Coupled Clustering of Time-Series and Networks

Yike Liu* Linhong Zhu† Pedro Szekely‡ Aram Galstyan‡ Danai Koutra*

Abstract

Motivated by the problem of human-trafficking, where it is often observed that criminal organizations are linked and behave similarly over time, we introduce the problem of *Coupled Clustering of Time-series and* their underlying *Network*. The goal is to find tightly connected subgroups of nodes that also have similar node-specific time series (temporal—not necessarily structural—behavior). We formulate the problem as a coupled matrix factorization for the time series, combined with regularization for network smoothness. We propose CCTN, and an incrementally-updated counterpart, CCTN-INC, which efficiently handles network updates. Extensive experiments show that CCTN is up to 4× more accurate than baselines that consider graph structure or time series alone, and CCTN-INC is up to 55× faster than CCTN. As an application, we explore an exclusive database with millions of online ads on human trafficking, and successfully deploy our technique to detect criminal organizations.

1 Introduction

Clustering, or finding groups of similar entities, is a fundamental task in data mining and machine learning, with applications in human mobility analysis, sensor data analytics in healthcare, intelligent urban systems, climate monitoring, and more. In many applications, time series and networks co-occur and may need to be clustered jointly instead of individually [5, 17, 12, 15, 6].

In this work we introduce the challenging problem of **Coupled Clustering of** (entity-specific) **Time-series and their underlying Network**, where we aim to group entities into '*temporally*' and '*structurally*' coherent clusters (Fig. 1). Our rationale is that jointly considering network-centric and temporal (but not necessarily structural) behavioral features should lead to better clustering results than treating each data modality separately. We give two motivating examples.

Example 1. (Human trafficking) *The Internet plays a key role in both enabling and combating human-trafficking. For instance, classified ads that have contact information for the interested parties have been shown to be useful for discovering criminal networks. Based on our analysis of human-trafficking data, we assume that (1) phone numbers belonging to the same criminal organization often co-appear in ads; and (2) ads from the same orgs have similar content (e.g., sentences, expressions). Thus, the problem of detecting criminal organizations can be framed as coupled clustering of (1) the phone number co-occurrence network and (2) a set of phone number-specific time series that capture ad content similarity (e.g., per-day average content similarity between the posted ads that mention the same phone number, shown as* $\mathbf{X}_1$ *in Fig. 1).*

Example 2. (Social Networks) *In many cases, the topology of interactions between users is augmented with information about their temporal behaviors. For instance, in location-based social networks, the users can be characterized by their mobility or check-in patterns, whereas in scientific collaboration networks they can be described by their temporal topical interests.*

In these examples, there are two constituent problems: time-series clustering and graph clustering. Both of them have been studied extensively but *separately* in the literature [5, 23, 17, 12, 15, 6, 2, 11]. However, in these and other real scenarios, the time series correspond to entities that do *not* occur in isolation, but are *related* via an underlying network (e.g., the phone number co-occurrence network in Example 1). On one hand, most existing time-series clustering methods simply ignore this underlying structure or lack a principled way of incorporating it. On the other hand, most graph clustering methods aim to find tightly connected subgraphs or communities in static and dynamic networks [27] by optimizing structural quantities, such as modularity or conductance [15, 6]. Our proposed problem of coupled time-series and network clustering differs from dynamic graph clustering [19]: the former does not necessarily require snapshots of a graph over time; it can operate on a single network with multiple node-specific temporal behaviors or time series. It is also different from prior work that detects correlated changes in dynamic networks [8] based on the graph structure alone.

In this work we aim to fill in this gap. Specifically,

*University of Michigan
†Facebook
‡USC Information Sciences Institute

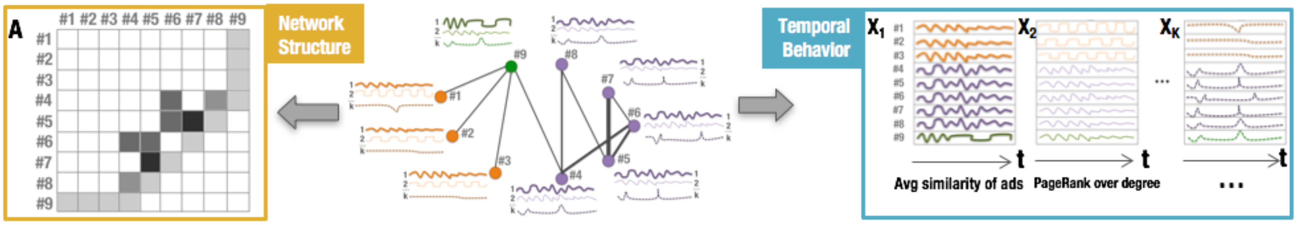


Figure 1: Human trafficking example: Nine phone numbers (nodes) form three clusters with tight connections, and similar temporal behaviors. The matrix on the left illustrates the representation of the network structure ($\mathbf{A}$) and the ones on the right show the different types of node-specific behaviors over time ($\mathbf{X}_k$).

we introduce the coupled clustering problem which aims to group nodes such that the similarity of their *time-series behaviors* and their *structural connectivity* is maximized per cluster. We formulate the problem as an intuitive latent time-series clustering problem joint with graph regularization, and show that it admits a standard quadratic programming solution. In more detail, our contributions are:

- **Novel Formulation:** Motivated by real applications, we propose the problem of finding groups of nodes that are both densely connected (network structure) and temporally coherent (time series of, potentially, non-structural behaviors). § 2
- **Principled Methods:** To effectively solve the problem, we propose CCTN and its counterpart CCTN-INC that efficiently handles updates (e.g., new nodes/edges and observations in the time series). § 2-3
- **Extensive Experiments:** We perform experiments on synthetic and real-world networks with up to 6.9 million edges, and show the effectiveness and efficiency of our proposed methods over the baseline methods. § 4
- **Application:** We explore an exclusive database of millions of online ads on human-trafficking, and show the potential of CCTN in detecting criminal organizations. § 4

The code and the supplementary material is hosted at `https://github.com/yikeliu/CCTN`.

2 Proposed Problem

Let $G = (\mathcal{V}, \mathcal{E})$ be a weighted graph with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges, and $\mathbf{A}$ be its weighted adjacency matrix. We assume that each node i is associated with K different types of time series (e.g., K different behavioral patterns). We denote $\mathbf{X}_k \in \mathbb{R}^{n \times T}$ as the stacked matrix where row i corresponds to the k^{th} type time-series of node i in the network.

In Example 1 (Fig. 1), the network based on human trafficking activity consists of phone numbers (nodes) that are linked if they appeared in the same online ad, with link or edge weight equal to their number of co-appearances. The first temporal behavior of each node is the per-day average content similarity between the posted ads that mention the phone number. The second behavior is its per-day $\frac{\text{PageRank}}{\text{degree}}$-ratio, which can capture unusual structural connectivity patterns [18].

Table 1: Major symbols and definitions.

Notation	Description
$G(\mathcal{V}, \mathcal{E})$	graph
$\mathcal{V}$, $n = \lvert\mathcal{V}\rvert$	node-set and number of nodes of G, resp.
$\mathcal{E}$, $m = \lvert\mathcal{E}\rvert$	edge-set and number of edges of G, resp.
$\mathbf{A}$	adjacency matrix of G, with entries $a(i,j)$
$\mathbf{D}$	diagonal degree matrix, $d(i,i) = \sum_j a(i,j)$ & $d(i,j) = 0$ o/w
$\mathbf{L}$	Laplacian matrix, $\mathbf{L} = \mathbf{D} - \mathbf{A}$
$\mathbf{X}_k$	$n \times T$ stacked matrix of the q^{th} time series type per node
$\mathbf{C}$	$n \times d$ embedded-clustering matrix (with node embeddings)
$\mathbf{W}$	$d \times T$ stacked basis matrix of temporal patterns
$\mathbf{c}$	$n \times 1$ vector with the cluster assignments per node
d	dimensionality of embedding for time-series patterns
t, T	timestamp and total number of timestamps, respectively

2.1 Problem Formulation As we mentioned above, we focus on the problem of clustering nodes that also have similar observed temporal behaviors. For consistency with the typical time-series or graph clustering problems, we make the following explicit assumptions:

(A1) **Node Temporal Behavior Similarity.** Nodes from the same cluster have similar patterns in the k^{th} time-series type (for types $k = 1, \ldots, K$).

(A2) **Graph Smoothness.** If two nodes are connected, their cluster assignments are similar. Also, the stronger the connection between them, the more likely they are to belong to the same cluster.

These assumptions align well with our motivating example. In human-trafficking, it is believed that linked phone numbers that have similar neighbors (A2) and behave similarly over time (A1)—e.g., with high average content similarity in their ads—may belong to the same criminal organization.

