DYMOND: DYnamic MOtif-NoDes Network Generative Model

Giselle Zeno
Purdue University
West Lafayette, IN, USA
gzenotor@purdue.edu

Timothy La Fond
Lawrence Livermore National Laboratory
Livermore, CA, USA
lafond1@llnl.gov

Jennifer Neville
Purdue University
West Lafayette, IN, United States
neville@purdue.edu

ABSTRACT

Motifs, which have been established as building blocks for network structure, move beyond pair-wise connections to capture longer-range correlations in connections and activity. In spite of this, there are few generative graph models that consider higher-order network structures and even fewer that focus on using motifs in models of dynamic graphs. Most existing generative models for temporal graphs strictly grow the networks via edge addition, and the models are evaluated using static graph structure metrics—which do not adequately capture the temporal behavior of the network. To address these issues, in this work we propose DYnamic MOtif-NoDes (DYMOND)—a generative model that considers (i) the dynamic changes in overall graph structure using temporal motif activity and (ii) the roles nodes play in motifs (e.g., one node plays the hub role in a wedge, while the remaining two act as spokes). We compare DYMOND to three dynamic graph generative model baselines on real-world networks and show that DYMOND performs better at generating graph structure and node behavior similar to the observed network. We also propose a new methodology to adapt graph structure metrics to better evaluate the temporal aspect of the network. These metrics take into account the changes in overall graph structure and the individual nodes' behavior over time.

CCS CONCEPTS

• **Theory of computation** → **Dynamic graph algorithms**; **Network formation**; *Social networks*; • **Computing methodologies** → **Network science**; **Motif discovery**; *Statistical relational learning*; • **Networks** → *Topology analysis and generation.*

KEYWORDS

Dynamic Networks, Temporal Graphs, Motifs, Network Evolution

ACM Reference Format:
Giselle Zeno, Timothy La Fond, and Jennifer Neville. 2021. DYMOND: DYnamic MOtif-NoDes Network Generative Model. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3442381.3450102

1 INTRODUCTION

Network models provide a way to study complex systems from a wide range of domains, such as social, biological, computing and communication networks. The ability to generate synthetic networks is useful for evaluating systems on a wide(r) range of structure and sharing without divulging private data, for example.

Many generative models for static graphs have aimed to generate synthetic graphs that can simulate real-world networks [3]. Random graph models [5] were among the first proposed. These were adapted to produce degree distributions similar to those of social networks [4, 31], but still failed to generate the clustering of real-world networks. Block models [13, 19, 20] were later proposed for creating communities observed in social networks. However, these methods focus on capturing either global or local graph properties.

Historically, complex networks with temporal attributes have been studied as static graphs by modeling them as growing networks or aggregating temporal data into one graph. In reality, most of these networks are dynamic in nature and evolve over time, with nodes and edges constantly being added and removed. Initial models for temporal or dynamic networks (where links appear and disappear, such as social-network communication [6, 9, 27]) focused on modeling the edges over time, ignoring higher-order structures.

Although traditionally most graph models have been edge-based, motifs have been established as building blocks for the structure of networks [17]. Modeling motifs can help to generate the graph structure seen on real-world networks and capture correlations in node connections and activity. Following work that studied the evolution of graphs using higher-order structures [1, 8, 22, 32, 33], recently [26] proposed a generative model using temporal motifs to produce networks where links are aggregated over time. However, this approach assumes that edges will not be removed once they are added (i.e., placed).

In this paper, we propose a dynamic network model, using temporal motifs as building blocks, that generates dynamic graphs with links that appear and disappear over time. Specifically, we propose DYnamic MOtif-NoDes (DYMOND), a generative model that first assigns a motif configuration (or motif type) and then samples inter-arrival times for the motifs. One challenge that comes with this is sampling the motif placement. To this end, we define motif node roles and use them to calculate the probability of each motif type. The motifs and node roles can capture correlations in node connections and activity.

Another key challenge is *how to evaluate dynamic graph models?* Previous work has focused on evaluating models using the structure metrics defined for *static* graphs without incorporating measures that reflect the temporal behavior of the network. To address this, we adapt previous metrics to consider temporal structure and node behavior over time. Using both sets of metrics (i.e., static and dynamic), we evaluate our proposed model on five real-world datasets, comparing against three recent alternatives. The results show that DYMOND generates dynamic networks with the closest graph structure and similar node behavior as the real-world data.

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450102

To summarize, we make the following contributions:

- We conduct an empirical study of motif behavior in dynamic networks, which shows that motifs do not change/evolve from one timestep to another, rather they keep re-appearing in the same configuration
- Motivated by the above observation, we develop of a novel statistical dynamic-graph generative model that samples graphs with realistic structure and temporal node behavior using motifs
- We outline a new methodology for comparing dynamic-graph generative models and measuring how well they capture the underlying graph structure distribution and temporal node behavior of a real graph

The rest of the paper is organized as follows. First, we go over related work and discuss where our model fits in Section 2. Then in Section 3, we present our empirical study of the evolution of motifs in temporal graphs. In Section 4, we propose our generative model DYMOND. In Section 5, we present the datasets and baseline models used in our evaluation. We also describe our evaluation metrics and how to adapt them to dynamic networks. Finally, we discuss the results of our evaluation (Section 6) and present our conclusions (Section 7).

2 RELATED WORK

Most models for temporal or dynamic networks have focused on modeling the edges over time [6, 9, 27]. A straightforward approach to generating temporal networks is to generate first a static graph from some model, and for each link generate a sequence of contacts [7]. Holme [6] uses an approach where they draw degrees from a probability distribution and match the nodes in random pairs for placing links. Then, for every link, they generate an active interval duration from a truncated power-law distribution and uniform random starting time within that time frame. Rocha and Blondel [27] use a similar method where the active interval of a node starts when another node's interval ends. Another approach is to start with an empty graph. Then, every node is made active according to a probability and connected to m random nodes. Perra et al. [23] use this approach with a truncated power-law distribution for each node's probability of being active. Laurent et al. [12] extend this model to include memory driven interactions and cyclic closure. Other extensions include aging effects [18] and lifetimes of links [28]. Vestergard et al. [29] model nodes and links as being active or inactive using temporal memory effects. All of these node and edge-based models do not consider higher-order structures and fail to create enough clustering in the networks generated.

Motivated by the work that established motifs as building blocks for the structure of networks [17], the definition of motifs was extended to temporal networks by having all the edges in a given motif occur inside a time period [8, 22, 33]. Zhang et al. [32] study the evolution of motifs in temporal networks by looking at changes in bipartite motifs in subsequent timesteps. Benson et al. [1] study higher-order networks and how 3-node motifs evolve from being empty to becoming a triangle in aggregated temporal graphs. Purohit et al. [26] propose a generative model that creates synthetic temporal networks where links are aggregated over time (i.e., no link deletions). Zhou et al. [34] propose a dynamic graph neural network model that takes into account higher-order structure by using node-biased temporal random walks to learn the network topology and temporal dependencies. The models that use temporal motifs are not designed for dynamic networks. We propose the first motif-based dynamic network generative model.

3 MOTIF EVOLUTION

Related work on modeling temporal networks showed the evolution of motifs using a triadic closure mechanism (e.g., wedges becoming triangles) [1]. However, these works make the assumption that edges will remain in the network once they are added [1, 26, 32]. This assumption would hold on growing networks, but does not apply to dynamic networks where links can also be removed.

We make the distinction that we are interested in the evolution of motifs in dynamic networks, where edges can appear and disappear. In our initial study below, we investigate if similar motif behavior occurs in dynamic networks across subsequent time windows (e.g., if the motifs appear, merge, split and/or disappear over time). Specifically, we investigate 3-node motifs and look for changes from one motif type to another (for example, wedges becoming triangles and vice versa).

3.1 Definitions

Here we introduce our main definitions used in this paper. The rest of notations and symbols are summarized in Table 1.

Definition 3.1 (Graph Snapshot). A graph snapshot is a time-slice of a network at time t, defined as $G_t = (V_t, E_t, S_t)$, where $V_t \subseteq V$ is the set of active nodes, $E_t \subseteq E$ is the set of edges at time t, and $S_t \subseteq S$ are the edge timestamps.

Definition 3.2 (Dynamic Network). A dynamic network (or graph) $\mathbf{G} = \{G_1, \ldots, G_T\}$ is a sequence of graph time-slices where T is the number of timesteps.

