1 Counting motifs in dynamic graphs

In this Secion, we describe basic insights regarding motifs in dynamic graphs. Then, we describe *StreaM*, a new stream-based algorithm for counting undirected 4-vertex motifs in dynamic graphs, and discuss its runtime complexity.

1.0.1 Basic insights

Whenever an edge $e = \{a, b\}$ is added to a graph G_t, i.e., update $u_{t+1} = add(e)$, two things happen: existing motifs are changed and new motifs are formed. First, consider an existing motif m_i that consists of a, b, and 2 other vertices. The addition of e causes the motif to change into a different motif m_j which contains one more edge. We denote this operation as $(i \to j)$. Its execution decreases the occurrences of m_i and increases the occurrences of m_j, i.e.,

$$(i \to j) : \quad \mathcal{F}_{t+1}(m_i) := \mathcal{F}_t(m_i) - 1,\ \mathcal{F}_{t+1}(m_j) := \mathcal{F}_t(m_j) + 1$$

Second, consider vertices c and d that do not form a connected component with a and b without e's existence. In case e connects the four vertices, a new motif m_k is formed. We denote this operation as $+(k)$. Its execution increases the occurrences of m_k, i.e.,

$$+(k) : \quad \mathcal{F}_{t+1}(m_k) := \mathcal{F}_t(m_k) + 1$$

In case an existing edge is removed, i.e., $u_{t+1} = rm(e)$, the inverse happens: some motifs are changed and others are dissolved. We denote these operation as $(i \to j)^{-1}$ and $+(i)^{-1}$.

$$\begin{aligned} (i \to j)^{-1} : &\quad \mathcal{F}_{t+1}(m_i) := \mathcal{F}_t(m_i) + 1,\ \mathcal{F}_{t+1}(m_j) := \mathcal{F}_t(m_j) - 1 \\ +(k)^{-1} : &\quad \mathcal{F}_{t+1}(m_k) := \mathcal{F}_t(m_k) - 1 \end{aligned}$$

Adding or removing a vertex with degree 0 has no effect on the motif count.

Each motif $m_i \in \mathcal{M}$ contains at least 3 and at most 6 edges. The addition and removal of edges leads to transitions between them (cf. Figure 1). For example, adding the missing edge to m_5 changes it to m_6 ($(5 \to 6)$) while removing any edge from m_6 changes it to m_5 ($(5 \to 6)^{-1}$). Adding edge $\{b, d\}$ to the disconnected set of nodes x creates a new motif m_1 ($+(1)$) which is dissolved by the removal of any of its 3 edges ($+(1)^{-1}$).

The main idea behind our new stream-based algorithm is to find and apply these operations to correctly update $\mathcal{F}$ for each edge addition and removal.

1.0.2 *StreaM*

Assume an update (addition or removal) of edge $e = \{a, b\}$. To correctly adapt $\mathcal{F}$, we need to consider all 2-vertex sets $\{c, d\} \in CD(a, b)$ such that a, b, c, and d form a motif if e exists. Either both vertices are connected to a or b directly or d is a neighbor of c which is connected to a or b. With

$$N(a, b) := (n(a) \cup n(b)) \backslash \{a, b\},$$

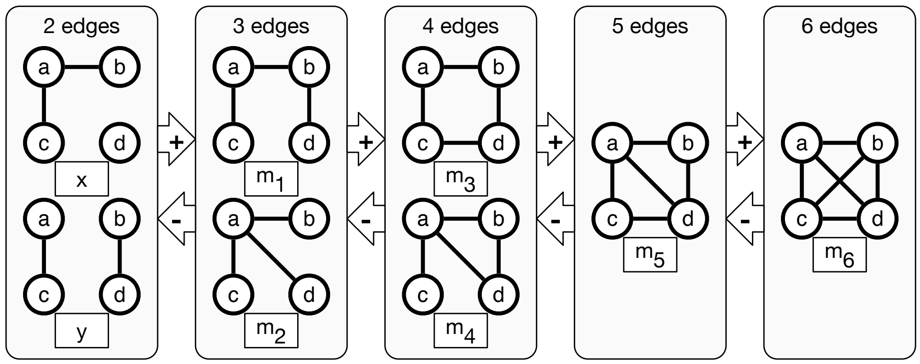
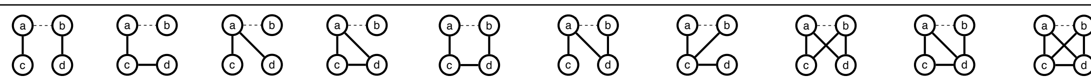