With these assumptions, we establish our model for the coupled clustering of time-series and network problem. We propose to embed the node-specific time-series patterns in a latent d-dimensional space using two factors: (1) a *basis* consisting of d time-series patterns which are stacked by row in matrix $\mathbf{W} \in \mathbb{R}^{d \times T}$, and (2) a matrix $\mathbf{C} \in \mathbb{R}^{n \times d}$ that describes the temporal behavior of each node as a weighted combination of the basis. We call $\mathbf{C}$ the embedded-clustering matrix, since similarities among its rows represent similar temporal behaviors among the corresponding nodes. Based on this representation, we can summarize the time-series patterns of all the nodes as follows:

$$\widetilde{\mathbf{X}}_k = \mathbf{C} \cdot \mathbf{W}, \tag{2.1}$$

where the same factors $\mathbf{W}$ and $\mathbf{C}$ are used to describe the various types of true temporal behaviors, $\mathbf{X}_k$. This model has several advantages: (1) it allows us to couple the various temporal behaviors and express them in terms of the *same basis* of temporal patterns, and (2) it constrains and guarantees a solution to our proposed problem. As shown below, if the temporal behaviors were modeled independently (i.e., via a different basis of temporal patterns $\mathbf{W}_k$ per type k), our formulation would admit *any arbitrary* solution for the embedded-clustering matrix $\mathbf{C}$, which is *not* meaningful.

By combining this model together with graph regularization, we formulate the **coupled clustering of time-series and network** problem.

Problem 1. Let $G(\mathcal{V}, \mathcal{E})$ be a network where each node u is associated with K different types of time-series denoted as $\mathbf{X}_k(u) \in \mathbb{R}^{1\times T}$. Then, the coupled clustering problem aims to assign to each node u a latent feature vector $\mathbf{C}(u) \in \mathbb{R}^{1\times d}$, which can then be projected to a cluster assignment $c(u) \in \mathbb{N}$, such that:

$$\arg\min_{\mathbf{C},\mathbf{W}} \{\sum_{k=1}^{K} a_k \cdot \|\mathbf{X}_k - \widetilde{\mathbf{X}}_k\|_F^2 + \lambda \cdot Tr(\mathbf{C}^T \mathbf{L} \mathbf{C})\} \tag{2.2}$$

where a_k controls the importance of the k^{th} time-series behavior, $\widetilde{\mathbf{X}}_k = \mathbf{C} \cdot \mathbf{W}$, $\mathbf{W}$ is the basis time-series matrix, $|| \cdot ||_F$ is the Frobenius norm of the enclosed matrix, $\mathbf{L}$ is the Laplacian matrix of G, $Tr(\cdot)$ is the trace of the corresponding matrix, and λ is a regularization parameter.

The **first term** of Eq. (2.2) represents the coupled clustering of the nodes based on their K *types of temporal behavior*. The clustering is given by our proposed model in Eq. (2.1). Our goal is to find the matrices $\mathbf{W}$ and $\mathbf{C}$ that best represent coherent time-series clusters across all behavior types. We note that a model with different basis patterns $\mathbf{W}_k$ would lead to trivial clustering solutions: for any fixed $\mathbf{C}$, it would be possible to find a set of $\mathbf{W}_1, \ldots, \mathbf{W}_K$ that satisfy Eq. (2.2). In the human-trafficking example, the first term finds a joint latent representation for the temporal content similarity of ads ($\mathbf{X}_1$ in Fig. 1) and the temporal structural patterns ($\mathbf{X}_2$ or the $\frac{\text{PageRank}}{\text{degree}}$-ratio per phone number and day). The **second term** of Eq. (2.2) imposes a graph smoothness constraint over the cluster assignments of the nodes (regularization). Intuitively, it forces the temporally coherent nodes to also have strong connectivity, thus satisfying assumption (A2). The influence of each term on the final clustering depends on the **parameters** a_k, λ, which we discuss in detail in the supplemental material. In Example 1, this constraint 'refines' the candidate criminal organizations that have temporally coherent behaviors by attaching well-connected phone numbers to them (thus, leading to also structurally coherent clusters).

2.2 Proposed Algorithm: CCTN Optimizing Problem 1 requires minimization with respect to two matrices, $\mathbf{C}$ and $\mathbf{W}$. To render the problem tractable, we devise an alternating process: (1) We fix $\mathbf{C}$ and turn Problem 1 into a relatively easier quadratic problem; (2) We fix $\mathbf{W}$ and turn Problem 1 into a mixed integer programming problem (Thm 1).

Next, we give the equations that need to be solved for the coupled clustering of time-series and network problem. These will be the building blocks of our proposed algorithm, CCTN.

Theorem 1. For a fixed clustering embedding $\mathbf{C}$, the basis time-series matrix $\mathbf{W}$ is the solution to:

$$(\textstyle\sum_k a_k \mathbf{C}^T \mathbf{C}) \mathbf{W} = (\textstyle\sum_k a_k \mathbf{C}^T \mathbf{X}_k) \tag{2.3}$$

where $\mathbf{C}^T$ denotes the transpose of matrix $\mathbf{C}$.
For a fixed basis time-series matrix $\mathbf{W}$, the clustering embedding $\mathbf{C}$ can be found by solving the equation:

$$\mathbf{C} \textstyle\sum_k a_k \mathbf{W}\mathbf{W}^T + \lambda \mathbf{L}\mathbf{C} = \textstyle\sum_k a_k \mathbf{X}_k \mathbf{W}^T, \tag{2.4}$$

which corresponds to a Mixed-Integer Programming problem. Equation (2.4) is a Sylvester equation.

Proof. See Appendix A in the supplemental material.

The linear system in Eq. (2.3) can be solved by randomized Kaczmarz algorithm [28]. This randomized iterative method can find $\mathbf{W}$ with expected exponential rate of convergence. To solve the Sylvester equation (2.4), we employ the scheme in [3] and rewrite it as an equation with Kronecker product (denoted as $\otimes$):

$$(\mathbf{I}_f \otimes \lambda \mathbf{L} + (\sum_k a_k \mathbf{W}\mathbf{W}^T)^T \otimes \mathbf{I}_n) \otimes vec(\mathbf{C}) = vec(\sum_k a_k \mathbf{X}_k \mathbf{W}^T)$$

where $vec()$ is the vectorization operator that takes a matrix and converts it to a vector by stacking its columns. The solution of $\mathbf{C}$ can be computed numerically by the Bartels-Stewart [3] algorithm. This algorithm first computes the Schur decomposition of the two matrices $\lambda\mathbf{L}$ and $-\sum_k a_k \mathbf{X}_k \mathbf{W}^T$ in Eq. (2.4) using a QR algorithm, and then solves the resulting triangular system via back-substitution.

Lemma 1. The MIP problem of Eq. (2.4) has a unique solution iff the $nf \times nf$ matrix $\mathbf{I}_f \otimes \lambda\mathbf{L} + (\sum_k a_k \mathbf{W}\mathbf{W}^T)^T \otimes \mathbf{I}_n$ is invertible—i.e., if $\mathbf{L}$ and $\sum_k a_k \mathbf{X}_k \mathbf{W}^T$ do *not* have common eigenvalues.

The computational cost of the original Bartels-Stewart [3] algorithm is $O(n^3)$. However, faster parallel solvers of large-scale Sylvester equations have been proposed, such as the Hessenberg-Schur method [14], and $\mathcal{H}$-matrix based sign function iteration [4], where large matrices are represented by sparse hierarchical matrices. The latter method is $O(n \log^2 n)$.

Algorithm. Based on Theorem 1 and the transformations of its main equations described above, we propose the CCTN method, whose pseudocode is given in Algorithm 1. Lines 5-9 describe the main part of our method, which seeks the solution in an iterative process, until convergence (line 9). In the absence of other information, the initialization of the matrices $\mathbf{C}$ and $\mathbf{W}$ is random (lines 3-4). After finding the embedded-clustering matrix $\mathbf{C}$, CCTN treats each row as an observation (which corresponds to a node) and applies a clustering technique in order to find similar nodes based on their latent representations in Eq. 2.2. In practice, any choice for clustering works for this step (e.g., k-means). We discuss our choices in the experiments.

