Mining subgroups from temporal data : from the parts to the whole

Alexander Gorovits
University at Albany, State University of New York, gorovits.alexander@gmail.com

The University at Albany community has made this article openly available. Please share how this access benefits you.

Follow this and additional works at: https://scholarsarchive.library.albany.edu/legacy-etr

Part of the Computer Sciences Commons

Recommended Citation
https://scholarsarchive.library.albany.edu/legacy-etr/2690

This Dissertation is brought to you for free and open access by the The Graduate School at Scholars Archive. It has been accepted for inclusion in Legacy Theses & Dissertations (2009 - 2024) by an authorized administrator of Scholars Archive. Please see Terms of Use. For more information, please contact scholarsarchive@albany.edu.
MINING SUBGROUPS FROM TEMPORAL DATA
FROM THE PARTS TO THE WHOLE

by

Alexander Gorovits

A Dissertation
Submitted to the University at Albany, State University of New York
in Partial Fulfillment of
the Requirements for the Degree of
Doctor of Philosophy

College of Engineering and Applied Sciences
Department of Computer Science
Spring 2021
To family, old and new, for believing in me through it all.
To those that came before and those that will come to add their own parts.
ABSTRACT

A variety of dynamic systems can be broken down into potentially overlapping subcomponents with varying temporal behavior, ranging from communities in networks, to clusters of trajectories in spatiotemporal data, to co-evolving subsets within multivariate time series. Using explicit regularization on various temporal behaviors within a tensor factorization framework, I demonstrate means to mine these subgroups along with their temporal activities, as well as how that yields information about the overall systems. Additionally, I adapt this notion of temporal communities to the spatiotemporal setting to develop a reinforcement learning approach for optimizing co-ordinated communication between independent agents.
ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Petko Bogdanov, for taking me in new to the field, humoring occasionally wild ideas, and pushing many a late night submission. Without his support, the projects he exposed me to, and the (frequently necessary) criticism, this would have been much less achievable.

I’m grateful to my thesis committee Dr. Evangelos Papalexakis, Dr. Charalampos Chelmis, and Dr. Mariya Zheleva for support and understanding through my proposal, publication attempts, and defense. Thanks for being available and willing to help, throughout.

A heartfelt thank you to my colleagues and collaborators at the DMM/UbiNet labs, including Lin Zhang, Dan DiTursi, Maxwell J McNeil, Xhuliano Brace, Karyn Doke, Daniel DiTursi, Francisco Cancedda, Ravi Sood, Carolyn Kaminski, Boya Ma, and James Oswald. I’m proud of all the work we’ve done as a team, and thanks for all the feedback (and time spent listening to long-winded talks) throughout.

Thanks as well to Dr. Stacy Copp and other collaborators and co-authors on a set of biomaterials projects that, while not directly related to the work herein, allowed me to participate in some exciting applications of data-driven methods to science with some exciting, novel results.

Finally (but certainly not least), thank you to Allison for dealing with the aforementioned hare-brained schemes and late nights, and to my parents and grandparents for holding out hope despite the long road I took to get here.
CONTENTS

ABSTRACT ......................................................... iii

ACKNOWLEDGMENTS .............................................. iv

1. Introduction .................................................. 1
   1.1 Motivation ................................................. 1
   1.2 Limitations of prior work ................................. 2
   1.3 Thesis Statement ........................................... 3
   1.4 Outline of Dissertation .................................... 3

2. Learning Activity Regularized Communities ....................... 4
   2.1 Introduction ................................................. 4
   2.2 Background and Related work .............................. 7
   2.3 Problem formulation ....................................... 8
       2.3.1 Tensor factorization model (TF) ....................... 9
       2.3.2 Affiliation generative model in time (GM) .......... 10
   2.4 Algorithms ................................................ 10
       2.4.1 Regularized Tensor Factorization LARC-TF .......... 11
       2.4.2 Affiliation Model Solver LARC-GM ................ 14
       2.4.3 Learning the number of communities $k$ ......... 15
       2.4.4 Learning $\lambda_c$ and $\lambda_d$ using MDL ....... 17
   2.5 Experimental evaluation .................................. 18
       2.5.1 Data .................................................. 18
       2.5.2 Experimental Setup .................................. 20
       2.5.3 Quality, scalability, and parameter selection .... 20
       2.5.4 LARC at work ........................................ 24
3. Varying Temporal Behavioral Assumptions — Bursty Subgroups

3.1 Introduction

3.2 Related Works

3.3 Preliminaries and notation

3.4 Problem formulation

3.4.1 Enforcing bursty structure: designing $S$

3.4.2 Tensor model with background activity

3.5 Optimization solution

3.5.1 Alternating optimization

3.5.2 Decoupling objectives via ADMM

3.5.3 Optimization for the shape subproblems

3.5.4 The overall algorithm: MYRON

3.5.5 Datasets

3.5.6 Experimental setup

3.5.7 Detection of ground truth groups

3.5.8 Scalability

3.5.9 Burst detection

3.5.10 Estimating $K$

3.5.11 Case study: analysis of a Flickr dataset

3.6 Discussion

3.7 Conclusion

3.8 Extending to other behaviors

4. Connectivity Optimization via Reinforcement Learning
CHAPTER 1

Introduction

1.1 Motivation

An increase in data collection capabilities across a variety of modalities has resulted in access to numerous high-resolution temporal datasets including dynamic networks, multivariate time series, and spatiotemporal traces. In order to properly understand and utilize these data, the systems they describe must be understood. Herein, I propose that many of these dynamic systems are driven by the activities of subgroups contained within them. These subgroups are made up of individual entities (nodes in a network, mobile agents in a spatiotemporal trace setting, etc.) and can overlap. A key assumption is that the overall system evolution is driven by temporally-varying activity at the group level — individuals actions may create events, such as edges within the network, but activity is best modeled through groups of individuals, subsets of the entire system. Faithfully modeling and understanding the behavior of these subgroups, specifically in their temporal aspect, is key to understanding the entire system.

Such subgroups have meaningful interpretation, depending on the setting. In dynamic networks, subgroups are overlapping communities of nodes that may create different numbers of edges in different time slices — examples are groups of connected friends on a social network, colleagues on a work project, or discussants on a web forum. In spatiotemporal data, groups correspond to co-located individuals moving together — herds animals undergoing seasonal migrations (or nightly herding), car- or bus-pooling commuters, or floating objects driven by ocean currents. Multivariate time series data resulting from an array of sensors can similarly contain groups of co-evolving univariates determined by location or shared environment. Again, a key component is a shared element (interest, driving force, environment, or task) within these groups that is responsible for the shape of the temporal activity.

Potential applications of detecting groups and understanding their behaviors are numerous. Understanding the overall system through the function of subgroups or communities
allows for targeting demand, with examples including identifying and reinforcing high-load regions in a cellular communication network (e.g. by placing cell towers) or targeting particular groups of people or times of day for social media advertising. A wide variety of networks can see such applications, from adding additional stations or devices to a bike sharing network, to identifying compromised devices or key nodes in a botnet, to capturing and harnessing meaningful subgroups and interactions in neuronal or protein networks. Spatiotemporal data also admits various useful downstream applications including load prediction (e.g. for traffic routing or road construction) or interaction prediction. The temporal behaviors of groups themselves may be potentially useful as well, serving as data or useful features for classification, for example in the case of neuronal subnetworks that may activate differently in normal and disease states.

In addition to modeling the way in which an overall system is composed of these temporally active subgroups, it is important to understand and properly model the temporal behaviors themselves. In this work, I seek in part to describe meaningful temporal patterns and how to utilize this structural knowledge within our overall subgroup framework.

1.2 Limitations of prior work

Subgroup detection is an important ongoing field of research, including avenues such as community detection on networks [1] or trajectory clustering for spatiotemporal mobility data [2]. On the network end, many of these approaches are static, though dynamic networks are becoming a larger area of research in recent years [3, 4, 5]. A large focus is in evolutionary clustering, where groups themselves shift in membership over time [6, 7]. In these cases it is difficult to track what a consistent system subgroup is, much less what its behavior is. Therefore, many settings may benefit from considering instead groups that are static in membership but not behavior; when these subgroups overlap, this allows for growth or die-off as well.

Temporal behavior itself is an obvious and well-studied aspect of data as well [8], but the majority of works are quite limited in the forms and extent of various behaviors. On networks, for instance, much of the current literature focuses on different levels of temporal aggregation or a generic time dimension without meaningful regularization [5, 9, 10, 11], or on the evolution of the graph within some pre-defined time scale [12, 6, 13], as in evolutionary
clustering above. More complex behaviors are often studied in time series contexts, but there is still significant space for a more unified concept of differing temporal behaviors among subsets of series (certainly as opposed to standard autoregression or de-trending and seasonality approaches).

In general, there is a need for understanding the role of properly modeling temporal behavior when mining and describing dynamic systems in general, and for the role of within-system variation in the behaviors that may need careful approaches to properly represent. Initially, the approach herein focuses on static-membership groups whose behaviors vary in time, as distinct from the evolutionary approaches mentioned above.

1.3 Thesis Statement

*Modeling the temporal activity of subgroups within dynamic systems improves understanding of the overall system as well as improving quality in downstream tasks.*

1.4 Outline of Dissertation

In chapter 2 I present a general approach for discovering temporally regularized, stationary-membership subgroups within network data, and a simple but effective way of modeling subgroup temporal behavior. In chapter 3 I expand the introduced temporal behavior to more complex and more practically relevant forms, and investigate how incorporating such behaviors affects performance and structure of the modeling. Chapter 4 extends the notion of subgroup-driven system behavior to spatiotemporal settings that may be online, distributed, and with limited information — how can we effectively learn a model, and what role do the subcomponents play?
CHAPTER 2
Learning Activity Regularized Communities

2.1 Introduction

Mapping the community structure of a network is essential for understanding its underlying system and processes. Social circles, gene pathways and cliques of scientific collaborators are all examples of functional units in large networks that can be modeled as overlapping communities. Common approaches for community detection rely on a static network [14, 15, 1], however, temporal interaction data is becoming increasingly available and, thus, holds the potential to inform better community detection methods. Consider, for instance, a community among professionals in the workplace, in which communication is likely to occur during work hours and in week days. If we attempt to detect work groups from communications within this network, we would have to discover and take into account this community activation pattern. The problem becomes even more challenging when a node communicates with connections in other (non-professional) circles, e.g. family members and friends, within the same communication medium and with varying intensity and temporal resolution. How can we exploit the timings of these interactions by enforcing appropriate smoothness on their time course to tease apart overlapping communities and their activity periods?

