Abstract
Dynamic interaction data is often aggregated in a sequence of network snapshots before being employed in downstream analysis. The two common ways of defining network snapshots are i) a fixed time interval or ii) fixed number of interactions per snapshot. The choice of aggregation has a significant impact on subsequent analysis, and it is not trivial to select one approach over another for a given dataset. More importantly, assuming snapshot regularity is data-agnostic and may be at odds with the underlying interaction dynamics.

To address these challenges, we propose a method for community-aware detection of network states (CADENCE) based on the premise of stable interaction time-frames within network communities.

We simultaneously detect network communities and partition the global interaction activity into scale-adaptive snapshots where the level of interaction within communities remains stable. We model a temporal network as a node-node-time tensor and use a structured canonical polyadic decomposition with a piece-wise constant temporal factor to iteratively identify communities and their activity levels. We demonstrate that transitions between network snapshots learned by CADENCE constitute network change points of better quality than those predicted by state-of-the-art network change point detectors. Furthermore, the network structure within individual snapshots reflects ground truth communities better than baselines for adaptive tensor granularity. Through a case study on a real-world Reddit dataset, we showcase the interpretability of CADENCE motivated snapshots as periods separated by significant events.

1 Introduction
Temporal interaction data from various domains are often collected at high temporal resolution. Examples include comment exchanges among Reddit users [19], physical contact detected by wearable RFID tags [4] and source-to-destination trips in ride sharing or taxi services [1]. While individual pair-wise events may be recorded at milliseconds granularity, many algorithms for downstream analysis including deep learning on dynamic graphs [14, 20, 28], tensor decomposition [9, 17], and evolutionary clustering [5] typically expect that interactions are aggregated into network snapshots.

Defining appropriate snapshots without supervision is not trivial. Existing works that use temporal interaction data define snapshots based on regular temporal intervals (e.g., hourly, daily, weekly) or based on a fixed number of interactions within a snapshot [14]. However, this assumes regularity that may not align well with the natural evolution of within-community interaction activity which could speed up, slow down or abruptly change due to global events affecting the network.

In this work we propose a data-driven approach to partition dynamic interaction data into network snapshots based on periods of stable activity within natural network communities. Fig. 1 presents an illustrative example of the key assumptions behind our methodology. We model temporal interactions as a 3-way node-node-time tensor. We assume that the majority of observed interactions are generated within communities of nodes, \( C_i \), which may overlap. This assumption is in line with affiliation generative models [29] and proposed extensions to temporal communities [9].

In this work we propose a data-driven approach to partition dynamic interaction data into network snapshots based on periods of stable activity within natural network communities. Fig. 1 presents an illustrative example of the key assumptions behind our methodology. We model temporal interactions as a 3-way node-node-time tensor. We assume that the majority of observed interactions are generated within communities of nodes, \( C_i \), which may overlap. This assumption is in line with affiliation generative models [29] and proposed extensions to temporal communities [9].

We also assume that the network transitions between temporal states (or time-frames) during which interaction patterns remain stable [18]. Thus, network snapshots are defined over periods of stable levels of interaction within communities. The points in time where interaction patterns shift are change points, which can be triggered by regular day/night or week/weekend boundaries, but also global events such as storms in...
transportation networks or sports events in social communities of sport fans. The example interaction tensor from Fig. 1 has 5 network states $T_1$ to $T_5$ with the majority of interactions residing in the colored sub-tensors corresponding to communities during a network state. Given a high-resolution interaction tensor, our goal is to identify network snapshots corresponding to stable activity states of underlying network communities.

We propose a method for community-aware detection of network states (CADENCE) based on the premise of stable interaction regimes within stationary network communities. CADENCE detects the underlying communities and partitions the global interaction activity into scale-adaptive snapshots in which community interactions levels remain stable. We derive a scalable solution based on structured non-negative tensor decomposition in which the temporal factor is sparse-coded via a multi-resolution over-complete dictionary. CADENCE’s snapshot transitions correspond to change points of better quality than those predicted by state-of-the-art network change point detectors and the network structure within individual snapshots reflects ground truth communities better than baselines for adaptive tensor granularity. Through a case study on a Reddit dataset, we also showcase the interpretability of snapshots as periods separated by significant sports events.