Algorithm 1 CCTN: Coupled Clust. of Time-series & Network

```
    Input: Graph $G(\mathcal{V}, \mathcal{E})$; stacked matrices of $k$-type time-series $\{\mathbf{X}_k\}$ with $T$ timesteps, parameters $a_k$ and $\lambda$, dimensionality $d$
    Output: Vector with cluster assignments $\mathbf{c}$
 1: $\epsilon = 10^{-6}, \tau_{max} = 100$    // Constants for convergence
 2: $\tau = 0$    // Iteration # initialization
 3: $\mathbf{C}_{(\tau)} = rand(n, d)$ // $n = |\mathcal{V}|$
 4: $\mathbf{W}_{(\tau)} = rand(d, T)$
 5: repeat
 6:     $\tau = \tau + 1$
    // Step 1: Update W using Eq. (2.3)
 7:     $\mathbf{W}_{(\tau)} = (\sum_k a_k \mathbf{C}_{(\tau-\mathbf{1})}{}^T \mathbf{C}_{(\tau-1)})^{-1} (\sum_k a_k \mathbf{C}^T_{(\tau-1)} \mathbf{X}_k)$
    // Step 2: Update C by solving Eq. (2.4) following [4]
 8:     $\mathbf{C}_{(\tau)} \sum_k a_k \mathbf{W}_{(\tau-1)} \mathbf{W}^T_{(\tau-1)} + \lambda \mathbf{L} \mathbf{C}_{(\tau)} = \sum_k a_k \mathbf{X}_k \mathbf{W}^T_{(\tau-1)}$
 9: until $(||\mathbf{C}_{(\tau)} - \mathbf{C}_{(\tau-1)}||_1 < \epsilon$ & $||\mathbf{W}_{(\tau)} - \mathbf{W}_{(\tau-1)}||_1 < \epsilon)$ or $\tau > \tau_{max}$
    // Step 3: Assign the nodes to clusters based on the inferred embeddings in C (each row is an 'observation').
10:  $\mathbf{c} = \text{cluster\_rows}(\mathbf{C}_{(\tau)})$
11: return c
```

2.3 Complexity Analysis In each iteration of Algorithm 1, the computation is composed of two steps: update $\mathbf{W}$ (Step 1) and update $\mathbf{C}$ (Step 2). $\mathbf{W}$ is updated by randomized Kaczmarz algorithm [28]. The computational complexity of solving a linear system $\mathbf{Mx} = \mathbf{b}$ is $O(n_K t_K)$, where $\mathbf{M} \in \mathbb{R}^{m_K \times n_K}$ and t_K is the number of iterations of random Kaczmarz update. For $m_K \neq n_K$, we have $t_K = \frac{2n_K}{(1-\sqrt{y})^2} \log \frac{1}{\epsilon_K}$, where $y := \frac{n_K}{m_K}$, and ϵ_K is the accuracy of the randomized Kaczmarz algorithm. In CCTN, we have a square matrix ($\mathbf{M} = \sum_k a_k \mathbf{C}^T \mathbf{C}$), but since it has exponential convergence [28], t_K will be very small ($t_K \sim 5$ in practice). Updating $\mathbf{C}$ with the $\mathcal{H}$-matrix based sign function iteration [4] has complexity $O(n \log^2 n)$. Hence, the complexity of CCTN is $O((dt_K + (n \log^2 n))$.

3 CCTN-inc: Incremental Updates

In many applications, including our motivating application of human-trafficking, the data are changing over time: new nodes (i.e., phone numbers) and edges (new co-occurrences) are added to the network, and new timestamps are added to the behavioral time series for the existing nodes (e.g., content similarity on the new days). In our experiments, we observed that for synthetic data (Kronecker graphs) with more than 6.6k nodes, the runtime of CCTN exceeds a week. Moreover, clustering millions of nodes is quite expensive. Thus, we propose the problem of *incremental* coupled clustering.

3.1 Problem Formulation The problem of **Incremental Coupled Clustering** seeks to efficiently handle incremental updates in the network structure *and* the time series, so that the computation that needs to be performed per timestamp is minimized.

Problem 2. Let $\mathbf{A}' \in \mathbb{R}^{n' \times n'}$ and $\mathbf{X}'_k \in \mathbb{R}^{n' \times T'}$ be the augmented adjacency matrix with n' nodes and the stacked matrix of the k^{th} type time series data with T' timestamps, respectively. Let also $\mathbf{C}$ and $\mathbf{W}$ be the solutions of Problem 1. The Incremental Coupled Clustering problem aims to find the perturbations $\Delta\mathbf{W}$ and $\Delta\mathbf{C}$ in the matrices of basis temporal behaviors and cluster embeddings s.t. the new solutions are expressed as $\mathbf{W}' = \mathbf{W} + \Delta\mathbf{W}$ and $\mathbf{C}' = \mathbf{C} + \Delta\mathbf{C}$, respectively:

$$\arg\min_{\mathbf{C}', \mathbf{W}'} \{ \sum_{k=1}^{K} a_k \cdot \|\mathbf{X}'_k - \widetilde{\mathbf{X}}'_k\|_F^2 + \lambda \cdot Tr(\mathbf{C}'^T \mathbf{L}' \mathbf{C}') \}$$

where a_k and λ remain the same as in CCTN, $\mathbf{L}'$ is the updated Laplacian matrix of $\mathbf{A}'$, and $\widetilde{\mathbf{X}}'_k = \mathbf{C}' \cdot \mathbf{W}'$.

Thus, this problem seeks to incrementally update the cluster assignment vector $\mathbf{c}'$, obtained by projecting the new embedded-clustering matrix $\mathbf{C}'$.

3.2 Incremental Algorithm: CCTN-inc To derive the solution of the incremental problem, we leverage small perturbations $\Delta\mathbf{Z}$ for each matrix $\mathbf{Z}$ that is involved in the derivations. Specifically, we rewrite the incremental adjacency matrix as (1) the original matrix and (2) the difference-matrix with the difference in weights between existing nodes and the connections to new nodes: $\mathbf{A}' = \mathbf{A}_{(n' \times n')} + \Delta\mathbf{A}$. In the human-trafficking example, $\Delta\mathbf{A}$ contains new phone numbers that appeared in ads published after time T, and new edges between numbers that co-appeared in ads after T.

Similarly the stacked matrix of k^{th}-type time series and the Laplacian of the new graph can be written as: $\mathbf{X}'_k = \mathbf{X}_{k,(n' \times T')} + \Delta\mathbf{X}_k$ and $\mathbf{L}' = \mathbf{D}' - \mathbf{A}' = \mathbf{L} + \Delta\mathbf{L}$. In our example, $\mathbf{X}'_k$ has additional rows for the new nodes (past the original n) and more columns for the new timestamps (past T).

Based on the above definitions, we can compute the incremental matrix $\mathbf{W}'$ by simply computing $\Delta\mathbf{W}$ and adding it to the solution of the non-incremental version. As in the solution of Problem 1, in the second step, we fix $\mathbf{W}'$ and find the new solution for $\mathbf{C}'$.

Theorem 2. For a fixed clustering embedding $\mathbf{C}'$, the difference $\Delta\mathbf{W}$ in the stacked matrix of the d base time series patterns is given by:

$$\Delta\mathbf{W} = \left(\textstyle\sum_k a_k \mathbf{C}'^T \mathbf{C}'\right)^{-1} \textstyle\sum_k a_k \mathbf{C}'^T \Delta\mathbf{X}_k \tag{3.5}$$

For a fixed basis time-series matrix $\mathbf{W}'$ (defined in Eq. (3.5)), the difference in the embedded-clustering matrix $\Delta\mathbf{C}$ is approximated by solving the following Sylvester equation:

$$\begin{aligned} &\Delta\mathbf{C} \sum_k a_k \mathbf{W}\mathbf{W}^T + \lambda\mathbf{L}\Delta\mathbf{C} = \sum_k a_k(\mathbf{X}_k(\Delta\mathbf{W})^T \\ &+ (\Delta\mathbf{X}_k)\mathbf{W}^T - \mathbf{C}(\Delta\mathbf{W})\mathbf{W}^T - \mathbf{C}\mathbf{W}(\Delta\mathbf{W})^T + \lambda(\Delta\mathbf{L})\mathbf{C}). \end{aligned} \tag{3.6}$$

Proof. See Appendix A in the supplemental material.

Algorithm. Based on Theorem 2, we propose CCTN-INC, an effective and fast approximation of CCTN, which handles incremental updates in the network structure, the introduction of new nodes, and changes in the behavioral patterns of existing nodes. We give the high-level pseudocode of CCTN-INC in Algorithm 2 in the supplemental material (Appendix B).

3.3 Complexity Analysis Although Eq. (3.5) involves inverting a matrix, the computation is not prohibitive due to its very small size. Matrix $\mathbf{C}'$ has size $n \times d$, and thus $\mathbf{C}'^T\mathbf{C}'$ (which is the matrix that needs to be inverted) is a $d \times d$ matrix. In practice, the dimensionality d of the embedding is significantly smaller than the number of nodes n, and most likely is in the order of 10-20 features. Equation (3.6) in Theorem 2 is still a Sylvester equation that can be solved with the Bartels-Stewart algorithm [3], as in CCTN. Due to the significant sparsity of $\Delta\mathbf{C}$, the computation of the incremental matrix is sub-quadratic, $O(n \log^2 n)$ [4].

4 Experiments

Our experiments are geared toward answering the following questions: **(1)** How effective is CCTN in terms of identifying temporally and structurally coherent clusters? **(2)** How well does CCTN-INC approximate CCTN? **(3)** Do CCTN and CCTN-INC scale well to large datasets? **(4)** Does CCTN generate intuitive clusters in real applications? **(5)** How robust is CCTN to different parameter settings? Before we present our results, we discuss our datasets, baselines and experimental setup. We answer question (5) in Appendix E.