Definition 3.3 (Motif). We define a motif as a 3-node subgraph $\{u, v, w\}$ and its *motif type* is determined by the number of edges between the nodes (i.e., *empty* has 0, *1-edge* has 1, *wedge* has 2, *triangle* has 3 edges).

3.2 Empirical Study

We test the hypothesis that motifs changing configurations is driven by a time-homogeneous Markov process, where the graph structure at the next timestep $t + 1$ depends on the current timestep t. Each timestep corresponds to a time window of the temporal graph. Then, we consider all 3-node motifs at each timestep to either transform from one motif type to another or remain the same. Note isomorphisms are combined into the same configuration.

We study the effectiveness of this approach on the Enron Emails and EU Emails datasets, described in Subsubsection 5.2.1 and Subsubsection 5.2.2 respectively. Additionally, we use a Wikipedia Links dataset, which shows the evolution of hyperlinks between Wikipedia articles [11, 25]. The nodes represent articles. Edges include timestamps and indicate that a hyperlink was added or removed depending on the edge weight (-1 for removal and +1 for addition). The transition probability matrices for both email datasets (Enron Emails and EU Emails) show that the motifs with edges (i.e., 1-edge, wedge, and triangle) will either keep their current

(a) Enron Emails **(b) EU Emails** **(c) Wikipedia Links**

Figure 1: Observed Motif Transition Probabilities

motif type, or become empty with almost equal probability (Figures 1a, 1b). For each motif type with edges, the count of times it stayed is very close to that of becoming empty at the next time period. In contrast, the Wikipedia Links dataset is a growing network, with more links between articles being added and very few removed. This makes it unlikely to see any motif with edges becoming empty (Figure 1c).

In the dynamic network datasets we investigated: (1) we do not observe motifs with edges changing from one motif type to another (e.g., wedges becoming triangles and vice versa), even when selecting different time windows to create the timesteps, and (2) motifs stay as the same type or disappear in the next time window. This motivates our use of motifs and inter-arrival rates in our proposed generative model for dynamic networks, which we outline next.

4 DYNAMIC MOTIF-NODES MODEL

We formally define the problem of dynamic network generation as follows:

Problem 1: ***Dynamic Network Generation***

Input: *A dynamic network* $\boldsymbol{G} = \{G_1, \ldots, G_T\}$

Output: *A dynamic network* $\boldsymbol{G'} = \{G'_1, \ldots, G'_{T'}\}$, *where the distribution of graph structure for* $\boldsymbol{G'}$ *matches* $\boldsymbol{G}$ *and node behavior is aligned across* $\boldsymbol{G'}$ *and* $\boldsymbol{G}$ *(i.e., the node behavior of a specific node* $v_{i'}$ *in* $\boldsymbol{G'}$ *should be similar to a specific node* v_i *in* $\boldsymbol{G}$*).*

Specifically, consider an arbitrary graph statistic $s(G)$ (e.g., average path length). Then the distribution of statistic values observed in the input dynamic network $\mathbf{s}_{in} = \{s(G_1), \ldots, s(G_T)\}$ should match the distribution of statistic values observed in the output dynamic network $\mathbf{s}_{out} = \{s(G'_1), \ldots, s(G'_{T'})\}$. Similarly, take any node statistic $\mathbf{s}(v_i|\mathbf{G})$ (e.g., node degree). Then, using the temporal distribution of values for a node $\mathbf{s}(v_i|\mathbf{G}) = \{s(v_i|G_1), \ldots, s(v_i|G_T)\}$, the distribution of values for all nodes in the input dynamic network $\{\mathbf{s}(v_j|\mathbf{G})\}_{j \in \mathbf{G}}$ should match the distribution of values for all nodes in the output dynamic network $\{\mathbf{s}(v_{j'}|\mathbf{G'})\}_{j' \in \mathbf{G'}}$.

To generate dynamic networks as specified above, we propose the DYnamic MOtif-NoDes (DYMOND) model[1]. Our model makes the following assumptions about the graph generative process:

(1) nodes in the graph become active and remain that way,
(2) nodes have a probability distribution over role types that they play in different motifs,
(3) node triplets have a single motif type over time,
(4) there is a distribution of motif types over the set of graphs,
(5) motif occurrences over time are distributed exponentially with varying rate.

First we describe DYMOND's generative process below. Then we describe our approach to estimate model parameters from an observed dynamic network. We model the time until nodes become active as Exponential random variables with the same rate λ_V. Since all possible 3-node motifs are considered, there will be edges shared among them. Therefore, to estimate the inter-arrival rate for each motif, we weigh the count of times a motif appeared by the number of edges shared with other motifs in a timestep. For each motif type with edges (i.e., triangle, wedge, and 1-edge), the model fits an Exponential distribution with the motif inter-arrival rates of that type. Motivated by our findings in Section 3, when a motif is first sampled it will keep the same configuration in the future.

In the generation process, the motifs are sampled from a probability distribution based on the probability of the nodes in a triplet participating in a particular motif type, while also ensuring the motif type proportions in the graph are maintained. The motif type probability for a triplet considers the roles each node would play in a motif. For example, in a wedge one node would be a hub and the other two would be spokes (Figure 2). The node role probabilities are learned from the input graph's structure and the motifs that the node participates in.

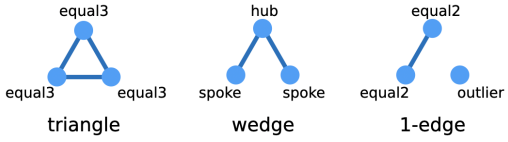


Figure 2: Motif Types and Node Roles

The motivation for this modeling approach is based on the following conjectures: (1) by modeling higher-order structures (i.e., motifs), the model will capture the underlying distribution of graph

[1]Code is available at https://github.com/zeno129/DYMOND

structure, and (2) by using the motif roles that nodes take in the dynamic network, the model will also capture correlations in node connections and activity.

4.1 Generative Process

The overall generative process is described in Alg. 1: In line 2, we first get the nodes that are active at each timestep using the node arrival rate λ_V (see Alg. 4, Appendix A). Whenever new nodes become active, we calculate the new triplets of active nodes that are now eligible to be sampled as a motif in line 5. In line 6, we proceed to sample the motifs, based on the node role probabilities p_R for each motif type, and the timesteps the motifs will appear using the motif inter-arrival rates λ_M (see Alg. 2). In line 8, we place the motifs' edges (Alg. 6, Appendix A) and in line 12 we construct the graph (Alg. 7, Appendix A).

Algorithm 1: `GenerateDynamicGraph`

input: $T, N, \lambda_V, p_M, \lambda_M, p_R, c_R$
output: $\mathbf{G}' = \{G'_1, \ldots, G'_T\}$

1 **begin**
2 $V \leftarrow$ `GetActiveNodes`$(T,\ N,\ \lambda_V)$
3 $M \leftarrow \emptyset, M^S \leftarrow \emptyset, M^E \leftarrow \emptyset$
4 **for** $t \in [1, \ldots, T]$ **do**
`// New active triplets at timestep` t
5 $U_t \leftarrow \{m = \{u, v, w\} \subseteq V_t \mid u < v < w,\ m \notin U_{t-1}\}$
6 $M_t, M_t^T, M_t^S, M_t^R, p_R, c_R \leftarrow$ `SampleMotifs`$(U_t, p_M, \lambda_M, p_R, c_R)$
7 $M \leftarrow M \cup M_t$ `// save new motifs`
8 $M_t^E \leftarrow$ `PlaceMotifEdges`(M_t, M^T, M_t^R)
9 $M^T \leftarrow M^T \cup M_t^T$ `// store their types`
10 $M^E \leftarrow M^E \cup M_t^E$ `// store their edges`
11 $M^S \leftarrow M^S \cup M_t^S$ `// store their timestamps`
12 $\mathbf{G}' \leftarrow$ `ConstructGraph`(M, M^E, M^S)

In Alg. 2 line 4, the model calculates the expected count of motifs $n^{(i)}$ to be sampled for each motif type i using the motif type proportions p_M. Then in line 5, the expected number of motifs for each type is sampled, given the probability p_T that the nodes in the triplet take on the roles needed (Eq. 6). Each node will have an expected count c_R of times they will appear having each role, for the total number of timesteps T to be generated. For this reason, in line 8 we sample the timesteps each motif will appear in (Alg. 5, Appendix A), and in line 10 we use the timestep counts to sample the node roles (Alg. 3).

