[n] = \{1, \dots, n\}$ and edges E
- vertex i located at $\mathbf{v}_i \in \mathbb{R}^2$
- edge $e = (i, j)$ represents bar, or **truss element**, from \mathbf{v}_i to \mathbf{v}_j
- weight $\gamma(e) > 0$ determined by bar's material and cross-section
- triangular faces of truss are called **truss faces**
- every edge must be part of a truss face



Rigidity Graph

rigidity graph Q_T

- vertices are truss faces of \mathcal{T}
- edges connect faces that share an element



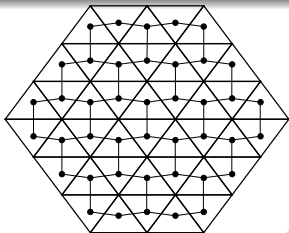
Rigidity Graph

rigidity graph $Q_{\mathcal{T}}$

- vertices are truss faces of \mathcal{T}
- edges connect faces that share an element

\mathcal{T} is stiffly-connected if

- $Q_{\mathcal{T}}$ is connected
- for every vertex i , graph induced by $Q_{\mathcal{T}}$ on the set of faces that contain i is connected



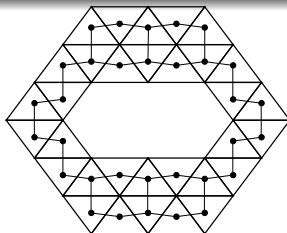
Rigidity Graph

rigidity graph $Q_{\mathcal{T}}$

- vertices are truss faces of \mathcal{T}
- edges connect faces that share an element

\mathcal{T} is stiffly-connected if

- $Q_{\mathcal{T}}$ is connected
- for every vertex i , graph induced by $Q_{\mathcal{T}}$ on the set of faces that contain i is connected



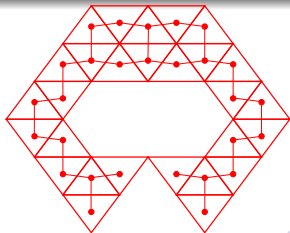
Rigidity Graph

rigidity graph $Q_{\mathcal{T}}$

- vertices are truss faces of \mathcal{T}
- edges connect faces that share an element

\mathcal{T} is stiffly-connected if

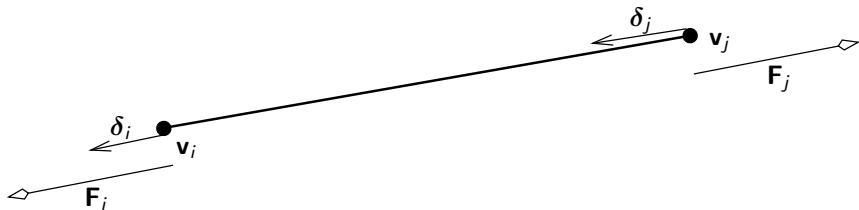
- $Q_{\mathcal{T}}$ is connected
- for every vertex i , graph induced by $Q_{\mathcal{T}}$ on the set of faces that contain i is connected



Stiffness Matrix

Consider the forces on a truss element (i, j)

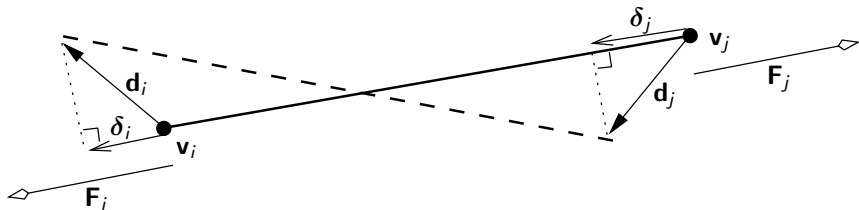
- Hooke's Law: $\mathbf{F}_i = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} (\delta_i - \delta_j) = -\mathbf{F}_j$



Stiffness Matrix

Consider the forces on a truss element (i, j)

- Hooke's Law: $\mathbf{F}_i = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} (\delta_i - \delta_j) = -\mathbf{F}_j$
- For infinitesimal displacements $\mathbf{d}_i, \mathbf{d}_j$,
we project $\delta_i = (\mathbf{u}_{ij} \mathbf{u}_{ij}^T) \mathbf{d}_i$ and $\delta_j = (\mathbf{u}_{ij} \mathbf{u}_{ij}^T) \mathbf{d}_j$
where $\mathbf{u}_{ij} = \frac{\mathbf{v}_i - \mathbf{v}_j}{|\mathbf{v}_i - \mathbf{v}_j|}$ is a unit vector parallel to the bar.

