1 Definitions and Notation

A *graph* $G = (V, E)$ is an ordered pair of *vertices* $V = \{v_1, v_2, \dots v_{|V|}\}$ and *edges* E. Here, we only consider *undirected graphs without loops*, i.e., $E \subseteq \{\{v, w\} : v, w \in V, v \neq w\}$. Two vertices v and w are called *connected* in case there exists a path between them. A graph is called *connected* in case any pair of vertices (v, w) is connected. For a subset V' of the vertex set V, we refer to $G^{V'} = (V', E'), E' := \{\{v, w\} \in E : v, w \in V'\}$ as the V'-induced subgraph of G.

As the *k-Neighborhood* $N_k(e)$ of an edge $e = \{a, b\}$, we denote the set of all $(k-2)$-tuples of vertices $v \in V, a \neq v \neq b$ that are connected to vertices a or b in the induced subgraph. We call each element $N \in N_k(e)$ a *neighbor set* of e. Each of them corresponds to the subgraph $G^{\{a,b\} \cup N}$ of G that contains a and b as well as $k-2$ other vertices.

For a graph G, its *adjacency matrix* $A(G) = A_{i,j}$ is an $n \times n$ matrix defined as follows:

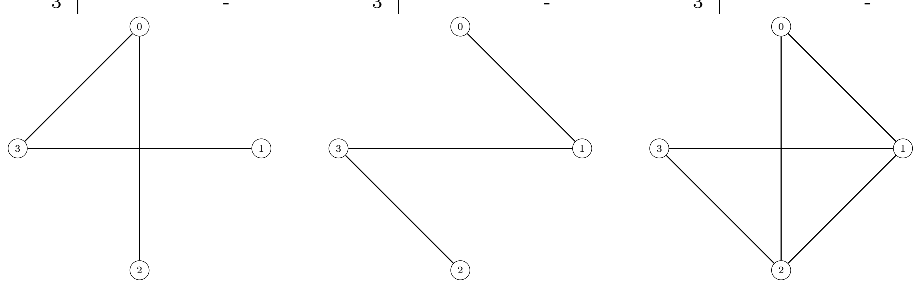
$$A^u_{ij} = \begin{cases} 1 \text{ (true)} & \text{if } i > j \wedge \{v_i, v_j\} \in E \\ 0 \text{ (false)} & \text{if } i > j \wedge \{v_i, v_j\} \notin E \\ \text{undefined} & \text{otherwise} \end{cases}$$

We denote the *set of all adjacency matrices* of size k as $\mathcal{A}_k$, $|\mathcal{A}_k| = 2^{\frac{n \cdot (n-1)}{2}}$. The set of the adjacency matrices of *connected graphs* of size k is denoted as $\mathcal{A}^{con}_k \subset \mathcal{A}_k$. As examples consider the following adjacency matrices:

	0	1	2	3
0	-	0	1	1
1		-	0	1
2			-	0
3				-

	0	1	2	3
0	-	1	0	0
1		-	0	1
2			-	1
3				-

	0	1	2	3
0	-	1	1	0
1		-	1	1
2			-	1
3				-



(a) $G_i, |E_i| = 3, n(G_i) = 22$ (b) $G_j, |E_j| = 3, n(G_j) = 49$ (c) $G_k, |E_k| = 5, n(G_k) = 59$

Figure 1: Examples of connected 4-vertex graphs

Assume a to be the sequence of all defined entries of an adjacency matrix A, i.e., $a = (a_1, a_2, \dots a_{\frac{k \cdot (k-1)}{2}}) := (A_{1,2}, A_{1,3}, \dots A_{k-1,k})$ Then, we define the *key* of an adjacency matrix A and the corresponding graph G as follows:

$$n(A) = n(G) := \sum_{i=1}^{\frac{k\cdot(k-1)}{2}} a_i \cdot 2^{i-1}$$

Then, $\mathcal{N}_k^{con} \subset [0, 2^{\frac{k\cdot(k-1)}{2}}]$ denotes the set of all keys of connected graphs of size k.

As a dynamic graph, we consider a graph whose set of edges E changes over time. We assume that in each time step, a single edge is either added to or removed from E. This change is denoted as an update: either $add(e)$ or $rm(e)$. A graph is transformed from G_i to G_{i+1} by the application of update u_{i+1}.

add example of graph transformation over time...

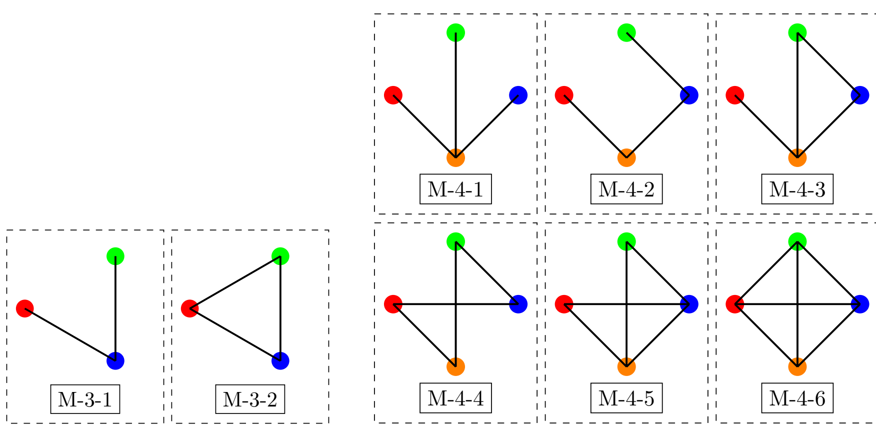
2 Motifs

As motifs of size k, also called k-vertex motifs or k-motifs, we consider the equivalence classes of isomorph connected k-vertex graphs which we denote as $\mathcal{M}_k$.