4.1 Data We use both synthetic and real datasets.
Synthetic Data. We generate three cliques of different sizes (50, 100, and 200 nodes), with random, sparse connections between them. For simplicity we treat the graph as unweighted, and keep the graph structure constant over time (i.e., we use one static network). For the node-specific temporal behavior, we either randomly generate time series, or extract time series of content similarity from the real human-trafficking HT-1 data (described below), and add six types of noise (Table 2).

Real Data. We also use 3 exclusive real datasets: two in human-trafficking domain (HT-1/HT-1M, HT-2), and one in the military domain (MITRE).

- **Human-trafficking data 1 (HT-1, HT-1M)**: This is a **labeled** dataset of advertisements assigned to clusters. For each advertisement, we have all or part of the following information: {region, phone number, text, title, post time, age, user location, city, cluster id}. The phone numbers are assigned cluster ids by domain experts (ground truth).

We create the co-occurrence graph (temporal and aggregated) on phone numbers by adding edges between phone numbers that appear in the same ad, and weighing them by the frequency of their co-occurrence. HT-1M denotes the manipulated graph of HT-1 where attackers randomly connect their phone numbers to public phone numbers such as AT&T service number. The node-specific temporal behaviors consist of: (1) Structural time series of $\frac{\text{PageRank}}{\text{degree}}$-ratio, which can capture anomalous patterns [18] and is obtained from the temporal graphs—i.e., $\mathbf{X}_1 = \mathbf{X}_{struc}$; (2) Content time series for the average pairwise Jaccard index across the ads with the same phone number (during the same time interval)—i.e., $\mathbf{X}_2 = \mathbf{X}_{cont}$. Although we use $k = 2$ temporal behaviors, CCTN can scale with greater k.

Table 2: Synthetic data: Description of the six cases that we designed for evaluation. For each cluster, we generate for its constituent nodes a specific type of time series per case (e.g., identical, correlated, noisy).

Case No.	Time series per cluster	Description of the clusters in terms of their nodes' time series
1	Random identical	$C_i = \{x_i(t), \ldots, x_i(t)\}$, $i \in \{1,2,3\}$, x_i randomly generated
2	Informed identical	$C_i = \{x_i(t), \ldots, x_i(t)\}$, $i \in \{1,2,3\}$, x_i extracted from real data
3	Correlated	$C_i = \{a_1(x_i(t) + b_1), \ldots, a_{\lvert C_i \rvert} x_i(t + b_{\lvert C_i \rvert})\}$, $i \in \{1,2,3\}$, x_i extracted from real data ($\lvert C_i \rvert$: size of i^{th} cluster)
4	Noisy	$C_i = \{x_i(t) + e_i(t), \ldots, x_i(t) + e_i(t)\}$, $i \in \{1,2,3\}$, x_i extracted from real data, $e_i(t) \sim \mathcal{N}(1,0)$
5	Anti-correlated	$C_i = \{x_i(t), \ldots, -x_i(t)\}$, $i \in \{1,2,3\}$, x_i extracted from real data, C_i split at half
6	Informed split	$C_{1,2} = \{x_{1,2}(t), \ldots, x_{1,2}(t)\}$, $C_3 = \{x_1(t), \ldots, x_2(t)\}$, $x_{1,2,3}$ extracted from real data, C_3 split at half

Table 3: Real data

Dataset	Nodes	Edges	Timestamps	Description
HT-1	60 437	1 241 773	13	labeled, human-trafficking
HT-2	61 155	129 196	60	unlabeled, human-trafficking
MITRE	3 813	6 892 425	335	labeled, Twitter data

• **Human-trafficking data 2 (HT-2)**: This is an **unlabeled** dataset of ads, for which we have all or part of the following information: {description, uri, date of creation, contact information, name, location}. Graphs and time series are generated the same way as for HT-1.

• **MITRE data**: This **labeled** Twitter dataset [29] is annotated with both GEO-location and social event forecasting results. Each location is assigned to a cluster by domain experts, based on event types and users. We generate a graph that consists of locations (nodes) connected by edges that are weighted by their geographic distance (based on longitude and latitude). Edges with distances above $d = 1000$ miles are pruned. The temporal behaviors include: (1) the structural time-series of the $\frac{\text{PageRank}}{\text{degree}}$-ratio—i.e., $\mathbf{X}_1 = \mathbf{X}_{struc}$; and (2) the activity-specific behavior consisting of the count of events at each location over time—i.e., $\mathbf{X}_2 = \mathbf{X}_{event}$.

4.2 Baselines We choose methods that fall into the two subproblems that we solve: (1) time-series clustering, from which we pick two recent, best-performing works, k-Shape [25] and TADPole [5]; and (2) graph clustering on the aggregated graph, from which we choose spectral clustering [15] and Louvain [6]. We describe the baselines in Appendix C.

4.3 Experimental Setup For CCTN and CCTN-INC, practitioners can choose any clustering method that takes in feature vectors of the CCTN node embeddings in $\mathbf{C}$ (line 10 of Alg. 1). For simplicity, we exploit the widely-used k-means clustering [24] for labeled data, and x-means [16] for unlabeled data with unknown number of clusters. To show the effect of parameters on the performance of CCTN, we report the results on two variants: (1) the '*default*' case, where we set $a_1 = a_2 = 1$, $\lambda = 0.01$, $d = 3$; and (2) the '*best*' case, where the parameters are chosen via grid search on a small random subset of the data. The sample we used in our experiments consists of $1/10^{th}$ of the data for MITRE, and $1/100^{th}$ of the HT-1 and HT-1M data. We performed grid search for the following ranges: $a_1 \in (0, 10], a_2 \in (0, 10], \lambda \in (0, 10], d \in (0, 10]$ with an interval of two, hence the 'best' is not the globally best result. Results on CCTN's robustness to parameter settings are in Appendix E. We ran the CCTN until convergence (line 9 in Alg. 1) for all the datasets except for HT-1, for which we set $\tau_{max} = 10$ iterations.

For the baselines we used the default values of their parameters: resolution $\tau_L = 10^{-4}$ for Louvain, band size 0.08 for DTW, and cut off distance 1.4619 for TADPole.

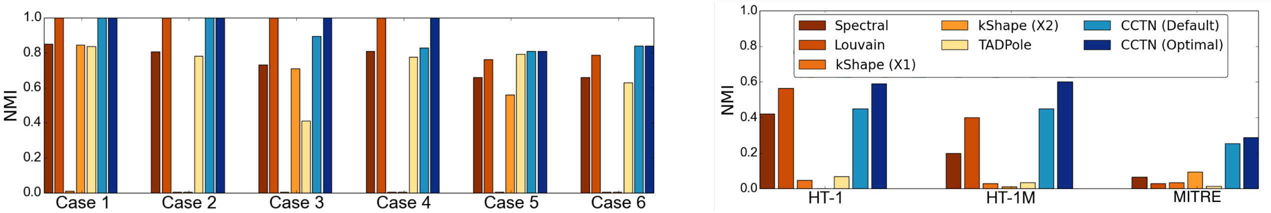
All the experiments were run on Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz with 264GB RAM.

Evaluation Since our objective is to find coherent groups of nodes, we focus on the evaluation of the embedded-clustering matrix $\mathbf{C}$ and the clustering results. We omit a detailed analysis of the auxiliary variable $\mathbf{W}$ due to space limitations. For the performance of cluster recovery, we use (1) normalized mutual information (NMI) and (2) Rand index, which measure the agreement between the found and ground-truth clusters, but capture different information. The results are generally consistent across the two metrics; for brevity we give the results based on Rand index in Appendix D.

4.4 Accuracy - (1) CCTN. First, we investigate how CCTN compares to the baselines in terms of effectiveness in identifying temporally and structurally coherent clusters. We present the NMI of the clusterings that all methods produce on the synthetic datasets in Fig. 2a and on the real datasets in Fig. 2b. We report the results based on the Rand index measure in Fig. 6 (Appendix D). For the synthetic data, where the structural time series are constant (due to the static network), TADPole is less effective when leveraging $\mathbf{X}_{struc}$ in addition to $\mathbf{X}_{cont}$. For that reason, we only report its results using $\mathbf{X}_{cont}$.

OBSERVATION 1. CCTN *outperforms the baselines on the real data, and is sometimes tied with Louvain on the synthetic data. The baselines have more variable performance across datasets.*

Louvain, which usually outperforms spectral clustering, gives equally good results for HT-1, but has significantly inferior performance on HT-1M and MITRE. Since it only relies on the graph structure, it



(a) Synthetic data: CCTN performs better than the baselines on difficult cases or equally well on easy cases.

(b) Real data: Overall, CCTN performs consistently better than or equally well as the baseline methods.

Figure 2: Accuracy of CCTN and CCTN-inc on synthetic and real data.

