Sketch-Based Anomaly Detection in Streaming Graphs

Siddharth Bhatia, Mohit Wadhwa, Kenji Kawaguchi, Neil Shah, Philip S. Yu, and Bryan Hooi

Raina Zhang, COMP3801

Real-World Problem & Motivation

Intrusion Detection

- ➔ Edge anomalies: suspicious individual connections between machines
- ➔ Subgraph anomalies: groups of attackers coordinating attacks on target machines
- ➔ Catching both allows full understanding of the attack and see patterns

Financial System

- ➔ Edge anomalies: unusually large or frequent transactions
- ➔ Subgraph anomalies: rings of individuals/businesses involved in coordinated fraud
- ➔ Detecting both protects against financial loss and help detect fraudulent activity

Streaming Data and Anomalous Behaviour

streaming data involving interactions between entities can be modeled as a dynamic graph

2 Types of Anomalous Behaviour

1. Edge Anomalies (Individual/Point Anomalies)

   ➔ Individual connections (edges) that deviate significantly from expected behaviour

2. Subgraph Anomalies (Collective/Structural Anomalies)

   ➔ Groups of nodes and their connections that form dense or unusual structural patterns compared to the rest of the graph

Detect Anomalous Edges

To detect whether the edge is part of an anomalous subgraph in an online manner. Being able to detect anomalies at the finer granularity of edges allows early detection so that recovery can be started as soon as possible and the effect of malicious activities is minimized.

Detect Anomalous Graphs

To detect the presence of an unusual subgraph (consisting of edges received over a period of time) in an online manner, since such subgraphs often correspond to unexpected behavior, such as coordinated attacks.

Constant Memory and Time

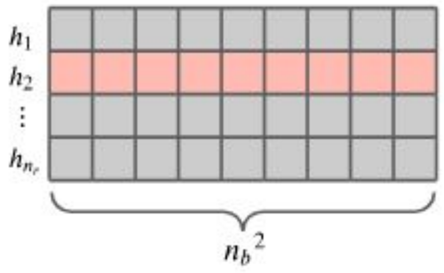
To ensure scalability, memory usage and update time should not grow with the number of nodes or the length of the stream. Thus, for a newly arriving edge, our algorithm should run in constant memory and update time.

Given a stream of graph edges from a dynamic graph, how can we assign anomaly scores to both edges and subgraphs in an online manner?

For the purpose of detecting unusual behaviour, using constant memory and update time