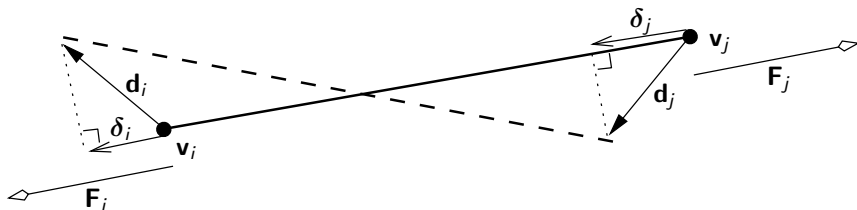


Stiffness Matrix

Consider the forces on a truss element (i, j)

- Hooke's Law: $\mathbf{F}_i = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} (\delta_i - \delta_j) = -\mathbf{F}_j$
- For infinitesimal displacements $\mathbf{d}_i, \mathbf{d}_j$, we project $\delta_i = (\mathbf{u}_{ij} \mathbf{u}_{ij}^T) \mathbf{d}_i$ and $\delta_j = (\mathbf{u}_{ij} \mathbf{u}_{ij}^T) \mathbf{d}_j$ where $\mathbf{u}_{ij} = \frac{\mathbf{v}_i - \mathbf{v}_j}{|\mathbf{v}_i - \mathbf{v}_j|}$ is a unit vector parallel to the bar.

- $$\begin{bmatrix} \mathbf{F}_i \\ \mathbf{F}_j \end{bmatrix} = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} \begin{bmatrix} \mathbf{u}_{ij} \mathbf{u}_{ij}^T & -\mathbf{u}_{ij} \mathbf{u}_{ij}^T \\ -\mathbf{u}_{ij} \mathbf{u}_{ij}^T & \mathbf{u}_{ij} \mathbf{u}_{ij}^T \end{bmatrix} \begin{bmatrix} \mathbf{d}_i \\ \mathbf{d}_j \end{bmatrix}$$



Stiffness Matrix

$$\bullet \begin{bmatrix} \mathbf{F}_i \\ \mathbf{F}_j \end{bmatrix} = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} \begin{bmatrix} \mathbf{u}_{ij}\mathbf{u}_{ij}^T & -\mathbf{u}_{ij}\mathbf{u}_{ij}^T \\ -\mathbf{u}_{ij}\mathbf{u}_{ij}^T & \mathbf{u}_{ij}\mathbf{u}_{ij}^T \end{bmatrix} \begin{bmatrix} \mathbf{d}_i \\ \mathbf{d}_j \end{bmatrix}$$

Stiffness Matrix

$$\bullet \begin{bmatrix} \mathbf{F}_i \\ \mathbf{F}_j \end{bmatrix} = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} \begin{bmatrix} \mathbf{u}_{ij}\mathbf{u}_{ij}^T & -\mathbf{u}_{ij}\mathbf{u}_{ij}^T \\ -\mathbf{u}_{ij}\mathbf{u}_{ij}^T & \mathbf{u}_{ij}\mathbf{u}_{ij}^T \end{bmatrix} \begin{bmatrix} \mathbf{d}_i \\ \mathbf{d}_j \end{bmatrix}$$

