Exploring Truss Maintenance in Fully Dynamic Graphs: A Mixed Structure-Based Approach

Qi Luo, *Student Member, IEEE*, Dongxiao Yu, *Senior Member, IEEE*, Xiuzhen Cheng, *Fellow, IEEE*, Hao Sheng, *Senior Member, IEEE* and Weifeng Lyu, *Member, IEEE*

Introduction

- **Problem**: Truss Maintenance with multiple edge/vertex insertions/deletions

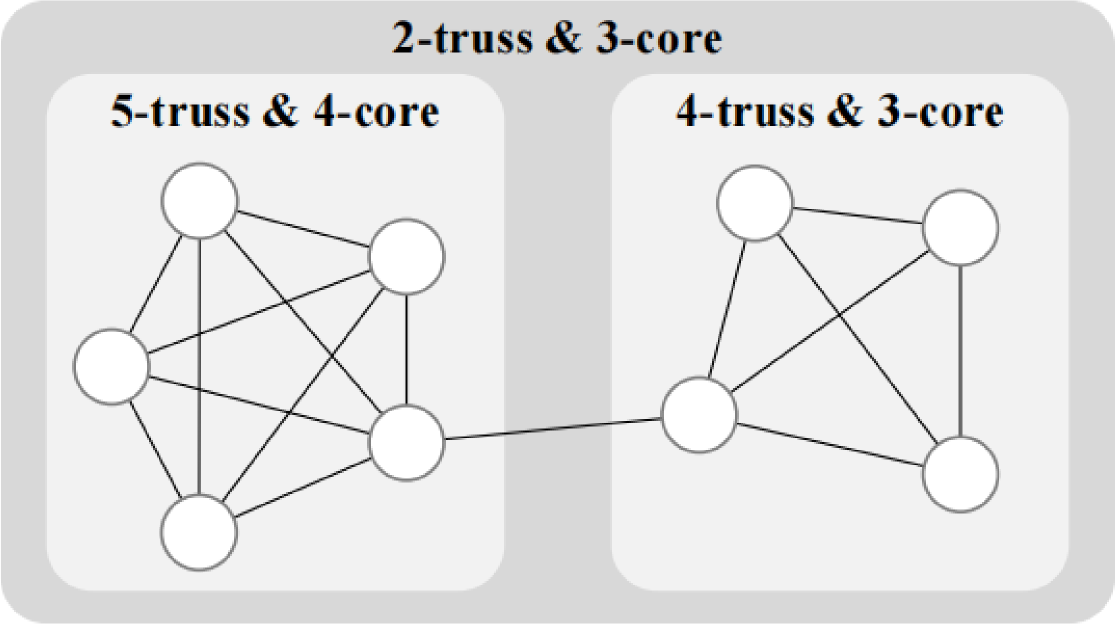


Fig. 1. An example of k-truss and k-core.

Introduction

- **Challenges**:

(1) determining which edge’s trussness will change

(2) the amount of trussness change after vertex/edge insertions/deletions

Methods

- Propose a structure called *Mixed Structure*
- Prove that the insertions/deletion of a mixed structure only makes each edge change its trussness by at most one
- Propose index called triangle support
- Admit parallel implementations

Problem Definitions

- $G = (V, E) \qquad N(v) = \{u \in V : (v, u) \in E\} \qquad d(v) = |N(v)| \qquad G[S]\ for\ S \subseteq V$
- $\triangle_{uvw} \qquad \triangle_{[u]} \qquad \triangle_{[e]}$
- $e(u, v)\ \sup\limits_{G}(e) = |\{\triangle_{uvw} : \triangle_{uvw} \in G\ \wedge\ \forall w \in V\}|\ = |N_G(u) \cap N_G(v)|$

Definition 1 (*k*-Truss). A connected subgraph H of G is a k-truss if H is maximal and satisfies the constraint that $sup_H(e) \geq k - 2$ for $\forall e \in H$.

Problem Definitions

- $\tau(H) = \min_{H}\{\sup(e) + 2 : e \in E(H)\}$ $\tau(e) = \max\{\tau(H) : e \in H, H \subset G\}$

$$
\begin{aligned}
&\tau(e) = \max\{k, 2\} \text{ s.t.} \\
&|\{w : \min\{\tau((u,w)), \tau((v,w))\} \geq k\}| \geq k-2, \qquad (1) \\
&w \in N_G(u) \cap N_G(v).
\end{aligned}
$$

Mixed Structure

- *Edge-Based triangle:* $E_{\triangle}(e) = \{e' | e' \in \triangle_{|e|}\}$
- *Triangle-Independent edge set:* $E_{TI} \subset E$ $E_{\triangle}(e_i) \cap E_{\triangle}(e_j) = \emptyset$ for any two edges $e_i, e_j \in E_{TI}$
- *Vertex-Based Triangle:* $E_{\triangle}(v) = \{e | e \in \triangle_{[v]}\}$
- *Triangle-Independent vertex set:* $V_{TI} \subset V$ $v_i, v_j \in V_{TI}$ it holds that $E_{\triangle}(v_i) \cap E_{\triangle}(v_j) = \emptyset$

Mixed Structure

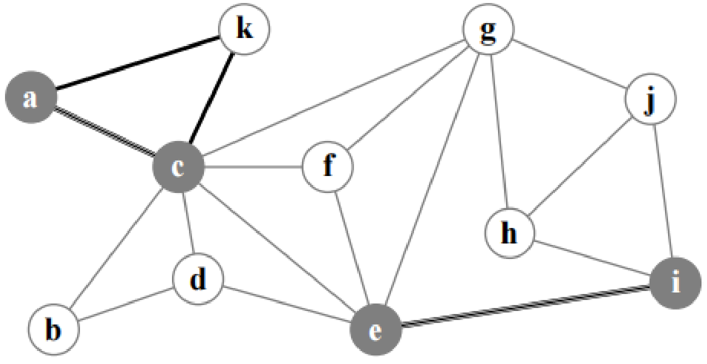
Definition 2 (Mixed Structure). Given a graph $G = (V, E)$, a mixed structure $H_{MS} = \{V_{MS}, E_{MS}\}$ of G satisfies the following requirements.

1) V_{MS} is a triangle-independent vertex set of G.
2) $E_{MS} = E_{MS1} \cap E_{MS2}$, where E_{MS1} contains all edges incident to the vertices in V_{MS}, and E_{MS2} is a triangle-independent edge set. Furthermore, for each pair of edges $e_1 \in E_{MS1}$ and $e_2 \in E_{MS2}$, $E_{\triangle}(e_1) \cap E_{\triangle}(e_2) = \emptyset$.