An illustrative example of communities and their activity detected by our methods in bike trips data from Boston, MA is presented in Fig. 2.1(a). Nodes in this network are bike rental stations and trips between stations (check-out to drop-off) are modeled as temporal interactions. We detect the overlapping structure of groups of stations with heavy within-group traffic in contiguous intervals. The moving activation average over almost 2 years (top-right in Fig. 2.1(a)) reveals an important change point of expanding the bike rental service from downtown Boston (red) to the outskirts resulting in two previously inactive communities: green — rides within downtown and popular new stations (Harvard and MIT); and blue — rides including the outskirts. Such an analytic tool can inform activity-aware improvements by city planners as well as resource provisioning for the the bike rental
Figure 2.1: (a) A visualization of 3 color-coded groups of bike rental stations in Boston that observe strong within-group traffic in contiguous intervals of time detected by our algorithm LARC. Beyond the overlapping community structure, we identify interpretable community activity profiles (visualized top-right) elucidating a timepoint of the service expansion as well as university break dips. (b) LARC’s quality in detecting ground truth (GT) communities is consistently and up to 2.6x superior to baselines (lower divergence (DIV) is better) when the temporal interactions are heterogeneously spread in time—simulating data sampled at high temporal resolution (V=200,T=100...20k).

Our goal in this chapter differs from evolutionary clustering [16, 6] where the objective is to study the long-term evolution (growth, splits, merges) of network clusters. Instead, we focus on the short-term (repeated) activity within relatively stationary communities, a setting in which the interactions within the communities are much more common than changes in the overall community structure. Considering temporal information, however, is a knife that cuts both ways. If aggregated inadequately, temporal network data can reveal unrealistic patterns [17, 4, 18, 19]. Furthermore, the appropriate window for aggregation may not be uniform for the whole timeline [20]. Hence, in order to capitalize on temporal
network information, it is important to jointly learn an appropriate temporal resolution of the data as well as the actual community structure. Community detection is notoriously hard under multiple popular measures including conductance [4], modularity [21] and ratio cut [22]. Allowing for overlap among communities makes the search space even larger [14]. Furthermore, employing the dynamic interaction behavior — our goal in this chapter — adds yet another dimension to the already computationally challenging problem.

We propose a general framework for *Learning Activity-Regularized Communities (LARC)* which detects jointly the overlapping community structure and the activity periods of communities, directly from temporal network data. Our framework simultaneously optimizes community fit and activation profiles that are close to piece-wise constant functions, thus ensuring interpretability, resilience to noisy interactions, and temporal oversampling. To quantify community fit we consider both reconstruction and generative models and derive efficient solvers for the resulting optimization problems. In addition, we derive unsupervised solutions for automatic selection of the number of communities based on activation-aware core consistency diagnostic as well as for parameter learning based on the minimum description length (MDL) principle. Our solutions scale to large real-world instances and can be employed in a variety of applications.

Our contributions in this chapter are as follows:

- **Novel Problem Formulation**: We propose the novel problem of jointly learning the overlapping community structure and smooth activation profiles from dynamic interaction data.
- **Flexible Optimization Framework**: We develop a general framework LARC for the problem imposing fused lasso regularization on the activation profiles and demonstrate that it can be coupled with both generative and reconstruction models for community fit.
- **Real-World Utility**: Our extensive evaluation on synthetic and real-world data demonstrates LARC’s 2.6x quality improvement over baselines in high temporal resolution, and superior quality with temporal aggregation and varying community overlap. LARC also reveals botnet attack activity and transportation change points. It remains practical—completes in minutes—for large datasets.

LARC’s implementation, data generator and evaluation datasets are available at [http://www.cs.albany.edu/~petko/lab/code.html](http://www.cs.albany.edu/~petko/lab/code.html).
2.2 Background and Related work

**Static communities:** Our work is different from static (overlapping) community detection [14, 23, 24, 1, 25] in that we consider dynamic interactions to improve the quality and also provide an interpretable temporal activation profile for each community. Furthermore, our goal is complementary to static community detection approaches, as objective functions from the static setting can be generalized within our framework to the dynamic one. We demonstrate this for the affiliation generative model in [15]. In addition, we compare to static methods by aggregating the temporal interactions and demonstrate that utilizing the temporal information is advantageous for recovering ground truth communities.

**Temporal communities:** Various temporal subgraph detection methods have also been considered in the literature: communities [4, 5], dense subgraphs [26, 27, 28, 29], heavy-weight subgraphs [30, 31] and persistent subgraphs [32, 33]. Many of these methods detect one subgraph over one “active interval” at a time, as opposed to recurring activation of multiple subgraphs [4, 30, 31, 32, 33]. These methods do not consider overlap, cannot ensure stable community membership if iteratively run to extract multiple communities, and are sensitive to the temporal resolution (aggregation) of the interaction data. Other methods enforce user-defined consistency by introducing parameters such as number of occurrences and time span [26] or some notion of persistence (e.g. time-to-live interval) for interaction edges [27, 28, 33, 29]. Different from all the above, we detect multiple overlapping communities over time and let the data define the natural periods of activity which may vary with communities, application and across time. Finally, Gauvin et al. [5] detect both overlapping and data-driven dynamics of communities using tensor factorization (TF), however, as we show in our evaluation, employing TF without activity regularization is sensitive to the temporal resolution, and thus, results in sub-par quality.

**Evolving communities:** The goal in evolutionary clustering [16, 6] and evolving community and heavy subgraph detection [34, 35] is different from ours as they characterize how community membership changes in the long term. Instead we focus on detecting the stationary overlapping community structure in a short time frame in which membership is relatively stable.

**Tensor methods:** Tensor factorization (TF) has also been employed to detect communities from temporal [5, 36, 37] and multi-view [38, 39] network data. To the best of our knowl-
edge, this is the first work that enforces temporal smoothness constraints in the factorization model, in order to uncover more accurate communities. We demonstrate how tensor-based reconstruction error can be used as a goodness-of-fit metric in our framework and further extend tensor factorization approaches to handle our activity regularization objective. We compare LARC-TF experimentally with popular TF approaches, and demonstrate its superior performance.

2.3 Problem formulation

We next formalize the problem of joint detection of communities and their activity profiles. We expect that, as time progresses, communities (i) exhibit relatively stationary membership and (ii) alternate between active (multiple internal interactions) and inactive states. Thus, interactions in time can be utilized to improve detection as communities will be more discernible at an appropriate temporal scale as opposed to in a fully-aggregated graph. We represent the observed symmetric interactions among a finite node set \( V, |V| = n \) over a finite interval of discrete time steps \([1,T]\) as a 3-way tensor \( X \in \mathbb{R}^{n \times n \times T} \), where each element is a value modeling the number of interactions between a pair of nodes at a given time. The tensor face of observed interactions at time \( t \) is denoted as \( X(t) \) and can be viewed as a weighted undirected network snapshot for that time. Let \( C \in \mathbb{R}^{n \times k} \geq 0 \) be a community matrix specifying the strength of affiliation of nodes to each of \( k \) communities. Let also \( A \in \mathbb{R}^{t \times k} \geq 0 \) be an activation matrix, modeling the activation profiles of communities over time. A high value of \( A_{tk} \) denotes high level of activity of community \( k \) at time \( t \), i.e. many internal interactions.

Let \( J(X,C,A,k) \) be an error-of-fit function for \( k \) communities for given \( C \) and \( A \) matrices. Our goal is to minimize \( J \) while enforcing contiguous active and inactive periods for each community. To this end, we impose a smoothness regularization on the community activation profiles, i.e. the columns of \( A \). In particular, we incorporate a fused lasso [40] regularizer, which has been shown effective for piece-wise constant signal approximations [41]. In our setting we enforce shrinkage by an \( L_1 \) penalty on the activation matrix \( A \) and total-variation denoising by an \( L_1 \) penalty on the difference of consecutive values in \( A \)'s columns.
Formally, our regularization has the following matrix form:

$$R(A) = \lambda_c ||A||_1 + \lambda_d ||DA||_1,$$

(2.1)

where $D$ is a row-wise difference matrix with $D_{i,i} = -1, D_{i,i+1} = 1$ and 0 for all other elements, and $\lambda_c$ and $\lambda_d$ are regularization parameters controlling the importance of the regularization objectives.

**Definition 1 Dynamic Overlapping Communities:** Given a dataset $\mathbf{X}$, number of communities $k$, $\lambda_c$ and $\lambda_d$, solve:

$$\min_{A,C} J(X,C,A,k) + R(A).$$

(2.2)

The error fit function $J$ can be instantiated based on different models reflecting how “good overlapping communities” should manifest in terms of inter-node interactions. Next, we demonstrate two possible realizations for $J$ based on (i) tensor factorization (TF) and (ii) a temporal extension of the affiliation generative model (GM).