Our contributions in this paper are as follows:

- **Novelty:** We propose CADENCE: a community-aware dynamic network state detector, which to the best of our knowledge, is the first method for unsupervised adaptive aggregation of interactions into snapshots based on periods of stable community activity.
- **Accuracy and Scalability:** CADENCE identifies ground truth network states up to 40% more accurately than state-of-the-art methods. It scales to hundreds of thousands of nodes and millions of time steps when interactions are considered at high temporal resolution.
- **Applicability:** The network states discovered by CADENCE correspond to network snapshots in which ground truth communities have small conductance and the majority of interactions residing in the colored sub-tensors from Fig. 1 has 5 network states.

2 Related work

**Tensor decomposition** extends matrix decomposition to multi-way data. The Canonical Polyadic Decomposition (CPD) (or PARAFAC) represents a tensor as a sum of three way outer products of factors $\mathbf{A}$, where Tucker decomposition involves a core tensor in addition to factor matrices $\mathbf{T}$. Extensions of the basic CPD model regularize the temporal factor to enforce bursty periodic behavior and smooth temporal behavior. The assumptions of smoothness in $\mathbf{A}$ is similar to our idea of network states, but smoothing is applied to each temporal factor independently, and thus, the resulting fit cannot be interpreted as global network snapshots.

**Change point detection** in dynamic graphs focuses on anomalous time points in which interaction trends change significantly. Some approaches employ Bayesian frameworks to model interactions and declare change points when model parameters change significantly [23]. Others assume a Hidden Markov Model (HMM) generating interactions and detect change points based on changes in the HMM states [22]. A third group tracks the eigenspectrum of the snapshots’ graph Laplacians [12, 13]. The method in [15] employs a standard CPD decomposition but applies post processing of the fitted temporal factor to detect change points. We compare our method to those from [12] and [15] and demonstrate its superior performance in detecting ground truth change points.

**Snapshot formation** for dynamic network data seeks to define an optimal resolution for aggregation. The method in [24] aggregates sequences of streamed edges into graphs optimizing various network properties. Other approaches detect optimal resolution based on assumptions for underlying information cascade and network growth models [7, 6]. The authors of [27] utilize nonuniform time slices of dynamic graphs to improve visualization. The method in [14] demonstrated that using a constant number of edges per network snapshot can improve link prediction accuracy. The work closest to ours seeks to aggregate a high resolution temporal tensor into frames based on temporally-local goodness heuristics [21]. We compare to the methods from [14] and [21] in our experimental evaluation.

3 Problem formulation

The input to our problem is a set of time ordered (potentially weighted) interaction triplets $(v_i, v_j, t_s, w_{ij,s})$, where $v_i, v_j \in V$ are nodes from a finite node set $V$ of size $|V| = n$, $t_s \in [0, t]$ is a timestamp of the interaction measured at some high resolution (e.g., milliseconds) and $w_{ij,s}$ is the weight (or strength) of the interaction. Note that the weight can also correspond to a count of interactions if the data is pre-aggregated at some level. We model such data as a 3-way tensor $X \in \mathbb{R}^{n \times n \times t}$.

Instead of assuming fixed and regular network snapshots [14], our key assumption is that snapshots correspond to states of the underlying network communities during which the level of within-community interactions is stable. A sketch of this intuition is presented in Fig. 1. The colored sub-tensors correspond to within-community interactions of a stable level. Multiple communities can be active at different levels during each state. The transitions between states can be viewed as change points (red partitions of the temporal mode)
caused by a global event that affects levels of activity in communities.

Given raw interaction data, our goal is to identify:
i) overlapping interaction communities of nodes,
ii) network states of stable activity, and
iii) the levels of community interaction during states.

To operationalize the above intuition we adopt a structured low-rank and non-negative tensor decomposition model \[ (3.1) \] with a basic form as follows:

\[
\mathcal{X} = [U, V, W] \\
\text{s.t. } U, V \geq 0,
\]

where the \( U \in \mathcal{R}^{n \times k} \) and \( V \in \mathcal{R}^{n \times k} \) factor matrices represent \( k \) overlapping node communities with column loadings that can be interpreted as association strength of each node to the corresponding community. The factor matrix \( W \in \mathcal{R}^{t \times k} \) represents the temporal activity trend for each community. The notation \( [U, V, W] \) stands for a tensor product of factor matrices that produces a rank-\( k \) tensor \[ (3.1) \].

The basic CPD model from Eq. \[ (3.1) \] allows for arbitrary interaction levels over time in \( W \) and does not readily model temporal states delineated by change points, with piece-wise constant levels within a state (Fig. 1). To this end, we model the temporal factor \( W \) as a product of i) an over-complete aggregation dictionary matrix \( H \in \mathcal{R}^{k \times m} \) which expresses all possible states (contiguous intervals of time) and ii) an encoding matrix \( A \in \mathcal{R}^{m \times k} \) which encodes the levels of community interactions in a given state.