Mixed Structure

$E_{\Delta}((a,c)) = \{(\mathrm{a, c}), (\mathrm{a, k}), (\mathrm{c, k})\}$

$E_{\Delta}((e,i)) = \{(\mathrm{e, i})\}$

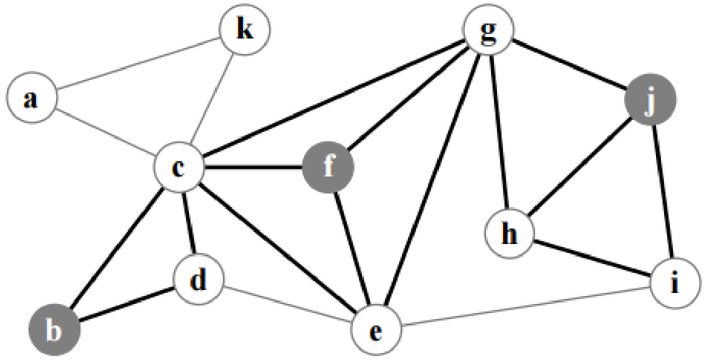


(a) Edge-based triangle edge set

$E_{\Delta}(b) = \{(\mathrm{b, c}), (\mathrm{b, d}), (\mathrm{c, d})\}$

$E_{\Delta}(f) = \{(\mathrm{f, c}), (\mathrm{f, e}), (\mathrm{f, g}), (\mathrm{c, e}), (\mathrm{c, g}), (\mathrm{e, g})\}$

$E_{\Delta}(j) = \{(\mathrm{j, g}), (\mathrm{j, h}), (\mathrm{j, i}), (\mathrm{g, h}), (\mathrm{h, i})\}$

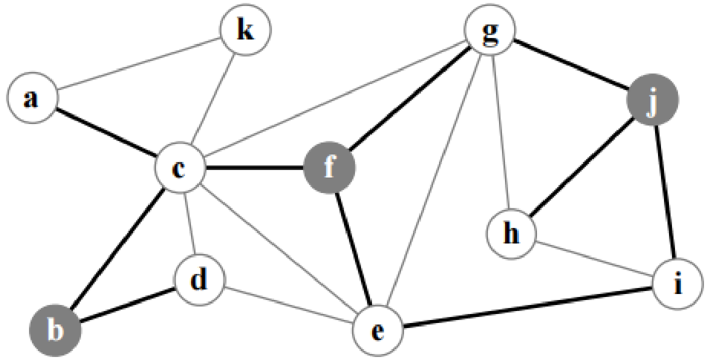


(b) Vertex-based triangle edge set

$V_{MX} = \{\mathrm{b, f, j}\}$

$E_{MX1} = \{(\mathrm{b, c}), (\mathrm{b, d}), (\mathrm{f, c}), (\mathrm{f, e}), (\mathrm{f, g}), (\mathrm{j, g}), (\mathrm{j, h}), (\mathrm{j, i})\}$

$E_{MX2} = \{(\mathrm{a, c}), (\mathrm{e, f})\}$



(c) Mixed structure

$E_{\triangle}(e)$	the edge-based triangle edge set of edge e
$E_{\triangle}(v)$	the vertex-based triangle edge set of vertex v
E_{TI}	a triangle-independent edge set
V_{TI}	a triangle-independent vertex set
H_{MS}	a mixed structure
E_{MS}	all edges in a mixed structure

Quantifying Trussness Change

Lemma 1 (Mixed Structure Deletion). Given a graph G and a mixed structure H_{MS} of G, let $G' = G \setminus E_{MS}$ be the graph obtained after deleting E_{MS} from G. Then the trussness of each edge in G' is decreased by at most one.

Quantifying Trussness Change

Lemma 2 (Mixed Structure Insertion). Given a graph G and a mixed structure H_{MS} of G, let $G' = G \cup H_{MS}$ be the graph obtained after inserting H_{MS} into G. Then the trussness of each edge in G is increased by at most one.

Quantifying Trussness Change

Definition 3 ((k, d)-Neighborhood [24]). Given a graph G, the (k, d)-neighborhood of a vertex v, denoted by $G_v^{k,d}$, is the maximal subgraph $H(V_H, E_H) \subseteq G[N(v)]$ such that

1) $\tau_G(e) \geq k, \forall e \in E_H$;
2) $deg_H(u) \geq d, \forall u \in V_H$.

Quantifying Trussness Change

Definition 4 (Pre-Truss for Edges in E_{MS1}). The pre-truss of an edge $e(v,w) \in E_{MS1}$ is defined as

$$\bar{\tau}_{G'}(e_{MS1}) = \max\{k, 2\} \text{ s.t. } \{k : w \in G_v^{k,k-2}\}. \quad (2)$$

Definition 5 (Pre-Truss for Edges in E_{MS2}). The pre-truss of an edge $e(u,v) \in E_{MS2}$ is defined as

$$\bar{\tau}_{G'}(e_{MS2}) = \max\{2, k\} \text{ s.t. } |\{w : \min\{\tau_G((u,w)), \tau_G((v,w))\} \geq k\}| \geq k-2. \quad (3)$$

Quantifying Trussness Change

Lemma 3. Given a graph G and a mixed structure H_{MS} of G‘, with $G' = G \cup H_{MS}$. For any edge $e \in E_{MS}$, it holds that

$$\bar{\tau}_{G'}(e) \leq \tau_{G'}(e) \leq \bar{\tau}_{G'}(e) + 1 \tag{4}$$

Quantifying Trussness Change

1) We prove $\bar{\tau}_{G'}(e) \leq \tau_{G'}(e)$ by considering the following two cases.
Case 1. $e \in E_{MS1}$. Let $e = (v, w)$ and $\hat{k} = \bar{\tau}_{G'}(e)$, where v, w are the endpoints of e. Because $\hat{k} = \bar{\tau}_{G'}(e) = \max_{k \geq 2}\{k : w \in G_v^{k,k-2}\}$, there exists a $\hat{k}$-truss $H_k = (V_H, E_H)$ in G. This means $\forall e' \in H_k$, $sup_{H_k}(e') \geq k - 2$. Let $H^* = (V_H \cup \{v\}, E_H \cup \{(w', v) | w' \in G_v^{k,k-2}\})$, which adds vertex v and v's incident edges (w', v) to H. By Definition 3,for each edge $(w', v) \in H^* \setminus H_k$, w' and v have at least $k-2$ common neighbors in H^*, indicating that $sup_{H^*}((v, w')) \geq k - 2$, i.e. H^* is at least a k-truss. Hence, $\tau_{G'}(e) \geq \bar{\tau}_{G'}(e)$.

Quantifying Trussness Change

Case 2. $e \in E_{MS2}$. By Definition 5, there are $k-2$ triangles containing e such that $sup_{G'}(e) \geq k-2$. All edges in these triangles, except e, have trussness not smaller than k. In other words, there exists a k-truss containing these edges except e and we denote this k-truss as H_k. Then we obtain that for $e \in H_k \cup \{e\}$, $sup_{G'}(e) \geq k-2$. Hence, $\tau_{G'}(e) \geq k = \bar{\tau}_{G'}(e)$.

$$\begin{aligned} &\bar{\tau}_{G'}(e_{MS2}) = \max\{2, k\} \text{ s.t.} \\ &|\{w : \min\{\tau_G((u,w)), \tau_G((v,w))\} \geq k\}| \geq k-2. \end{aligned} \tag{3}$$

Quantifying Trussness Change

$$\tau_{G'}(e) \leq \bar{\tau}_{G'}(e) + 1$$

2) Let $e = (v, w)$ and $\hat{k} = \bar{\tau}_{G'}(e) = \max\{k : w \in G_v^{k,k-2}\}$. Assume $\tau_{G'}(e) = \hat{k} + x$, where $x \geq 2$. Then there exists a $(\hat{k} + x)$-truss $H_{\hat{k}+x}$ containing e in G'. We delete v and all its incident edges $E(v)$ from $H_{\hat{k}+x}$, and the remaining graph is denoted as H^*. Then the trussness of each edge in H^* is decreased by at most 1 as per Lemma 1. Hence, for any edge $e* \in H^*$, it satisfies $\tau(e^*) \geq \hat{k} + x - 1 \geq \hat{k} + 1$, i.e., $\max\{k : w \in G_v^{k,k-2}\} \geq \hat{k} + 1$ before v is deleted, which contradicts with the given condition that $\hat{k} = \{k : w \in G_v^{k,k-2}\}$.

Scoping Edge Traversal

Definition 6 ***(k-Triangle).*** Given a triangle $\triangle_{uvw} \subset G$, if all edges in $\triangle_{uvw}$ have trussness of at least k, i.e., $\min\{\tau((u,v)), \tau((v,w)), \tau((u,w))\} = k$, $\triangle_{uvw}$ is called a k-triangle, denoted as $\triangle^k_{uvw}$.

