

# Spatial Social Community

Lu Chen

Swinburne University of Technology

*luchen@swin.edu.au*

July 15, 2016

## 1 $\sqrt{e}$

- Listing all triangles in a Graph
- Time Complexity
- Truss Decomposition
- Time Complexity
- Self Analysis
- Sort edges according to their support rapidly: data structures
- Sort algorithm
- Truss Decomposition ALG with detailed datastructure

# Listing triangles

## 1 **Procedure** Tree()

- 2 | Find a rooted spanning tree for each non-trivial connected component of  $G$ ;
- 3 | If any tree edge is contained in a triangle the procedure terminates(Problem);
- 4 | Delete the tree edges from  $G$ ;

## 1 **Algorithm** Triangle()

- 2 | Repeat Tree until all edges of  $G$  are deleted

# Time Complexity

- Let  $c$  denote the number of connected components. During the execution of Triangle, the value of  $c$  increases ( $c=1$  at the initialization ).
- If  $c \leq n - e^{\frac{1}{2}}$ : each iteration of Tree causes the deletion of  $n - c$ ;  $n - c \geq n - (n - e^{\frac{1}{2}}) = e^{\frac{1}{2}}$  edges; since there are total  $e$  edges in  $G$ , the Triangle may at most call  $\frac{e}{e^{\frac{1}{2}}}$  times of Tree.
- If  $c > n - e^{\frac{1}{2}}$ : the degree of each vertex is at  $n - c$ ;  $n - c \leq n - e^{\frac{1}{2}} = e^{\frac{1}{2}}$ ; since each iteration of Tree decreases the degree of each non-isolated vertex, there may be at most  $e^{\frac{1}{2}}$  such iterations.

# Truss Decomposition

```
1  $k \leftarrow 2$ ;  
2 compute  $\text{sup}(e)$  for each edge  $e \in E_G$ ;  
3 sort all the edges in ascending order of their support;  
4 while  $\exists e$  such that  $\text{sup}(e) \leq (k - 2)$  do  
5   let  $e = (u, v)$  be the edge with lowest support;  
6   assume, w.l.o.g.,  $\text{deg}(u) \leq \text{deg}(v)$ ; for each  $w \in \text{nb}(u)$  do  
7     if  $(u, w) \in E_G$  then  
8       decrease  $\text{sup}((u, w))$  by 1;  
9       decrease  $\text{sup}((v, w))$  by 1;  
0       reorder  $(u, w)$  and  $(v, w)$  according to their new support;  
1     end  
2   end  
3    $\tau(e) = \text{sup}(e)$ ;  
4   remove  $e$  from  $G$ ;  
5 end  
6 if not all edges in  $G$  are removed then  
7   increase  $k$  by 1;
```

# Time Complexity

- Let  $nb_{\geq u}(u)$  be the neighbors of  $u$  that have degrees no less than degree of  $u$
- Prove that for any  $u \in V_G$ ,  $|nb_{\geq}(u)| \leq 2\sqrt{m}$
- If  $\deg(u) \leq \sqrt{m}$ , then  $|nb_{\geq}(u)| \leq 2\sqrt{m}$
- If  $\deg(u) > \sqrt{m}$  and suppose  $|nb_{\geq}(u)| > 2\sqrt{m}$ , then  $\sum_{u \in V_G} \deg(u) > 2E$ , which is impossible ( $\sum_{v \in nb_{\geq}(u)} \deg(v) \geq |nb_{\geq}(u)| \times \deg(u)$ ).

**Input:**  $G = (V, E)$

**Output:**  $\tau(e)$  for  $e \in E$

```
1  $k \leftarrow 2$ ;  
2 compute  $\text{sup}(e)$  for each edge  $e \in E$ ;  
3 sort all the edges in ascending order of their support;  
4 while  $\exists e$  such that  $\text{sup}(e) \leq (k - 2)$  do  
5     let  $e = (u, v)$  be the edge with the lowest support;  
6     assume, w.o.l.g,  $\text{deg}(u) \leq \text{deg}(v)$ ;  
7     for each  $w \in N(u)$  and  $(v, w) \in E$  do  
8          $\text{sup}((u, w)) \leftarrow \text{sup}((u, w)) - 1$ ;  
9          $\text{sup}((v, w)) \leftarrow \text{sup}((v, w)) - 1$ ;  
0         reorder  $(u, w)$  and  $(v, w)$  according to their new support;  
1     end  
2      $\tau(e) \leftarrow k$ , remove  $e$  from  $G$ ;  
3 end  
4 if not all edges in  $G$  are removed then  
5      $k \leftarrow k + 1$  ;  
6     goto line 4 ;  
7 end
```

## Important data structures

- an auxiliary array  $a[0...n_a]$ ,  $n_a = \text{Max}(\{\text{sup}(e) | e \in E\})$
- a array  $s[0, ..., n_s]$ ,  $n_s = |E| - 1$ , it stores all edge (reused by both input and sorted edges)
- a hash table: given a edge, it returns its position in  $s$  (this is for fast truss decomposition)



**Input:**  $E, \{sup(e) | e \in E\}$

**Output:** a permutation of  $E$ , in which edges are sorted by their support in ascending order

```
1  $n_a \leftarrow \text{Max}(\{sup(e) | e \in E\});$ 
2 for  $i = 1$  to  $n_a$  do
3   |  $a[i] \leftarrow 0;$ 
4 end
5 for  $e \in E$  do
6   |  $a[sup(e)]++;$ 
7 end
8  $l \leftarrow 0;$ 
9 for  $i = 1$  to  $n_a$  do
10  |  $t \leftarrow a[i], a[i] \leftarrow l, l \leftarrow l + t;$ 
11 end
12 let  $s$  be a array with size of  $|E|;$ 
13 for  $e \in E$  do
14   |  $s[a[sup(e)]] \leftarrow e;$ 
15   |  $a[sup(e)]++;$ 
16 end
17 return  $s;$ 
```

**Input:**  $h, a, s, E, \{sup(e) | e \in E\}$

**Output:**  $\{\tau(e) | e \in E\}$

```
1 for all edges with support of 0, directly set their trussness to be 2;
2  $k \leftarrow 3$ ;
3 for  $i \leftarrow a[0]$  to  $|E| - 1$  do
4   if  $i > a[k - 2]$  then
5      $k++$ ;
6   end
7   let  $e = (u, v)$  be  $a[i]$ ;
8    $a[sup(e)-1] ++$ ;
9   assume, w.l.o.g,  $deg(u) \leq deg(v)$ ;
10  for each  $w \in N(u)$  and  $(v, w) \in E$  do
11     $sup((u, w)) --$ ;
12     $sup((v, w)) --$ ;
13     $reOrder((u, w), reOrder((v, w))$ ;
14  end
15   $\tau(e) \leftarrow k$ ;
16  remove  $e$  from  $g$ ;
17 end
18 return  $\{\tau(e) | e \in E\}$ ;
19 Procedure  $reOrder(e)$ 
20    $p = a[sup(e) - 1]$   $e' \leftarrow s[p + 1]$ ;
21    $s[p + 1] \leftarrow e$ ;
22    $s[h(e)] \leftarrow e'$ ;
23    $h(e') \leftarrow h(e)$ ;
24    $h(e) \leftarrow p + 1$ ;
25    $a[sup(e) - 1] ++$ ;
```