Stiffness Matrix $A_{\mathcal{T}}$

$$A_{(i,j)} = \frac{\gamma(i,j)}{|\mathbf{v}_i - \mathbf{v}_j|} \begin{bmatrix} 0 & \dots & 0 & \dots & 0 \\ \vdots & \mathbf{u}_{ij}\mathbf{u}_{ij}^T & \vdots & -\mathbf{u}_{ij}\mathbf{u}_{ij}^T & \vdots \\ 0 & \dots & 0 & \dots & 0 \\ \vdots & -\mathbf{u}_{ij}\mathbf{u}_{ij}^T & \vdots & \mathbf{u}_{ij}\mathbf{u}_{ij}^T & \vdots \\ 0 & \dots & 0 & \dots & 0 \end{bmatrix}$$

$$A_{\mathcal{T}} = \sum_{(i,j) \in E} A_{(i,j)}$$

Our Result

For stiffness matrix $A_{\mathcal{T}}$ of a truss \mathcal{T} with angles, element lengths, and weights in constant-bounded ranges we can find in time $\tilde{O}(n)$ a preconditioner B such that:

- $\kappa(A, B) = \tilde{O}(n^{1/2})$
- we can Cholesky factorize B in time $\tilde{O}(n^{5/4})$

Our Result

For stiffness matrix $A_{\mathcal{T}}$ of a truss \mathcal{T} with angles, element lengths, and weights in constant-bounded ranges we can find in time $\tilde{O}(n)$ a preconditioner B such that:

- $\kappa(A, B) = \tilde{O}(n^{1/2})$
- we can Cholesky factorize B in time $\tilde{O}(n^{5/4})$

Theorem

Our algorithm solves systems in $A_{\mathcal{T}}$ within relative error ϵ in time $\tilde{O}(n^{5/4} \log \frac{1}{\epsilon})$

- 1 Overview of Concepts
 - Support Theory
 - Trusses
 - Fretsaw Extensions

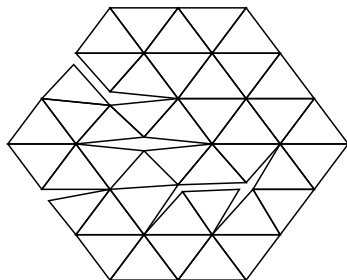
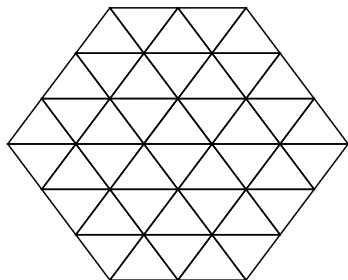
- 2 Path Lemma

- 3 Open Questions

Fretsaw Extensions

Definition (Shklarski Toledo '06)

The **fretsaw extension** of a truss \mathcal{T} is a truss \mathcal{T}' with the same faces as \mathcal{T} , but with some vertices split into multiple copies.

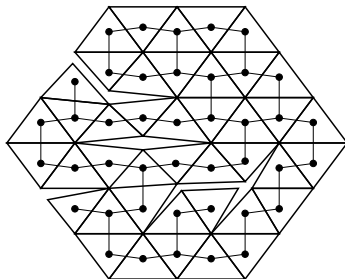
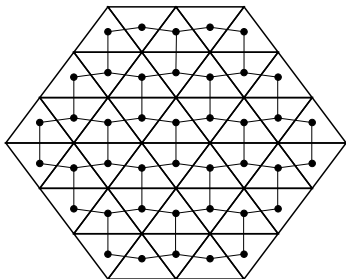


Fretsaw Extensions

Definition (Shklarski Toledo '06)

The **fretsaw extension** of a truss \mathcal{T} is a truss \mathcal{T}' with the same faces as \mathcal{T} , but with some vertices split into multiple copies.

- rigidity graph of \mathcal{T}' subgraph of rigidity graph of \mathcal{T} .