- **K-triangle support of edge e(u, v) :**

$$S(e) = |\{\triangle^k_{[e]} : \tau(e) = k,\ \triangle^k_{[e]} \in G\}|$$

Scoping Edge Traversal

***Lemma* 4.** Let G' be the graph obtained after inserting a mix structure H_{MS} into graph G, then the trussness of an edge $e \in G'$ would not increase if $S(e) \leq \tau_G(e) - 2$.

***Lemma* 5.** Let G' be the graph obtained after deleting a mix structure H_{MS} from graph G, then the trussness of an edge $e \in G'$ would decrease by one if $S(e) < \tau_G(e) - 2$.

Scoping Edge Traversal

Definition 7 (k-Triangle Connectivity). Given two k-triangles $\triangle^{(s)}$ and $\triangle^{(t)}$ in G, they are k-triangle connected, denoted as $\triangle^{(s)} \overset{\triangle}{\leftrightarrow} \triangle^{(t)}$, if there exists a sequence of $n \geq 2$ k-triangles $\triangle^{(1)}, \cdots, \triangle^{(n)}$ s.t. $\triangle^{(s)} = \triangle^{(1)}$, $\triangle^{(t)} = \triangle^{(n)}$, and for $1 \leq i \leq n$, $\triangle^{(i)} \cap \triangle^{(i+1)} = \{e|e \in E_G\}$ and $\tau(e) = k$. Analogously, we say two edges e,$e' \in E$ are k-triangle connected, denoted as $e \overset{k}{\leftrightarrow} e'$, if and only if (1) e and e' belong to the same k-triangle, or (2) $e \in \triangle^{(s)}$, $e' \in \triangle^{(t)}$, s.t. $\triangle^{(s)} \overset{\triangle}{\leftrightarrow} \triangle^{(t)}$.

Scoping Edge Traversal

Lemma 6 (Insertion Propagation). Let G' be the graph obtained after inserting a mixed structure $H_{MS} = (V_{MS}, E_{MS})$ into the graph $G = (V, E)$. An edge e may increase its trussness only if it satisfies one of the following conditions:

1) e is in a triangle with an edge $\hat{e} \in E_{MS}$ and $\tau_G(e) \leq \bar{\tau}(\hat{e})$;
2) e is k-triangle connected with an edge e' satisfying Condition 1), where $\tau_G(e) = k$.

Scoping Edge Traversal

Case 1. e^* is in a triangle with an edge $\hat{e} \in E_{MS}$, but $\tau(e^*) > \bar{\tau}(\hat{e})$;

For *Case 1*, let $\hat{k} = \bar{\tau}(\hat{e})$ and $\tau_G(e^*) = \hat{k} + x$, where $x \geq 1$. By Lemma 2, the trussness of every edge can increase by at most one. We assume that $\tau_{G'}(e^*) = \hat{k} + x + 1$. Then there would be at least $\hat{k} + x - 1$ triangles in which the trussness of every edge is not smaller than $\hat{k} + x + 1$. However, the trussness of $\hat{e}$ can be at most $\hat{k}+1$ by Lemma 3. The contradiction implies that the trussness of e^* cannot increase.

Scoping Edge Traversal

Case 2. e^* is not in a triangle with any edge in E_{MS}, and is not k-triangle connected with any edge $\hat{e}$ that satisfies condition 1.

For *Case* 2, suppose that $\hat{k} = \tau_G(e^*)$. We assume the trussness of e^* increases by one, i.e., $\tau_{G'}(e^*) = \hat{k} + 1$. If an edge has its trussness increased, it must hold that either there is a new triangle (increased support) containing it, or in one of the triangles $\triangle^{\hat{k}}_{[e^*]}$, there is an edge e_1 other than e^* whose trussness changes from $\hat{k}$ to $\hat{k} + 1$.

Scoping Edge Traversal

Lemma 7 (Deletion Propagation). Let G' be the graph obtained after deleting a mixed structure $H_{MS} = (V_{MS}, E_{MS})$ from graph $G = (V, E)$. An edge e may decrease the trussness only if e satisfies one of the following conditions:

1) e is in a triangle with an edge $\hat{e} \in E_{MS}$ and $\tau_G(e) \leq \tau(\hat{e})$;
2) e is k-triangle connected with an edge e' satisfying 1), where $\tau_G(e') = k$.

Incremental Truss Maintenance

Algorithm 1 Incremental Truss Maintenance

Input: $G = (V, E)$, $\{\tau(e) | e \in E\}$, $H = \{\Delta V, \Delta E\}$
Output: $\{\tau(e) | e \in E \cup \Delta E\}$

while $\Delta E \neq \emptyset$ **do**
  $E_{\triangle}(MS) \leftarrow \emptyset$; ▷ triangle edges
  $V_{MS} \leftarrow \emptyset$; ▷ vertices in mixed structure
  while $\Delta V \neq \emptyset$ **do**
    for $v \in \Delta V$ **do**
      if $E_{\triangle}(MS) \cap E_{\triangle}(v) = \emptyset$ **then**
        $V_{MS}.add(v)$;
        $E_{\triangle}(MS).add(E_{\triangle}(v))$;
  $E_{MS1} \leftarrow \bigcup_{v \in V_{MS}} E_G(v)$;
  $\Delta E \leftarrow \Delta E \setminus E_{MS1}$; ▷ update ΔE
  $E_{MS2} \leftarrow \emptyset$;
  for $e \in \Delta E$ **do**
    if $E_{\triangle}(e) \cap E_{\triangle}(MS) = \emptyset$ **then**
      $E_{MS2}.add(e)$;
      $E_{\triangle}(MS).add(E_{\triangle}(e))$;
  $\Delta V \leftarrow \Delta V \setminus V_{MS}$; ▷ update ΔV
  $\Delta E \leftarrow \Delta E \setminus E_{MS2}$; ▷ update ΔE
  $G \leftarrow (V \cup V_{MS}, E \cup E_{MS1} \cup E_{MS2})$;
  $M \leftarrow \emptyset$; ▷ a map stores same trussness of edges
  for $e_0 = (u, v) \in E_{MS1} \cup E_{MS2}$ **do**
    compute pre-$\tau(e_0)$;
    $\tau(e_0) \leftarrow$ pre-$\tau(e_0)$;
    for $w \in N(u) \cap N(v)$ **do**
      $k = \min\{\tau((v, w)), \tau((u, w))\}$;
      if $k \leq \tau(e_0)$ **then**
        if $\tau(e) = k$, $e \in \{(u, w), (v, w)\}$ **then**
          $M[k].add(e)$;
  IncrementalTraversal(G, τ, M) ;

Incremental Truss Maintenance

Algorithm 2 IncrementalTraversal(G, τ, M)

for $k = \max(M.keys())$ decrease to $\min(M.keys())$ **do**
  $Q \leftarrow \emptyset$; $Q.push(M[k])$;
  while $Q \neq \emptyset$ **do**
    $(x, y) \leftarrow Q.pop()$;
    $S((x, y)) \leftarrow 0$; ▷ number of support triangles
    for $z \in N(x) \cup N(y)$ **do**
      if $\min\{\tau((x, z)), \tau((y, z))\} < k$ **then continue**;
      $S((x, y)) \leftarrow S((x, y)) + 1$;
      if $\tau((z, x)) = k$ **and** $(z, x) \notin M[k]$ **then**
        $Q.push((z, x))$; $M[k].add((z, x))$;
      if $\tau((z, y)) = k$ **and** $(z, y) \notin M[k]$ **then**
        $Q.push((z, y))$; $M[k].add((z, y))$;
  while $\exists S((x, y)) \leq k - 2$ in $M[k]$ **do**
    $M[k].remove((x, y))$;
    for $z \in N(x) \cap N(y)$ **do**
      if $\min\{\tau((x, z)), \tau((y, z))\} < k$ **then continue** ;
      if $\tau(e \in \{(z, x), (z, y)\}) = k$ **and** $e \notin M[k]$ **then**
        continue ;
      if $(e \in \{(z, x), (z, y)\}) \in M[k]$ **then**
        $S(e) \leftarrow S(e) - 1$;
  for $e \in M[k]$ **do**
    $\tau(e) \leftarrow k + 1$; ▷ update trussness

Lemma 4. Let G' be the graph obtained after inserting a mix structure H_{MS} into graph G, then the trussness of an edge $e \in G'$ would not increase if $S(e) \leq \tau_G(e) - 2$.