### 2.3.1 Tensor factorization model (TF)

Since the observed interaction data $\mathbf{X}$ in our setting is a tensor, tensor factorization models are a natural fit for learning the communities $C$ and temporal profiles $A$ as corresponding tensor factors. We focus on CANDECOMP/PARAFAC [42], which decomposes a tensor into a sum of rank-one tensors $\mathbf{X} \approx \sum_k c_k \circ c_k' \circ a_k$, which we also denote in matrix form as $\mathbf{X} \approx [C, C', A]$, where $C'$ is equivalent to our community matrix $C$ and $A$ holds the temporal dimension factors corresponding to our activation matrix. Note that since temporal snapshots $X(t)$ are symmetric matrices, solutions for community factor matrices will be equivalent $C \approx C'$, and thus, we further simplify the reconstruction notation to $\mathbf{X} \approx [C, A]$. PARAFAC minimizes the factorization reconstruction error for a fixed number of factors $k$, which we adopt as error-fit function:

$$J_{TF}(\mathbf{X}, C, A, k) = ||\mathbf{X} - [C, A]||_F^2.$$

(2.3)
2.3.2 Affiliation generative model in time (GM)

Probabilistic affiliation generative models (GM) [43, 44] provide an alternative to reconstruction (TF) models. In static network GMs, nodes have affiliations to communities $C$ which drive randomly observed connections among affiliated nodes [15]. Here, we extend this model to overlapping communities over time, arriving at an alternative error-of-fit function $J_{GM}$. Specifically, we define the probability of observing an edge $(i,j)$ at time $t$ due to community $k$ as: $P_k(i,j,t) = 1 - e^{-C_{ik}A_{tk}C_{jk}}$. Note, that if the temporal profile of the community is fixed $A_k = 1$, the likelihood of interaction reduces to that in the static affiliation model [15]. The temporal profile $A$ acts as a selector enabling interactions when the community is active ($A_{tk} > 0$) and inhibiting them during periods of inactivity. The probability of interaction due to any community is then:

$$P(i,j,t) = 1 - \prod_k (1 - P_k(i,j,t)) = 1 - e^{-C_iA(t)C_j^T}, \quad (2.4)$$

where $A(t) = \text{diag}(A_k)$ is a diagonal matrix of community activation states at time $t$. Assuming independence of interactions, the conditional likelihood $L(\mathbf{X}|C,A)$ of observing $\mathbf{X}$ is then:

$$L(\mathbf{X}|C,A) = \prod_t \prod_{\mathbf{X}(i,j,t) \neq 0} P(i,j,t) \prod_{\mathbf{X}(i,j,t) = 0} [1 - P(i,j,t)]. \quad (2.5)$$

Note that the basic probabilistic affiliation model, unlike the reconstruction model (TF), assumes binary data, i.e. interactions are either 1 or 0. To make it applicable to non-binary data, one can consider thresholding schemes. We define the GM error-of-fit function based on the log-likelihood $LL(\mathbf{X}|C,A)$ of observed interactions:

$$J_{GM}(\mathbf{X},C,A,k) = -LL(\mathbf{X}|C,A) = 
\sum_t \left[ \sum_{\mathbf{X}(i,j,t) = 0} \log (1 - e^{-C_iA(t)C_j^T}) - \sum_{\mathbf{X}(i,j,t) \neq 0} C_iA(t)C_j^T \right].$$

2.4 Algorithms

In this section, we propose solvers that minimize the two objectives corresponding to the error-of-fit functions $J_{TF}$ and $J_{GM}$ combined with the same fused lasso regularization term $R(A)$. As we demonstrate experimentally, both models show comparable quality
in recovering ground truth communities that is superior to baselines due to their shared regularization approach which explicitly models smooth active/inactive community behavior. We further derive estimators for (i) the optimal number of communities \( k \) and (ii) the regularization parameters \( \lambda_c \) and \( \lambda_d \).

### 2.4.1 Regularized Tensor Factorization LARC-TF

The objective for our reconstruction model is as follows:

\[
\min_{C,A} \| X - [C,A]\|^2_F + \lambda_c \| A \|_1 + \lambda_d \| DA \|_1.
\]  

(2.6)

It can be viewed as a regularized tensor factorization problem in which we have imposed a fused lasso penalty \( R(A) \) to enforce piecewise-constant and sparse solution for the time path \( A_{:,i} \) of each community. Furthermore, in our solutions for Eq. 2.6 we seek to obtain non-negative factors \( C \) and \( A \) as they model affiliation and temporal activation respectively, thus ensuring interpretability within our problem. We extend a commonly-used non-negative factor method for solving the PARAFAC problem, namely alternating least squares (ALS) with non-negative factors [42]. An ALS solution for PARAFAC keeps two factors fixed, and takes advantage of the convexity and existence of a closed-form analytical solution for the third. Iterative updates for any factor \( U_1 \), assuming the remaining two \( U_2 \) and \( U_3 \) are fixed, have the following form:

\[
U_1 \leftarrow \arg \min_{U_1} \| X^{(1)} - (U_3 \odot U_2)U_1 \|^2_F,
\]  

(2.7)

where \( X^{(1)} \) is the tensor unfolding on the updated dimension, the factors \( U_i \) correspond to our community \( C \) or activation \( A \) matrices, and \( \odot \) is the Khatri-Rao product [45]. We cannot use this framework directly as we need to incorporate the fused lasso regularization \( R(A) \). Our formulation, however, retains some of the advantageous properties allowing an ALS-like solution, namely simplicity of updates and convexity.

To solve the updates efficiently, we adopt the Alternating Direction Method of Multipliers (ADMM) which has recently been employed in a number of high-dimensional large-scale problems for efficiently utilizing batch updates typically occurring in ALS [46]. In particular, we devise an Alternating Optimization ADMM (AOADMM) which combines the alternat-
Algorithm 1 LARC-TF

Require: Tensor $\mathbf{X}$, number of factors $k$, regularization parameters $\lambda_c$, $\lambda_d$

Ensure: Community $C \approx C'$ and activation $A$ matrices.

1: Initialize $C, C', A$ randomly
2: Initialize residual matrices $R_C, R_{C'}, R_A$ to 0
3: while The factors $C, C', A$ have not converged do
4:   for Each factor $H$ in $\{C, C', A\}$ do
5:      Let $\mathbf{R}_H$ and $\text{dim}(H)$ be the residual and tensor dimension of factor $H$
6:      Let $H_1$ and $H_2$ be the other two fixed factors
7:      $\tilde{H} \leftarrow H_1 \odot H_2$
8:      $\rho \leftarrow \text{tr}((\tilde{H}^T \tilde{H} + \rho I))$
9:      $L \leftarrow \text{Lower Cholesky decomposition of } (\tilde{H}^T \tilde{H} + \rho I)$
10:     $F = \text{MTTKRP}(\mathbf{X}, \tilde{H}, \text{dim}(H))$
11:    while Not converged do
12:       $\tilde{H} \leftarrow (L^T)^{-1}L^{-1}(F + \rho(H + R_H))$ \hspace{1cm} \triangleright Optimized Eq. 2.9
13:       $H \leftarrow \text{proxOpp}(H, \tilde{H}, R_H, \rho, \text{dim}(H))$
14:       $R_H \leftarrow R_H + H - \tilde{H}^T$ \hspace{1cm} \triangleright Eq. 2.11
15:    end while
16:   end for
17: end while
18: return $C, C', A$

ing least squares and the ADMM framework [45]. Intuitively, the main idea is to divide the problem into simpler-to-update blocks before reconciling these partial solutions. We enforce a non-negativity constraint on the updates of both community factors $C, C'$ and design a custom update for the activation factor $A$ that handles the fused lasso penalty and also enforces non-negativity. The objective in our ADMM update for $A$ is to solve the following convex sub-problem (adding $R(A)$ maintains convexity):

$$
\min_{A, \tilde{A}} \|X_{(3)} - \tilde{C}\tilde{A}\|_F^2 + \lambda_c\|A\|_1 + \lambda_d\|DA\|_1 \text{ s.t. } A = \tilde{A}^T, A \geq 0,
$$

(2.8)

where $X_{(3)}$ is the tensor unfolding on the third temporal dimension, $\tilde{C} = C' \odot C$ is the Khatri-Rao product of the community factors and $\tilde{A}$ is an auxiliary ADMM variable used to update $A$. The minimization can be solved by iterating over the following update sequence:

$$
\tilde{A} \leftarrow (\tilde{C}^T \tilde{C} + \rho I)^{-1}(\tilde{C}^T X_{(3)} + \rho(A + R_A)^T)
$$

(2.9)

$$
A \leftarrow \arg \min_{\tilde{A}} \lambda_c\|A\|_1 + \lambda_d\|DA\|_1 + \rho/2\|R_A + A - \tilde{A}^T\|_F^2
$$

(2.10)

$$
R_A \leftarrow R_A + A - \tilde{A}^T,
$$

(2.11)

where $\rho = \text{tr}(\tilde{C}^T \tilde{C})$ is the trace of the Khatri-Rao product of the community factors and
its transpose and $R_A$ is a running residual matrix for factor $A$. The first “fit” update (Eq. 2.9) and third “residual” update (Eq. 2.11) are common for all factors (i.e. $C$ and $C'$ as well as $A$) and there exist fast solutions for them based on Lower Cholesky decomposition and Matricized Tensor Times Khatri-Rao Product (MTTKRP) in non-regularized ADMM methods [45] detailed further in the Alg. 1. The second “regularization update” (Eq. 2.10) is also referred to as the proximity operator (proxOpp) of the trace-scaled regularization function $1/\rho R(A)$. While it involves minimization of a convex function of $A$, there is no closed-form analytic solution for it, so we employ coordinate descent with a non-negativity constraint for this step.

Algorithm 1 shows the steps of our AOADM approach LARC-TF for fused lasso tensor factorization. After initialization of the factors and their corresponding residual matrices (Steps 1,2), we iteratively update the factors one at a time while keeping the other two fixed (Steps 3-17) until convergence. In the ADMM update step for each factor $H$ (Steps 4-16) we first pre-compute several matrices and scalars that let us speed up the fit update from Eq. 2.9. Namely, the Khatri-Rao product of the fixed factors $\tilde{H}$ (Step 7); the trace $\rho$ (Step 8); a lower Cholesky decomposition of the first inverted matrix $(\tilde{H}^T \tilde{H} + \rho I)$ of Eq. 2.9 (Step 9), and the MTTKRP (Step 10). Note, that all the above are constant during the repeated updates of $H, \tilde{H}, R_H$, and thus precomputing them only once saves time. In the updates of $H$ (Step 13) we employ the appropriate proximity operator for each factor. For factors $C, C'$, since we enforce non-negativity the proximity operator simply replaces negative elements with 0, i.e. element-wise $\max(H, 0)$. If the update is for the activity factor $A$, however, we need to solve the minimization problem in Eq. 2.10. To this end, we perform a coordinate descent with line search to determine an appropriate learning rate $\beta$, where the main update is along the gradients for each timestep:

$$\frac{\partial}{\partial A_t} = \lambda_c I + \lambda_d sgn(A_t - A_{t-1}) - sgn(A_{t+1} - A_t) + \rho(R_A + A_t - \tilde{A}_t),$$

where $A_t$ is a short hand for the $t$-th row of $A$, and $sgn()$ is the element-wise sign function setting elements to $\{+1, -1\}$ depending on their sign. Non-negative projection, similar to those for $C$ and $C'$, is also applied at the end of the gradient descent for $A$.

The gradient descent in Step 13 has the highest computational footprint, being nested
in two convergence loops and further depending on $T$ to ensure smoothness of $A$’s columns. However, all operations preserve sparsity and, thus, are expected to scale almost linearly with the sizes of the input. Additionally, allowing a fixed update size instead of a full line search results in significant speedup at minimal quality expense. Our experimental evaluation reveals that with increasing $T$ the number of iterations for $A$’s convergence grows slightly, however, the overall running time remains practical for our largest instances.