4.2 Learning

Given an observed dynamic graph $\mathbf{G}$, we estimate the input parameters for our generative process as outlined in Alg. 8, Appendix A.

4.2.1 Node Arrivals. We begin by estimating the node arrival rate $\widehat{\lambda}_V$, which will determine when nodes become active in the dynamic network, by using the first timestep in which each node had its first edge.

$$\widehat{\lambda}_V = \frac{\sum_{v \in V} \left(\arg\min_t \mathbb{1}(v \in V_t) \right)}{|V|} \tag{1}$$

Table 1: Notations and Symbols

Symbol	Description
$\mathbf{G} = \{G_1, \ldots, G_T\}$	dynamic temporal network
$G_t = (V_t, E_t, S_t)$	graph snapshot at time t
$t \in [1, \ldots, T]$	timestep (time-window)
T	number of timesteps
$N = \lvert V \rvert$	number of nodes
$V_t \subseteq V$	set of active nodes at timestep t
$E_t \subseteq E$	set of edges at timestep t
$S_t\big((u', v')\big)$	list of timesteps for edge (u', v')
λ_V	node arrival rate (see Eq. 1)
$p_M = (p_M^{(1)}, p_M^{(2)}, p_M^{(3)})$	motif type proportions (see Eq. 2)
$\lambda_M\big(\{u, v, w\}\big)$	motif inter-arrival rate (see Eq. 3a)
$\lambda_M = (\lambda_M^{(1)}, \lambda_M^{(2)}, \lambda_M^{(3)})$	motif type rates distr. (see Eq. 3b)
C^M	count times motifs appear (see Eq. 4)
$c_t(u', v')$	number of times $(u', v') \in E_t$
$r_t^{(i)}(u', v')$	remaining count $(u', v') \in E_t$
$\lvert N^{(i)}(u', v') \rvert$	number of motifs type i
$p_T^{(i)}\ (\{u, v, w\})$	probability of motif i (see Eq. 6)
p_R	node role probabilities (see Eq. 7)
c_R	node role counts (see output Alg. 10)
M	set of motifs
$M^{(i)}$	motifs of type i
M^T	motif types
M^E	motif edges
M^S	motif timesteps
M^R	motif node roles

4.2.2 Motif Proportions. In Alg. 9 (Appendix A), we find the motifs in G_t for each time window t. For each 3-node motif $\{u, v, w\}$, we find its motif type i at timestep t (line 8). If we have previously seen $\{u, v, w\}$ and the motif type i is of higher order at the current timestep t, then we update the type stored to be i (line 13). For example, if we observe the triplet $\{u, v, w\}$ is a triangle at timestep t and we previously saw it as a wedge, we update $M^T\big(\{u, v, w\}\big)$ as a triangle.

Then, we calculate the motif proportions $\widehat{p}_M^{(i)}$ of each type in the graph, where i corresponds to the number of edges in the motif (i.e., $i = 1$ for a 1-edge, $i = 2$ for a wedge, and $i = 3$ for a triangle motif).

$$\widehat{p}_M^{(i)} = \frac{\left\{\{u, v, w\} \in M \;\middle|\; M^T\big(\{u, v, w\}\big) = i\right\}}{\binom{N}{3}}, \quad \text{for } i \in [1, 2, 3]$$

$$\widehat{p}_M^{(0)} = 1 - \sum_{i=1}^{n} \widehat{p}_M^{(i)} \tag{2}$$

where M is the set of motifs, and $\{u, v, w\}$ is a motif consisting of the nodes u, v, w.

4.2.3 Motif Inter-Arrivals. We estimate the inter-arrival rates of each observed motif $\{u, v, w\}$ using weighted edge counts (Eq. 3a). Their rates are then used to learn a rate of inter-arrival rates $\widehat{\lambda}_M^{(i)}$ from the motifs of each type i (Eq. 3b). Note that we do not need to estimate rates for the empty motif type ($i = 0$).

$$\widehat{\lambda}_M\big(\{u, v, w\}\big) = \frac{\sum_{t=1}^{T} C_t^M\big(\{u, v, w\}\big)}{T} \tag{3a}$$

Algorithm 2: SampleMotifs

input: $U_t, p_M, \lambda_M, p_R, c_R$
output: M_t // sampled motifs
M_t^T // motif types
M_t^S // motif timestamps
M_t^R // motif node roles
p_R // node role probabilities
c_R // node role counts

begin
  $L = U_t$ // triplets to sample from
  for $i \in [3, 2, 1]$ **do**
    // sample motifs
    $n^{(i)} = |U_t| \cdot p_M^{(i)}$ // num. motifs to sample
    $M^{(i)} \sim$
    $$Mult\left(\left[\frac{p_T^{(i)}(\{u,v,w\})}{\sum_{\{u',v',w'\}\in L} p_T^{(i)}(\{u',v',w'\})} \;\middle|\; \{u,v,w\} \in L\right],\ n^{(i)}\right)$$
    $M_t^T = i,\ \forall \{u,v,w\} \in M^{(i)}$ // store motif types
    $L = L - M^{(i)}$ // triplets left to sample from
    $S^{(i)} \leftarrow \texttt{SampleMotifTimesteps}(t, M^{(i)}, i, \lambda_M)$
    $M^{(i)'} = \left\{m \;\middle|\; m \in M^{(i)},\ |S^{(i)}(m)| > 0\right\}$
    $R^{(i)}, p_R, c_R \leftarrow \texttt{SampleNodeRoles}(M^{(i)'}, i, p_R, c_R, S^{(i)})$
    $M_t^S \leftarrow M_t^S \cup S^{(i)}$ // store timesteps
    $M_t^R \leftarrow M_t^R \cup R^{(i)}$ // store roles

$$\widehat{\lambda}_M^{(i)} = \frac{\sum_{\{u,v,w\}\in M^{(i)}} \left(\widehat{\lambda}_M(\{u,v,w\})\right)}{|M^{(i)}|} \tag{3b}$$

where $M^{(i)}$ is the set of all motifs of type i.

Since edges might be shared by more than one motif, we use edge-weighted Poisson counts C_t^M, per timestep t, to estimate the inter-arrival rate for each motif $\{u,v,w\}$ (Eq. 4). The weights $W_t^{(i)}$ will depend on the motif type i of $\{u,v,w\}$ and are calculated for each edge of the motif (Eq. 5).

$$C_t^M(\{u,v,w\}) = \frac{\sum_{(u',v')\in E_t(\{u,v,w\})} W_t^{(i)}((u',v'))}{|E_t(\{u,v,w\})|} \tag{4}$$

For a motif $\{u,v,w\}$, we calculate the weight of its edge (u',v') using the count for the edge in the timestep window and considering its motif type i (Eq. 5). We give larger edge-weight to motif types with more edges, since they are more likely to produce the observed edges. This also ensures that motif types with smaller proportion $p_M^{(i)}$ (Eq. 2) have a high enough inter-arrival rate to show up (i.e., triangles).

$$W_t^{(i)}(u',v') = \frac{r_t^{(i)}(u',v')}{|N^{(i)}(u',v')|} \tag{5a}$$

$$r_t^{(i-1)}(u',v') = \min\left(r_t^{(i)}(u',v'),\ |N^{(i-1)}(u',v')|\right) \tag{5b}$$

where $|N^{(i)}(u',v')|$ is the number of motifs of type i, the number of times (u',v') appears in E_t is $c_t(u',v')$, the remaining edge count is $r_t^{(i)}(u',v')$, and for triangles $r_t^{(i+1)}(u',v') = c_t(u',v')$.