Incremental Truss Maintenance

Theorem 1. After inserting a set of vertices ΔV and edges ΔE into G, Algorithm 1 can correctly update the trussness of edges in $O(\mathcal{R}_+ \cdot (d^2_{max} \cdot (|\Delta V| + |\Delta E|) + \mathcal{A}_{max} \cdot (d^2_{max} + \mathcal{P}_{max})))$ time.

$$\mathcal{R}_+ = \max_{T \subseteq H} \max_{v \in \Delta V, e \in \Delta E} \{|V^v_\triangle(T)| + |E^e_\triangle(T)|\}.$$

$$\mathcal{A}_{max} = \max_{T \subseteq H, T' \subseteq H \setminus T} |A^T_{T'}|.$$

$$\mathcal{P}_{max} = \max_{T \subseteq H, T' \subseteq H \setminus T, e \in G_{T'}} P^T_{T'}(e).$$

$$P^T_{T'}(e) = S(e) - \tau_{G_{T'}}(e) + 2$$

Decremental Truss Maintenance

Algorithm 3 Decremental Truss Maintenance

Input: $G = (V, E)$, $\{\tau(e) | e \in E\}$, $H = \{\Delta V, \Delta E\}$
Output: $\{\tau(e) | e \in E \setminus \Delta E\}$

while $\Delta E \neq \emptyset$ **do**
  $E_{\triangle}(MS) \leftarrow \emptyset$;
  $V_{MS} \leftarrow \emptyset$;
  while $\Delta V \neq \emptyset$ **do**
    for $v \in \Delta V$ **do**
      if $E_{\triangle}(v) \cap E_{\triangle}(MS) \neq \emptyset$ **then**
        $V_{MS}.add(v)$;
        $E_{\triangle}(MS).add(E_{\triangle}(v))$;
  $E_{MS1} \leftarrow \bigcup_{v \in V_{MS}} E_G(v)$;
  $\Delta E \leftarrow \Delta E \setminus E_{MS1}$; ▷ update ΔE
  $E_{MS2} \leftarrow \emptyset$;
  for $e \in \Delta E$ **do**
    if $E_{\triangle}(e) \cap E_{\triangle}(MS) = \emptyset$ **then**
      $E_{MS2}.add(e)$;
      $E_{\triangle}(MS).add(E_{\triangle}(e))$;
  $\Delta V \leftarrow \Delta V \setminus V_{MS}$; ▷ update ΔV
  $\Delta E \leftarrow \Delta E \setminus E_{MS2}$; ▷ update ΔE
  $G \leftarrow (V \setminus V_{MS}, E \setminus (E_{MS1} \cup E_{MS2}))$;
  $M \leftarrow \emptyset$; ▷ a map stores same trussness of edges
  for $e_0 = (u, v) \in E_{MS1} \cup E_{MS2}$ **do**
    for $w \in N(u) \cap N(v)$ **do**
      $k = \min\{\tau((v, w)), \tau((u, w))\}$;
      if $k \leq \tau(e_0)$ **then**
        if $\tau(e) = k$, $e \in \{(u, w), (v, w)\}$ **then**
          $M[k].add(e)$;
  DecrementalTraversal(G, τ, M) ;

Decremental Truss Maintenance

Algorithm 4 DecrementalTraversal(G, τ, M)

for $k = \min(M.keys())$ decrease to $\max(M.keys())$ **do**
  $Q \leftarrow \emptyset$; $Q.push(M[k])$;
  while $Q \neq \emptyset$ **do**
    $(x, y) \leftarrow Q.pop()$;
    $S((x, y)) \leftarrow 0$; ▷ number of support triangles
    for $z \in N(x) \cup N(y)$ **do**
      if $\min\{\tau((x, z)), \tau((y, z))\} \geq k$ **then**
        $S((x, y)) \leftarrow S((x, y)) + 1$;
      if $\tau((z, x)) = k$ **and** $(z, x) \notin M[k]$ **then**
        $Q.push((z, x))$; $M[k].add((z, x))$;
      if $\tau((z, y)) = k$ **and** $(z, y) \notin M[k]$ **then**
        $Q.push((z, y))$; $M[k].add((z, y))$;
  while $\exists S((x, y)) < k - 2$ in $M[k]$ **do**
    $\tau(x, y) \leftarrow k - 1$; ▷ update trussness
    for $z \in N(x) \cap N(y)$ **do**
      if $\min\{\tau((x, z)), \tau((y, z))\} < k$ **then continue** ;
      if $\tau(e \in \{(z, x), (z, y)\}) = k$ **and** $e \notin M[k]$ **then**
        continue ;
      if $(e \in \{(z, x), (z, y)\}) \in M[k]$ **then**
        $S(e) \leftarrow S(e) - 1$;

Parallel Implementations

Algorithm 5 ParallelIncrementalTraversal(G, τ, M)

for each k **in** $M.keys()$ **in parallel do**
  $Q.push(M[k])$; ▷ a stack stores edges
  $V \leftarrow \emptyset$; ▷ visited edges
  $TS \leftarrow \emptyset$; ▷ triangle support
  while $Q \neq \emptyset$ **do**
    $(x, y) \leftarrow Q.pop()$;
    if $V.contains((x, y))$ **then continue**;
    $V.add((x, y))$;
    for each $z \in N(x) \cap N(y)$ **do**
      $t \leftarrow \min\{\tau((x, z)), \tau((y, z))\}$;
      if $(t > k)$ **or** $(\tau(e') = k$ **and** $TS(e') > k - 2$ for $e' \in \{(x, z), (y, z)\})$ **then**
        if $TS((x, y)) = null$ **then**
          $TS((x, y)) = 1$;
        else
          $TS((x, y)) \leftarrow S((x, y)) + 1$;
        if $\tau((x, z)) = k$ and $!V.contains((x, z))$ **then**
          $Q.push((x, z))$;
        if $\tau((y, z)) = k$ and $!V.contains((y, z))$ **then**
          $Q.push((y, z))$;
  while $\exists TS((x, y)) \leq k - 2 \in E_k$ **do**
    $E_k.remove((x, y))$;
    for $z \in N(x) \cup N(y)$ **do**
      if $\min\{\tau((x, z)), \tau((y, z))\} < k$ **then continue** ;
      if $\tau(e \in \{(z, x), (z, y)\}) = k$ **and** $e \notin E_k$ **then**
        continue ;
      if $(x, z) \in E_k$ **then**
        $TS((x, z)) \leftarrow TS((x, z)) - 1$;
      if $(y, z) \in E_k$ **then**
        $TS((y, z)) \leftarrow TS((y, z)) - 1$;
  for $e \in E_k$ **do**
    $\tau(e) \leftarrow k + 1$; ▷ update trussness