2.4.2 Affiliation Model Solver LARC-GM

To minimize the objective $f_{GM} = J_{GM}(X, A, C, k) + R(A)$, we consider block coordinate descent methods. The gradient with respect to a community membership vector $C_i$ is:

$$
\frac{\partial f_{GM}}{\partial C_i} = \sum_t \left( \sum_{(i,j,t) = 1} C_j A(t) e^{-C_i A(t) C_j^T} - \sum_{(i,j,t) = 0} C_j A(t) \right).
$$

Similarly, differentiating with respect to $A_{tk}$, we get:

$$
\frac{\partial f_{GM}}{\partial A_{tk}} = \sum_{(i,j,t) = 1} C_{ik} C_{jk} e^{-C_i A(t) C_j^T} - \sum_{(i,j,t) = 0} C_{ik} C_{jk} - \lambda_s - \lambda_d (\text{sgn}(A_{tk} - A_{(t+1)k}) - \text{sgn}(A_{(t+1)k} - A_{tk})),
$$

which can be combined into a single update for the block $A_{ti}$. Direct coordinate or gradient descent will not scale for large instances $X$, hence we seek to scale our solutions by avoiding re-computation of the full gradient. Particularly, similar to the static network affiliation model solutions [15], only a small number of elements are updated in the unobserved edges component. Hence, one can re-write the no-edge portion of the update as:

$$
\sum_{X_{i,j,t} = 0} C_j A(t) = \sum_j C_j A(t) - C_i A(t) - \sum_{j \in N^t_i} C_j A(t), \quad (2.12)
$$

where $N^t_i$ is the set of neighbors of $i$ (i.e. nodes with which $i$ interacted) at time $t$. We can thus, compute and store $\sum_j A(t) C_j$ at each iteration (over all $i$), leading to faster updates of $C_i$ over the much smaller set of neighbors. A similar approach can be adopted to speed up $A$’s gradient as well. First, we notice that we can update either entire faces (fixed $t$) or communities (fixed $k$) at a time. In the first case, the $\sum_{i,j \notin X(t)} C_{ik} C_{jk}$ term can be similarly
Algorithm 2 LARC-CCD: Detect $k^*$ with Time-Warped CCD

Require: Tensor $\mathbf{X}$, factorization $[C, C', A]$ produced by LARC

Ensure: Activity-aware optimal $k^*$

1: $[\mathbf{U}, \Sigma, \mathbf{V}] \leftarrow \text{SVD}(A)$
2: for $r=1: k$ do
3:   $\Pi_r = \mathbf{U}(:,1:r) \ast \mathbf{U}(:,1:r)^T$
4:   $\mathbf{X}_r = \mathbf{X} \times_3 \Pi_r$
5:   $A_r = \Pi_r \ast A$
6:   $c(r) = \text{efficient_corcondia}(\mathbf{X}_r, C, C', A_r, 1_k)$
7: end for
8: $k^* = \text{AutoTen}(\max(c))$
9: return $k^*$

decomposed by storing $\sum_j C_j k$ in vector form for all $k$, as $C_\Sigma = \sum_j C_j$. We can then iterate over $i$ (instead of $i,j$) and compute:

$$ \frac{\partial f_{\text{GM}}}{\partial A}(t) = -\lambda_s I - \lambda_d * (\text{sgn}(A(t) - A(t-1)) - \text{sgn}(A(t+1) - A(t)))$$

$$+ \text{diag}\left( \sum_i \left( \sum_{j \in N_i^t} \frac{C_i \circ C_j e^{-C_i A(t) C_j}}{1 - e^{-C_i A(t) C_j}} - C_i \circ (C_\Sigma - C_i - \sum_{j \in N_i^t} C_j) \right) \right),$$

where $\circ$ denotes the Hadamard (element-wise) product of the two matrix rows, and $\text{diag}()$ is the diagonal matrix of the argument vector. LARC-GM then iterates between coordinate descent steps on for $C$ and $A$ using also a line search for an appropriate learning rate. The update optimizations in LARC-GM increase its speed compared to direct coordinate descent significantly, however, as we show experimentally LARC-TF scales much better than than LARC-GM, while they both produce better quality communities compared to baselines.

### 2.4.3 Learning the number of communities $k$

An important question when analyzing a new dataset is how to set $k$. We extend TF consistency approaches to our activity regularized objectives and develop a method LARC-CCD for selecting $k$ which outperforms regularization-oblivious alternatives. Finding the rank of a tensor is a NP-hard problem, however, there exist heuristic models such as the Core Consistency Diagnostic (CCD) algorithm \[47, 48\]. Given a tensor $\mathbf{X}$ and its PARAFAC factorization $[C, C', A]$, CCD provides a number indicative of the factorization quality, thus allowing for selection of maximum number of good-quality communities \[49\]. In our case, however, using CCD as a black-box may lead to bad estimates of $k$ as our solution’s factorization $[C, C', A]$ is a product of an activity smoothness regularizer, which as we
Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>V</th>
<th>T</th>
<th>nnz</th>
<th>k</th>
<th>LARC-CCD</th>
<th>AutoTen[49]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Synthetic</td>
<td>200</td>
<td>20k</td>
<td>61k</td>
<td>5</td>
<td>32%</td>
<td>71%</td>
</tr>
<tr>
<td>Football</td>
<td>115</td>
<td>2k</td>
<td>17k</td>
<td>12</td>
<td>13%</td>
<td>42%</td>
</tr>
<tr>
<td>Reality Min.</td>
<td>94</td>
<td>8k</td>
<td>0.1m</td>
<td>5</td>
<td>12%</td>
<td>56%</td>
</tr>
<tr>
<td>Reddit-sports</td>
<td>120k</td>
<td>267</td>
<td>1.1m</td>
<td>15</td>
<td>27%</td>
<td>57%</td>
</tr>
<tr>
<td>Reddit-news</td>
<td>140k</td>
<td>267</td>
<td>0.7m</td>
<td>5</td>
<td>21%</td>
<td>40%</td>
</tr>
<tr>
<td>Bike Rides</td>
<td>145</td>
<td>628</td>
<td>0.8m</td>
<td>-</td>
<td>11%*</td>
<td>45%*</td>
</tr>
<tr>
<td>Botnet</td>
<td>20k</td>
<td>6k</td>
<td>0.5m</td>
<td>-</td>
<td>20%*</td>
<td>20%*</td>
</tr>
</tbody>
</table>

Table 2.1: Dataset statistics (cols 1-5) and success in the estimation of $k$ in datasets with ground truth (GT) communities (cols 6-7). Note that due to the lack of ground truth communities in the Bike Rides and Botnet datasets, only variance in the estimation of $k$ is reported.

Table 2.2: Comparison of quality (DIV and NMI) and running time in seconds for all competing methods and datasets. Note that, due to the lack of GT communities in the Bike Rides and Botnet datasets, only running time is reported.

demonstrate experimentally, is different from no-regularization PARAFAC factorization.

In order to make CCD amenable to our regularization, we need to “compress” the temporal mode of the tensor in a way that respects the temporal smoothness discovered by LARC. If $A$ yields very smooth latent factors, we aim to compress the temporal mode accordingly, so that we adjust the tensor to that smoothness. The key to that compression is the row space of the activity matrix $A$: if there exists a subspace of that row space which, when we project both the tensor and matrix $A$, yields a higher core consistency than simply using the computed factors and the uncompressed tensor, we choose that subspace to generate the core consistency that characterizes the quality of our solution.

Algorithm 2 summarizes our method for learning $k$ called LARC-CCD. Given $\mathbf{X}$ and a candidate decomposition $[C, C', A]$, we first compute the the Singular Value Decomposition (SVD) of $A = U\Sigma V^T$, where $U$ is a basis for $A$’s row space. Then we quantify the activity-aware CCD $c(r)$ for all $r \leq k$ (Steps 2-8) and maintain the $r$ that maximizes $c(r)$. For each $r$
we create a projector matrix $\Pi_r$ for the subspace defined by the dominant $r$ singular values of $A$. We use $\Pi_r$ to compress the tensor $\mathbf{X}$ by taking its $3$-mode product $\times_3$ with $\Pi_r$ (Step 4). The $n$-mode product multiplies a tensor and a matrix that match on the $n$-th mode of the tensor, in the same fashion as matrix-matrix multiplication. We similarly compress $A$ to obtain $A_r$ (Step 5) and compute the core consistency of $\mathbf{X}_r$ using $[C, C', A_r]$, employing efficient_corcondia [48] (Step 7). The highest core consistency value $c(r)$ is supplied to AutoTen [49] which estimates the rank $k^*$, which reveals the number of communities in the data (Step 9). While the optimal $k$ detection is tensor-oriented, one can easily adopt it for GM and other error-of-fit models, relying on a fixed natural number of communities in the data.

2.4.4 Learning $\lambda_c$ and $\lambda_d$ using MDL

The regularization parameters $\lambda_c, \lambda_d$ control the relative importance of the fused lasso regularization in our objective functions. Thus, it is important to set them appropriately to balance the contributions of $J$ and $R(A)$. We propose to set $\lambda_c$ and $\lambda_d$ based on the Minimum Description Length (MDL), where we aim to minimize the number of bits needed to encode errors due to the fit and the number of “switches” between active and inactive community states in $A$. Intuitively, the higher the deviation of each element of $\mathbf{X}(i,j,t)$ from its reconstruction, the more bits are needed to encode this error in a lossless compression employing the characterization.

For the case of LARC-TF, we formalize the average bits to encode the the error of reconstruction as:

$$B_{TF}^{\{\lambda_c, \lambda_d\}} = -\log \left( ||\mathbf{X} - \mathbf{X}_{\{\lambda_c, \lambda_d\}}||_F^2 / ||\mathbf{X}|| \right),$$

where $\mathbf{X}_{\{\lambda_c, \lambda_d\}}$ is the reconstruction obtained by employing LARC-TF with parameters set to $\lambda_c$ and $\lambda_d$, and $||\mathbf{X}||$ is the number of elements of the tensor. Similarly we define the average bits to encode error due to LARC-GM as:

$$B_{GM}^{\{\lambda_c, \lambda_d\}} = -\log \left( \sum_{i,j,t} [\mathbf{X}(i,j,t) - P_{\{\lambda_c, \lambda_d\}}(i,j,t)]^2 / ||\mathbf{X}|| \right),$$

where $P_{\{\lambda_c, \lambda_d\}}(i,j,t)$ is the probability of observing an edge according to the model learned by LARC-GM using $\lambda_c$ and $\lambda_d$. 

