Temporal Graph Motifs

François Théberge*

1 Introduction

In this note, we describe the algorithms to look for *temporal* $K_{2,h}$ motifs in a graph or network of *events*. We assume that events are time indexed and grouped under some **root event** which happens first via a tree structure (which could also simply be a linear sequence of events). Many types threaded conversation (e.g. tweets and retweets, Reddit comments, email chains) can be described in this format, which are instances of the W3 paradigm: (i) Who (the actors), (ii) What (the events) and (iii) When (time of the events).

1.1 Data Format

The data consists of k temporal 4- or 5-tuples:

- $\{(event_i, actor_i, t_i, root_i)\}_{i=1}^k$
- $\{(event_i, actor_i, t_i, root_i, parent_i)\}_{i=1}^k$

Where:

- $event_i$: is a unique identifier for each event
- $actor_i$: is the identifier of the actor/user generating the event
- t_i: is the time of the event, typically in UTC format
- $root_i$: is the identifier of the root event that $event_i$ is part of
- $parent_i$: is the identifier of the parent of this event when events with same $root_i$ have a tree structure

*Tutte Institute for Mathematics and Computing, Ottawa, Canada; email: theberge@ieee.org

We assume that:

- $event_i = root_i$ for the **root event**; in this case, $parent_i$ is meaningless
- all events with same $root_i$ can not happen before the root event itself
- if $parent_i$ are used, the parent event can not happen before one of its child event

1.2 Temporal $K_{2,h}$ motifs

A $K_{2,h}$ motif is a complete bipartite subgraph with vertices from 2 subsets V_1 and V_2 where $|V_1| = 2$ and $|V_2| = h \geq 2$. In our context, nodes in V_1 represent *actors*, such as users in a social network, while nodes in V_2 are *events*, for example a message in a thread, or a re-tweet.

Temporal $K_{2,h}$ motifs are parameterized by 3 values:

- Δt: the reaction time, in seconds;
- ΔT: the repetition time, in seconds;
- h: the number of distinct events forming the motif.

We describe 2 families of motifs next. The pseudo-codes to identify those motifs are detailed in the Appendix.

1.2.1 root-based motifs

A temporal root-based $K_{2,h}$ motifs given $(\Delta t, \Delta T)$ occurs when:

(1) an actor A submits a root event $root_i$ and a different actor B submits an event under that root event (i.e. with same $root_i$) within Δt seconds (the *reaction* time), and

(2) the above scenario happens h times, for h distinct root events, all within ΔT seconds (the *repetition* time).

1.2.2 hop-based motifs

A temporal hop-based $K_{2,h}$ motifs given $(\Delta t, \Delta T)$ occurs when:

(1) an actor A submits an event with identifier $event_i$ (the event needs not be the *root event* but it can be), and a different actor B submits another event with parent identifier $parent_j$ such that $parent_j = event_i$, within Δt seconds (the *reaction* time), and

(2) the above scenario happens h times, for h distinct root events, all within ΔT seconds (the *repetition* time).

In cases where (1) above happens more than once under the same *root* event, the first instance is retained.

2 The Python Code

Installing the `temporal graph mining (tgm)` library is straightforward with: `pip install tgm`. The only dependencies are the `igraph` python library as well as the standard `numpy` and `pandas` libraries.

For both types of temporal motifs, root-based or hop-based, the algorithm to find the motifs is broken up in two steps:

(a) given the list of event tuples, build a *bipartite graph* representation of the data, and

(b) enumerate all *motifs* using this bipartite graph.

The advantage of this decomposition is that if look for motifs multiple times for the same dataset, for example using different parameter values $(\Delta t, \Delta T)$, then step (a) is only run once, which speeds up the process. The pseudocodes are detailed in the appendix, namely:

- Algorithm 1: compute bipartite graph for root-based motifs;
- Algorithm 2: use the bipartite graph from Algorithm 1 to find root-based motifs;
- Algorithm 3: compute bipartite graph for hop-based motifs;
- Algorithm 4: use the bipartite graph from Algorithm 3 to find hop-based motifs.

Examples running the `tgm` on toy and real datasets are given in the Jupyter notebook which can be found at `https://github.com/ftheberge/tgm`.