Algorithm 3: SampleNodeRoles

input: $M^{(i)'}, i, p_R, S^{(i)}$
output: $R^{(i)}$ // node roles for motifs in $M^{(i)}$
p_R // updated role probabilities
c_R // updated role counts

begin
  for $m \in M^{(i)'}$ **do**
    if $i = 3$ **then** // triangle
      $R^{(i)}(m, \texttt{equal3}) \leftarrow m$
      $c_R(v, \texttt{equal3}) \mathrel{-}= |S^{(i)}(m)|,\ \forall v \in m$
    else if $i = 2$ **then** // wedge
      // hub node
      $p_h = \left[\frac{p_R(v,\,\texttt{hub})}{\sum_{v'\in m} p_R(v',\,\texttt{hub})} \;\middle|\; v \in m\right]$
      $v_h \sim Bin(m,\ p_h)$
      $R^{(i)}(m, \texttt{hub}) \leftarrow v_h$
      $c_R(v_h, \texttt{hub}) \mathrel{-}= |S^{(i)}(m)|$
      // spoke nodes
      $R^{(i)}(m, \texttt{spoke}) \leftarrow m \setminus \{v_h\}$
      $c_R(v, \texttt{spoke}) \mathrel{-}= |S^{(i)}(m)|,\ \forall \{v \in m : v \neq v_h\}$
    else if $i = 1$ **then** // 1-edge
      // outlier node
      $p_o = \left[\frac{p_R(v,\,\texttt{outlier})}{\sum_{v'\in m} p_R(v',\,\texttt{outlier})} \;\middle|\; v \in m\right]$
      $v_o \sim Bin(m,\ p_o)$
      $R^{(i)}(m, \texttt{outlier}) \leftarrow v_o$
      $c_R(v_o, \texttt{outlier}) \mathrel{-}= |S^{(i)}(m)|$
      // equal2 nodes
      $R^{(i)}(m, \texttt{equal2}) \leftarrow m \setminus \{v_o\}$
      $c_R(v, \texttt{equal2}) \mathrel{-}= |S^{(i)}(m)|,\ \forall \{v \in m : v \neq v_o\}$
    for $v \in m$ **do** // update role distr (Eq. 7)
      $p_R(v, \texttt{role}) = P[v \text{ is role}],\ \forall r \in R$

4.2.4 Motif Types. The probability of a node triplet becoming a triangle, wedge, or 1-edge motif is based on the probability that each node takes on the roles needed to form that motif type. The roles for each motif type are shown in Figure 2. Specifically, a triangle requires all three nodes to have the equal3 role, a wedge requires one node to be a hub and the rest to have the spoke role, a 1-edge requires two nodes to have the equal2 role and the remaining one the outlier role (Eq. 6).

$$p_T^{(i)}(\{u,v,w\}) = P[u \text{ is } r_1 \wedge v \text{ is } r_2 \wedge w \text{ is } r_2] \tag{6}$$

$$P[u \text{ is } r] = \frac{count(u,r)}{\sum_{r'\in R} count(u,r')} \tag{7}$$

where $R = \{\texttt{equal3}, \texttt{hub}, \texttt{spoke}, \texttt{equal2}, \texttt{outlier}\}$ is the set of possible roles, and $count(u,r)$ is the weighted count of times that node u had role r (see Alg. 10, Appendix A). The weights are used to avoid over-counting the roles for motifs of the same type with a shared edge.

5 METHODOLOGY

We first describe the baseline models (Subsection 5.1) and datasets (Subsection 5.2) used in our evaluation. Then, we introduce the

metrics for evaluating graph structure, our novel approach for evaluating node behavior, and implementation details of all models (Subsection 5.3).

5.1 Baselines

The related work using motif-based models for temporal graphs focuses on the aggregated temporal graph and not its dynamic changes over time [26]. With that in mind, we picked baselines that aim to model the changes in dynamic graphs. We compare our model with three baselines: a temporal edge-based model (SNLD), a model based on node-activity (ADN), and a graph neural network (GNN) model based on temporal random walks (TagGen).

5.1.1 Static Networks with Link Dynamics Model (SNLD). We used an approach based on [6], where they begin by generating a static graph and then generate a series of events. Their procedure begins by sampling degrees from a probability distribution. They refer to these degrees as "stubs" and they create links by connecting these "stubs" randomly. Finally, for each link, they assign a time-series from an inter-event distribution.

In our implementation of the SNLD model, we start by sampling the degrees from a Truncated Power-law distribution. Since our starting point is a static graph, we assume all the nodes to be active already. Then, we sample inter-event times for every edge. We found that we could best model the edge inter-event times in the real data using an Exponential distribution. To learn the Truncated Power-law parameters, we aggregated and simplified the real graph.

5.1.2 Activity-Driven Network Model (ADN). We use the approach in [12], which extends the model in [23] by adding memory effects and triadic closure. The triadic closure takes place when node i connects to node k forming a triangle with its current neighbor j. Adding a triadic closure mechanism helps to create clustering (communities) [2]. The memory effect is added by counting the number of times that the nodes have connected up to the current time t. The procedure starts by creating an empty graph G_t at each timestep. Then, for each node i: delete it with probability p_d or mark it as active with probability a_i. If the node is "deleted", then the edges in the current timestep are removed, the counts of connections set to zero, and another degree is sampled to estimate a new a_i. If a node i is sampled as active, we connect it to either: (1) a neighbor j, (2) a neighbor of j, or (3) a random node.

In our implementation of the SNLD model, we base the probability to create new edges a_i on the degree of node i, which we sample from a Truncated Power-law distribution. We estimate the parameters using the average degree across timesteps for the nodes in the real graph. There is a fixed probability p_d for any node being "deleted" (losing its memory of previous connections and sampling a new a_i). We estimate this probability using the average ratio of nodes becoming disconnected in the next timestep. To estimate the probability for triadic closure (forming a triangle), we use the average global clustering coefficient across timesteps.

5.1.3 TagGen. TagGen is a deep graph generative model for dynamic networks [34]. In their learning process they treat the data as a temporal interaction network, where the network is represented as a collection of temporal edges and each node is associated with multiple timestamped edges at different timestamps. It trains a bi-level self-attention mechanism with local operations (addition and deletions of temporal edges), to model and generate synthetic temporal random walks for assembling temporal interaction networks. Lastly, a discriminator selects generated temporal random walks that are plausible in the input data, and feeds them into an assembling module. We used the available implementation of TagGen[2] to learn the parameters from the input graph and assemble the dynamic network using the generated temporal walks.

5.2 Datasets

We use the datasets described below, with more detailed statistics shown in Table 4 and Figure 4 of Appendix A.

5.2.1 Enron Emails. The Enron dataset is a network of emails sent between employees of Enron Corporation [10, 11]. Nodes in the network are individual employees and edges are individual emails. Since it is possible to send an email to oneself, loops were removed.

5.2.2 EU Emails. The EU dataset is an email communication network of a large, undisclosed European institution [11, 14]. Nodes represent individual persons and edges indicate at least one email has been sent from one person to the other. All edges are simple and spam emails have been removed from the dataset.

5.2.3 DNC Emails. The DNC dataset is the network of emails of the Democratic National Committee that were leaked in 2016 [11, 24]. The Democratic National Committee (DNC) is the formal governing body for the United States Democratic Party. Nodes in the network correspond to persons and an edge denotes an email between two people. Since an email can have any number of recipients, a single email is mapped to multiple edges in this dataset.

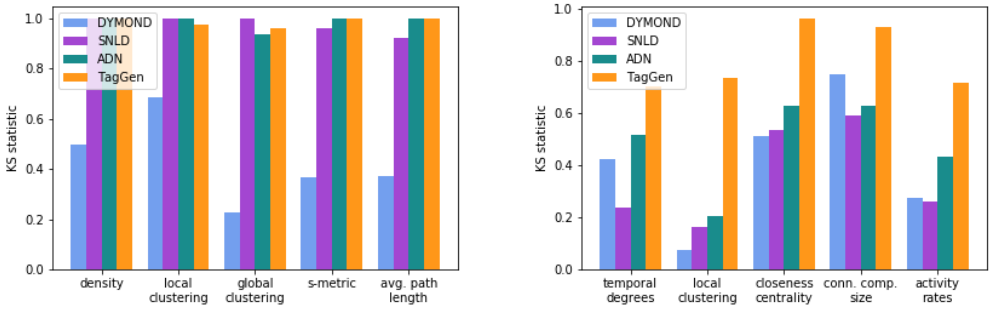
5.2.4 Facebook Wall-Posts. The Facebook dataset is a network of a small subset of posts to other users' walls on Facebook [11, 30]. The nodes of the network are Facebook users, and each directed edge represents one post, linking the users writing a post to the users whose wall the post is written on. Since users may write multiple posts on a wall, the network allows multiple edges connecting a single node pair. Since users may write on their own wall, loops were removed.

5.2.5 CollegeMsg. The CollegeMsg dataset is comprised of private messages sent on an online social network at the University of California, Irvine [15, 21]. Users could search for other users in the network, based on profile information, and then begin conversation. An edge (j, k, t) means that user j sent a private message to user k at time t.