Therefore, each connected adjacency matrix $A \in \mathcal{A}_k^{con}$ is element of exactly one equivalence class represented by a motif $m \in \mathcal{M}_k$. We express this property as a function that maps the key $n(A)$ of a connected adjacency matrix A to a motif $m \in M_k$, i.e.,

$$r : \mathcal{N}_k^{con} \to \mathcal{M}_k$$

This assignment can be computed by enumerating all connected adjacency matrices and determining their equivalence class by performing an isomorphism check with all existing motifs.



(a) 3-vertex motifs: $\mathcal{M}_3$ (b) 4-vertex motifs: $\mathcal{M}_4$

Figure 2: Examples for the set of motifs $\mathcal{M}_k$ for different sizes

3 Implementation

For simplicity, we store the function r as integer pairs (n, m) where n is the key of a connected adjacency matrix and m the index of the equivalence class, or motif, it belongs to.

4 Algorithm

Whenever an edge $e = \{a, b\}$ is added to or removed from a graph G_i, each subgraph $G_i^{\{a,b\} \cup N}, N \in N_k(e)$ represents a motif that is created, transformed, or dissolved.

In case $u_i = rm(e)$, $n(A_i^{\{a,b\} \cup N})$ is the key of the adjacency matrix for $N \in N_k(e)$ before the removal. After the removal, the key will be $n(A_{i+1}^{\{a,b\} \cup N}) = n(A_i^{\{a,b\} \cup N}) - 1$, i.e., the same adjacencies except for the missing edge between the first two vertices. If $n(A_i^{\{a,b\} \cup N}) - 1 \notin \mathcal{N}_k^{con}$, the existing motif will be dissolved and is transformed otherwise.

Similarly, in case e is added to the graph G_i, $n(A_{i+1}^{\{a,b\} \cup N}) = n(A_i^{\{a,b\} \cup N}) + 1$ is the key of the adjacency matrix for the neighbor set $N \in N(a, b)$ afterwards. If $n(A_i^{\{a,b\} \cup N}) \notin \mathcal{N}_k^{con}$, a new motif is created and an existing one transformed otherwise.

From this, we can define an algorithm that updates the motif frequency $\mathcal{F}$ for the application of an update u_i. In addition, all changes to motifs in the graph can be listed:

5 Complexity of Algorithm

define d_{max} here?!?

When processing an update u_{i+1}, i.e., $add(\{a, b\})$ or $rm(\{a, b\})$, we must iterate over all elements of $N_k(a, b)$ with $|N_k(a, b)| \leq d_{max}^{k-2}$. Processing each neighborhood $N \in N_k(a, b)$ can be done in $O(1)$ as it only requires the generation of the key $n(A^{\{a,b\} \cup N})$, its lookup in the pre-computed assignment r, and the adaptation of $\mathcal{F}$. Therefore, the complexity for the execution of the algorithms is $O(d_{max}^{k-2})$.

6 Statistics about motifs

Data: $G_i, e = \{a, b\}, type \in \{add, rm\}, print \in \{true, false\}$
begin
    for $N \in N_k(e)$ **do**
        $n_i = n(A_i^{\{a,b\} \cup N})$; `/* key before */`
        if $type = add$ **then**
            $n_{i+1} = n_i + 1$; `/* key after addition */`
        else
            $n_{i+1} = n_i - 1$; `/* key after removal */`
        if $n_i \in \mathcal{N}_k^{con}$ **then**
            $\mathcal{F}(r(n_i)) -= 1$; `/* decr old motif */`
        if $n_{i+1} \in \mathcal{N}_k^{con}$ **then**
            $\mathcal{F}(r(n_{i+1})) += 1$; `/* incr new motif */`
        if *print* **then**
            if $n_i \in \mathcal{N}_k^{con} \wedge n_{i+1} \in \mathcal{N}_k^{con}$ **then**
                print 'transformed: a, b, N ($r(n_i) \rightarrow r(n_{i+1})$)'
            else if $n_i \in \mathcal{N}_k^{con}$ **then**
                print 'dissolved: a, b, N ($r(n_i)$)'
            else
                print 'created: a, b, N ($r(n_{i+1})$)'
    end
end

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

k	2	3	4	5	6	7
$\lvert\mathcal{A}_k\rvert$	2	8	64	1,024	32,768	2,097,152
$\lvert\mathcal{A}_k^{con}\rvert$	1	4	38	827	26,704	1,866,256
$\lvert\mathcal{M}_k\rvert$	1	2	6	21	112	853

Table 1: Statistics about adjacency matrices and motifs of different sizes