Figure 1: Transitions between the motifs $m_i \in \mathcal{M}$ when adding and removing edges

Table 1: Operation mapping $\mathcal{O}$ from signatures $\mathcal{S}(a, b, c, d)$ to operations

$\mathcal{S}$	10010 01100	10001 01001 00101 00011	11000 00110	11001 00111	10011 01101	11100 11010 10110 01110	10101 01011	11110	11101 11011 10111 01111	11111
$\mathcal{O}$	$+(1)$	$+(1)$	$+(2)$	$+(4)$	$(1 \to 3)$	$(1 \to 4)$	$(2 \to 4)$	$(3 \to 5)$	$(4 \to 5)$	$(5 \to 6)$



we can define $CD(a, b)$ as follows:

$$CD(a, b) = \{\{c, d\} : (c, d \in N(a, b), c \neq d) \vee (c \in N(a, b), d \in n(c) \backslash \{a, b\})\}$$

Besides $\{a, b\}$, 5 edges are possible between a, b, c, and d. We denote their existence as a quintuple $\mathcal{S}(a, b, c, d) = (ac, ad, bc, bd, cd)$, called their *signature*. At least two distinct edges must exist, the first connecting c and the second connecting d. Therefore, there are $2^5 - 2 \cdot 2^2 = 24$ possible signatures.

Each signature corresponds to a specific operation that must be executed to update $\mathcal{F}$. We define a function $\mathcal{O}$ that maps a signature $\mathcal{S}$ on the corresponding operation. The complete assignment of signatures to operations is given in Table 1. In case the edge $\{a, b\}$ is removed instead of added, the inverse operation must be executed. As an example consider the signature (10010) which is isomorph to (01100). The addition of $\{a, b\}$ creates the motif m_1. Its removal dissolves the motif as a, b, c, and d are no longer connected.

Based on $\mathcal{S}$ and $\mathcal{O}$, we can now describe the stream-based algorithm *StreaM* for updating the motif frequency in an undirected graph (cf. Algorithm 1). For an edge $\{a, b\}$ that is added or removed (described by *type*), we first determine the set $CD(a, b)$ of all pairs of vertices connected to a and b. For each pair $\{c, d\} \in CD(a, b)$, the required operation $o = \mathcal{O}(\mathcal{S}(a, b, c, d))$ is determined from the signature of a, b, c, and d. If $\{a, b\}$ is added, the operation o is executed. Otherwise, the inverse operation o^{-1} is executed.

```
Data: G, {a,b}, type ∈ {add, rm}
begin
    for {c,d} ∈ CD(a,b) do
        o = O(S(a,b,c,d)) ;                      /* operation */
        if type = add then
            execute o ;                          /* edge is added */
        else if type = rm then
            execute o^{-1} ;                     /* edge is removed */
    end
end
```

Algorithm 1: *StreaM* for maintaining $\mathcal{F}$ in dynamic graphs

1.0.3 Complexity discussion

StreaM iterates over the $|CD(a, b)| \leq 5 \cdot (d_{max})^2$ elements of $CD(a, b)$. For each element $\{c, d\}$, it computes the signature which can be done in $5 \cdot O(1)$ time, assuming hash-based datastructures are used for adjacency lists. In addition, $\mathcal{F}$ is incremented or decremented which has time complexity of $O(1)$ as well. Therefore, processing a single edge addition or removal with *StreaM* has time complexity of

$$O((d_{max})^2) \cdot (O(1) + O(1)) = O((d_{max})^2)$$