5.3 Evaluation

We use two sets of metrics in our evaluation for graph structure and node behavior. The majority of graph structure metrics we selected are widely used to characterize graphs. With these first set of metrics we aim to measure if the overall graph structure of the generated graph $\mathbf{G}'$ is similar to the dataset graph $\mathbf{G}$. For the second set, we propose to use node-aligned metrics to capture node behavior.

[2] https://github.com/davidchouzdw/TagGen



(a) Graph Structure Metrics **(b) Node Behavior Metrics**

Figure 3: EU Emails - KS Statistics

5.3.1 Graph Structure Metrics. We use the following graph metrics: density, average local clustering coefficient, global clustering coefficient, average path length of largest connected component (LCC), and s-metric. *Density* measures ratio of edges in the graph versus the number of edges if it were a complete graph. The *local clustering coefficient* quantifies the tendency of the nodes of a graph to cluster together, and the *global clustering coefficient* measures the ratio of closed triplets (triangles) to open and closed triplets (wedges and triangles). The *average (shortest) path length*, for all possible pairs of nodes, measures the efficiency of information transport. The *s-metric*, which is less well-known, measures the extent to which a graph has hub-like structure [16]. The s-metric reflects large star structures in a graph. Together with local and global clustering, these metrics provide insight into graph structure like tightly knit-groups and large star structures.

To compare the graph structure generated by the models against that of the datasets, we calculate the graph structure metrics for each time-slice of the generated graph and the input graph. Specifically, for each graph structure metric s, we calculate the distribution of values $\mathbf{s}_{gen}$ of the generated graph and $\mathbf{s}_{in}$ of the input graph (where $\mathbf{s}_{in} = \{s(G_1), \ldots, s(G_T)\}$, G_t is a time-slice of $\mathbf{G}$, and $t \in [1, \ldots, T]$). Given that we aim to model the distribution of graph structure, and not just generate the same graph sequence, we calculate the Kolmogorov-Smirnov (KS) test statistic on $\mathbf{s}_{gen}$ and $\mathbf{s}_{in}$ to evaluate $\mathbf{G}'$ against $\mathbf{G}$.

5.3.2 Node Behavior Metrics. We propose a new approach to analyze a node's behavior over time. For this, we use the following *node-aligned, temporal* metrics: activity rate, temporal degree distribution, clustering coefficient, closeness centrality, and the size of its connected component. The *activity rate* of a node measures how often it participates in an edge. The *temporal degree distribution* of a node u is the set of degrees of u over all snapshots G_t; it shows how many nodes it interacts with. The *local clustering coefficient* of a node measures how close its neighbors are to becoming a clique. The *closeness centrality* of a node u in a (possibly) disconnected graph is the sum of the reciprocal of shortest path distances to u over all other reachable nodes. The node's closeness centrality and size of its connected component indicate the location of the node relative to others.

To compare the temporal node behavior of the generated graphs against the datasets, we calculate the node-aligned temporal metrics for every node in the dataset $\mathbf{G}$ and in the generated graphs $\mathbf{G}'$. Node-alignment refers to assumption that node ids are aligned over graph snapshots, within a graph sequence. Based on this, we can measure the distribution of values a node has over time $\mathbf{s}(v_i|\mathbf{G}) = \{s(v_i|G_1), \ldots, s(v_i|G_T)\}$ for each metric s. Since the nodes in $\mathbf{G}$ do not necessarily correspond to those in $\mathbf{G}'$, we consider the inter-quartile range (IQR) of values over time $\{\mathbf{s}(v_j|\mathbf{G})\}_{j \in \mathbf{G}}$. We then perform a 2-dimensional KS test using the Q_1 and Q_3 values of all nodes in $\mathbf{G}$ and $\mathbf{G}'$. In this way, we capture each node's individual behavior and their joint behavior.

In the past, people have used the mean and median of the values, but these statistics do not capture characteristics of the distribution of values and can be misleading. For example, a synthetic graph $\mathbf{G}'$ could have mean and median values of a metric s very close to those of an observed graph $\mathbf{G}$, but have a much larger dispersion of s values than observed in $\mathbf{G}$. We use the KS test on the inter-quartile range (i.e., Q_1 and Q_3) because it does not make assumptions about the distribution of values and can capture variability or dispersion.

5.3.3 Implementation Details. We estimate initial motif configurations and all parameters of our DYMOND model as described in Subsection 4.2 from dataset graphs. Similarly, we estimate all parameters of the SNLD and ADN baselines as described in Subsubsections 5.1.1 and 5.1.2. For the TagGen baseline, we use the available implementation from [34], which is generally described in Subsubsection 5.1.3.

6 RESULTS

6.1 Evaluation of Generated Graphs

In Figure 3, we show the KS statistic (lower is better) for the graph structure and node behavior of the EU Emails dataset, as an example. The full set of results of all the other datasets, for the five graph metrics and the five node metrics, are in Appendix A. In order to compare the models more easily, we calculated the mean reciprocal rank (MRR) of the KS statistic for the graph structure and the node behavior metrics. To calculate the MRR, we ranked the model results by using the average KS statistic.

In Table 2, we can observe that our DYMOND model outperforms the baselines when considering all the graph structure metrics together using the MRR (higher is better). In Table 3, our model performs best on the node behavior for two of the datasets (Enron

Emails and Facebook). SNLD performs better on the EU Emails dataset, with our model being a close second, and the CollegeMsg dataset. Finally, ADN performs best on the DNC Emails dataset, but DYMOND significantly outperforms the other two baselines.

Table 2: Graph Structure Mean Reciprocal Rank

Model	Enron	EU	DNC	Facebook	CollegeMsg
DYMOND	**0.57**	**1.00**	**0.80**	**0.77**	**0.90**
SNLD	0.52	0.61	0.29	0.47	0.45
ADN	0.46	0.67	0.34	0.50	0.43
TagGen	0.53	0.58	0.64	0.30	0.30

Table 3: Node Behavior Metrics Mean Reciprocal Rank

Model	Enron	EU	DNC	Facebook	CollegeMsg
DYMOND	**0.93**	0.70	0.50	**0.85**	0.48
SNLD	0.40	**0.73**	0.35	0.65	**0.95**
ADN	0.47	0.37	**0.93**	0.42	0.48
TagGen	0.25	0.25	0.27	0.25	0.25

6.2 Discussion

SNLD creates a static graph with a degree distribution learned from the input graph and models the edge inter-event times independently. This fails to create graph structure similar to the datasets due to little clustering. The CollegeMsg dataset has low clustering (local and global) but a high s-metric, which indicates large star structure in the graph (i.e., high degree nodes), as seen in Figure 4. In this case, SNLD is able to better match the clustering than the other datasets (Figure 6d).

ADN models node activation rates using sampled node degrees from a Power-law distribution. However, during sampling a node might be "deleted" and have its rate changed. When evaluating node-aligned metrics over time, these rate modifications will change the behavior of a node. This explains the poor performance of ADN on the node activity rates metric of all the datasets (Figures 6a to 6d) except the EU Emails dataset (Figure 3b). Lastly, this model incorporates a triadic closure mechanism to create clustering in the graph structure, which actually helps it perform better on datasets with high clustering (Figure 6b).

Though TagGen doesn't perform well on most of the graph structure metrics, it manages to create graph clustering comparable to our model in three of the datasets: Enron Emails, DNC Emails, and Facebook (Figures 5a to 5c). These three datasets have various star structures (very high degree nodes) across timesteps and shorter diameter than the other datasets. TagGen performs biased temporal random walks and, according to the authors, high degree nodes tend to be highly active resulting in a weak dependence between the topology and temporal neighborhoods. This would explain why it performs better in these three datasets than the others.

Using motifs helps our DYMOND model create better clustering in the graph than the other baselines (Figures 5a, 5c and 5d), which also impacts the graph density and s-metric that is generated. The motif inter-arrival times are able to capture the node-aligned behavior in the graph (Figures 6a to 6c). The addition of motif node roles determines the placement of the motifs in the graph, which in turn impacts the node closeness and shortest path lengths produced. These roles also help capture the node-aligned temporal degree distribution even though we do not directly optimize it.

7 CONCLUSION

Our proposed dynamic network generative model, DYnamic MOtif-NoDes (DYMOND), is the first motif-based dynamic network generative model. DYMOND not only considers the dynamic changes in overall graph structure using temporal motif activity, but also considers the roles the nodes play in motifs (e.g., one node plays the hub role in a wedge, while the remaining two act as spokes). We note that using motifs helps our DYMOND model create better graph structure overall, while the motif node roles can better represent the temporal node behavior.