Parallel Implementations

Algorithm 6 ParallelDecrementalTraversal($G, \{\tau(e) | e \in E \cup E_{MS}\}, M$)

for each k in $M.keys()$ in parallel **do**
    $Q.push(M[k]);$ ▷ a stack stores edges
    $V \leftarrow \emptyset;$ ▷ visited edges
    $S \leftarrow \emptyset;$ ▷ triangle support of edges
    while $Q \neq \emptyset$ **do**
        $(x, y) \leftarrow Q.pop();$
        if $V.contains((x, y))$ **then continue**;
        $V.add((x, y));$
        for each $z \in N(x) \cap N(y)$ **do**
            if $\min\{\tau((x, z)), \tau((y, z))\} \geq k$ **then**
                if $S((x, y)) = null$ **then**
                    $S((x, y)) = 1;$
                else
                    $S((x, y)) \leftarrow S((x, y)) + 1;$
                if $\tau((x, z)) = k$ and $!V.contains((x, z))$ **then**
                    $Q.push((x, z));$
                if $\tau((y, z)) = k$ and $!V.contains((y, z))$ **then**
                    $Q.push((y, z));$
    $E_k \leftarrow S.keys();$ ▷ candidate edges
    while $\exists S((x, y)) < k - 2$ in E_k **do**
        $E_k.remove((x, y));$
        $\tau(e) \leftarrow k - 1;$ ▷ update trussness
        for $z \in N(x) \cup N(y)$ **do**
            if $\min\{\tau((x, z)), \tau((y, z))\} < k$ **then continue** ;
            if $\tau(e \in \{(z, x), (z, y)\}) = k$ **and** $e \notin E_k$ **then**
                continue ;
            if $(x, z) \in E_k$ **then**
                $S((x, z)) \leftarrow S((x, z)) - 1;$
            if $(y, z) \in E_k$ **then**
                $S((y, z)) \leftarrow S((y, z)) - 1;$

Datasets

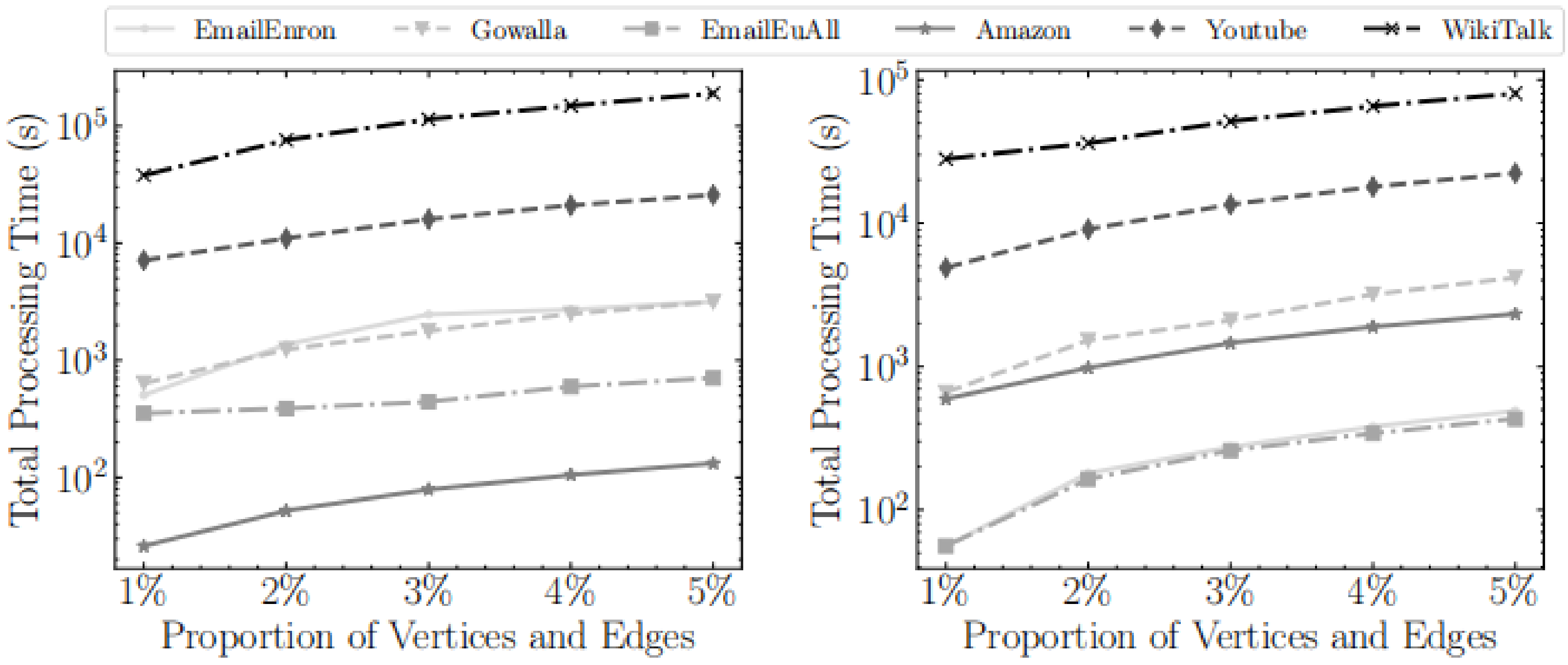
TABLE 2
Statistic of Real-world Graphs

Dataset	$\|V\|$	$\|E\|$	$\|\triangle\|$	$Truss_{max}$
EmailEnron	36K	183K	727K	423
Gowalla	196K	950K	2273K	1300
EmailEuAll	265K	420K	267K	723
Amazon	334K	952K	667K	164
YouTube	1135K	2988K	3056K	4037
WikiTalk	2394K	5021K	9203K	1634

TABLE 3
Statistic of Temporal Graphs

Dataset	$\|V\|$	Static Edges	Temporal Edges
SuperUser	197K	1.44M	924.9K
WikiTalk	1.14K	7.8M	3.3M
Overflow	2.6M	63.5M	36.2M

Performance Evaluation



(a) Insertion (b) Deletion

Fig. 3. The processing time of our truss maintenance algorithms at different scales to update vertices and edges.