CMS

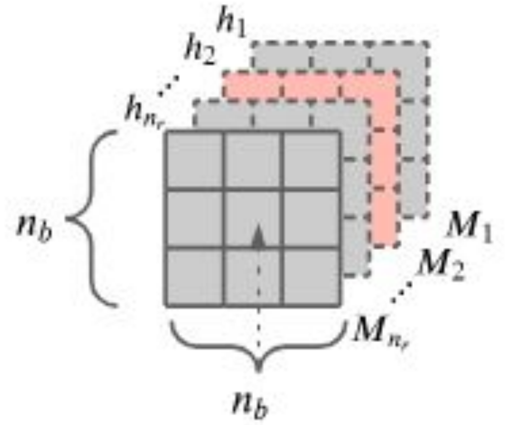
hash
functions



buckets

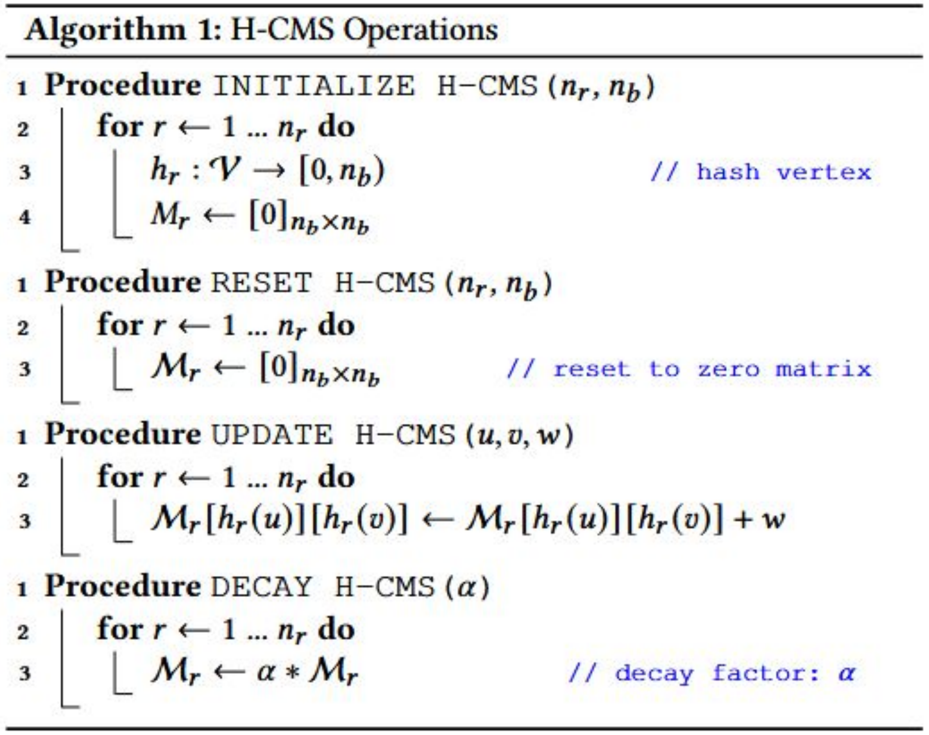
- Hashes 1D items
- An edge (u, v) is hashed like a single token “u,v”
  - Loses directionality and structure

Higher Order-CMS



- Hashes an edge into a 2D location
  - Hash the source into a row index
  - Hash the destination into a column index
- Each hash function has its own n_b x n_b adjacency matrix
- Dense subgraph detection becomes dense submatrix detection
- Keeps the same bounds as CMS