A sketch of the structured temporal factor \( W \) is presented in Fig. 2. The non-zero positions of column atoms in the aggregation dictionary \( H \) represent all contiguous intervals that may correspond to networks states (colored time segments in \( H \)). In this example, the timeline is partitioned into 4 states (or snapshots) with durations \( T_1 = 1, T_2 = 2, T_3 = 3 \) and \( T_4 = 2 \). Their corresponding atoms in \( H \) are colored red and the community interaction levels are encoded in corresponding non-zero rows of matrix \( A \).

For \( W \) to encode network states that do not overlap in time, \( H \) atoms selected for encoding should be mutually orthogonal and span the timeline. Let \( H_{A, \neq 0} \) denote the principle sub-matrix of \( H \) comprised of columns with non-zero corresponding rows in a learned encoding \( A \). Then a valid encoding should satisfy \( H_{A, \neq 0}^TH_{A, \neq 0} = I \), assuming that all atoms in \( H \) have a norm of 1.

Based on the above definition we can formalize our problem as a constrained non-negative tensor factorization as follows:

\[
\min_{U,V,A} \frac{1}{2} ||X - [U, V, HA]||_F^2 \\
\text{s.t. } U, V \geq 0, \quad ||A||_0 < \theta, \\
H_{A, \neq 0}^TH_{A, \neq 0} = I,
\]

where the minimization term seeks a low-rank representation of the input interaction tensor \( X \) and the constraints ensure that i) the community factors \( U, V \) are non-negative, ii) the encoding \( A \) via the over-complete dictionary \( H \) is sparse employing a bounded \( L_0 \) sparsity norm and iii) the atoms of \( H \) used to encode interaction levels in \( A \) form an orthonormal basis.

4 Optimization framework

The overall optimization problem in Eq. \[ (3.2) \] in structure closely resembles non-negative CPD tensor factorization prompting potential solutions based on well-established alternating least-squares methods \[ [3] \]. However, the constraint on selected atoms from \( H \) in the encoding of the temporal factor is in essence combinatorial. Relaxations of this constraint will render our main goal of detecting temporal network states unachievable as it is unclear how to reconcile a partitioning of the timeline from overlapping atoms selected for encoding. Another challenge is the exhaustive nature of the atoms in \( H \) whose number grows quadratically with the length of the original timeline \( t \) (see Fig. 2). To address the above challenges, we combine ideas from non-negative tensor factorization for the community factors and sparse dictionary coding via Orthogonal Matching Pursuit (OMP) \[ [22] \] for the temporal factor. Specifically, to optimize equation \[ (3.2) \] we adopt an alternating optimization scheme that updates each of the variables \( U, V \) and \( A \) while keeping the other two fixed.

Solutions for community factors \( U \) and \( V \). Both community factors can be updated in the same manner employing a least squares procedure as they have the same role in the optimization function and share the same constraint. Furthermore, when the input interactions are undirected (i.e., \( w_{i,j,s} = w_{j,i,s} \)) the two factors should converge to be the same. For the sake of com-
The sub-problem with respect to $U$ is:

$$
\begin{aligned}
(4.3) \quad \min_{U} & \frac{1}{2} ||X_U^T - U[(HA) \odot V]^T||_F^2, \text{ s.t. } U > 0,
\end{aligned}
$$

where $X_U^T$ is the tensor unfolding on the updated mode $U$ and $\odot$ is the Khatri-Rao product \[11\]. Setting the gradient of Eq. (4.3) with respect to $U$ to 0 and letting $B = [(HA) \odot V]$ we obtain:

$$
(4.4) \quad -X_U^T B + U B^T B = 0
\quad U = X_U^T B (B^T B)^{-1}
$$

To ensure non-negative entries in $U$, we replace negative values with zero in line with ALS solutions for non-negative CPD \[9\]. The solution for $V$ is analogous to that for $U$.