17
The second part of our MDL encoding is the number of “switches” between active and inactive community states which we quantify as 
$$\Delta\{\lambda_c, \lambda_d\} = -\log(||DA_{\{\lambda_c, \lambda_d\}}||^2)/||A_{\{\lambda_c, \lambda_d\}}||,$$
where $A_{\{\lambda_c, \lambda_d\}}$ is the community activation matrix learned using the corresponding regularization parameters by either of the models. To find the parameters, we then minimize the total number of bits:
$$\{\lambda_c, \lambda_d\} = \arg\min B^{\{\lambda_c, \lambda_d\}} + \Delta^{\{\lambda_c, \lambda_d\}},$$
where $B^{\{\lambda_c, \lambda_d\}} \in \{B_{GM}^{\{\lambda_c, \lambda_d\}}, B_{TF}^{\{\lambda_c, \lambda_d\}}\}$. Minimizing the objective depends on invoking LARC-TF or LARC-GM, thus, we perform a grid search over possible values and pick the configuration that minimizes MDL. We demonstrate that in a variety of synthetic and real datasets setting the parameters according to the MDL principle results in optimal quality of detecting ground truth communities.

A similar heuristic can be utilized to learn a variety of parameters, including our rank. In that case, however, we would need a more complex encoding to account for the additional information held by an increased rank; we therefore prefer the more established rank method above.

### 2.5 Experimental evaluation

We evaluate the quality, scalability, and real-world utility of LARC on both real and synthetic datasets. All experiments are for single core execution of our methods implemented in Matlab. For tensor manipulations we use the Tensor Toolbox for Matlab [51, 52].

#### 2.5.1 Data

We summarize the datasets used for evaluation in Table 2.1.

**Synthetic:** We generate the community structure and a smooth temporal activation. $|V|$ nodes are randomly assigned to $k$ overlapping communities $C$ while fixing the average community overlap (Jaccard similarity). Piece-wise constant activation profiles $A_{tk}$ are sampled from a Markov chain with active 1 and inactive 0 states for length $T$ and state-change probability of $p = 0.2$. We next generate temporal interactions for each snapshot $X(t)$ using $A, C$ and based on the GM specified in Sec 2.3.2. We also “stretch” the temporal dimension of instances in a controlled fashion to simulate temporal oversampling and varying
Figure 2.2: Comparison of LARC-TF’s quality with that of competing techniques for varying levels of temporal aggregation on synthetic (V=200,T=20000)\(^{(a)}\), and Reality Min. (b) datasets. Quality comparison for increasing community overlap (measured in Jaccard Similarity) on synthetic data (V=200,T=2500)\(^{(c)}\). Comparison of scalability with increasing V and fixed nnz=const (T=50)\(^{(d)}\), increasing V and nnz=1\% (T=50)\(^{(e)}\), and increasing T (V=100)\(^{(f)}\). Quality of estimating the GT number of communities \(k\) for increasing average stretch on synthetic (T=1k,V=100)\(^{(g)}\). DIV and MDL from Eq. 2.15 as a function of the regularization parameters \(\lambda_c\) and \(\lambda_d\)\(^{(h)}\).

rates of interactions within communities across time. We map each original time step \(t\) to \(s \sim \text{Poisson}(\lambda_{st})\) time steps, where the interactions in \(X(t)\) are uniformly distributed across the \(s\) new snapshots. This makes recovering the original profile and communities increasingly challenging.

Real-world: We employ several real-world datasets from different domains: Football \[53\]; Reality Min. \[54\] captures the temporal interactions (calls, texts, BT proximity) among students and faculty with self-reported friendship relations which we use to get ground truth communities employing BigCLAM \[15\]; Reddit datasets contain exchanges between users (posts and replies) on reddit.com, where the ground truth communities are based on subset of sports and news subreddits and their participants \[55\]; Bike Rides consists of rides between bike rental stations in Boston, MA over 2 years \[56\]; and Botnet contains inter-IP flows of university normal and botnet machines and includes several time-annotated DDoS attacks involving traffic of 10 bot and 1 victim IP \[57\].
2.5.2 Experimental Setup

**Baselines:** We evaluate our methods’ quality in retrieving ground truth (GT) overlapping communities and running time in comparison to three baselines. *TF* is a tensor factorization method proposed for overlapping temporal community detection by Gauvin et al. [5] and can also be viewed as a special case of our LARC-TF, where regularization in time is turned off. *BigCLAM* is the state-of-the-art method for overlapping community detection based on the affiliation generative model, which we extend to time to obtain LARC-GM [15]. *Non-negative matrix factorization (NMF)* is another popular approach for overlapping communities [50]. Since they operate on static graphs, we employ both BigCLAM and NMF on an aggregated temporal interactions static graph.

**Metrics:** For datasets with GT, we compare the level of agreement between GT and learned communities by all competing techniques. The *Kullback—Leibler divergence (KL-Div)* has been previously used to evaluate overlapping cluster solutions [58], where both GT and learned communities are treated as distributions over the nodes and the measure quantifies the differences between them. KL-Div is, however, not symmetric and also not defined when the distributions have regions of 0 density. Hence, we apply a metric alternative *Jansen-Shannon divergence (DIV)* [59] that handles 0 probabilities and varies between 0 (no divergence) and 1. We also adopt the normalized mutual information (NMI) [60] to compare learned and GT communities, where higher NMI corresponds to better detection of GT. This measure requires 0/1 membership, hence, we threshold community vectors using values ranging in $[10^{-3}, 1]$ in order to obtain the best NMI score for each method.

2.5.3 Quality, scalability, and parameter selection

**Quality.** The main advantage of LARC is in its treatment of time: enforcing solutions in which communities alternate between active/inactive in contiguous periods. To test this experimentally, we simulate temporal oversampling by stretching a smooth instance with expected on/off behavior by “stretching” it in time and spreading a given time slice’s temporal interactions to several new time slices controlled by an average stretch parameter. Fig. 2.1(b) (Sec. 2.1) shows a quality comparison in terms of Divergence (DIV) from GT communities of all competing techniques for increasing average stretch of a Synthetic dataset. For small stretch, i.e. a well-behaved smooth instance, LARC-TF and LARC-GM outperform TF by
a small margin and static methods (NMF and BigCLAM) by a factor of 3 in DIV. For increasing stretch, LARC-TF maintains a near constant quality, while TF deteriorates to the quality level of BigCLAM (2.6x deterioration). This behaviour is due to regularization forcing LARC-TF to continue considering a temporal segmentation of time at which communities are most discernible, while TF is affected by only observing partially the interactions of a community in individual timeslices and not employing any smoothness in time. At the same time, static methods suffer from an opposite extreme - over-aggregation in which the communities become also hard to discern. While LARC-GM performs very well in terms of quality, it does not scale to long timelines due to its reliance on coordinate descent methods for both A and C.

Since communities are elusive to the baseline methods, at both high and low (full aggregation) temporal resolution, we next investigate the feasibility of varying regular aggregations to get better communities in a synthetic Fig. 2.2(a) and the Reality Mining Fig. 2.2(b) datasets. The trends in both figures reveal the aforementioned challenges at both ends of aggregation and a slightly better performance for TF for medium aggregation. LARC, however, maintains a consistently better quality than the baselines at high resolutions (on synthetic) and as the aggregation coarsens deteriorates to the performance of TF as at these levels the useful temporal information is lost. Note, that in Reality mining (Fig. 2.2(b)), at the highest resolution even LARC suffers from fragmented communities in time and regularization cannot really attain the best quality at some reasonable small aggregation level. Note also that the GT in Reality mining relies on BigCLAM-extracted overlapping communities from the user-reported friendship graph and thus may not be ideal, resulting in relatively high DIV values. Comparison to LARC-GM is again omitted due to limited scalability for high temporal resolutions and similar to LARC-TF’s performance for low-resolutions.

Naturally, the quality of community detection deteriorates with the amount of overlap among communities as evident in Fig. 2.2(c). However, thanks to the timing of interactions coupled with smoothness regularization LARC-TF consistently outperforms all baselines. Interestingly, the temporal information loses its utility when there is very small overlap and BigCLAM performs on par with LARC-TF although using fully aggregated as opposed to temporal data. As demonstrated in previous studies, NMF performs consistently worse than BigCLAM and thus all other competing methods.
**Scalability.** While exhibiting good quality on small instances LARC-GM does not scale well with both \( V \): Figs. 2.2(d), 2.2(e) and \( T \): Fig. 2.2(f) due to its reliance on coordinate methods for both \( A \) and \( C \). LARC-TF, on the other hand, scales similar to TF and both of them slightly slower than the static baselines for increasing \( V \) while keeping the number of non-zeroes (nnz) in the input constant Fig. 2.2(d). NMF and BigCLAM’s time increases on par with the TF methods when the nnz is kept at 1% of the tensor size for increasing \( V \) Fig. 2.2(e), since the resulting aggregate graphs densify with the nnz. As expected, since LARC-TF performs smoothing via a descent method on the activation profiles \( A \), its running time increases faster than that of TF—computational time invested to enable its superior quality performance. It, nevertheless, completes within a few minutes on our largest real-world and synthetic datasets Tbl 2.2 and Fig. 2.2(f). It is worth to note that since it is an AOADDM method, it is amenable to parallel and distributed implementations, which can enable its feasible adoption for analysis of even larger timelines.

**Parameter selection.** We evaluate the ability of our estimation approach LARC-CCD to recover the GT number of communities \( k_{GT} \) and compare it to a regularization-oblivious approach from the literature AutoTen [49]. Since both approaches require a maximum \( k_{max} \) to probe, we set this value to \( 2k_{GT} \) for both methods in synthetic data and \( k_{max} = 20 \) for all real-world datasets. Fig. 2.2(g) shows the comparison of the two competing methods, where the quality measure on the vertical axis is the percent deviation of the estimated \( k_{EST} \) from the GT one: \( \frac{|k_{EST} - k_{GT}|}{k_{GT}} \). We report the average of 10 increasing average stretch of a synthetic dataset. LARC-CCD outperforms AutoTen consistently by at least a factor of 2 in terms deviation. The reason for this performance is the regularization-enabled compression we perform on the input tensor detailed in Alg. 2.4.3. This superior performance in estimating the number of communities is also evident in real-world datasets as reported in columns 6 and 7 of Tbl. 2.1.