H-CMS Functions

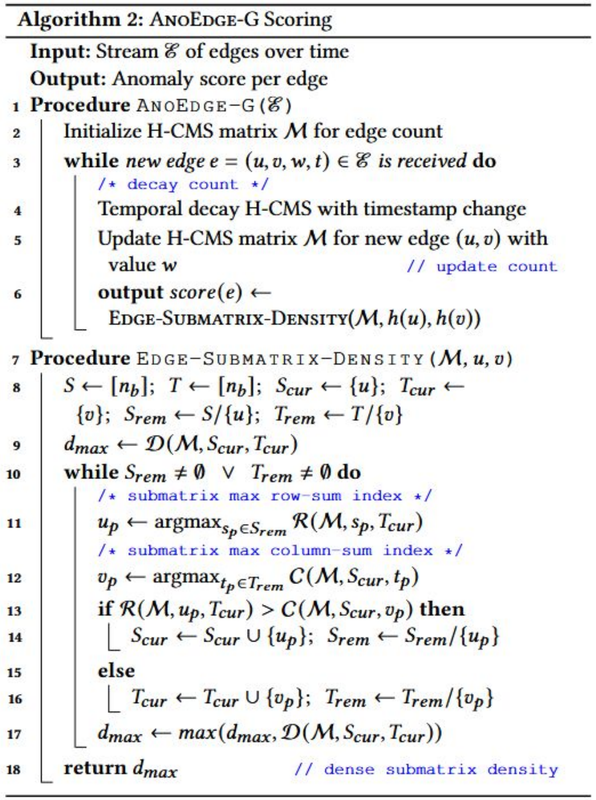


Algorithm 1: H-CMS Operations

```
Procedure INITIALIZE H-CMS ($n_r, n_b$)
    for $r \leftarrow 1 \dots n_r$ do
        $h_r : \mathcal{V} \rightarrow [0, n_b)$                // hash vertex
        $M_r \leftarrow [0]_{n_b \times n_b}$
Procedure RESET H-CMS ($n_r, n_b$)
    for $r \leftarrow 1 \dots n_r$ do
        $\mathcal{M}_r \leftarrow [0]_{n_b \times n_b}$     // reset to zero matrix
Procedure UPDATE H-CMS ($u, v, w$)
    for $r \leftarrow 1 \dots n_r$ do
        $\mathcal{M}_r[h_r(u)][h_r(v)] \leftarrow \mathcal{M}_r[h_r(u)][h_r(v)] + w$
Procedure DECAY H-CMS ($\alpha$)
    for $r \leftarrow 1 \dots n_r$ do
        $\mathcal{M}_r \leftarrow \alpha * \mathcal{M}_r$          // decay factor: $\alpha$
```

- Decay is to forget old edges

AnoEdge-G Scoring

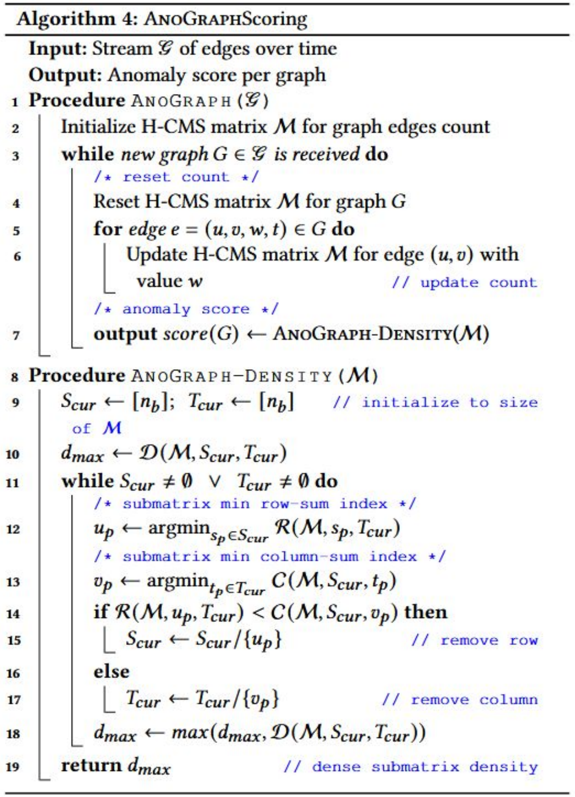


Algorithm 2: AnoEdge-G Scoring

```
Input: Stream $\mathcal{E}$ of edges over time
Output: Anomaly score per edge
Procedure AnoEdge-G($\mathcal{E}$)
    Initialize H-CMS matrix $\mathcal{M}$ for edge count
    while new edge $e = (u, v, w, t) \in \mathcal{E}$ is received do
        /* decay count */
        Temporal decay H-CMS with timestamp change
        Update H-CMS matrix $\mathcal{M}$ for new edge $(u, v)$ with
          value $w$                                // update count
        output $score(e) \leftarrow$
          Edge-Submatrix-Density($\mathcal{M}, h(u), h(v)$)
Procedure Edge-Submatrix-Density($\mathcal{M}, u, v$)
    $S \leftarrow [n_b]$;  $T \leftarrow [n_b]$;  $S_{cur} \leftarrow \{u\}$;  $T_{cur} \leftarrow$
      $\{v\}$;  $S_{rem} \leftarrow S/\{u\}$;  $T_{rem} \leftarrow T/\{v\}$
    $d_{max} \leftarrow \mathcal{D}(\mathcal{M}, S_{cur}, T_{cur})$
    while $S_{rem} \neq \emptyset \ \vee \ T_{rem} \neq \emptyset$ do
        /* submatrix max row-sum index */
        $u_p \leftarrow \text{argmax}_{s_p \in S_{rem}} \mathcal{R}(\mathcal{M}, s_p, T_{cur})$
        /* submatrix max column-sum index */
        $v_p \leftarrow \text{argmax}_{t_p \in T_{rem}} \mathcal{C}(\mathcal{M}, S_{cur}, t_p)$
        if $\mathcal{R}(\mathcal{M}, u_p, T_{cur}) > \mathcal{C}(\mathcal{M}, S_{cur}, v_p)$ then
            $S_{cur} \leftarrow S_{cur} \cup \{u_p\}$;  $S_{rem} \leftarrow S_{rem}/\{u_p\}$
        else
            $T_{cur} \leftarrow T_{cur} \cup \{v_p\}$;  $T_{rem} \leftarrow T_{rem}/\{v_p\}$
        $d_{max} \leftarrow max(d_{max}, \mathcal{D}(\mathcal{M}, S_{cur}, T_{cur}))$
    return $d_{max}$                      // dense submatrix density
```