**Solution for $A$.** When the community factors $U, V$ are fixed, the problem with respect to $A$ with only the sparsity constraint is similar to sparse coding by an over-complete dictionary $H$:

$$
(4.5) \quad \min_{A} \frac{1}{2} ||X_W^T - HA[V \odot U]^T||_F^2, \text{ s.t. } ||A||_0 < \theta,
$$

where $X_W^T$ is the unfolding of the input tensor on its temporal mode. The Orthogonal Matching Pursuit (OMP) \[22\] has been widely adopted in signal processing for sparse coding problems. Within OMP, atoms from $H$ are iteratively selected based on their alignment with the residual of the signal not represented by previously selected atoms. In order to formulate Eq. (4.5) such that an OMP-like algorithm can be employed, we introduce an intermediate variable $W = HA$:

$$
(4.6) \quad \min_{W} \frac{1}{2} ||X_W^T - W[V \odot U]^T||_F^2, \text{ s.t. } W \approx HA,
$$

where the 0-gradient solution with respect to $W$ just like in ALS is:

$$
(4.7) \quad W = X_W^T (V \odot U)^T (U^T U \odot V^T V)^{-1} \approx HA.
$$

In other words, we can approximate an unconstrained temporal factor $W$ by a sparse coding solution $HA$.

**Algorithm 1 O^2MP: sparse orthogonal atom coding**

**Input:** Temporal factor $W$, Budget of snapshots $b_n$

**Output:** Selected atom and sparse codes $H_T, A_T$

1: $H_T = \{1_{1 \times 1}\}$
2: $C = \{\text{atom indices of prefixes and suffixes of } 1_{1 \times 1}\}$
3: $S = \emptyset$
4: for $i = 2$ to $b_n$ do
5: // OMP encoding and residual update
6: $A_T = (H_T^T H_T)^{-1} H_T^T W$
7: $R = W - H_T A_T$
8: // OMP-like atom scoring and selection
9: Add $(c, S_c = \sum_{i=1}^k |h_i^T R_i|)$ to $S$, $\forall c \in C$
10: Let $m$ be the atom index of maximum $S_m$
11: Let $h_m$ be the atom at index $m$
12: Let $h_s$ be the atom enclosing $h_m$
13: Let $h_m$ be the atom in $C$ satisfying $h_m = h_s - h_m$
14: // Dictionary Updates
15: $H_T = H_T \setminus h_s$
16: $H_T = H_T \cup h_m \cup h_m$
17: $C = \{\text{atom indices of prefixes and suffixes of } h_m \text{ and } h_m\}$
18: Remove $(m, S_m)$ and $(m, S_{\bar{m}})$ from $S$
19: end for

The only remaining challenge is the additional constraint $H_{A_i \neq 0} H_{A_i \neq 0} = I$ from Eq. (3.2) on using orthogonal atoms from $H$. Since our dictionary $H$ contains orthogonal atoms corresponding to non-intersecting intervals of time, we restrict the greedy selection of new atoms in the OMP steps to ensure orthogonality of the candidate set.

We call the resulting algorithm O^2MP (Alg. \[1\]) as it ensures both orthogonality of the representation and orthogonality of selected atoms. Intuitively O^2MP partitions the timeline in a top-down manner by iteratively selecting atoms that ensure best representation of the residual of the signal based on its current representation. This process is demonstrated pictorially in Fig. \[3\]. At any point we maintain a set of non-overlapping atoms in $H_T$ starting with the atom spanning the whole time line. Atoms are represented by horizontal time segments which correspond to their non-zero elements Fig. \[3\]. The atoms in the current solution $H_T$ are represented with red segments that span the whole timeline and are orthogonal to each other as they do not overlap, i.e., $H_T$ satisfies the orthogonality constraint $H_T^T H_T = I$. Candidate atoms considered in the next iteration are prefixes and suffixes of intervals corresponding to atoms in $H_T$. For example, the candidates considered due to $h_s \in H_T$ are depicted below $h_s$ (grey and green atoms). Assuming that candidate atom $h_m$ is selected as the one that best improves the current representation, we substitute its enclosing solution interval $h_s$ with $h_m$ and its complement $h_{\bar{m}} = h_s - h_m$ in the solution set $H_T$.

We list all steps of O^2MP in Alg. \[1\]. The algorithm takes as an input the temporal tensor factor $W$ to
approximate via the dictionary encoding and a budget of $b_s$ snapshots to create which corresponds to the number of atoms to select for encoding. The output is the selected atoms $H_T$ and a corresponding encoding $A_T$. We initialize the selected atom set $H_T$ with the last atom in $H$, namely a vector of all ones spanning the whole timeline (Step 1). Note, that atoms in $H$ in our problem formulation are norm one. However, in our implementation we simply focus on orthogonality, i.e. considered atoms correspond to time intervals of value 1. We use atoms and intervals interchangeably.