We also evaluate the utility of our MDL approach for automatically selecting the regularization parameters \( \lambda_c \) and \( \lambda_d \) Fig. 2.2(h). In this experiment, we vary the two parameters in exponential steps and compare the shape of the DIV surface and that for MDL cost, employing the TF bit cost \( B^{(\lambda_c,\lambda_d)}_{TF} \). Since the global minima of both functions are attained for the same region of values of the parameters, MDL can successfully be employed as a proxy for parameter estimation in conjunction with calls to LARC. We observe similar behavior on
other datasets as well. It is worth noting that while DIV has multiple local minima, MDL is much more smooth, hence line-search approaches can be adopted to speed-up the estimation without covering the full grid of parameters.

**Overall evaluation and discussion.** A comparison of the quality and running time for all datasets is presented in Tbl. 2.2. Here we show two measures of quality: DIV and NMI, and report all running times in seconds. In terms of LS divergence (DIV), LARC-TF outperforms baselines on all datasets with ground truth except for the Football dataset on which BigCLAM and NMC perform slightly better. The reason for this behavior is the very low overlap between communities in this dataset rendering temporal interactions disadvantageous compared to a full aggregation. Similar behavior is observed in our quality with increasing overlap experiment Fig. 2.2(c) where for the minimum overlap BigCLAM similarly performs on par with LARC-TF. The pattern is similar for NMI with the exception of the Reality mining dataset, where BigCLAM has a slightly higher, though very close to 0 NMI. As we discussed earlier, the ground truth for this dataset is based on static overlapping friendship communities detected by BigCLAM, which may not align well with the multi-mode user interactions observed in the data, thus resulting in relatively low quality on all datasets.

In terms of running time, while LARC-TF is slower than alternatives, its running time exceeds 1 min only on the long \( T = 20k \) synthetic instance, making it practical to employ on large real-world datasets. An important observation here is that while the aggregate methods are typically much faster than TF and LARC-TF, their running time increases even beyond that of the temporal methods on the Reddit datasets due to the aggregate matrix they operate on being significantly denser than the individual time snapshots on which TF and LARC-TF operate.

LARC-GM’s running time increases faster with the input size (Figs. 2.2(d),2.2(e),2.2(f)) due to the more expensive gradient solver. However, its quality on small instances is promising (see first two points in Fig. 2.1(b)) due to the same temporal regularization we employ in LARC-TF. Better solvers (e.g. stochastic alternatives) for \( J_{GM} \) may render it a better fit for some datasets in practice. In addition, alternative *error-of-fit* functions for communities (i.e., beyond \( J_{TF} \) and \( J_{GM} \)) may also be advantageous within our general regularization framework. We plan to investigate the above questions in future research.
2.5.4 LARC at work

Beyond our analysis of the temporal changes in the Bike Rides dataset in Fig. 2.1(a), we also consider a sample community activity profile from the Reality mining dataset in Fig. 2.3. Weekday mid-nights are marked by grey lines, while the weekend is enclosed between red vertical lines. Outside of a Saturday morning spike, variance in activity seems lower overall during the weekend, though interpretation of this pattern may require more thorough community information. The diversity of potential options for ground truth and interaction types in the Reality Mining data (friendships, work, and external contacts; proximity and calls) makes assigning definitive communities and behaviors difficult, which may explain relatively higher DIV values. While these patterns are more or less expected, the activity profiles may be employed to detect abnormal activity movements, thus enabling anomaly detection at the community level. We detected such a big change in the bike service data in Fig. 2.1(a) which coincided with the geographical expansion of the service. The activity profiles can also be used to inform appropriate temporal aggregations of network data which can then be employed for other tasks: e.g. temporal link prediction, partitioning and others.

While we do not have an exact GT community structure for the Botnet data, the metadata specifies the set of bot IPs and that of the victim which is flooded by packets several times during the trace in coordinated DDoS attacks. We employ LARC-TF on this dataset, setting $k = 3$, (based on the recommendation from our $k$ estimation approach LARC-CCD) and examine the resulting communities. The entire botnet and the victim are consistently included in one of the reported communities. While in terms of total number of network flows, the botnet traffic does not stand out in this trace, the coordinated timings of the attack allow LARC-TF to group participants in the attack, demonstrating its potential as a network traffic analysis tool for security professionals.
2.6 Conclusion

We proposed LARC, a novel dynamic overlapping community detection framework to learn jointly the community structure and the temporal community activity profile. It enforces interpretable piece-wise constant activity profiles via a temporal smoothness regularization. We demonstrated that our framework can successfully accommodate different measures of community fit. Our proposed algorithms, by virtue of effectively leveraging the temporal aspect of the data, demonstrate $2.6\times$ quality improvement over state-of-the-art baselines on data with ground truth communities. We demonstrate the importance of regularizing time when dealing with dynamic networks, and suggest that similar or alternative regularizations can be implemented on top of other community detection methods. Furthermore, LARC produced interpretable and intuitive results when applied “in the wild”, to a variety of real-world scenarios (botnet attacks, change points in urban transportation patterns, and public forum interaction trends), demonstrating its wide applicability and practicality as a data mining tool. In addition to our optimization techniques, we provided a comprehensive set of tools for choosing estimating all method parameters in an unsupervised manner, rendering LARC useful for researchers and practitioners alike.
3.1 Introduction

Data from many domains can be represented as an evolving network where edges correspond to interactions between actors and/or items. Examples abound: message exchanges in social networks, reviews of products on Amazon or businesses on Yelp, and packet exchange events among internet hosts. Frequently, network interactions are generated from an underlying group structure: related individuals, groups of related products or interests, or botnet IPs. Understanding such groups and their temporal behaviors yields key insights about network structure and aids applications such as recommender systems, forecasting, or anomaly detection.

In all but the simplest static graphs, event timing yields critical information about network structure and evolution [61, 62]. Intuitively, nodes within a group will have coherent in-group connectivity as well as temporal behavior. In particular, interactions frequently behave in a self-exciting manner, driving further in-group activity [63, 64]. Comments on a post on a forum typically elicit comments in reply, or product reviews might invite further discussion; such “bursty” behavior has been described in settings from social to biological [65, 66].

Importantly, activity of any nodes in the group serves to drive overall group-level, as opposed to individual, self-excitation, so tracking the temporal activity of the group over time is crucial to identifying its members.

There are several key challenges in bursty group detection. First, explicitly modeling self-exciting behavior among all observed interactions through widely-adopted point process models such as Hawkes processes[67] is prohibitive for large datasets with millions of interactions [68]. Alternatively, aggregating the data as tensors or evolving snapshots [69] can offer scalability at the expense of losing inter-temporal influence. A second challenge is that groups may overlap and exhibit varying degrees of burstiness, with some spiking faster
Figure 3.1: Six nodes are members of two overlapping groups, with temporal activities displayed in the central panel. The overall, dual-thrust approach of MYRON involves simultaneously fitting tensor factorization-based group structure and enforcing bursty temporal behavior on the temporal mode of said groups or higher [70]. Additionally, group-level bursty activity may be “buried” in background or other-group activity. Finally, groups present in a system may exhibit variable temporal activity, even effectively disappearing or reappearing in the data - an effective method must be able to track such a discontinuous group as a consistent entity as opposed to clusters of nodes in individual bursts.

Figure 3.1 presents an example of our setting and an overview of our approach. We seek to identify group membership of nodes alongside overall group temporal behavior from unlabeled, attribute-free weighted dynamic graphs representing node interactions over time. We propose a framework, called MYRON, which detects groups in unlabeled interaction datasets within a regularized tensor factorization framework enforcing group-level self-exciting structure on the temporal factors. Our discrete time approach allows for scalability to large datasets while enforcing long-range temporal influence among snapshots. Simultaneously, MYRON allows flexibility in modeling the bursty structure of groups: we demonstrate two alternatives based on wavelets and point processes. We demonstrate via experimentation in synthetic and real-world datasets that MYRON is advantageous compared to state-of-the-art techniques for both group detection and temporal burst detection.

Our contributions in this chapter are as follows:

- We introduce a general and robust framework for bursty group detection from interaction data.
- We propose two models for imposing burstiness via: i) an interpretable Poisson model, and ii) a light-weight relaxation based on wavelet decomposition.
• We perform exhaustive evaluation of our framework on synthetic and real-world datasets demonstrating its significant advantage in group and temporal burst detection, boosting accuracy in ground truth group detection over baselines by up to 30% in synthetic and 35% in real-world data.

3.2 Related Works

Community detection: Community detection is a fundamental problem in network analysis. While community detection in static networks has enjoyed sustained research interest [71, 72], community detection in dynamic networks is a newer field; for a recent review, see [3]. A variety of approaches consider evolving communities of (smoothly) varying membership [7, 61]. This group of approaches focuses on tracking communities’ memberships as opposed to their varying activity. Other methods, similar to our setting, focus on the evolving behavior of static-membership communities [62, 73, 4]. Similar to us, many of them leverage tensors to represent dynamic interaction data [39, 5, 36]. Others model the internal community structure as deviating from the global network behavior [74]. Global temporal network analysis for community detection has also received increased attention recently including methods employing local profiles [75] and uncovering anomalous temporal communities [36]. Recent works have demonstrated the importance of imposing constraints on community temporal behavior through Fused-LASSO regularization for temporal factors [62], wavelet-based treatment of the temporal mode in a Tucker decomposition [76], or periodic behavior [73]. In contrast to all the above, we explicitly model bursty self-exciting behavior and the influence of past in-group interactions on the future, resulting in superior group detection as we demonstrate empirically. In particular, bursty behavior results from long range dependencies that short-history models cannot capture.

Bursty behavior: Bursty processes in data streams or point processes have been modeled as Markovian [77] and through long-memory self-exciting point processes such as the Hawkes process [67]. Hawkes processes have been employed to uncover the underlying network structure among entities [78], to group event sequences [79], and for large-scale inference [80]. This line of work, however, typically suffers from limited scalability due to long-range inter-event relationships and the computational cost of maximum-likelihood methods typically employed. Work by Linderman and Adams [68] seeks to alleviate the limited scalability by
temporal aggregation within a Bayesian framework, though without grouping in the node domain. There is also recent research on continuous-time-aware tensor decomposition\cite{81, 69} which combines ideas from Hawkes point processes and tensor factorization methods, however the approaches in this group suffer from limited scalability to large-scale networks frequently encountered in the online domain.