- If an edge (u, v) lands in a dense part of the H-CMS, it may be part of a burst of edges
- Use submatrix density around the hashed location of (u,v) and use it as an anomaly score
- Start with the 1x1 cell of (u, v) and greedily expand by adding row/col with max sum
- Anomaly score is the maximum density across all expansions
- AnoEdge maintains a temporally decaying H-CMS to simulate the gradual “forgetting” of older (and therefore outdated) information

AnoGraph Scoring



Algorithm 4: AnoGraphScoring

```
Input: Stream $\mathcal{G}$ of edges over time
Output: Anomaly score per graph
1  Procedure AnoGraph ($\mathcal{G}$)
2      Initialize H-CMS matrix $\mathcal{M}$ for graph edges count
3      while new graph $G \in \mathcal{G}$ is received do
           /* reset count */
4          Reset H-CMS matrix $\mathcal{M}$ for graph $G$
5          for edge $e = (u, v, w, t) \in G$ do
6              Update H-CMS matrix $\mathcal{M}$ for edge $(u, v)$ with
                 value $w$                                  // update count
           /* anomaly score */
7          output $score(G) \leftarrow$ AnoGraph-Density($\mathcal{M}$)
8  Procedure AnoGraph-Density ($\mathcal{M}$)
9      $S_{cur} \leftarrow [n_b]$;  $T_{cur} \leftarrow [n_b]$       // initialize to size
         of $\mathcal{M}$
10     $d_{max} \leftarrow \mathcal{D}(\mathcal{M}, S_{cur}, T_{cur})$
11     while $S_{cur} \neq \emptyset \ \vee \ T_{cur} \neq \emptyset$ do
           /* submatrix min row-sum index */
12         $u_p \leftarrow \text{argmin}_{s_p \in S_{cur}} \mathcal{R}(\mathcal{M}, s_p, T_{cur})$
           /* submatrix min column-sum index */
13         $v_p \leftarrow \text{argmin}_{t_p \in T_{cur}} \mathcal{C}(\mathcal{M}, S_{cur}, t_p)$
14         if $\mathcal{R}(\mathcal{M}, u_p, T_{cur}) < \mathcal{C}(\mathcal{M}, S_{cur}, v_p)$ then
15             $S_{cur} \leftarrow S_{cur} / \{u_p\}$                 // remove row
16         else
17             $T_{cur} \leftarrow T_{cur} / \{v_p\}$              // remove column
18         $d_{max} \leftarrow max(d_{max}, \mathcal{D}(\mathcal{M}, S_{cur}, T_{cur}))$
19     return $d_{max}$                        // dense submatrix density
```

- Detect if the entire input graph G is anomalous
- Build an H-CMS by hashing all edges of graph G
- Find the densest submatrix by greedy row/col removal that removes min sum
- Return density as graph anomaly score
- After scoring, H-CMS is cleared before processing next graph

The End