In Step 2 we initialize the set of candidate atom indices with those of all prefix and suffix intervals of the all-one vector $1_{1\times t}$. An example of all prefixes and suffixes of an interval $h_s$ is shown in Fig. 3. Intuitively, these are all pairs of intervals that split an enclosing interval in two. We also initialize an empty priority queue $S$ of ordered pairs $(i, S_i)$, where $i$ is an atom index and $S_i$ is an alignment score of the interval used as a priority value. Each iteration of the main loop of the algorithm (Steps 5-18) replaces an atom $h_s \in H_T$ from the current solution with two atoms $h_m$ and $h_{\bar{m}}$ that split $h_s$ in two parts, i.e. $h_s = h_m + h_{\bar{m}}$. An example of such triplet of intervals is presented in Fig. 3. We first obtain the best encoding $A_T$ via the set of currently atoms $H_T$ (Step 6) and quantify the unrepresented residual $R$ of $W$ (Step 7). The inversion in Step 6 involves a matrix stacking only the selected atoms so far (at most $b_s \times b_s$ matrix) and not the complete dictionary $H$. Moreover, $H_T^T H_T$ is diagonal and thus its inverse involves finding the reciprocal of $b_s$ scalars.

Next we compute alignment scores $S_c$ for all atoms whose indices $c \in C$ are in the candidate set and add them to the priority queue $S$ (Step 9). The alignment score $S_c = \sum_{i=1}^{k} |h_i^T R_{si}|$ accumulates absolute values of inner products of the atom with columns of the residual. Intuitively, atoms aligned with the residual of each factor are preferred for encoding. This is also an essential step of OMP for sparse coding.

Next we select the atom $h_m$ with the largest alignment score $S_m$ among candidates (top of the the priority queue $S$). We also identify $h_m$’s unique enclosing atom $h_s \in H_T$ from the current solution and its complement $h_{\bar{m}} = h_s - h_m$ (Steps 10-13). We substitute $h_s$ with $h_m$ and $h_{\bar{m}}$ in the current solution $H_T$ (Steps 15-16) and create a new set of candidates $C$ to be scored comprising all prefixes and suffixes of both $h_m$ and $h_{\bar{m}}$ (Step 17). Finally, we remove the entries for $h_m$ and $h_{\bar{m}}$ from the priority queue $S$.

The complexity of the algorithm is $O(b_s (tk + b_s))$. The maintenance of the priority queue $S$ is logarithmic in $t$ assuming binary heaps as its implementation. The $tk$ factor in each iteration is due to the scoring of $O(t)$ new candidates from $C$ in Step 9. When the budget number of partitions is constant with respect to $t$, the algorithm is linear in the matrix $W$ it approximates. As we demonstrate experimentally, the overall CADENCE algorithm runs in minutes on sparse tensors with up to 100k nodes and 35k timestamps.

5 Experimental evaluation

We compare CADENCE to state-of-the-art baselines in terms of quality of snapshots, change point detection and node classification on dynamic graphs.

5.1 Datasets. We employ synthetic and three real-world datasets for evaluation with summary statistics listed in Tbl. 4 and detailed descriptions included in the extended version[4]. The synthetic $Syn$ dataset includes 30 stochastic block communities (sizes 15-25) whose network activity level changes over 50 temporal states of varying length. The Reddit $F1$ dataset contains interactions between active users in a Formula 1 racing subreddit ($r/formula1$) [19]. We align it to the 2019 Formula 1 Grand Prix calendar to define ground truth change points, at the start and end dates of races. The $Taxi$ dataset represents rides between neighborhoods in New York City in 2017. Nodes are neighborhoods and edges correspond to rides. We use working versus non-working days (including holidays) as the ground truth change points (e.g., most Saturdays are change points). Both Reddit $F1$ and Taxi do not feature ground truth community membership. We also employ the $Hospital$ dataset for experiments in Sec. 5.3.5.5.

Table 1: Summary statistics of datasets (see Section 5.1) and performance on the change point detection task in terms of covering metric (Section 5.4). #e denotes number of edges, #CP denotes the number of changepoints, and #C is the number of communities.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>n</th>
<th>t</th>
<th>#e</th>
<th>#CP</th>
<th>#C</th>
<th>Resolution</th>
<th>CADENCE</th>
<th>LAD</th>
<th>TSP-Dif</th>
<th>TSP-LOF</th>
<th>AGT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syn</td>
<td>500-600</td>
<td>50-2500</td>
<td>3M</td>
<td>49</td>
<td>30</td>
<td>NA</td>
<td>1.00</td>
<td>0.70</td>
<td>0.49</td>
<td>0.34</td>
<td>0.20</td>
</tr>
<tr>
<td>Taxi</td>
<td>266</td>
<td>8763</td>
<td>25M</td>
<td>106</td>
<td>/</td>
<td>24h-1min</td>
<td>0.72</td>
<td>0.60</td>
<td>0.47</td>
<td>0.53</td>
<td>0.14</td>
</tr>
<tr>
<td>Reddit F1</td>
<td>100852</td>
<td>365</td>
<td>2M</td>
<td>42</td>
<td>/</td>
<td>Daily</td>
<td>0.81</td>
<td>0.71</td>
<td>0.71</td>
<td>0.42</td>
<td>0.29</td>
</tr>
<tr>
<td>Hospital</td>
<td>75</td>
<td>17383</td>
<td>64848</td>
<td>/</td>
<td>5</td>
<td>20 Secs</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
<td>/</td>
</tr>
</tbody>
</table>