In our empirical study of dynamic networks, we demonstrated that motifs with edges: (1) generally do not change configurations (e.g., wedges becoming triangles and vice versa); (2) once they appear, they will continue during the next time window or disappear. Though we do not explicitly address higher-order motifs, using the node roles distribution with graphlets of size three captures some of the dependencies beyond pairwise links. We highlight that we consider all possible motifs from induced 3-node graphlets (i.e., there are overlaps). It is not clear that using higher-order than size three is necessary, but it should be a relatively simple extension to our model.

We also developed a novel methodology for comparing dynamic graph generative models to measure how well they capture: (1) the underlying graph structure distribution, and (2) the node behavior of a real graph over time. In the case of node behavior, using node-aligned metrics over the graph snapshots helps to evaluate the node's topological connectivity and temporal activity. Our use of the Kolmogorov-Smirnov (KS) test with the inter-quartile range, instead of the mean and median values, is an initial effort on adapting graph structure metrics designed for static graphs to the dynamic graph setting. In conclusion, when jointly considering graph structure and node behavior, DYMOND shines in our quantitative evaluation on five different real-world datasets.

ACKNOWLEDGMENTS

This research is supported by NSF under contract numbers CCF-0939370 and IIS-1618690. This work was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344. This document was prepared as an account of work sponsored by an agency of the United States government. Neither the United States government nor Lawrence Livermore National Security, LLC, nor any of their employees makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise does not necessarily constitute or imply its

endorsement, recommendation, or favoring by the United States government or Lawrence Livermore National Security, LLC. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States government or Lawrence Livermore National Security, LLC, and shall not be used for advertising or product endorsement purposes. LLNL-CONF-819670

REFERENCES

[1] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences of the United States of America* 115, 48 (2018), E11221–E11230. https://doi.org/10.1073/pnas.1800683115 arXiv:1802.06916

[2] Ginestra Bianconi, Richard K. Darst, Jacopo Iacovacci, and Santo Fortunato. 2014. Triadic Closure as a Basic Generating Mechanism of Communities in Complex Networks. *Physical Review E* 90, 4 (Oct. 2014), 042806. https://doi.org/10.1103/PhysRevE.90.042806

[3] Deepayan Chakrabarti and Christos Faloutsos. 2006. Graph Mining: Laws, Generators, and Algorithms. *Comput. Surveys* 38, 1 (June 2006), 2–es. https://doi.org/10.1145/1132952.1132954

[4] Fan Chung and Linyuan Lu. 2002. The average distances in random graphs with given expected degrees. *Proceedings of the National Academy of Sciences of the United States of America* 99, 25 (2002), 15879–15882. https://doi.org/10.1073/pnas.252631999

[5] P Erdös and a Rényi. 1959. On random graphs. *Publicationes Mathematicae* 6 (1959), 290–297. https://doi.org/10.2307/1999405 arXiv:1205.2923

[6] Petter Holme. 2013. Epidemiologically Optimal Static Networks from Temporal Network Data. *PLoS Computational Biology* 9, 7 (2013). https://doi.org/10.1371/journal.pcbi.1003142

[7] Petter Holme. 2015. Modern temporal network theory: a colloquium. *European Physical Journal B* 88, 9 (2015). https://doi.org/10.1140/epjb/e2015-60657-4 arXiv:1508.01303

[8] Yuriy Hulovatyy, Huili Chen, and T Milenković. 2015. Exploring the structure and function of temporal networks with dynamic graphlets. *Bioinformatics* 31, 12 (2015), i171–i180.

[9] Brian Karrer and M. E. J. Newman. 2009. Random graph models for directed acyclic networks. *Physical Review E* (2009). https://doi.org/10.1103/PhysRevE.80.046110 arXiv:0907.4346

[10] Bryan Klimt and Yiming Yang. 2004. The enron corpus: A new dataset for email classification research. In *European Conference on Machine Learning*. Springer, 217–226.

[11] Jérôme Kunegis. 2013. KONECT - The koblenz network collection. *Proceedings of the 22nd International Conference on World Wide Web* (2013), 1343–1350.

[12] Guillaume Laurent, Jari Saramäki, and Márton Karsai. 2015. From calls to communities: a model for time-varying social networks. *European Physical Journal B* 88, 11 (2015), 1–10. https://doi.org/10.1140/epjb/e2015-60481-x arXiv:1506.00393

[13] Jure Leskovec, D Chakrabarti, J Kleinberg, C Faloutsos, and Zoubin Ghahramani. 2010. Kronecker graphs: An approach to modeling networks. *Journal of Machine Learning Research* 11 (2010), 985–1042. https://doi.org/10.1145/1756006.1756039 arXiv:0812.4905

[14] Jure Leskovec, Jon Kleinberg, and Christos Faloutsos. 2007. Graph evolution: Densification and shrinking diameters. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 1, 1 (2007), 2.

[15] Jure Leskovec and Andrej Krevl. 2014. SNAP Datasets: Stanford Large Network Dataset Collection. http://snap.stanford.edu/data.

[16] Lun Li, David Alderson, John C Doyle, and Walter Willinger. 2006. Towards a Theory of Scale-Free Graphs : Definition , Properties , and Implications. 2 (2006), 431–523.

[17] R. Milo, S. Shen-Orr, S. Itzkovitz, N. Kashtan, D. Chklovskii, and U. Alon. 2002. Network motifs: Simple building blocks of complex networks. *Science* 298, 5594 (2002), 824–827. https://doi.org/10.1126/science.298.5594.824

[18] Antoine Moinet, Michele Starnini, and Romualdo Pastor-Satorras. 2015. Burstiness and Aging in Social Temporal Networks. *Physical Review Letters* 114, 10 (2015). https://doi.org/10.1103/PhysRevLett.114.108701 arXiv:1412.0587

[19] Sebastian Moreno and Jennifer Neville. 2009. An Investigation of the Distributional Characteristics of Generative Graph Models. *Win* (2009).

[20] Krzysztof Nowicki and Tom a. B Snijders. 2001. Estimation and Prediction for Stochastic Blockstructures. *J. Amer. Statist. Assoc.* 96, 455 (2001), 1077–1087. https://doi.org/10.1198/016214501753208735

[21] Pietro Panzarasa, Tore Opsahl, and Kathleen M. Carley. 2009. Patterns and Dynamics of Users' Behavior and Interaction: Network Analysis of an Online Community. *Journal of the American Society for Information Science and Technology* 60, 5 (2009), 911–932.

[22] Ashwin Paranjape, Austin R Benson, and Jure Leskovec. 2017. Motifs in temporal networks. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*. ACM, 601–610.

[23] N. Perra, B. Gonçalves, R. Pastor-Satorras, and A. Vespignani. 2012. Activity driven modeling of time varying networks. *Scientific Reports* 2 (2012), 1–7. https://doi.org/10.1038/srep00469 arXiv:1203.5351

[24] Rene Pickhardt. 2016. Extracting 2 social network graphs from the Democratic National Committee Email Corpus on Wikileaks. https://www.rene-pickhardt.de/extracting-2-social-network-graphs-from-the-democratic-national-committee-email-corpus-on-wikileaks/. [Online; accessed 14-October-2016].

[25] Julia Preusse, Jérôme Kunegis, Matthias Thimm, Steffen Staab, and Thomas Gottron. 2013. Structural Dynamics of Knowledge Networks. *ICWSM* 17 (2013), 18.

[26] Sumit Purohit, Lawrence B Holder, and George Chin. 2018. Temporal Graph Generation Based on a Distribution of Temporal Motifs. *Proceedings of the 14th International Workshop on Mining and Learning with Graphs* (2018), 7.

[27] Luis E.C. Rocha and Vincent D. Blondel. 2013. Bursts of Vertex Activation and Epidemics in Evolving Networks. *PLoS Computational Biology* 9, 3 (2013). https://doi.org/10.1371/journal.pcbi.1002974

[28] Albert Sunny, Bhushan Kotnis, and Joy Kuri. 2015. Dynamics of history-dependent epidemics in temporal networks. *Physical Review E* 92, 2 (2015), 022811.