A Pseudo-code

Algorithm 1 Directed Bipartite Graph for Root-based Motifs

Require: Events $(event_i, actor_i, t_i, root_i)$ in dataframe D

  Initialize empty directed bipartite graph B_r
  for each event $(event_i, actor_i, t_i, root_i)$ in D **do**
    if $event_i = root_i$ **then**
      add edge $actor_i \rightarrow root_i$ to B_r with attribute $(time = t_i, isRoot = True)$
    else
      add edge $actor_i \rightarrow root_i$ to B_r with attribute $(time = t_i, isRoot = False)$
    end if
  end for
  return B_r

Algorithm 2 Root-based Temporal $K_{2,h}$ Motifs

Require: B_r from Algorithm 1 and parameters $(h, \Delta t, \Delta T)$
  initialize empty list E
  for each vertex *root* in B_r with in-degree(*root*) > 1 (i.e. at least 2 events) **do**
    if there is an edge $actor \to root$ with attributes ($time = t$, $isRoot = True$) for some t **then**
      $t_{min} = t$ (time of the root event)
      $rootActor = actor$ (actor of the root event)
      for each $actor_i$ such that there is at least one edge $actor_i \to root$ **do**
        let t_i be the minimum of all time attributes from all edges $actor_i \to root$
        if $t_i - t_{min} \leq \Delta t$ and $actor_i \neq rootActor$ **then**
          append 4-tuple $(root, actor_i, t_{min}, rootActor)$ to E
        end if
      end for
    end if
  end for
  let $|E| = k$, re-label $E = (root_i, actor_i, t_i, rootActor_i)$ for $1 \leq i \leq k$
  initialize empty graph G
  for $1 \leq i \leq k$ **do**
    if there is already an edge $actor_i \to rootActor_i$ in G **then**
      append event $(root_i, t_i)$ as edge attribute
    else
      add edge $actor_i \to rootActor_i$ to G with event $(root_i, t_i)$ as edge attribute
    end if
  end for
  for all edges in G **do**
    order edge attributes $(root_i, t_i)$ with respect to time
    count the number of h-consecutive events happening within ΔT, store as edge *weight* (or drop edge if $weight = 0$)
  end for
  return weighted graph G where each edge represents an $actor \to rootActor$ pair and edge weight is the number of temporal $K_{2,h}$ motifs for this pair.

Algorithm 3 Directed Bipartite Graph for Hop-based Motifs

Require: Events $(event_i, actor_i, t_i, root_i, parent_i)$ in dataframe D

for each $(event_i, actor_i, t_i, root_i, parent_i)$ in D **do**
  if there exists j such that $event_j = parent_i$ and $actor_j \neq actor_i$ **then**
    (i) $parent\_actor_i = actor_j$
    (ii) $parent\_t_i = t_j$
    (iii) $\Delta t_i = t_i - parent\_t_i$
    (iv) add $parent\_actor_i, parent\_t_i, \Delta t_i$ to D
  else
    drop this event in D
  end if
end for
prune D keeping only the earliest instance with respect to t for every unique triple $(actor, root, parent\_actor)$
initialize empty directed bipartite graph B_h
for each $(event_i, actor_i, t_i, root_i, parent\_actor_i, parent\_t_i, \Delta t_i)$ in D **do**
  add edge $actor_i \rightarrow parent\_actor_i$ to B_h with attribute $(root = root_i, time = parent\_t_i, \Delta t = \Delta t_i)$
end for
return B_h

Algorithm 4 Hop-based Temporal $K_{2,h}$ Motifs

Require: B_h from Algorithm 3 and $(h, \Delta t, \Delta T)$

initialize empty graph G

for each edge $actor_i \rightarrow parent\_actor_i$ in B_h with attributes $(root_i, parent\_t_i, \Delta t_i)$ **do**

  if $\Delta t_i \leq \Delta t$ **then**

    if there is already an edge $actor_i \rightarrow parent\_actor_i$ in G **then**

      append event $(root_i, parent\_t_i)$ as edge attribute

    else

      add edge $actor_i \rightarrow parent\_actor_i$ to G with event $(root_i, parent\_t_i)$ as edge attribute

    end if

  end if

end for

for all edges in G **do**

  order edge attributes $(root_i, parent\_t_i)$ with respect to time

  count the number of h-consecutive events happening within ΔT, store as edge *weight* (or drop edge if $weight = 0$)

end for

return weighted graph G where each edge represents an $actor \rightarrow parent\_actor$ pair and edge weight is the number of temporal $K_{2,h}$ motifs for this pair.