### 3.3 Preliminaries and notation

We introduce key notation and concepts used throughout the paper in Table 3.1. The input data in our problem setting is a set of temporal interactions of the form \((i, j, t)\), where \(i \in V_1, |V_1| = N_1\) and \(j \in V_2, |V_2| = N_2\) are the interacting entities or nodes and \(t \in [1, T]\) is the discrete time of the interaction occurrence. Note that when analysing user-user interactions (i.e., \(V_1 = V_2\)) the problem becomes one of community detection, while in the general case the two sets could represent different entities, e.g., people and photos. We organize all interactions in a three-way tensor \(X \in \mathbb{R}_{\geq 0}^{N_1 \times N_2 \times T}\) with entities corresponding to the number of interactions, i.e., we allow for multiplicity of pairwise interactions within the same time point.

The canonical polyadic decomposition (CPD) \cite{82} is a tensor extension of SVD and factorizes a tensor \(X\) as a sum of \(K\) rank-one tensors \([C1, C2, A]\), where \(Ci \in \mathbb{R}_{\geq 0}^{N_1 \times K}\)
are factor matrices for the first two modes which can be interpreted as groupings of the corresponding entities, while $A \in \mathbb{R}_{\geq 0}^{T \times K}$ represents the activity profiles (temporal factors) corresponding to each of the $K$ groups. $C1$ and $C2$ are in general different (e.g. with directed or bipartite graphs), however, in the symmetric case they convey the same information.

### 3.4 Problem formulation

Our model is based on the following (informal) generative process for temporal interactions:

1. **Interactions arise within overlapping groups of entities whose strength of group association may vary**;
2. **Additional “non-group” background interactions of a fixed intensity arise between random pairs**;
3. **The intensities of within-group interactions exhibit a bursty structure over time**.

Our goal is to recover the bursty group structure corresponding to the above behavior from temporal interaction data, which is formalized as a constrained tensor factorization as follows:

---

**Bursty Group Detection.**

Given a tensor $X \in \mathbb{R}_{\geq 0}^{N1 \times N2 \times T}$ of weighted temporal interactions, group number $K$, and temporal fit parameter $\lambda$, find factor matrices $C_i \in \mathbb{R}_{\geq 0}^{Ni \times K}, i \in 1, 2$ representing node group membership and $A \in \mathbb{R}_{\geq 0}^{T \times K}$ representing corresponding group temporal behaviors based on:

$$
\min_{C_1,C_2,A,b_X,S} ||X - [[C1,C2,A]] - b_X 1||_F^2 + \lambda ||A - S||_F^2
$$

s.t. $C1,C2,A,b_X \geq 0$,

where $S$ is the bursty shape approximation matrix, and $b_x$ models the global average interaction level.

---

The first term is a tensor factorization loss augmented by a background interaction level controlled by $b_X$, while the second is a penalty enforcing alignment of the temporal factor with a bursty activity learned in the shape matrix $S$. Scalar $\lambda$ balances the fit and burstiness.
Our solution framework, called MYRON, is illustrated in Fig.3.1 and consists of two mutually-reinforcing components: *Burst modeling* and *Node grouping with background activity*. MYRON addresses all challenges in bursty community detection: limited scalability of continuous time methods, overlapping group structure with varied behaviors, and background noise.

### 3.4.1 Enforcing bursty structure: designing $S$.

One of our key design principles is that within-group activity over time is self-exciting and forms bursts of interactions, i.e. in-group interactions boost successive in-group interactions. In addition, these activity profiles are specific to individual groups. Within our objective we enforce this through a reference shape $S$ which we model as a group-specific bursty time series. A regularization-based approach for this objective allows for a controlled level of variation in valid profiles and penalizes non-bursty temporal fits for groups. We consider two models for this reference bursty shape: (i) a non-homogeneous Poisson process as an “aggregate” extension of a Hawkes process, and (ii) a time series with sparse wavelet reconstruction via the *Daubechies basis* which resembles bursts in time. The former is rooted in a long history of self-exciting process models; the latter serves as an efficient high-quality alternative.

**Non-homogeneous Poisson (NHP) burstiness:** Self-exciting point processes are a common, if computationally challenging, way to model burstiness of events. We adapt the exponential kernel Hawkes process formulation[67] for even intensity at time $t$, $\alpha_t = \alpha_0 + \sum_{\tau < t} \Phi(t - \tau)$, to introduce a *Non-homogeneous Poisson process (NHP)* for our aggregate group-level activity. We define the same intensity, or number of expected in-group interactions in interval $[t, t+1)$, $\alpha_t$, as $\alpha_t = \bar{\alpha} + (\alpha_{t-1} - \bar{\alpha}) \ast \beta(\Delta_{t-1}) + \gamma \hat{\alpha}_{t-1}$. Here $\bar{\alpha}$ is a persistent baseline intensity describing the arrival of within-group non-burst events, $\beta$ is a decay kernel, $\gamma$ is additional intensity from events in the previous period, and $\hat{\alpha}_t = \frac{N_t}{\Delta_t}$ is the empirical rate of events in that period. This per-interval intensity is described by a baseline, decayed intensity from the prior period, and additional intensity from recent events.

Assuming a constant time interval length (as in [68]), i.e. $\exists c : \forall t, \Delta_t = c$, yields a
constant decay kernel $\beta$. We can recursively express the intensity as:

$$\alpha_t = \bar{\alpha}(1 - \beta^t) + \beta^t \alpha_0 + \gamma \sum_{i=1}^t \beta^{i-1} \hat{\alpha}_{t-i},$$  \hspace{1cm} (3.1)

where $\alpha_0$ is the initial intensity. Note that since $\Delta_t$ is absent from the above equation, we can rescale time so that $\Delta_t = 1$ without loss of generality and think in units of "time windows". Our NHP model is an aggregate version of the exponential kernel Hawkes process and the two are asymptotically equivalent. Specifically one can show the following relationship:

**Theorem 3.1** Let $\beta(\Delta_t) = \beta \Delta_t$, then the model from Eq. 3.1 converges to a Hawkes process with exponential kernel as $\Delta_t \to 0$ and $t \to \infty$.

**Proof**: As the window size decreases, we reach a point where $\alpha_t \in \{0, 1\}$, i.e. there is at most one event per period. Now, as $t \to \infty$, we approach $\bar{\alpha} + \gamma \sum_{i=1}^t \beta^{i-1} 1_{\alpha_t = 1}$, which we can rewrite as $\bar{\alpha} + \sum_{\tau \leq t} \gamma \beta^{t-\tau}$, thus arriving at the standard Hawkes formulation.

For interpretability, we require several natural restrictions on the parameters in Eq. 3.1. All parameters should be non-negative: negative intensity has no meaningful interpretation, and $\gamma < 0$ will imply self-damping as opposed to self-excitation. In addition, $0 \leq \beta < 1$, as we expect a decaying influence with time, thus avoiding intensity buildup in the absence of events.

To impose an NHP-like structure on our temporal factors in our overall objective, the shape $S$ in the regularization term $||A - S||_F^2$ is instantiated with a matrix of time series resulting from bursty NHP fits of within-group events. In other words, $S_{k} = \{\alpha_t^{(k)}\}$, where $\{\alpha_t^{(k)}\}$ is the NHP intensity fit for the $k$-th group’s temporal component. Note that in addition to enforcing the bursty shape, a solution also yields the burstiness parameters characterizing group activity.

**Daubechies wavelets**: While the NHP-based burstiness definition is interpretable and rooted in a long-standing body of work on point process modeling, its recursive nature comes with a considerable computational cost. Thus, we also propose an efficient alternative based on “fitting” factor activity with an appropriate wavelet basis decomposition. The Discrete Wavelet transform [83] models a time series as a combination of scaled and shifted wavelet filters which form a multi-resolution representational basis. Informally the wavelet transform will represent an arbitrary temporal group profile in our framework as consisting of a small
number of scaled and shifted kernel "shapes". In particular, if the kernel shape is "burst-like", then the temporal profile will be represented by a number of appropriately placed and sized bursts. By retaining only high-energy wavelet coefficients, one can reduce the noise [84] in group activity and focus only on the bursts themselves. For the purposes of our objective function, $S$ in the wavelet case is a matrix of group-specific time series reconstructed by sparse wavelet transform. The shape of the Daubechies wavelet [85] captures a "rising front" behavior similar to the NHP formulation, hence we adopt it for regularization. Alternative wavelet bases can also be employed.

### 3.4.2 Tensor model with background activity.

In our formulation we augment CPD to include fixed-form rank-one factor representing a background activity level:

$$X \approx \sum_{K} [c_{1k} \otimes c_{2k} \otimes a_k] + b_X 1,$$

where $K$ is the number of bursty groups (or factorization rank), $c_{1k}, c_{2k}, a_k$ are vectors representing single-factor loadings in the respective tensor mode. This modeling decision explicitly allows for a fraction of node interactions to arise outside the context of recurring longitudinal groups. In the case of user-user interactions, this background behavior can be due to weak ties [86]. We considered evolving (as opposed to constant) background level models, but the simpler approach was advantageous across datasets, and thus, we focus on it for our presentation. The introduction of the baseline activity tensor $b_X 1$ is akin to removing the global mean of the input $X$. Another means to this end could be to preprocess the data by subtracting the mean from the tensor element-wise. Such an approach will densify typically sparse input data resulting in significant computational and memory overhead [87].

### 3.5 Optimization solution

The optimization of our objective requires solving for three factor matrices $C1, C2, A$, the baseline level $b_X$, and the detailed parameters $\theta$ of the temporal shape, which in the NHP case contains a set of $(\alpha_0, \bar{\alpha}, \gamma, \beta)$ per group, while in the wavelet case it consists of a set of per-group vectors of wavelet coefficients. We outline the steps for solving the compound problem below.
3.5.1 Alternating optimization.

Alternating Optimization is a common technique in a variety of settings, particularly in the case of tensor factorization [88, 45]. We update individual components of the problem in a cyclic manner while holding other components fixed, cycling through updates of $C_1, C_2, A, S, b_X$. Like coordinate descent, as long as each individual update is optimal this is a non-increasing operation to the objective, as each step minimizes an objective already minimized for another component by the previous step. Since the overall objective is bounded (even crudely at zero), this monotone behavior guarantees convergence. The structure of the problem leads to related updates for the node-dimension factor matrices. In particular:

- $C_1, C_2$ are nonnegative factors updated via
  \[
  \arg\min_{C_i} ||X - [[C_1, C_2, A]] - b_X 1||^2_F \quad \text{s.t.} \quad C_{i_{nk}} \geq 0 \forall n, k.
  \]

- $A$ is updated via a ”shape-based” constraint
  \[
  \arg\min_{A} ||X - [[C_1, C_2, A]] - b_X 1||^2_F + \lambda ||A - S||^2_F.
  \]

- $S$ is updated by fitting parameters to the current $A$ and using these to reconstruct approximation $S$.