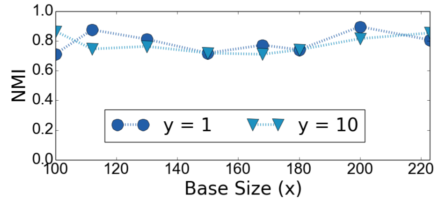


Figure 3: MITRE: CCTN-inc approximates CCTN well. They yield similar clusterings.

is heavily impacted by graph noise, as shown by its performance on the manipulated HT-1M dataset (Fig. 2b). K-Shape and TADPole, which ignore the graph structure, tend to perform worse than CCTN, especially on the real datasets ($NMI < 0.1$). By combining the temporal and structural aspects, CCTN is more robust: it achieves up to $NMI = 0.6$ for real data, and often perfect performance in the synthetic cases.

(2) CCTN-inc. Second, we explore how well CCTN-inc approximates the clusterings that our exact approach, CCTN, generates. To that end, we split the real labeled data into two parts: (1) the first x timestamps (or the 'base size'), which are used to create the input graph and time series, as described in Sec. 4.1; and (2) the future timestamps. Then, we compute the agreement between the clustering that CCTN-inc outputs after incremental updates for y timestamps, and the clustering of CCTN when applied to the $(x + y)$ timestamps all together. For brevity, in Fig. 3, we show the agreement (NMI) of the two methods on MITRE by varying the base size x, and for $y \in \{1, 10\}$ steps of incremental updates. The results based on Rand index are given in Fig. 7 (Appendix D). The results are consistent for other values of y and the other datasets.

Observation 2. CCTN-inc *is a good approximation of* CCTN *(wrt both NMI and Rand index), independent of the incremental interval y. The accuracy varies for different base size x, but remains relatively stable. As expected, performing more incremental updates (e.g., $y = 10$) leads to lower, but still high, agreement.*

We note that in this experiment we do not compare the CCTN-inc clustering with the ground-truth labels, because its performance is measured by how well it approximates CCTN and how efficient it is (Sec. 4.5).

4.5 Runtime - (1) CCTN. Third, we evaluate how well CCTN scales with the size of the data, and specifically with (i) the number of nodes in the aggregated graph, and (ii) the number of timestamps.

For experiment (i), we fix the number of timestamps to 10, and generate the aggregated graph by combining 10 Kronecker graphs [20] of different sizes (given random seed matrices). Assuming $\sqrt{n/2}$ clusters (which is the rule of thumb for choosing number of clusters [1]), we generate that many random time series, and randomly assign them to the n nodes. For experiment (ii), we generate synthetic data in the same way, but fix the number of nodes to 1 024 and vary the number of timestamps of the time series. We demonstrate the results in Fig. 4a.

Observation 3. *The runtime of* CCTN *increases subquadratically with the number of nodes, and smooths out to near-linear when the number of nodes becomes large. Its efficiency is independent of the number of timestamps T (right plot in Fig. 4a).*

(2) CCTN-inc. Finally, we compare CCTN-inc to CCTN to show its efficiency benefits. Figure 4b shows the runtime of the two methods on MITRE for different combinations of x and y (explained in Sec. 4.4).

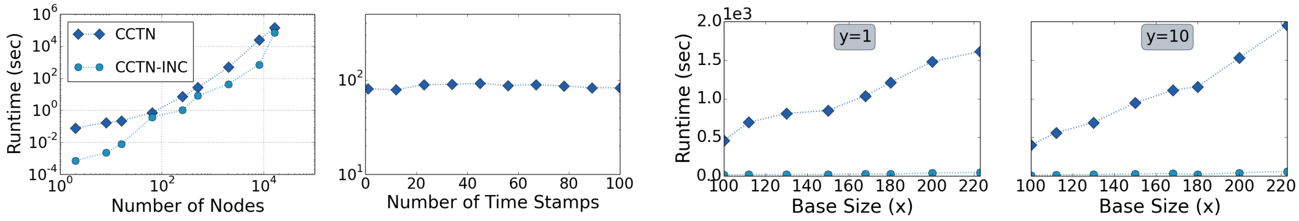
Observation 4. CCTN-inc *is up to* $30 - 55\times$ *faster than* CCTN, *due to the sparser matrix computations that it performs per update, and its faster convergence.*

CCTN-inc usually converges in $1/50 \sim 1/30$ of the iterations that CCTN needs for real data. This contributes to the significant reduction in runtime.

5 Case Study on Real Data

To evaluate whether CCTN finds intuitive clusters in real applications, we apply it to the human-trafficking HT-2 dataset. Among the 61 155 nodes in HT-2, CCTN identified 15 clusters, with size ranging from 129 to 31 060 nodes. All the clusters have numerous phone number pairs that have appeared in similar ads.

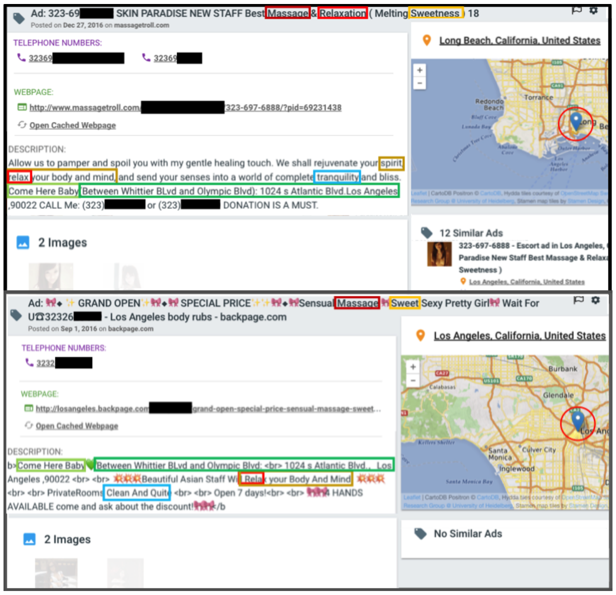
In Fig. 5, we show a randomly-picked example consisting of phone numbers 1-3236****** and 1-3232******. Across the 60 timestamps, these phone



(a) Runtime vs. number of nodes / number of time stamps

(b) MITRE: Runtime of CCTN-INC vs. CCTN.

Figure 4: Runtime analysis.



Ad: 323-69[redacted] SKIN PARADISE NEW STAFF Best Massage & Relaxation (Melting Sweetness) 18

Posted on Dec 27, 2016 on massagetroll.com

TELEPHONE NUMBERS:

☎ 32369[redacted] ☎ 32369[redacted]

WEBPAGE:

http://www.massagetroll.com/[redacted]323-697-6888/?pid=69231438

Open Cached Webpage

DESCRIPTION:

Allow us to pamper and spoil you with my gentle healing touch. We shall rejuvenate your spirit relax your body and mind, and send your senses into a world of complete tranquility and bliss. Come Here Baby (Between Whittier BLvd and Olympic Blvd): 1024 s Atlantic Blvd.Los Angeles ,90022 CALL Me: (323)[redacted] or (323)[redacted] DONATION IS A MUST.

2 Images

Long Beach, California, United States

12 Similar Ads

323-697-6888 - Escort ad in Los Angeles, Paradise New Staff Best Massage & Relax Sweetness)

Los Angeles, California, United States

Ad: 🎀◆ ✨ GRAND OPEN✨🎀◆🎀 SPECIAL PRICE✨✨🎀◆🎀Sensual Massage🎀 Sweet Sexy Pretty Girl🎀 Wait For U☎32326[redacted] - Los Angeles body rubs - backpage.com

Posted on Sep 1, 2016 on backpage.com

TELEPHONE NUMBERS:

☎ 3232[redacted]

WEBPAGE:

http://losangeles.backpage.com[redacted]grand-open-special-price-sensual-massage-sweet...

Open Cached Webpage

DESCRIPTION:

b>Come Here Baby💚 Between Whittier BLvd and Olympic Blvd:
 1024 s Atlantic Blvd., Los Angeles ,90022

 💥💥💥Beautiful Asian Staff Wil Relax your Body And Mind 💥💥💥

 PrivateRooms Clean And Quite

 Open 7 days!

 🎀🎀4 HANDS AVAILABLE come and ask about the discount!🎀🎀</b

2 Images

Los Angeles, California, United States

No Similar Ads

Figure 5: Advertisements related to the phone numbers 1-3236****** and 1-3232******.

numbers have co-occurred 2 656 times in different ads (high edge weight in the graph). Besides the high similarity between the ads they co-occurred in, we also observe that many of the ads in which each phone number appeared alone have largely similar content (both in text and images). In the bottom ad, which was posted on Sep 1, 'GRAND OPEN' suggests that the service was just opened. The top ad was posted several months later, on Dec 27, yet it has very similar text and pictures to the bottom ad (similar text is marked with the same color). Though the original data extraction process did not identify the ads to be at the same location (Long Beach vs. the general Los Angeles area), CCTN has identified two phone numbers pointing to exactly the same address in the ads, which is clear evidence that they belong to the same organization. Similar observations for many other pairs of phone numbers hold.