Performance Evaluation

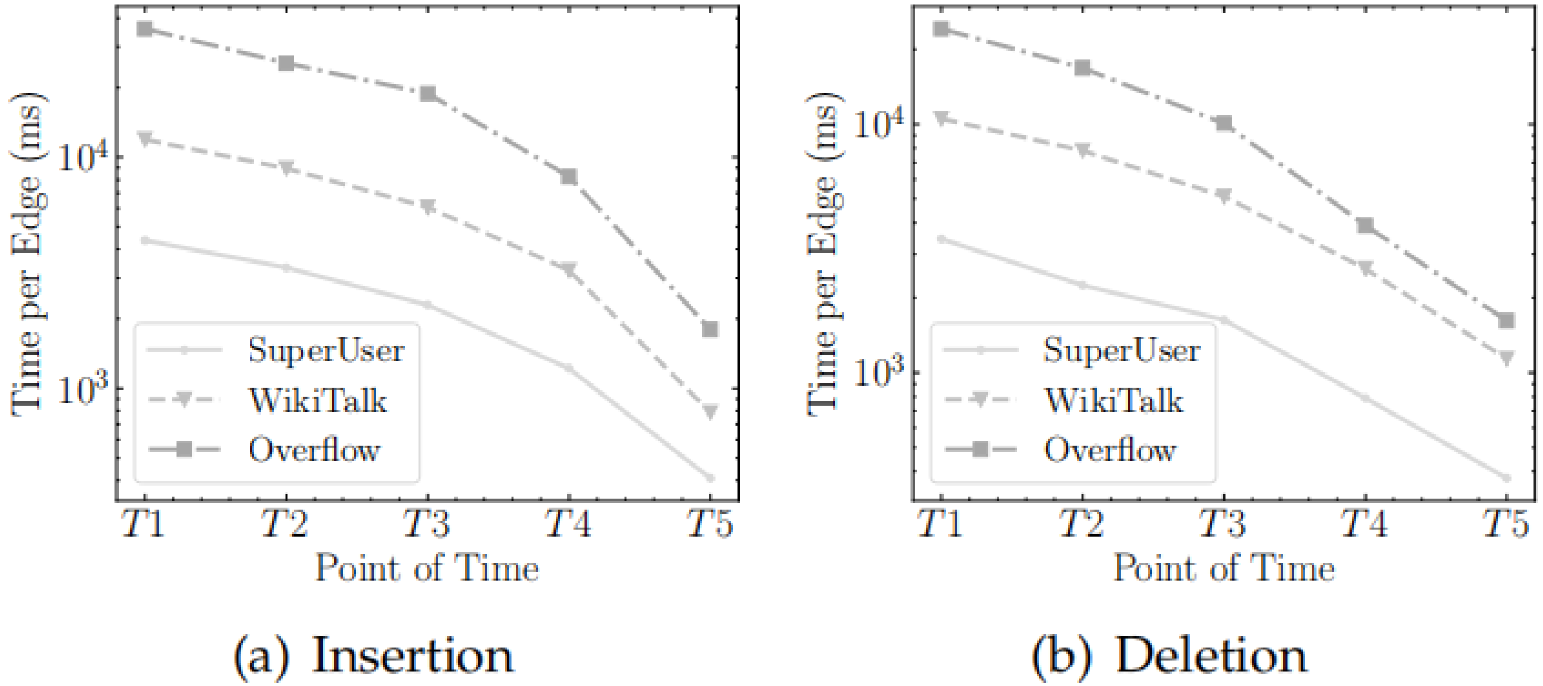
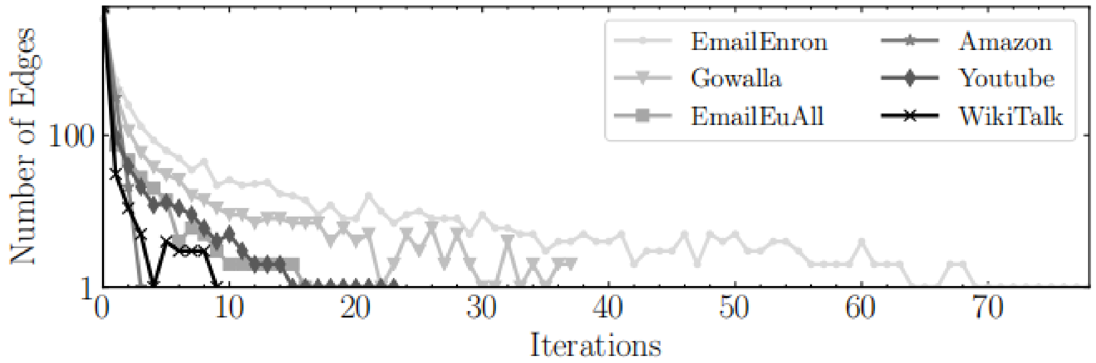
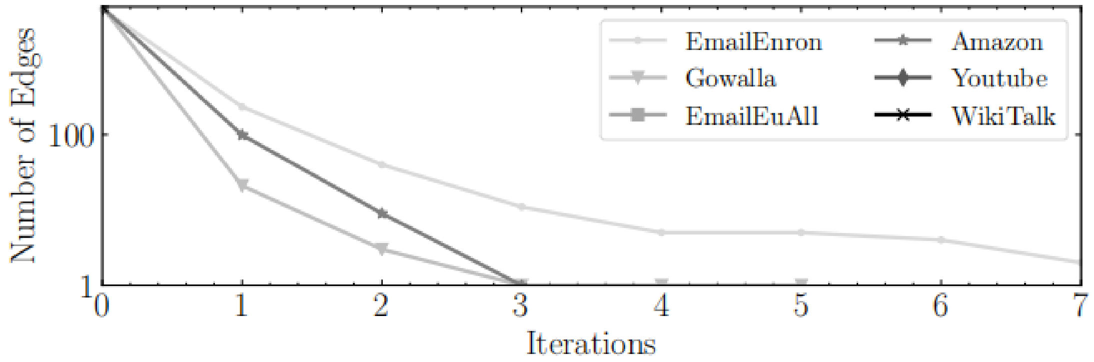


Fig. 4. Performance in temporal graphs.

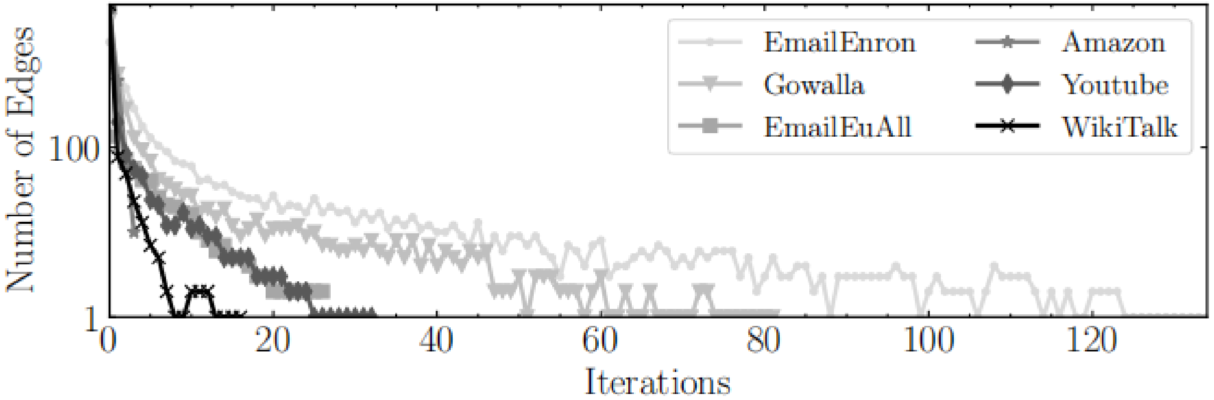
Performance Evaluation



(a) Average degree vertices



(b) Low degree vertices



(c) High degree vertices

Fig. 5. The number of edges in the mixed structures at iterations.