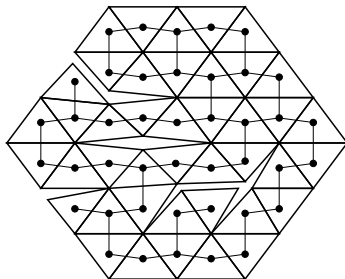
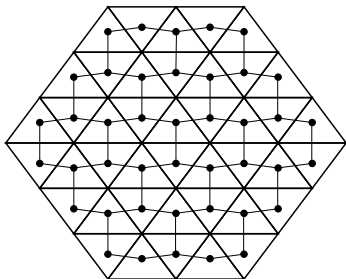


Fretsaw Extensions

Definition (Shklarski Toledo '06)

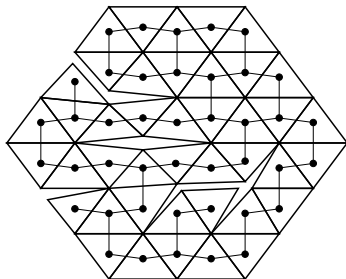
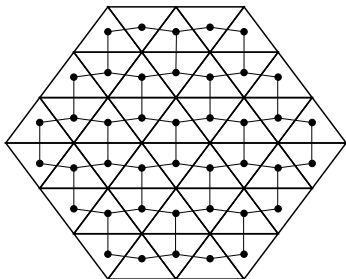
The **fretsaw extension** of a truss \mathcal{T} is a truss \mathcal{T}' with the same faces as \mathcal{T} , but with some vertices split into multiple copies.

- rigidity graph of \mathcal{T}' subgraph of rigidity graph of \mathcal{T} .
- rigidity graph close to tree \Rightarrow can be quickly Cholesky factorized



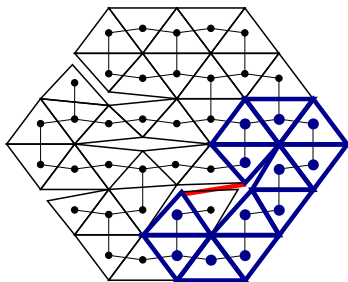
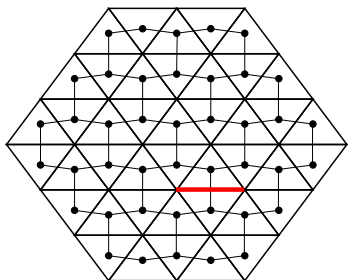
Fretsaw Extensions

- **Low-Congestion Augmented Spanning Trees** of [EEST '06, ST '06] yield good choice of fretsaw extension.



Fretsaw Extensions

- **Low-Congestion Augmented Spanning Trees** of [EEST '06, ST '06] yield good choice of fretsaw extension.
- each edge in \mathcal{T} is supported by “truss path” in \mathcal{T}' .



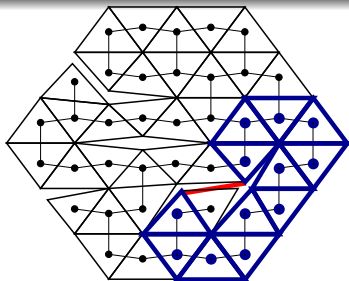
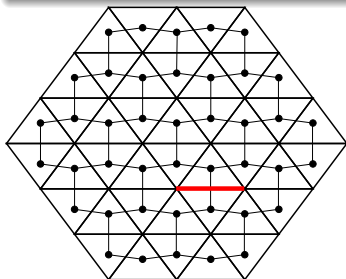
Fretsaw Extensions

Path Lemma

For truss element (i,j) and “truss path” $\mathcal{T}_{(i,j)}$ of length t connecting i and j , we have:

$$\lambda_{\max}(A_{(i,j)}, A_{\mathcal{T}_{(i,j)}}) = O(t^3)$$

provided the angles, elements lengths, and weights are in constant-bounded ranges.



- 1 Overview of Concepts
 - Support Theory
 - Trusses
 - Fretsaw Extensions

2 Path Lemma

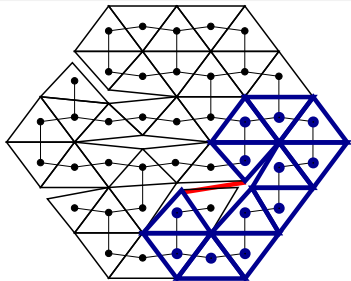
- 3 Open Questions

Path Lemma

For truss element (i,j) and “truss path” $\mathcal{T}_{(i,j)}$ of length t connecting i and j , we have:

$$\lambda_{\max}(A_{(i,j)}, A_{\mathcal{T}_{(i,j)}}) = O(t^3)$$

provided the angles, elements lengths, and weights are in constant-bounded ranges.