6 Related Work

We briefly review two related directions to our work.

Time-series Clustering. Time-series clustering [21] aims to group coherent time-series sequences together based on a certain similarity measure of either raw data or extracted features or representations [5, 10, 26]. Targeting imputation and prediction for co-evolving higher-order time series, [7] jointly models them in a 'network' of time series and follows a tensor decomposition approach. This model is more restrictive than ours that loosely couples the network with the time dimension. We present a set of other representative works (not exhaustive) in Appendix F. In a nutshell, our proposed approach has three main advantages compared to the state-of-the-art approaches: (1) it combines the strengths of both model-based similarity measurement and low-dimension representation learning (i.e., cluster embedding); (2) it is built upon a unified framework that performs representation learning, coherence evaluation, and clustering simultaneously; and (3) it utilizes rich meta-data information (e.g., co-occurrence graphs) to improve the quality of clustering.

Graph Clustering. For an extensive review on graph clustering, we refer the interested reader to two surveys [27, 22]. Recent work [13] converts the time-series clustering problem to graph clustering by first creating a similarity graph where each node corresponds to a time sequence and then applying modularity-based clustering [6]. Moreover, there has been increasing interest in jointly clustering attribute and relational data. For instance, Cho et. al. [9] proposed a shared latent space model for describing both network and behavioral data. However, their work focused on static behaviors/attributes only. Also, [8] extracts subgraphs with similarly evolving *structural* patterns, but do not take into account non-structural temporal behaviors. Our work, on the other hand, explicitly addresses the temporal aspect of the problem by representing each node as a multivariate time series.

7 Conclusion

Motivated by the need to identify criminal organizations involved in human-trafficking (which are often related to each other, and behave similarly over time), we introduced the problem of *coupled clustering of time-series and their underlying network.* We formulated it as an optimization over the time-series embeddings, coupled with graph regularization. To solve it, we proposed CCTN, an efficient method that combines matrix factorization and network embeddings, as well as an incrementally-updated counterpart that efficiently adjusts the discovered clusters to the graph and temporal changes over time. Our experiments on synthetic and large real data showed that our methods are up to 4× more accurate than the baselines that ignore either the graph structure or the time-series component. We also demonstrated that CCTN produces sensible results on real human-trafficking data and identifies temporally and structurally coherent clusters, which likely represent criminal organizations.

Acknowledgements

This material is based upon work supported by an Army Young Investigator Award, the National Science Foundation, an Adobe Digital Experience research faculty award, an Amazon research faculty award, and Trove AI. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the funding parties.

References

[1] clusterMaker: Creating and Visualizing Cytoscape Clusters. `http://www.cgl.ucsf.edu/cytoscape/cluster/clusterMaker.shtml`.

[2] Miguel Araujo, Stephan Günnemann, Spiros Papadimitriou, Christos Faloutsos, Prithwish Basu, Ananthram Swami, Evangelos E. Papalexakis, and Danai Koutra. Discovery of "comet" communities in temporal and labeled graphs com2. *KAIS*, 46(3):657–677, Mar 2016.

[3] Richard H. Bartels and GW Stewart. Solution of the matrix equation ax+ xb= c [f4]. *CACM*, 15(9), 1972.

[4] Ulrike Baur. Low rank solution of data-sparse sylvester equations. *Numer Linear Algebr*, 15(9):837–851, 2008.

[5] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *KDD*. ACM, 2015.

[6] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *JSTAT*, (10), 2008.

[7] Yongjie Cai, Hanghang Tong, Wei Fan, Ping Ji, and Qing He. Facets: Fast comprehensive mining of coevolving high-order time series. In *KDD*, pages 79–88. ACM, 2015.

[8] Jeffrey Chan, James Bailey, Christopher Leckie, and Michael Houle. ciforager: Incrementally discovering regions of correlated change in evolving graphs. *ACM TKDD*, 6(3):11, 2012.

[9] Yoon-Sik Cho, Greg Ver Steeg, Emilio Ferrara, and Aram Galstyan. Latent space model for multi-modal social data. In *WWW*, 2016.

[10] Marcella Corduas and Domenico Piccolo. Time series clustering and classification by the autoregressive metric. *CSDA*, 52(4):1860–1872, 2008.

[11] Pravallika Devineni, Evangelos E. Papalexakis, Danai Koutra, A. Seza Doğruöz, and Michalis Faloutsos. One size does not fit all: Profiling personalized time-evolving user behaviors. In *IEEE/ACM ASONAM*, pages 331–340. ACM, 2017.

[12] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 2016.

[13] Leonardo N Ferreira and Liang Zhao. Time series clustering via community detection in networks. *Information Sciences*, 326:227–242, 2016.

[14] Gene Golub, Stephen Nash, and Charles Van Loan. A hessenberg-schur method for the problem ax+ xb= c. *IEEE Trans Automat Contr.*, 24(6), 1979.

[15] Joao P Hespanha. An efficient matlab algorithm for graph partitioning. *UCSB, CA, USA*, 2004.

[16] Anil K Jain. Data clustering: 50 years beyond k-means. *Pattern Recognit Lett*, 31(8):651–666, 2010.

[17] Abhay Jha, Shubhankar Ray, Brian Seaman, and Inderjit S Dhillon. Clustering to forecast sparse time-series data. In *ICDE*. IEEE, 2015.

[18] Danai Koutra, Di Jin, Yuanshi Ning, and Christos Faloutsos. Perseus: An interactive large-scale graph mining and visualization tool. *PVLDB*, 8(12):1924–1927, 2015.

[19] Stefano Leonardi, Aris Anagnostopoulos, Jakub Lacki, Silvio Lattanzi, and Mohammad Mahdian. Community detection on evolving graphs. In *NIPS*, 2016.

[20] Jure Leskovec, Deepayan Chakrabarti, Jon Kleinberg, Christos Faloutsos, and Zoubin Ghahramani. Kronecker graphs: An approach to modeling networks. *JMLR*, 11(Feb):985–1042, 2010.

[21] T Warren Liao. Clustering of time series data – a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[22] Yike Liu, Tara Safavi, Abhilash Dighe, and Danai Koutra. Graph summarization methods and applications: A survey. *ACM CSUR*, 51(3):62, 2018.

[23] Yike Liu, Tara Safavi, Neil Shah, and Danai Koutra. Reducing large graphs to small supergraphs: a unified approach. *Social Netw. Analys. Mining*, 8(1):17, 2018.

[24] Stuart Lloyd. Least squares quantization in pcm. *ITIT*, 28(2), 1982.

[25] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, 2015.

[26] Tara Safavi, Chandra Sripada, and Danai Koutra. Scalable hashing-based network discovery. In *IEEE ICDM*, pages 405–414, 2017.

[27] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[28] Thomas Strohmer and Roman Vershynin. A randomized kaczmarz algorithm with exponential convergence. *JFAA*, 15(2), 2009.

[29] Liang Zhao, Qian Sun, Jieping Ye, Feng Chen, Chang-Tien Lu, and Naren Ramakrishnan. Multi-task learning for spatio-temporal event forecasting. In *KDD*. ACM, 2015.

Supplementary Material

A Derivations and Proofs

In this section we provide the proofs for the two main results behind our proposed methods, CCTN and CCTN-INC.

Proof. [**Theorem 1**] When fixing variable $\mathbf{C}$, we only need to focus on the parts of the objective (2.2) that depend on $\mathbf{W}$, so the objective becomes:

$$J(\mathbf{W}) = \textstyle\sum_{k=1}^{K} a_k \|\mathbf{X}_k - \mathbf{C}\mathbf{W}\|_F^2, \tag{A.1}$$

which corresponds to a quadratic optimization function. To optimize Eq. (A.1), we apply the first order optimality condition as follows:

$$\tfrac{\partial J(\mathbf{W})}{\partial \mathbf{W}} = 2(\textstyle\sum_k a_k \mathbf{C}^T\mathbf{C})\mathbf{W})\mathbf{W} - 2\sum_k a_k \mathbf{C}^T\mathbf{X}_k = \mathbf{0}$$

Hence, $\mathbf{W}$ is the solution of linear equation (2.3).