4 An extended version of the paper is available at https://www.cs.albany.edu/~petko/lab/papers/mmtb2023sdm.pdf
and administrators.

5.2 Experimental setup We next discuss briefly evaluation metrics and baselines and provide further details including definition for metrics and parameter tuning for baselines in the extended version.

Metrics. We employ conductance to measure the quality of snapshots identified by CADENCE and competitors with respect to ground truth communities [16]. To evaluate the quality of our state identifications we utilize the covering metric defined in [26]. Perfectly aligned predictions result in a value of 1. For the task of node classification, we employ commonly used metrics, namely the F1 measure and accuracy.

Baselines: For quality of snapshots and node classification we compare to the following approaches: regular aggregation in time (Reg) (e.g., every hour is a snapshot), constant number of edges per snapshot (Cons) and Adaptive Granularity in Time Evolving Graphs as Tensors (AGT) [21] which aggregates temporal slices greedily into a single snapshot. For change point detection and scalability we compare to AGT [21], LAD [12] which is a state-of-the-art change point detector for dynamic graphs based on the eigenspectra of consecutive snapshots; and Tensorsplat [13] which performs a CPD decomposition on the tensor to obtain a temporal representation of community activity. To identify change points in this temporal representation TSP-LOF follows the approach used in [12] and utilizes local outlier factor [2] while TSP-DIF, measures the absolute difference between consecutive time steps.

5.3 Snapshot quality. We measure the conductance of ground truth communities in Synthetic and Hospital within each snapshot obtained by competing temporal aggregations. Fig. 4 shows how many communities satisfy a given conductance threshold across all snapshots. Although the total number of snapshots are similar for all methods, CADENCE produces more snapshots at low conductance thresholds. Its snapshots exhibit stronger community structure than competing methods because our aggregation is based on finding stable regions of community activity. Naturally, since states correspond to periods of community activity (and thus have more internal edges), our aggregation produces snapshots following the same principle.

5.4 Change Point Detection. We next quantify the level of alignment of predicted state transitions with ground-truth change points in real-world networks. We compare the covering metric of solutions obtained by all competitors on Syn, Taxi, and Reddit F1 with respect to ground truth change points. First, in Tbl. 1 we list the maximum covering metric obtained by all methods. CADENCE exceeds the performance of all baselines. Because the right number of change points is not often known, we are also interested in the trend of covering metric for competitors as a function of the number of predicted change points. TSP-LOF and AGT are not appropriate for such a task as they do not provide ranking of predicted change points. For CADENCE we take the first $k$ state predictions, and for TSP-DIF and LAD we take the top $k$ ranked change points. The prediction quality of CADENCE consistently increases as the number of predicted states approaches the ground truth (Fig. 5). In contrast, other approaches stagnate or even decrease in performance as the number of predicted states increases.

Finally, we examine how CADENCE and competitors predict change points when the data is given at varying temporal resolutions. To this end, we represent the Taxi dataset at different granularities ranging from daily (24h) down to 30 second intervals (30sec). Results are presented in Fig. 5(d). CADENCE’s performance is remarkably consistent, indicating that it finds relatively similar change points regardless of the temporal resolution of the input tensor. Even at 30 second resolution which constitutes over a million timesteps CADENCE is able to find the ground truth change points with high accuracy. In contrast competitors’ performance vary widely between different resolutions without clear trends. Note that LAD and AGT were not able to complete for resolutions higher than 30 min and 30 sec resolutions due to scalability limitations (see Sec. 5.6).

5.5 Node Classification. Node classification is a central task in (deep) learning on graphs, and is also a relevant downstream task for gauging the quality of temporal aggregation methods for learning on dynamic graphs. In this experiment we demonstrate that a dynamic network represented by CADENCE’s snapshots allows for better performance as compared to those by competing approaches. Specifically, we employ AdaNN [28] (with default parameters) as the down-
stream framework for node classification. This method uses attention in a neural network to model temporal and spatial information, simultaneously. The focus of this comparison is on the synthetic and Hospital datasets, as they have ground truth class labels.