Performance Evaluation

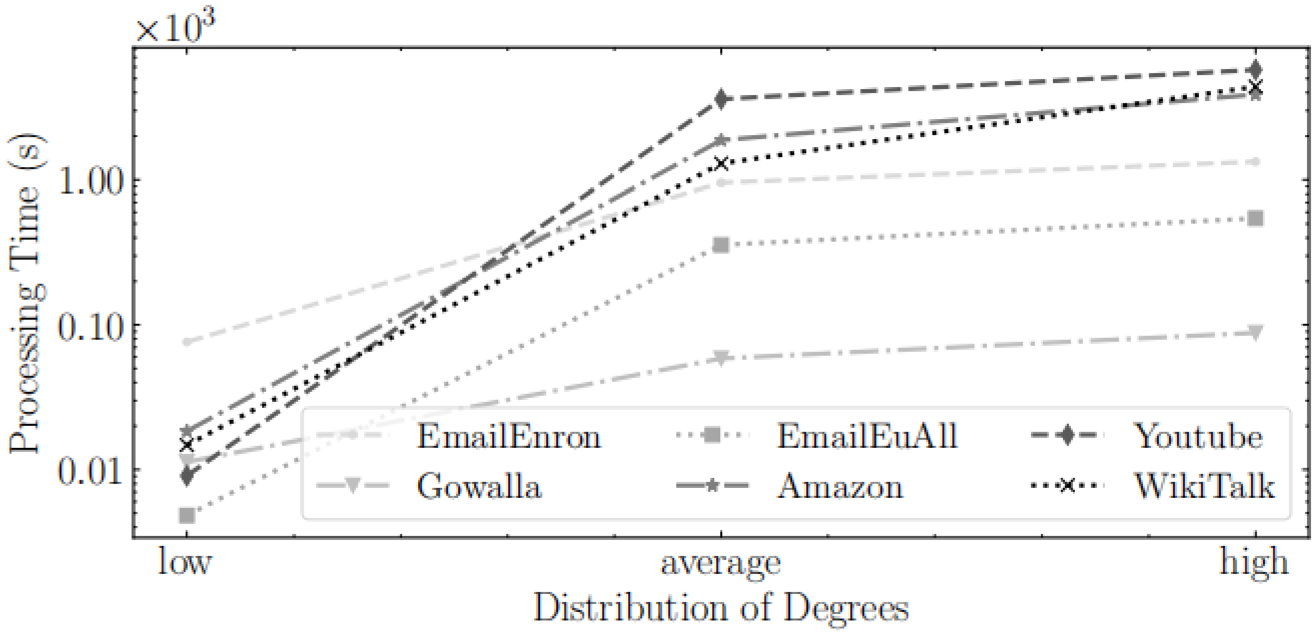
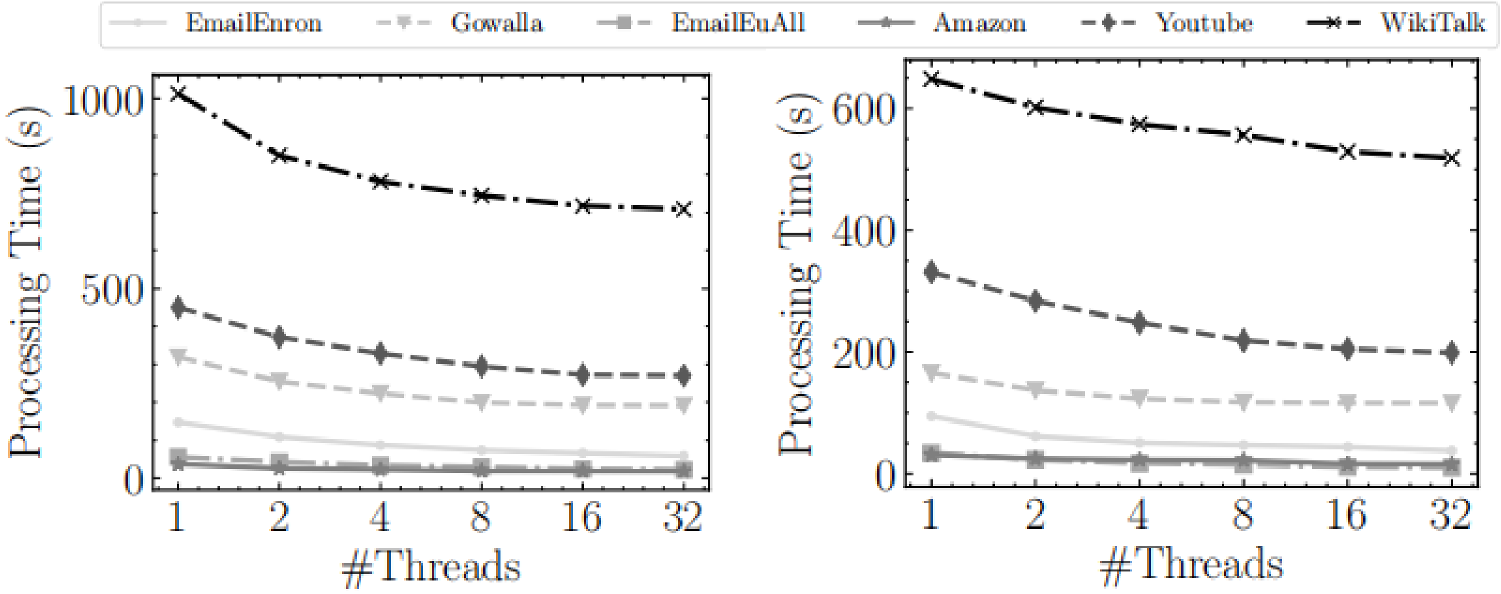


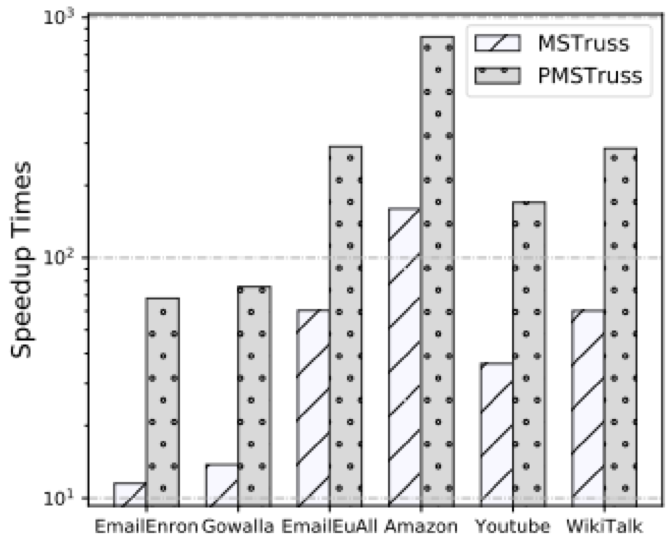
Fig. 6. The processing time under different degree distributions.