When we fix $\mathbf{W}$, we need to take into account the parts of the objective function that depend on $\mathbf{C}$:

$$Q(\mathbf{C}) = \textstyle\sum_{k=1}^{K} a_k \|\mathbf{X}_k - \mathbf{C}\mathbf{W}\|_F^2 + \lambda Tr(\mathbf{C}^T\mathbf{L}\mathbf{C}). \tag{A.2}$$

To minimize $Q(\mathbf{C})$ with respect to $\mathbf{C}$, we take the partial derivative and set it to 0:

$$\frac{\partial Q(\mathbf{C})}{\partial \mathbf{C}} = 0 \Rightarrow 2\sum_k a_k(\mathbf{C}\mathbf{W} - \mathbf{X}_k)\mathbf{W}^T + 2\lambda\mathbf{L}\mathbf{C} = 0.$$

By rearranging this equation, we obtain Eq. (2.4). ■

Proof. [**Thoerem 2**] The embedded clustering matrix $\mathbf{C}'$ is fixed and initialized to the solution of the non-incremental version, i.e., $\mathbf{C}' = \mathbf{C}$. According to Theorem 1, and using the notation that we introduced for perturbations, we have:

$$\mathbf{W}' = \left(\sum_k a_k \mathbf{C}'^T\mathbf{C}'\right)^{-1} \sum_k a_k \mathbf{C}'^T\mathbf{X}'_k \Rightarrow$$

$$\mathbf{W} + \Delta\mathbf{W} = \left(\sum_k a_k \mathbf{C}^T\mathbf{C}\right)^{-1} \sum_k a_k \mathbf{C}^T(\mathbf{X}_k + \Delta\mathbf{X}_k)$$

By using the formula for $\mathbf{W}$ from Theorem 1 and substituting it above, we obtain the expression in Eq. (3.5). Then, we apply Theorem 1 to find the 'ground truth' solution of $\mathbf{C}'$ (based on CCTN), when $\mathbf{W}'$ is fixed:

$$\mathbf{C}' \textstyle\sum_k a_k \mathbf{W}'\mathbf{W}'^T + \lambda\mathbf{L}'\mathbf{C}' = \sum_k a_k \mathbf{X}'_k\mathbf{W}'^T$$

By rewriting the last equation using the Δ-notation for all the matrices that are involved, we obtain

$$(\mathbf{C} + \Delta\mathbf{C})\sum_k a_k(\mathbf{W} + \Delta\mathbf{W})(\mathbf{W} + \Delta\mathbf{W})^T$$
$$+ \lambda(\mathbf{L} + \Delta\mathbf{L})(\mathbf{C} + \Delta\mathbf{C}) = \sum_k a_k(\mathbf{X}_k + \Delta\mathbf{X}_k)(\mathbf{W} + \Delta\mathbf{W})^T$$

Based on the assumption that the perturbations are all significantly sparser than the original matrices, we can approximate the above equation by ignoring their 2^{nd}-order terms. That is, we drop the terms of the form $(\Delta\mathbf{M})^2$, where $\mathbf{M} \in \{\mathbf{C}, \mathbf{W}, \mathbf{L}, \mathbf{X}_k\}$. By expanding the last equation and applying this matrix approximation, we obtain Eq. (3.6). ■

B CCTN-inc: Incremental Algorithm

In Sec. 3.2, we leveraged Theorem 2 to propose CCTN-INC, an effective and fast approximation of CCTN, which handles incremental updates in the network structure, the introduction of new nodes, and changes in the behavioral patterns of existing nodes. In Algorithm 2, we give the high-level pseudocode of CCTN-INC.

Algorithm 2 CCTN-INC: Incremental version of CCTN

```
Input: Graph $G'(\mathcal{V}', \mathcal{E}')$; stacked matrices of $k$-type time-series $\{\mathbf{X}'_k\}$ with $T'$ timesteps, basis matrix of temporal patterns $\mathbf{W}$ and embedded-clustering matrix $\mathbf{C}$ found by Alg. 1; parameters $a_k$ and $\lambda$, dimensionality $d$ (same as Alg. 1)
Output: Vector with cluster assignments $\mathbf{c}'$
$\epsilon = 10^{-6}, \tau_{max} = 100$    // Constants for convergence
$\tau = 0$    // Initialization of iteration #
// Initialization based on the solutions of Alg. 1
$\mathbf{C}'_{(\tau)} = \mathbf{C}$    // Random init of rows of new nodes
$\mathbf{W}'_{(\tau)} = \mathbf{W}$    // Random init of cols of new timestamps
repeat
    $\tau = \tau + 1$
    // Step 1: Compute $\mathbf{W}'$ via Eq. (3.5)
    $\mathbf{W}'_{(\tau)} = (\sum_k a_k \mathbf{C}'^T_{(\tau-1)}\mathbf{C}')^{-1}(\sum_k a_k \mathbf{C}'^T_{(\tau-1)}\mathbf{X}'_k)$
    // Step 2: Compute $\mathbf{C}'$ via Eq. (3.6)
    $\Delta\mathbf{C}_{(\tau)} \sum_k a_k \mathbf{W}_{(\tau)}\mathbf{W}^T_{(\tau)} + \lambda\mathbf{L}\Delta\mathbf{C} = \sum_k a_k(\mathbf{X}_k(\Delta\mathbf{W}_{(\tau)})^T$
        $+ (\Delta\mathbf{X}_k)\mathbf{W}^T_{(\tau)} - \mathbf{C}(\Delta\mathbf{W}_{(\tau)})\mathbf{W}^T_{(\tau)}$
        $- \mathbf{C}_{(\tau)}\mathbf{W}_{(\tau)}(\Delta\mathbf{W}_{(\tau)})^T + \lambda(\Delta\mathbf{L})\mathbf{C}_{(\tau)})$
until $(||\mathbf{C}'_{(\tau)} - \mathbf{C}'_{(\tau-1)}||_1 < \epsilon$ & $||\mathbf{W}'_{(\tau)} - \mathbf{W}'_{(\tau-1)}||_1 < \epsilon)$ or $\tau > \tau_{max}$
// Step 3: Assign the nodes to clusters based on the inferred embeddings in $\mathbf{C}$' (each row is an 'observation').
$\mathbf{c}' = $ cluster_rows$(\mathbf{C}'_{(\tau)})$
return $\mathbf{c}'$
```

C Baselines

Here we present some details about the clustering methods that we considered as baselines in our experiments.

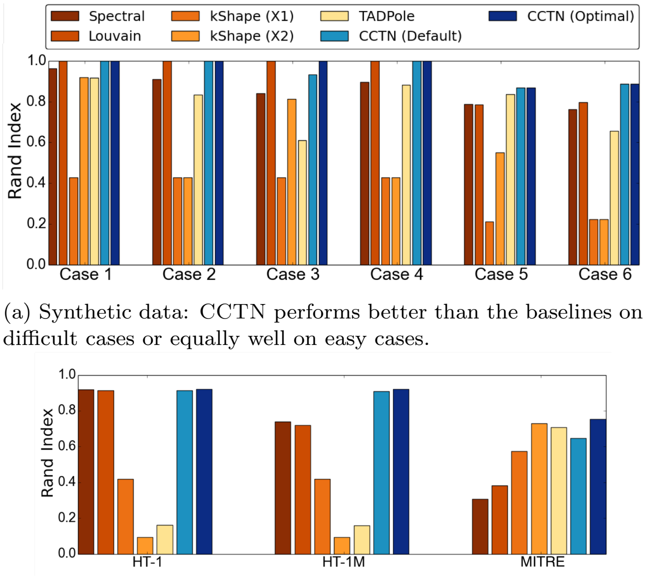
- **Spectral clustering** refers to a class of algorithms that utilize eigendecomposition to identify community structure. We utilize one such algorithm [15], which partitions a graph by performing k-means clustering on the top-k eigenvectors of the input graph. The idea behind this clustering is that nodes with similar connectivity have similar eigen-scores in the top-k vectors, and thus form clusters.
- **Louvain** is a modularity-based partitioning method for detecting hierarchical community structure. The method is iterative: (i) Each node is placed in its own community. Then, the neighbors j of each node i are considered, and i is moved to j's community if the move produces the maximum modularity gain. The process is applied repeatedly until no further gain is possible. (ii) A new graph is built whose supernodes represent communities, and superedges are weighted by the weighted sum of links between the two communities. The algorithm typically converges after a few passes.
- **k-Shape** relies on a scalable iterative refinement procedure, which creates homogeneous and well-separated clusters. As for the distance measure, k-Shape uses a normalized version of the cross-correlation in order to consider the shapes of time series while comparing them. Based on the properties of this distance measure, a method is developed to compute cluster centroids, which is used in every iteration to update the assignment of time series to clusters.
- Based on DTW, **TADPole** uses a pruning strategy that exploits both upper and lower bounds to remove a large fraction of the expensive distance calculations. This pruning strategy gives provably identical results to the brute force algorithm, but is at least an order of magnitude faster. It also uses a simple heuristic to order the calculations, thus casting the clustering as an anytime algorithm.

We discuss the parameter settings for the baselines in Sec. 4.3, and the setup for this experiment in Sec. 4.4.

D Accuracy of Cluster Recovery

In Sec. 4.4, we presented the results from comparing our method, CCTN, with the baselines in terms of accuracy (i.e., agreement with the ground-truth clusters in the labeled data). In the main paper, we included the plots that compare the methods in terms of NMI. Here we supplement our analysis with results on the Rand index in Fig. 6. The trends are similar to the ones we see for NMI, though the scores are consistently higher. Overall, CCTN has consistently strong performance across a wide variety of synthetic and real datasets, unlike the baselines which exhibit wide variability in their performance—for example, k-Shape works well on MITRE in terms of the Rand index, but performs poorly or worse than CCTN in the other real datasets and many of the synthetic cases.