We compare the quality of node classification for competing methods by remove varying percentages of class labels to use as testing. In each case, 10% of the labels are reserved for validation the remaining labels are used for training. The results are shown in Fig. 5 presented in terms of both the F1 score and accuracy. CADENCE in both datasets either matches or exceeds baselines in all measures across all parameter settings. We expect that this is because AdaNN can utilize community structure in snapshots to better represent the relationship between edges and node labels. As demonstrated in Sec. 5.3 snapshots based on the states identified by CADENCE create a dynamic network representation with strong community structure.

5.6 Scalability. We evaluate the scalability of CADENCE and competing approaches on real world datasets in terms of number of nodes, and number of timesteps using the parameters from Sec. 5.4. Results can be seen in Fig. 8. We use the largest dataset across the changed dimension (i.e., Reddit F1 for number of nodes and Taxi for number of timesteps) for this comparison.

To evaluate the impact of the number of nodes on running time we randomly subset the Reddit F1 dataset to 1000, 10000, and 100000 nodes (Fig. 8(a)). CADENCE scales similarly to TSP, taking only seconds to process the dataset even at its largest size. In contrast, for AGT and LAD the running time increases substantially as the size of the data increases. We perform a similar experiment in Fig. 8(b), now considering the Taxi dataset at varying temporal resolutions. In this instance, TSP’s running time actually exceeds that of CADENCE although both methods are based on CPD. We suspect that CADENCE has faster convergence because its temporal representation is simpler, having only as much temporal variation as the number of states, and thus converges to the fit criteria in fewer iterations.

5.7 Case Study: Reddit F1. To demonstrate the interpretive power of CADENCE we perform a case study on the Reddit F1 Dataset. We utilize the same parameters as those used from Sec. 5.4. Figure 7 illustrates the Reddit F1 daily interaction volume and the stable community activity states detected using CADENCE. Our method consistently identifies distinct Grand Prix states, with few exceptions.

Highlighted above the time series are three network snapshots identified by CADENCE that correspond to the stable activity states prior to, during, and following a typical Grand Prix. These states cover five, two, and thirteen days of activity on r/formula1, respectively. In the network visualizations, nodes are laid out using a force-directed algorithm that draws interacting users together and sends non-interacting users apart. Many nodes appear in more than one snapshot,
### 6 Conclusion

In this paper we introduced a framework for identifying states of stable network activity in dynamic networks called CADENCE. It frames a dynamic network as an interaction tensor and utilizes canonical polyadic decomposition with a piece-wise constant temporal factor to decompose an input interaction tensor of potentially high temporal resolution. The decomposition simultaneously detects network communities and partitions the global interaction activity into scale-adaptive snapshots in which community interactions levels remain stable.

We demonstrated that across multiple data sets the snapshots detected by CADENCE reflect the community structure and known change points, and facilitate increased performance in the task of node classification. Furthermore, these benefits do not come at the cost of scalability. CADENCE was able to process dataset with hundreds of thousands of nodes and up to a million time steps in less than 10 minutes. It scales similarly or better than the fastest available baselines and produces snapshots of the best quality compared to those of all baselines. We also demonstrated that discovered states are highly interpretable, capturing windows of diverse network activity associated with real world events in a case study from Reddit. The code for our method is available at [http://www.cs.albany.edu/~petko/lab/code.html](http://www.cs.albany.edu/~petko/lab/code.html).

### 7 Acknowledgements

The work was supported by the NSF Smart and Connected Communities (SC&C) grant CMMI-1831547.

### References


8 Supplemental material

In the supplemental content we add details additional details. First, an in depth explanations of the datasets, then formally define used metrics. Finally, we provide detailed explanations of competitors and discuss how their hyperparameters were set.

8.1 Dataset Details: Synthetic data generation details

The synthetic data, abbreviated Syn, consists of a series of disaggregated stochastic block model graphs stacked into a interaction tensor. In total there are 30 block communities ranging from in size from 15 to 25 nodes. Each community has a 30% chance of becoming "active" during a particular timestep. When a community is "active" 20% of possible internal community edges randomly are assigned and the community has no external edges. When a community is inactive every node has a 20% chance of linking with any node not in a active community. This process is repeated 49 times to create 49 base graph snapshots. Each graph snapshots is then randomly split into 1 to 50 desegregated snapshots. This disaggregation process uniformly in quantity but randomly in edge index distributes the original edges into the selected number of new snapshots. The change points in Syn reflect the time points where the graph snapshot which was disaggregated into said timepoints changes.

