Spatial Social Community

Lu Chen

Swinburne University of Technology

luchen@swin.edu.au

July 13, 2016

Overview

1. $\sqrt{e}$
   - Listing all triangles in a Graph
   - Time Complexity
   - Truss Decomposition
   - Time Complexity
   - Self Analysis
   - Sort edges according to their support rapiadly: datastructures
   - Sort algorithm
   - Truss Decomposition ALG with detailed datastructure

Listing triangles

Procedure `Tree()`

   Find a rooted spanning tree for each nontrivial connected component of G;

   If any tree edge is contained in a triangle the procedure terminates(Problem);

   Delete the tree edges from G;

Algorithm `Triangle()`

   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

$k \leftarrow 2$;
compute sup(e) for each edge $e \in E_G$;
sort all the edges in ascending order of their support;
while $\exists e$ *such that* $sup(e) \leq (k-2)$ **do**
    let $e = (u, v)$ be the edge with lowest support;
    assume, w.l.o.g., $deg(u) \leq deg(v)$; **for** *each* $w \in nb(u)$ **do**
        if $(u, w) \in E_G$ **then**
            decrease $sup((u, w))$ by 1;
            decrease $sup((v, w))$ by 1;
            reorder (u, w) and (v, w) according to their new support;
        end
    end
    $\tau(e) = sup(e)$;
    remove e from G;
end
if *not all edges in* G *are removed* **then**
    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($\sum_{v \in nb_{\geq}(u)} \geq |nb_{ge}(u)| \times deg(u)$).

Input: $G = (V, E)$
Output: $\tau(e)$ for $e \in E$

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

Important datastructures

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

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

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

```
n_a ← Max({sup(e)|e ∈ E});
for i = 1 to n_a do
    a[i] ← 0;
end
for e ∈ E do
    a[sup(e)]++;
end
l ← 0;
for i = 1 to n_a do
    t ← a[i] ,a[i] ← l, l ← l + t;
end
let s be a array with size of |E|;
for e ∈ E do
    s[a[sup(e)]] ← e;
    a[sup(e)] + +;
end
return s;
```

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

```
k ← 2;
for i ← 0 to |E| − 1 do
    if i > a[k − 2] then
        k++;
    end
    let e = (u, v) be a[i];
    assume, w.l.o.g, deg(u) ≤ deg(v);
    for each w ∈ N(u) and (v, w) ∈ E do
        p = a[sup((u, w)) − 1] e' ← s[p + 1];
        s[p + 1] ← (u, w);
        s[h((u, w))] ← e';
        h(e') ← h((u, w));
        h((u, w)) ← p + 1;
        a[sup((u, w)) − 1] + +;
        sup((u, w)) − −;
        p = a[sup((v, w)) − 1];
        e' ← s[p + 1];
        s[p + 1] ← (v, w);
        s[h((v, w))] ← e';
        h(e') ← h((v, w));
        h((v, w)) ← p + 1;
        a[sup((v, w)) − 1] + +;
        sup((v, w)) − −;
    end
    τ(e) ← k;
end
return {τ(e)|e ∈ E};
```