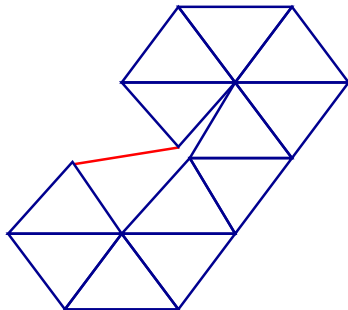


Path Lemma

For truss element (i,j) and “truss path” $\mathcal{T}_{(i,j)}$ of length t connecting i and j , we have:

$$\lambda_{\max}(A_{(i,j)}, A_{\mathcal{T}_{(i,j)}}) = O(t^3)$$

provided the angles, elements lengths, and weights are in constant-bounded ranges.

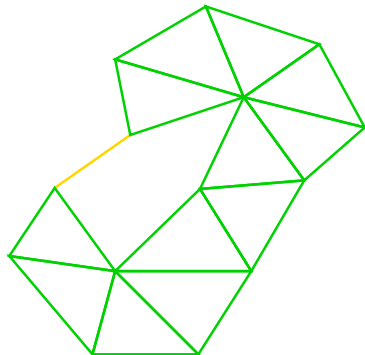
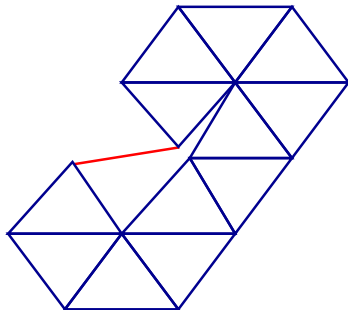


Path Lemma

For truss element (i,j) and “truss path” $\mathcal{T}_{(i,j)}$ of length t connecting i and j , we have:

$$\lambda_{\max}(A_{(i,j)}, A_{\mathcal{T}_{(i,j)}}) = O(t^3)$$

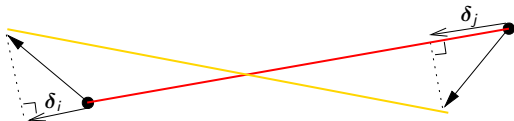
provided the angles, elements lengths, and weights are in constant-bounded ranges.



Definition

Distortion of element is change in length projected onto original axis.

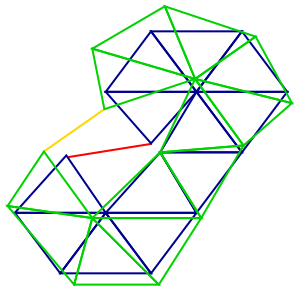
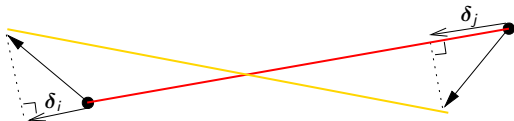
$$\text{distort}(i, j) = |\delta_i - \delta_j|:$$



Definition

Distortion of element is change in length projected onto original axis.

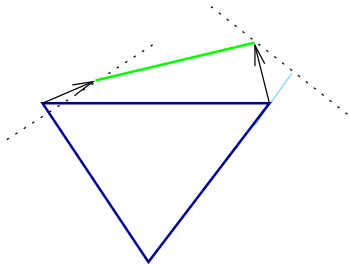
$$\text{distort}(i, j) = |\delta_i - \delta_j|:$$



$$\lambda_{\max} = \frac{\text{distort}(i, j)^2}{\sum_{e \in E} \text{distort}(e)^2}$$