[29] Christian L. Vestergaard, Mathieu Génois, and Alain Barrat. 2014. How memory generates heterogeneous dynamics in temporal networks. *Physical Review E - Statistical, Nonlinear, and Soft Matter Physics* 90, 4 (2014), 1–19. https://doi.org/10.1103/PhysRevE.90.042805 arXiv:1409.1805

[30] Bimal Viswanath, Alan Mislove, Meeyoung Cha, and Krishna P. Gummadi. 2009. On the Evolution of User Interaction in Facebook. In *Proceedings of the 2nd ACM Workshop on Online Social Networks*. 37–42.

[31] Stanley Wasserman and Philippa Pattison. 1996. Logit models and logistic regressions for social networks: I. An introduction to markov graphs and p. *Psychometrika* 61, 3 (1996), 401–425. https://doi.org/10.1007/BF02294547

[32] Xin Zhang, Shuai Shao, H Eugene Stanley, and Shlomo Havlin. 2014. Dynamic motifs in socio-economic networks. *EPL (Europhysics Letters)* 108, 5 (2014), 58001.

[33] Qiankun Zhao, Yuan Tian, Qi He, Nuria Oliver, Ruoming Jin, and Wang Chien Lee. 2010. Communication motifs: A tool to characterize social communications. *International Conference on Information and Knowledge Management, Proceedings* (2010), 1645–1648. https://doi.org/10.1145/1871437.1871694

[34] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. 2020. A Data-Driven Graph Generative Model for Temporal Interaction Networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD '20)*. Association for Computing Machinery, New York, NY, USA, 401–411. https://doi.org/10.1145/3394486.3403082

A APPENDIX

Table 4: Statistics of the Dynamic Network Datasets

Dataset	$\|V\|$	$\|E\|$	Unique	Timesteps
Enron Emails	785	5,794	2,517	20
EU Emails	784	68,091	6,878	79
DNC Emails	1,579	33,378	3,911	23
Facebook Wall-Posts	2,245	23,507	4,976	43
CollegeMsg	1,083	34,328	5,589	31

Algorithm 4: `GetActiveNodes`

input: T, N, λ_V
output: V `// active nodes per timestep`

begin
  $V_t \leftarrow \emptyset,\ \forall j \in [1, \ldots, T]$
  for $v \in [1, \ldots, N]$ **do**
    $a \sim Exp(\lambda_V)$ `// sample arrival time`
    for $t \in [a, \ldots, T]$ **do**
      $V_t \leftarrow V_t \cup \{v\}$ `// add to active nodes`
  $V = V_1 \cup \ldots \cup V_T$

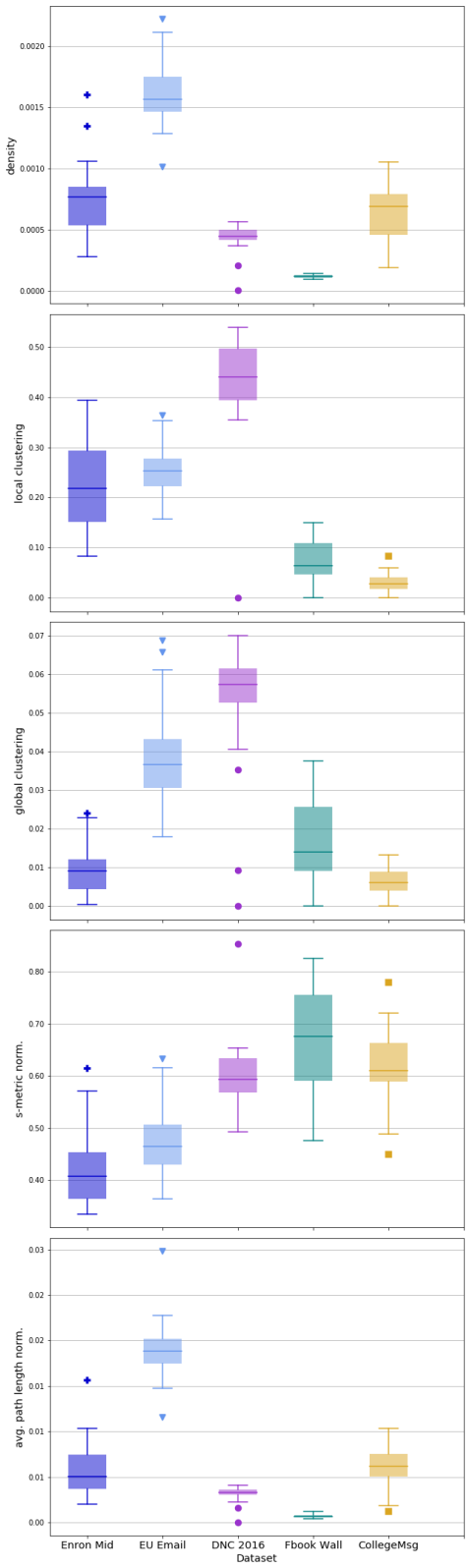


Figure 4: Dataset Graph Structure Metrics

Algorithm 5: SampleMotifTimesteps

input: $t, M_t, M^T, \lambda_M = (\lambda_M^{(1)}, \lambda_M^{(2)}, \lambda_M^{(3)})$
output: M_t^S // motif timesteps

begin
  for $\{u, v, w\} \in M_t$ **do**
    $i \leftarrow M^T(\{u, v, w\})$ // motif type
    // sample inter-arrival rate
    $\beta_M(\{u, v, w\}) \sim Exp(\lambda_M^{(i)})$
    $\lambda_M(\{u, v, w\}) \leftarrow \frac{1}{\beta_M(\{u,v,w\})}$
    $prev \leftarrow t$ // first time motif can appear
    // sample inter-arrival time
    $next \sim Exp(\lambda_M(\{u, v, w\}))$
    while $prev + next < T$ **do**
      // save timestamp to list
      $M_t^S(\{u, v, w\})$.append$(prev + next)$
      $prev \leftarrow prev + next$
      // sample next inter-arrival time
      $next \sim Exp(\lambda_M(\{u, v, w\}))$

Algorithm 6: PlaceMotifEdges

input: M_t, M_t^T, M_t^R
output: M_t^E // edges for motifs in $M^{(i)}$

begin
  for $m \in M_t$ **do**
    if $M_t^T(m) = 3$ **then** // triangle
      $M_t^E(m) \leftarrow \binom{m}{2}$
    else if $M_t^T(m) = 2$ **then** // wedge
      $h \leftarrow M_t^R(m, \text{hub})$
      $s_1, s_2 \leftarrow M_t^R(m, \text{spoke})$
      $M_t^E(m) \leftarrow \{(h, s_1), (h, s_2)\}$
    else if $M_t^T(m) = 1$ **then** // 1-edge
      $e_1, e_2 \leftarrow M_t^R(m, \text{equal2})$
      $M_t^E(m) \leftarrow \{(e_1, e_2)\}$

Algorithm 7: ConstructGraph

input: M, M^S, M^E
output: $\mathbf{G}' = \{G_1', \ldots, G_T'\}$ // where $G_t' = (V_t', E_t', S_t')$

begin
  $V_t' \leftarrow \emptyset$; $E_t' \leftarrow \emptyset$; $S_t' \leftarrow \emptyset$, $\forall t \in [1, \ldots, T]$
  for $\{u, v, w\} \in M$ **do**
    $t' \leftarrow \min(M^S(\{u, v, w\}))$ // first timestep
    $V_t' \leftarrow V_t' \cup \{u, v, w\}$, $\forall t \in [t', \ldots, T]$ // update
    for $t \in M^S(\{u, v, w\})$ **do**
      $E_t' \leftarrow E_t' \cup M^E(\{u, v, w\})$ // place edges
      // add timestamps to edges
      $S_t'((u', v')) = t$, $\forall (u'v') \in M^E(\{u, v, w\})$

Algorithm 8: LearnParameters

input: $\mathbf{G} = \{G_1, \ldots, G_T\}$ // graph to learn from
output: $\widehat{\lambda}_V$ // node arrival rate
$\widehat{\lambda}_M = (\widehat{\lambda}_M^{(1)}, \widehat{\lambda}_M^{(2)}, \widehat{\lambda}_M^{(3)})$ // motif rates distr.
$\widehat{p}_M = (\widehat{p}_M^{(1)}, \widehat{p}_M^{(2)}, \widehat{p}_M^{(3)})$ // proportions motifs
p_R // node roles probabilities
c_R // node roles counts

begin
  Estimate node arrival rate $\widehat{\lambda}_V$ // Eq. 1
  $M, M^T \leftarrow$ GetMotifsGraph($\boldsymbol{G}$)
  for $i \in [1, 2, 3]$ **do**
    Estimate proportions $\widehat{p}_M^{(i)}$ // Eq. 2
  for $\{u, v, w\} \in M$ **do**
    Estimate inter-arrival rate $\widehat{\lambda}_M(\{u, v, w\})$ // Eq. 3a
  for $i \in [1, 2, 3]$ **do**
    Estimate rates of inter-arrivals $\widehat{\lambda}_M^{(i)}$ // Eq. 3b
  $c_R \leftarrow$ GetNodeRoleCounts(M, M^T, M^E, c_R, T)
  for $v \in V$ **do**
    for $r \in R$ **do**
      Estimate role probability $\widehat{p}_R(v, r)$ // Eq. 7