Real world details: The Reddit F1 dataset contains interactions between users active on an online forum about Formula 1 racing (r/formula1), extracted from 2019 Formula 1 Grand Prix calendar to generate ground truth change points, at a high level, conductance approaches 0 when a community is well separated from the rest of the graph and approaches 1 when it is well connected to the rest of a graph.

To evaluate the quality of our state identifications in we utilize the covering metric defined in 20:

\[ C(\Theta, \tilde{\Theta}) = \frac{1}{T} \sum_{A \in \Theta} |A| \max_{\tilde{A} \in \tilde{\Theta}} J(A, \tilde{A}) \]

Where \(\Theta\) and \(\tilde{\Theta}\) are the ground truth and predicted change points, \(A\) and \(\tilde{A}\) represent the sets of time points which constitute states, and \(J(A, \tilde{A})\) is the Jaccard index of these sets. Perfectly aligned predictions will result in a value of 1.

8.3 Details on Baselines and Parameter Settings. Baselines: We compare CADENCE to approaches designed for two different tasks. The first is tensor aggregation. These approaches are designed to aggregate temporal slices into a smaller number of representative snapshots. We compare to these approaches on the task of producing network snapshots with interpretable community structure as well as node classification, a common downstream task.

Reg: regular aggregation of time points (e.g., every hour is a snapshot). Cons: constant number of edges. In this approach every snapshot has the same number of edges. This aggregation strategy was used in 14 to achieve state-of-the-art performance for the task of link prediction task. Adaptive Granularity in Time Evolving Graphs as Tensors (AGT) 21, seeks to learn temporal aggregations of a tensor that are smaller along the temporal dimension. This approach greedily aggregates
temporal slices into a single snapshot until a heuristic is met, at which point a new slice is formed and the process is repeated until the last time step is reached. We choose to compare against the so-called infinity-Norm heuristic, empirically shown to perform best[21]. In our network statistic experiments we set the AGT threshold to produce a similar number of snapshots to competitors, and use default parameters otherwise.

The second perspective we compare to is that of change point detection. These methods are not concerned with producing informative snapshots, focusing instead on finding changes in the structure of the dynamic network occurring at particular points in time.

LAD [12] is a state-of-the-art method for change point detection in dynamic graphs which compares the eigenvalues of the graph Laplacian, over time, to quantify how much a graph has changed from its "normal state". The position of the top $k$ values from this quantification are determined to be change points. We compare to [12] in our state detection experiments, employing a grid search across its parameters. Tensorsplat [15] performs a CPD decomposition on the representative tensors to obtain a temporal representation of community activity similar to our approach. However, the authors [15] do not offer an explicit way of identifying change points, thus we explore two different ways of identifying change points based on extracted features of Tensorplat (TSP).

TSP-LOF follows the approach used in [12] and utilize local outlier factor [2] to identify anomalies on the learned temporal representation. TSP-Dif, measures the absolute difference between consecutive time steps in its learned temporal representation. The $k$ timepoints that represent the greatest absolute difference are identified as change points. We compare to both TSP-LOF and TSP-Dif in state detection experiments and grid search parameters for both variations.

The AGT approach to tensor aggregation is also included in state-detection comparison, since its heuristics may also be able to accurately capture ground truth changes. However, we do not compare to Reg and Cons on this task. These approaches rely on simple statistics (number of timestamps, number of edges) to define states and produce snapshots, meaning they are agnostic to the structural information held in the tensor.

The optimal parameters found and used in change point detection experiments are included in Table 2.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>K</th>
<th>K</th>
<th>#N</th>
<th>Con</th>
<th>SW</th>
<th>LW</th>
<th>#EV</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syn</td>
<td>30</td>
<td>30</td>
<td>30</td>
<td>2 .001</td>
<td>5</td>
<td>50</td>
<td>500</td>
</tr>
<tr>
<td>Taxi</td>
<td>5</td>
<td>500</td>
<td>1000</td>
<td>2 .001</td>
<td>5</td>
<td>75</td>
<td>250</td>
</tr>
<tr>
<td>Reddit F1</td>
<td>500</td>
<td>5</td>
<td>20</td>
<td>5 .001</td>
<td>5</td>
<td>50</td>
<td>250</td>
</tr>
</tbody>
</table>

Table 2: Parameters used by methods in change point detection