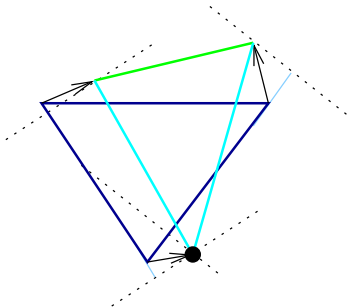
Fact

If *move edge of a triangle to any position,*



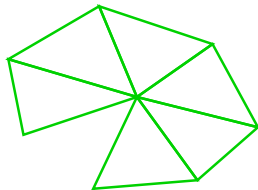
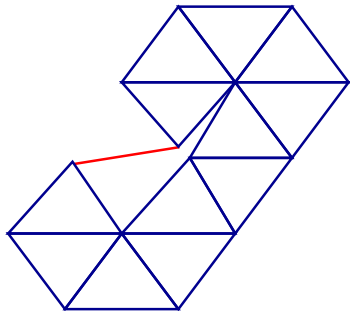
Fact

If *move edge of a triangle to any position,*
then *there is distortion free position for other two edges.*



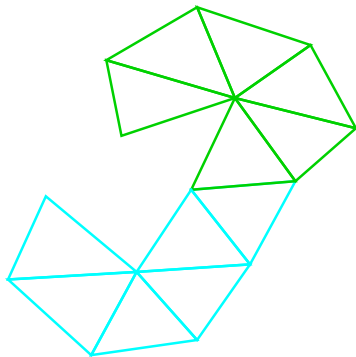
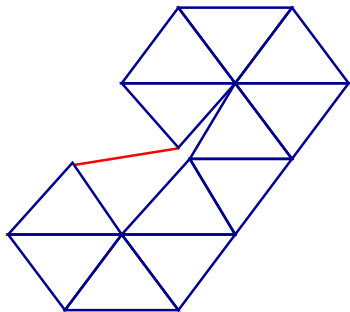
Fact

If move first k vertices to any position,



Fact

If move first k vertices to any position,
then there is distortion free position for remaining vertices.

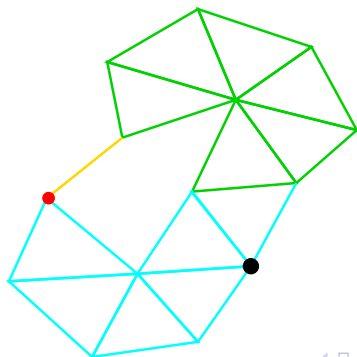


IDEA

Move vertices one by one to final position, keeping remaining edges distortion-free.

k th step:

- Start with first $k - 1$ vertices in final position.

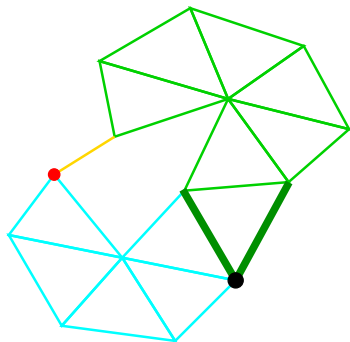


IDEA

Move vertices one by one to final position, keeping remaining edges distortion-free.

k th step:

- Start with first $k - 1$ vertices in final position.
- Move k th vertex some distance d_k to final position.

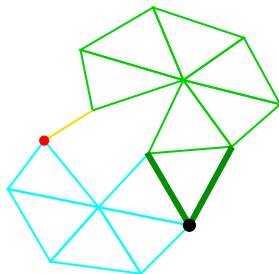
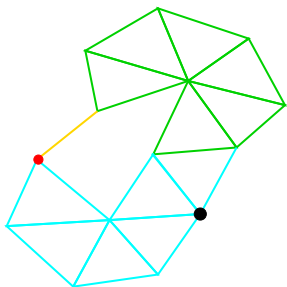


IDEA

Move vertices one by one to final position, keeping remaining edges distortion-free.

k th step:

- Start with first $k - 1$ vertices in final position.
- Move k th vertex some distance d_k to final position.