Algorithm 9: GetMotifsGraph

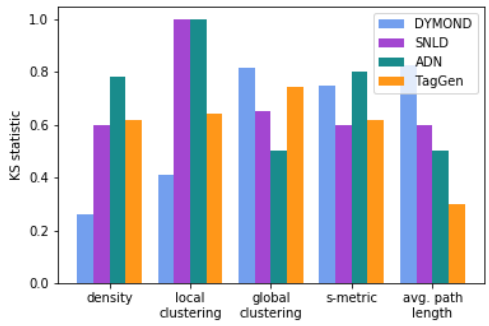
input: $\mathbf{G} = \{G_1, \ldots, G_T\}$
output: M // motifs in $\mathbf{G}$
M^T // motif types

begin
  $M \leftarrow \emptyset$; $M^T \leftarrow \emptyset$
  for $t \in [1, \ldots, T]$ **do**
    for $(u, v) \in E_t$ **do**
      if $u \neq v$ **then**
        for $w \in V_t$ **do**
          if $w \neq u$ **and** $w \neq v$ **then**
            // type i = num. unique edges
            $i = |E_t(\{u, v, w\})|$
            if $\{u, v, w\} \notin M$ **then**
              $M \leftarrow M \cup \{\{u, v, w\}\}$
              $M^T(\{u, v, w\}) \leftarrow i$
            else if $i > M^T(\{u, v, w\})$ **then**
              // prefer higher-order motif types
              $M^T(\{u, v, w\}) \leftarrow i$

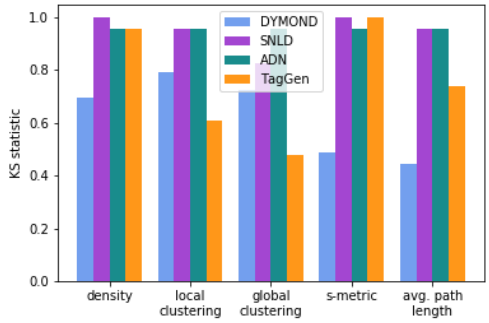
Algorithm 10: GetNodeRoleCounts

input: M, M^T, E, c_R, T
output: $c_R = count(v, r)$ // node role counts

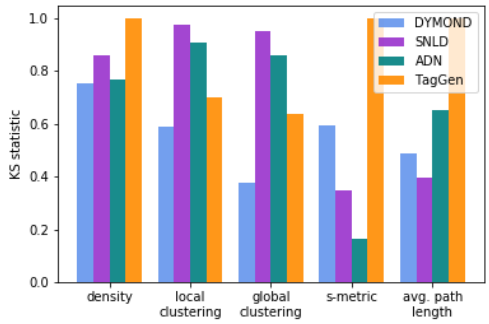
begin
  $count(v, r) = 0,\ \forall v \in V,\ \forall r \in roles$ // init. counts
  for $t \in [1, \ldots, T]$ **do**
    for $(u, v) \in E_t$ **do**
      if $|N^{(3)}(u, v)| > 0$ **then** // triangles
        $role\_weight \leftarrow \frac{min(c_t(u,v), |N^{(3)}(u,v)|)}{\frac{|N^{(3)}(u,v)|}{3}}$
        for $n \in [u, v, w]$ **do**
          $count(n, \text{equal3})$ += $role\_weight$
      if $|N^{(2)}(u, v)| > 0$ ***and*** $r_t^{(2)}(u, v) > 0$ **then**
       // wedges (Eq. 5b)
        $role\_weight \leftarrow \frac{min(r_t^{(2)}(u,v), |N^{(2)}(u,v)|)}{\frac{|N^{(2)}(u,v)|}{2}}$
        for $n \in [u, v, w]$ **do**
          $r \leftarrow$ GetRoleTimestep($E_t, \{u, v, w\}, n$)
          if $r =$ hub **then**
            $count(n, \text{hub})$ += $role\_weight$
          else
            $count(n, \text{spoke})$ += $\frac{role\_weight}{2}$
      if $|N^{(1)}(u, v)| > 0$ ***and*** $r_t^{(1)}(u, v) > 0$ **then**
       // 1-edge (Eq. 5b)
        $role\_weight \leftarrow \frac{min(r_t^{(1)}(u,v), |N^{(1)}(u,v)|)}{|N^{(1)}(u,v)|}$
        for $n \in [u, v, w]$ **do**
          $r \leftarrow$ GetRoleTimestep($E_t, \{u, v, w\}, n$)
          if $r =$ equal2 **then**
            $count(n, \text{equal2})$ += $min(r_t^{(1)}(u, v), |N^{(1)}(u, v)|)$
          else
            $count(n, \text{outlier})$ += $role\_weight$



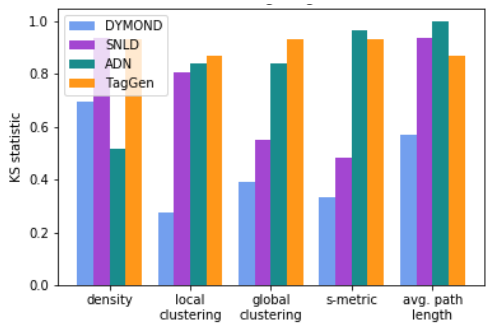
(a) Enron Emails



(b) DNC Emails

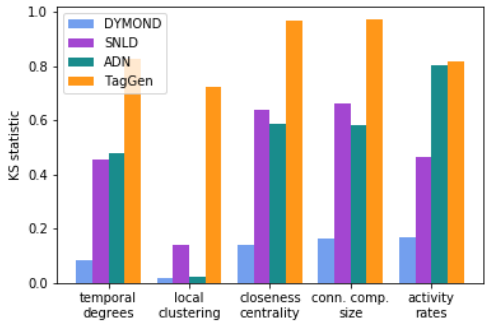


(c) Facebook Wall-Posts

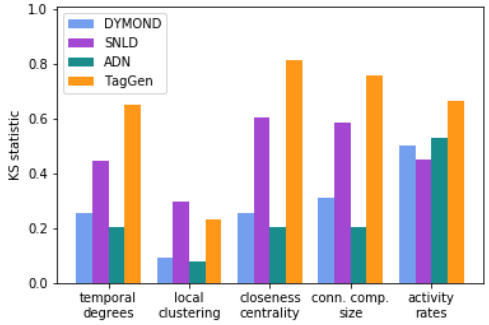


(d) CollegeMsg

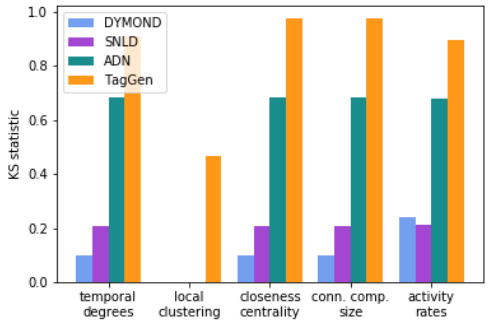
Figure 5: KS Statistic of Graph Structure Metrics



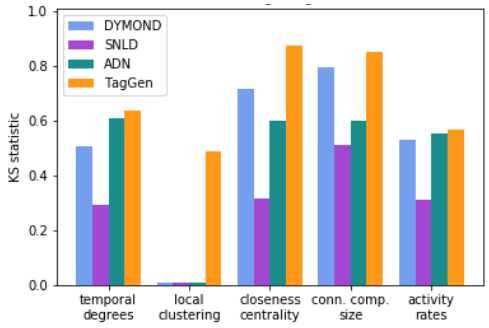
(a) Enron Emails



(b) DNC Emails



(c) Facebook Wall-Posts



(d) CollegeMsg

Figure 6: KS Statistic of Node Behavior Structure Metrics