In Fig. 7, we also present Rand index-based results about the approximation of CCTN by its incremental counterpart, CCTN-INC. The setup for this experiment is given in Sec. 4.4(2). We observe that, based on both metrics, CCTN-INC approximates CCTN very well in that it finds similar clusterings. The trends between NMI and Rand index are similar, with Rand index being consistently higher.



(a) Synthetic data: CCTN performs better than the baselines on difficult cases or equally well on easy cases.

(b) Real data: Overall, CCTN performs consistently better than or equally well as the baseline methods.

Figure 6: Rand index-based accuracy of CCTN and CCTN-INC on synthetic and real data.

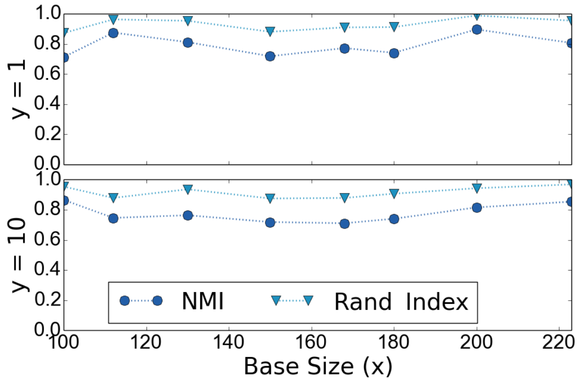
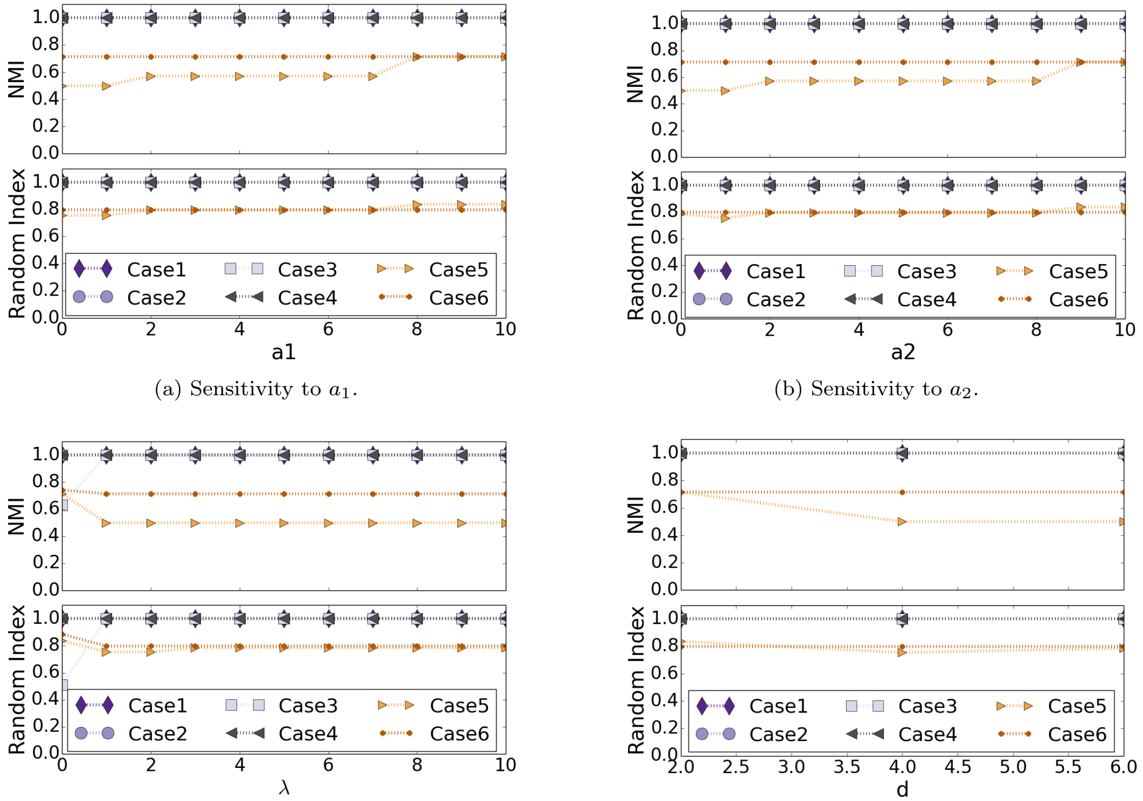


Figure 7: MITRE: CCTN-INC approximates CCTN well based on both the NMI and Rand index metrics.



(a) Sensitivity to a_1.

(b) Sensitivity to a_2.

(c) Sensitivity to λ.

(d) Sensitivity to d.

Figure 8: CCTN is robust with respect to the parameters. The different lines in the plots correspond to different synthetic cases, as described in the rightmost plot (d).

E Sensitivity of Clustering

The last question that we raised in Sec. 4 is about the robustness of CCTN to different parameter settings. To answer this question, we experiment with different combinations of parameters and evaluate the corresponding effect on clustering. In this experiment we call $a_1 = a_2 = \lambda = 1, d = 4$ 'default setting'. Note that these values are different from the default case that we discuss in Sec. 4.3. Then, we run CCTN on all the synthetic cases of Table 2 by fixing all the parameters to their default setting, with the exception of one parameter that we vary. The results are shown in Figure 8.

Observation 5. CCTN *is generally robust to different parameter settings. Based on the Rand index, it is more sensitive to λ than the other parameters. Practitioners may choose to tune λ primarily to achieve better performance.*

F Additional Related Work

We provide additional related work for our proposed coupled clustering of time-series and network problem.

Time-series Clustering. We discuss a set of representative works (not exhaustive) in Table 4 and refer to the survey [21] for more details. Various time-series clustering approaches focus on directly evaluating the similarity of raw data [38, 5] or extract features from raw data and then apply time-series clustering to the feature representation [34]. Regardless of data source, there are two completely different approaches of similarity measurements: distance-based and model-based. In the distance-based approach, coherence of time-series sequences is evaluated with distance functions, e.g. Euclidean distance function [41], Short time series distance [38], DTW distance [5], KL-divergence [32] etc. In model-based approaches, coherence is evaluated with model similarity, e.g. Hidden Markov Model [39] and auto-regression moving average (ARMA) [10, 42].

Table 4: Qualitative comparison of related approaches.

Methods	Data Source			Similarity	
	Raw	Feature	Representation	Dist	Model
[5, 38, 35, 32, 25]	✓			✓	
[10, 42]	✓				✓
[36, 41, 12]			✓	✓	
[34]		✓		✓	
[39]	✓			✓	✓

References

[30] Nurjahan Begum, Liudmila Ulanova, Jun Wang, and Eamonn Keogh. Accelerating dynamic time warping clustering with a novel admissible pruning strategy. In *KDD*. ACM, 2015.

[31] Marcella Corduas and Domenico Piccolo. Time series clustering and classification by the autoregressive metric. *CSDA*, 52(4):1860–1872, 2008.

[32] Rainer Dahlhaus. On the kullback-leibler information divergence of locally stationary processes. *Stochastic processes and their applications*, 62(1), 1996.

[33] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 2016.

[34] Cyril Goutte, Lars Kai Hansen, Matthew G Liptrot, and Egill Rostrup. Feature-space clustering for fmri meta-analysis. *Hum Brain Mapp*, 13(3):165–183, 2001.

[35] Konstantinos Kalpakis, Dhiral Gada, and Vasundhara Puttagunta. Distance measures for effective clustering of arima time-series. In *ICDM*. IEEE, 2001.

[36] Eamonn J Keogh and Michael J Pazzani. An enhanced representation of time series which allows fast and accurate classification, clustering and relevance feedback. In *KDD*, volume 98, 1998.

[37] T Warren Liao. Clustering of time series data – a survey. *Pattern recognition*, 38(11):1857–1874, 2005.

[38] Carla S Möller-Levet, Frank Klawonn, Kwang-Hyun Cho, and Olaf Wolkenhauer. Fuzzy clustering of short time-series and unevenly distributed sampling points. In *IDA*. Springer, 2003.

[39] Tim Oates, Laura Firoiu, and Paul R Cohen. Clustering time series with hidden markov models and dynamic time warping. In *IJCAI Workshop*, 1999.

[40] John Paparrizos and Luis Gravano. k-shape: Efficient and accurate clustering of time series. In *SIGMOD*, 2015.

[41] CT Shaw and GP King. Using cluster analysis to classify time series. *Physica D: Nonlinear Phenomena*, 58(1-4):288–298, 1992.

[42] Yimin Xiong and Dit-Yan Yeung. Time series clustering with arma mixtures. *Pattern Recognition*, 37(8):1675–1689, 2004.