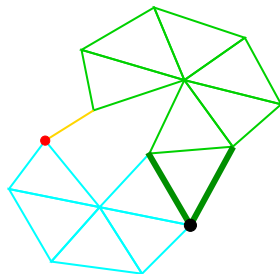
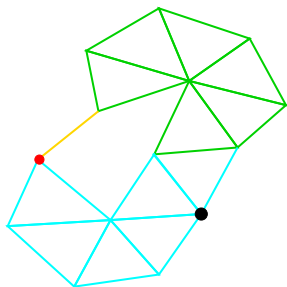


IDEA

Move vertices one by one to final position, keeping remaining edges distortion-free.

k th step:

- Start with first $k - 1$ vertices in final position.
- Move k th vertex some distance d_k to final position.
- Two green edges each get distortion $\Theta(d_k)$.

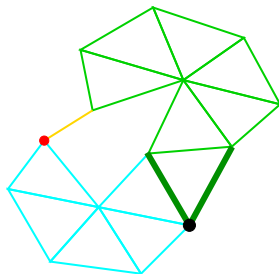
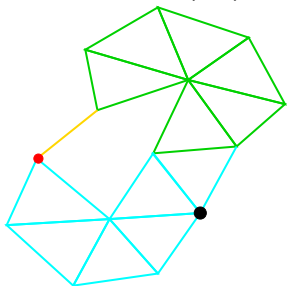


IDEA

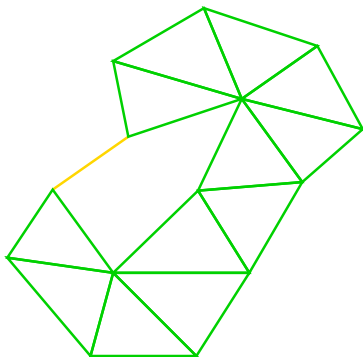
Move vertices one by one to final position, keeping remaining edges distortion-free.

k th step:

- Start with first $k - 1$ vertices in final position.
- Move k th vertex some distance d_k to final position.
- Two green edges each get distortion $\Theta(d_k)$.
- Yellow edge gets $O(td_k)$ additional distortion.



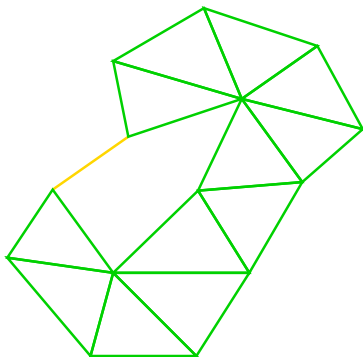
- k th pair of green edges get distortion $\Theta(d_k)$.
- Yellow edge (i, j) gets $O(td_k)$ additional distortion at step k .



- k th pair of green edges get distortion $\Theta(d_k)$.
- Yellow edge (i,j) gets $O(td_k)$ additional distortion at step k .

After all t steps:

- Total distortion on (i,j) : $O(t \sum d_k)$

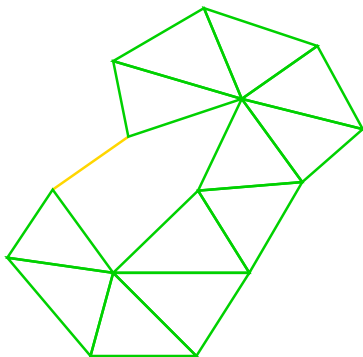


- k th pair of green edges get distortion $\Theta(d_k)$.
- Yellow edge (i, j) gets $O(td_k)$ additional distortion at step k .

After all t steps:

- Total distortion on (i, j) : $O(t \sum d_k)$

$$\lambda_{\max} = \frac{\text{distort}(i, j)^2}{\sum_{e \in E} \text{distort}(e)^2} \sim \frac{(t \sum d_k)^2}{\sum d_k^2} = O(t^3)$$



- 1 Overview of Concepts
 - Support Theory
 - Trusses
 - Fretsaw Extensions
- 2 Path Lemma
- 3 Open Questions

Open Questions

- remove assumptions, improve running time

Open Questions

- remove assumptions, improve running time
- extend to 3D trusses

Open Questions

- remove assumptions, improve running time
- extend to 3D trusses
- can this be implemented efficiently?

THANK YOU