Performance Evaluation

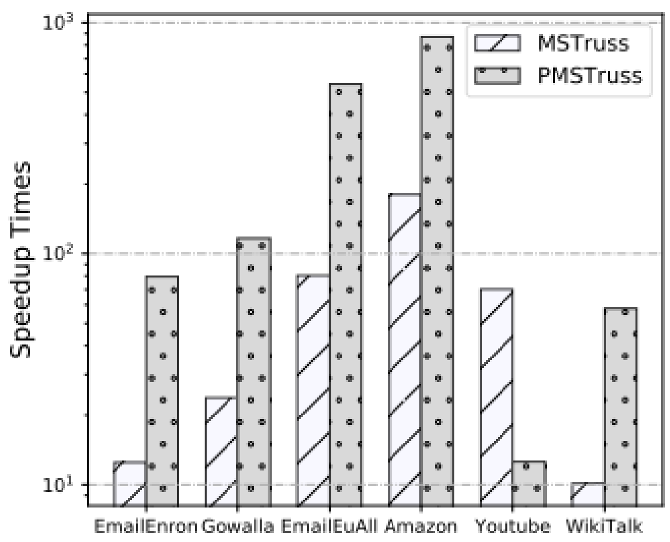


(a) Insertion (b) Deletion

Performance Evaluation



(a) Insertion



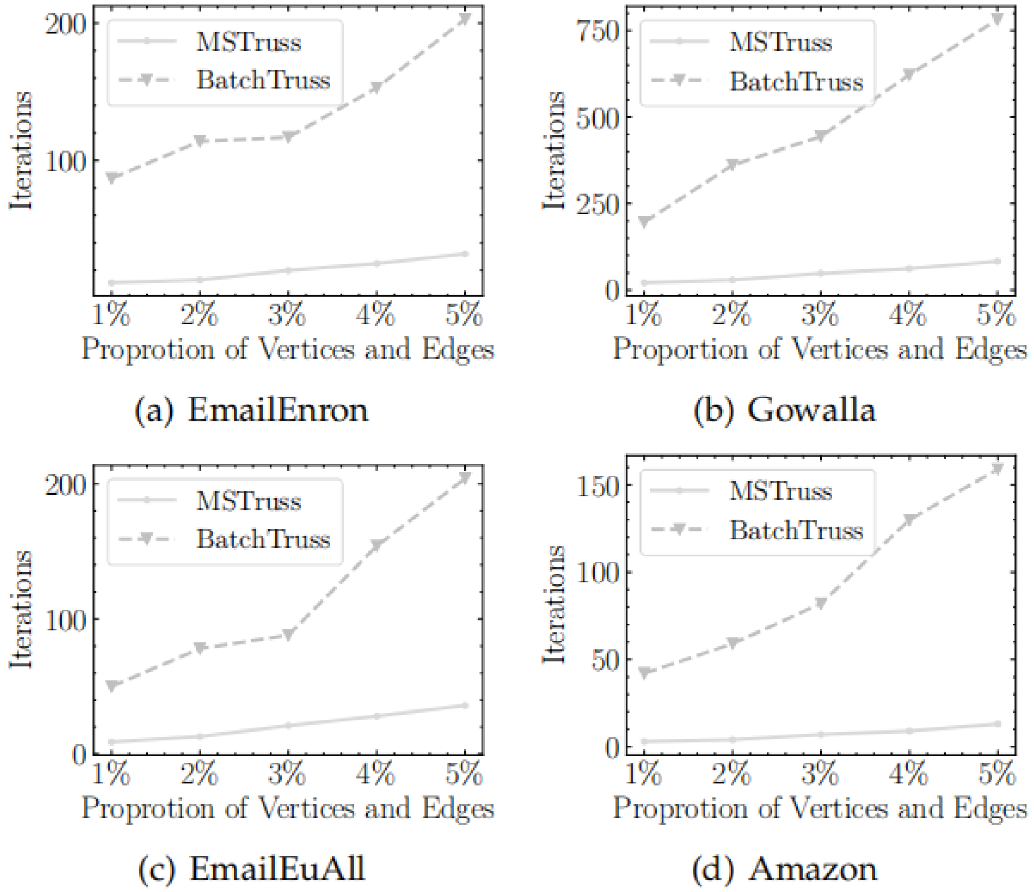
(b) Deletion

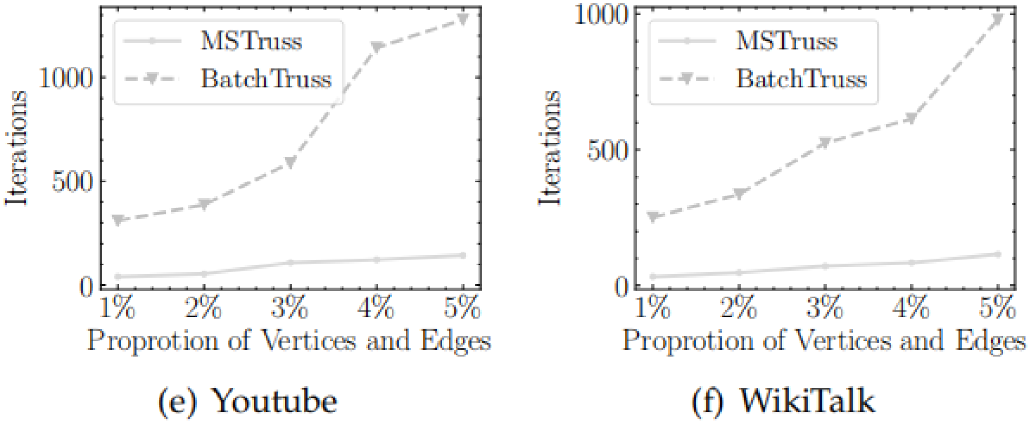
Fig. 8. The speedup times of our algorithms compared to PP&TCP-truss.

[22] X. Huang, H. Cheng, L. Qin, W. Tian, and J. X. Yu, "Querying k-truss community in large and dynamic graphs," in *International Conference on Management of Data, SIGMOD*, Snowbird, UT, USA, 2014, pp. 1311–1322.

[24] S. Ebadian and X. Huang, "Fast algorithm for k-truss discovery on public-private graphs," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI*, S. Kraus, Ed., 2019, pp. 2258–2264.

Performance Evaluation





[25] Q. Luo, D. Yu, X. Cheng, Z. Cai, J. Yu, and W. Lv, "Batch processing for truss maintenance in large dynamic graphs," *IEEE Transactions on Computational Social Systems*, pp. 1–12, 2020.

Fig. 9. The iterations of BatchTruss and MSTruss.

Performance Evaluation

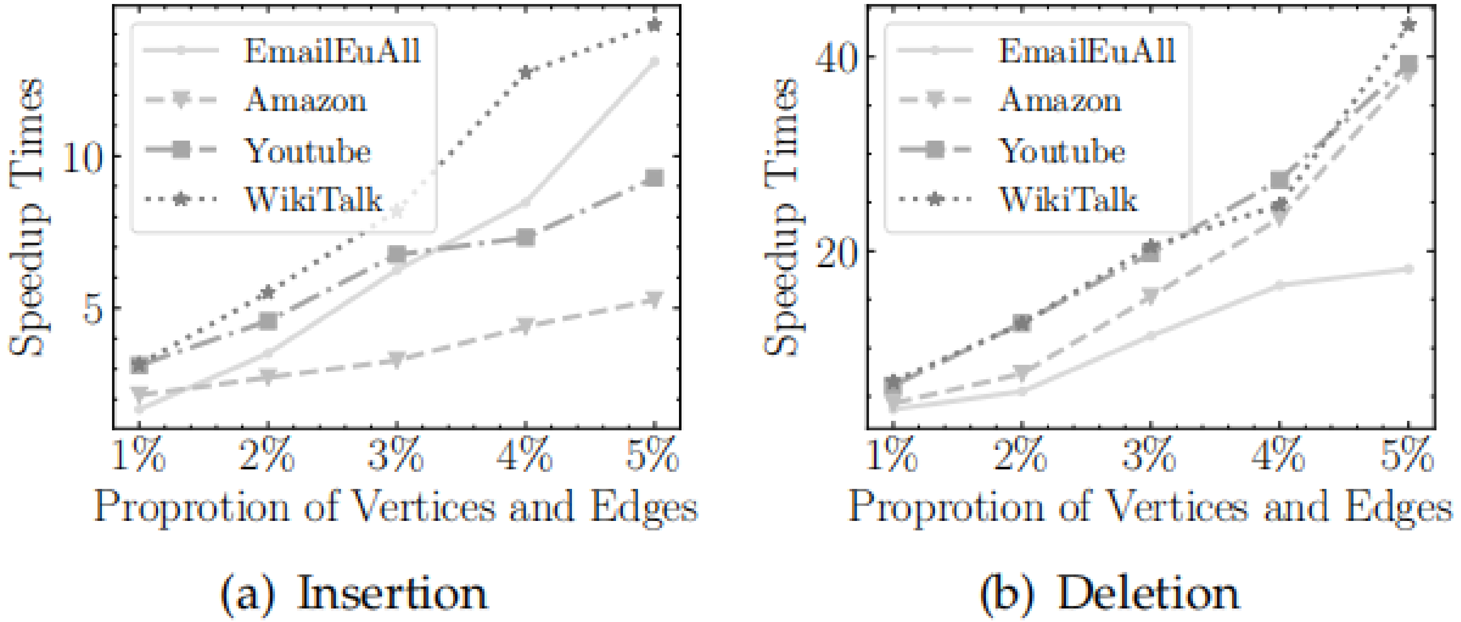


Fig. 10. The speedup times of our algorithms compared to BatchTruss.

谢谢大家！