Below, we discuss solutions for each subproblem.

3.5.2 Decoupling objectives via ADMM

The Alternating Direction Method of Moments (ADMM) is commonly used [89] method for dividing a complex optimization problem into simpler components. It has also been recently applied to regularized tensor decomposition objectives [62, 45]. Intuitively, ADMM works by solving each subproblem separately while enforcing similarity between the solutions. For the temporal factor $A$, the ADMM objective is:

\[
\min_{C_1, C_2, A, A, b_X} \quad ||X - [[C_1, C_2, \hat{A}]] - b_X 1||^2_F + \lambda ||A - S||^2_F \quad \text{s.t.} \quad C_1, C_2, A, \hat{A}, b_X \geq 0, \quad A = \hat{A} \tag{3.3}
\]

The ADMM update involves iterating over the three updates: the tensor objective with similarity to the shape solution, the shape objective with similarity to the tensor solution, and the error between the two. The updates themselves are given by the following, where $\hat{C}$
denotes the Khatri-Rao product of $C_1, C_2$:

\[
\begin{align*}
\hat{A} &\leftarrow (\hat{C}^T \hat{C} + \rho I)^{-1}(\hat{C}^T X_{(3)} + \rho (A + R_A)^T) \quad (a) \\
A &\leftarrow \arg\min_A \lambda ||A - S||^2_F + \rho/2 ||R_A + A - \hat{A}^T||^2_F \quad (b) \tag{3.4} \\
R_A &\leftarrow R_A + A - \hat{A}^T \\
\end{align*}
\]

The first update can be efficiently computed via the lower Cholesky decomposition of $\hat{C}^T \hat{C} + \rho I$ and the matricized tensor times Khatri-Rao product (MTTKRP) representation of $\hat{C}^T X_{(3)}$, similar to prior approaches [62, 45]. With $S$ known, the solution to 3.4b becomes $A = (\lambda S + \frac{\rho}{2} \hat{A})/(\lambda + \frac{\rho}{2})$. Note that the non-negativity constraint for $C_1$ and $C_2$ is also efficiently enforced via an ADMM objective; in this case $\hat{C}$ is the product of the other two factors, and the update in Eq. 3.4b simply retains the positive values from $\rho/2 ||R_A + Ci - \hat{C}^iT||^2_F$.

3.5.3 Optimization for the shape subproblems.

The solution to the shape subproblem depends on the specific constraint considered. In each case, however, the idea is similar - we fit the temporal profiles $A$ to a specific model of temporal activity, and use the resultant idealized approximation as our shape matrix $S$. When $A$ is known, solving for $S$ in either the NHP or wavelet formulation requires first fitting a set of parameters describing the burstiness of proposed traces.

**NHP:** In the NHP case, the problem consists of optimizing over the four parameters $\alpha_0, \bar{\alpha}, \beta, \gamma$. This can also be done independently for each group $k = 1...K$. We iterate through updates given by:

\[
\begin{align*}
\bar{\alpha}^* &= \frac{N - \sum_i \beta^i (1 - \beta^{i-1})\alpha_0 \sum_j \xi_j (1 - \beta^j)}{\sum_i (1 - \beta^i)^2} \\
\alpha_0^* &= \frac{(1 - \beta^2)(\sum_i \beta^i (1 - \beta^{i-1}) - \bar{\alpha} \sum_i \beta^i (1 - \beta^{i-1}) - \gamma \sum_i \beta^i \xi_i)}{\sum_i \beta^i \xi_i} \\
\gamma^* &= \sum_i \beta^i \xi_i (1 - \bar{\alpha}) \sum_i \xi_i (1 - \beta^i) - \alpha_0 \sum_i \beta^i \xi_i \\
\beta^* &= \arg\min_{\beta} \sum_i |A_t - (\bar{\alpha}(1 - \beta^i) + \beta^i \alpha_0 + \gamma \sum_{t'=1}^{i-1} \beta^{i-1} A_{t-t'})|^2, \\
\end{align*}
\]

where $\xi_i = \sum_{t=1}^i \beta^{i-1} A_{t-i}$ and $N = \sum_{t=0}^T A_t$. All but the $\beta^*$ update can be computed analytically from first derivatives. Per-iteration complexity for this method is thereby driven by said update and the contained computation of objective $(A_t - \alpha_t)$, the latter of which can be done in $O(T)$. Precomputing $\xi$ once per full iteration is done via single pass through the time series. Overall per-update complexity is $O(KT)$, which can be reduced by approximating
Figure 3.2: Mallat’s algorithm recursively constructs a coefficient vector using a wavelet form and temporal profile A. We retain the top values and use them to reconstruct the denoised bursty approximation S.

the objective - for instance, truncating the number of past periods considered for influence from \( A_{t-i} \) directly decreases the effective \( T \) when computing \( \beta^* \) while minimally affecting the overall value if \( \beta \) is small.

**Wavelets:** Given fixed A, we compute the complete wavelet decomposition via Mallat’s pyramid algorithm [90]. This efficient algorithm consists of repeatedly applying approximation and detail filters specific to the wavelet form. Filters are then reapplied recursively to the product of the approximation coefficient, yielding progressively higher level detail coefficients that effectively represent scaled and shifted wavelet shapes, eventually perfectly reconstructing an arbitrary temporal profile through a vector of coefficients.

The procedure of imposing a wavelet structure on the temporal tensor modes is schematically illustrated in Fig. 3.2. A top fraction of coefficients are retained, which is a tunable parameter. This thresholding has been theoretically demonstrated to be optimal in terms of minimizing mean squared error (MSE) for the sparse wavelet reconstruction [91]. A fraction of 2% yielded good performance across datasets in our evaluation. To obtain the shape matrix \( S \), we reconstruct each time series from the sparsified coefficient vector by running the reverse transform.

### 3.5.4 The overall algorithm: MYRON.

We combine the above updates in the steps of MYRON outlined in Alg. 3. The outer loop of MYRON (Alg. 3) iterates over each of the three factor matrices with the remainder of the parameters fixed (lines 3-5) following the ADMM process described in Section 3.5.2.
Algorithm 3 MYRON

Require: Tensor $X$, group count $K$
Ensure: Factorization $\{C_1, C_2, A\}$, scaling $s$

1: Initialize $C_1, C_2, A$
2: while Factors not converged do
3:   $C_1, s \leftarrow $ FACTOR$(X(1), b_X, C_2 \odot A, s, K, \text{nonnegative})$
4:   $C_2, s \leftarrow $ FACTOR$(X(2), b_X, C_1 \odot A, s, K, \text{nonnegative})$
5:   $A, s \leftarrow $ FACTOR$(X(3), b_X, C_2 \odot C_1, s, K, \text{shape})$
6:   $S, \text{params} \leftarrow \text{ShapeApprox}(A)$
7:   $b_X \leftarrow \frac{\sum_{ijt} X_{ijt} - \sum_k (\sum_i C_{1ik}) (\sum_j C_{1jk}) (\sum_t A_{tk})}{N_1 \times N_2 \times T}$
8: end while

In line 6, ShapeApprox() refers to the shape-specific fitting described in Section 3.5.3. For the NHP fit, ShapeApprox() consists of running the iterative optimization of the four NHP parameters, and reconstructing $S$ based on said parameters and the current $A$. In the wavelet case, we perform wavelet decomposition, sparsify, and reconstruct the sparse wavelet representation $S$. The mean of the residual tensor is computed in Step 7, as the difference of the sum of all entries of $X$ and the sum of the factor-based reconstructed tensor, which can be computed via the sum-product in Step 7. In addition to the steps described in the prior sections, we make two observations that aid computation and convergence: Fitting Sparse Global Behavior: Although using the mean of the overall adjacency tensor as global behavior is a simple pre-processing step on paper, simply subtracting the mean of the overall $X$ would yield a large dense tensor which will be inefficient to decompose. Instead, we represent the aforementioned mean-scaled constant Kruskal tensor in its decomposed form (i.e. three vectors of ones and the single scalar $b_X$) and utilize the associativity of the MTTKRP to compute $\tilde{W}^T(X(\ast) - b_X \mathbb{1}_{(\ast)})$ as follows:

\[
\text{MTTKRP}(\tilde{W}^T, X_{(\ast)}) - \text{MTTKRP}(\tilde{W}^T, b_X \mathbb{1}_{(\ast)}).
\]

Enforcing Scale: If we simply maintain the approach described in the preceding sections, we may run into scaling problems between the individual factors. Namely, if the total energy in the activity of some factor is greater than that of others, the Frobenious norm shape regularizer will favor smooth fitting of the $k$-th factor while essentially ignoring all others, thus yielding poor fits for less active communities or potentially splitting a larger one and missing others entirely. This problem can be exacerbated by the scaling freedom of CPD. In particular, different initialization of $C_1, C_2, A$ can lead to varying amounts of energy assigned to the temporal mode of any given group. We can solve this problem
Algorithm 4 FACTOR

Require: Tensor $X$, mean $b_X$, product of known factors $W$, scale $s$, group count $K$, constraint type $cons$
Ensure: Fitted factor $H$, scale $s$

1: Initialize $H, R$
2: $\hat{W} \leftarrow s \odot W$
3: $G \leftarrow \hat{W}^T \hat{W}$
4: $\rho \leftarrow \text{trace}(G)/k$
5: $L \leftarrow \text{LowerCholesky}(G + \rho I)$
6: $F = W^T (X - b_X \mathbb{1})$
7: while Not Converged do
8: $\hat{H} \leftarrow (L^T)^{-1}L^{-1}[F + \rho(H + U)^T]$
9: if $cons == \text{nonnegative}$ then
10: $H \leftarrow \max(0, \hat{H}^T - U)$
11: else if $cons == \text{shape}$ then
12: $H \leftarrow \left[\lambda S + \frac{\rho}{2} (\hat{H}^T - U)\right] / (\lambda + \frac{\rho}{2})$
13: end if
14: $R \leftarrow R + H - \hat{H}^T$
15: end while
16: $H \leftarrow H./\text{colmax}(H)$
17: $s \leftarrow s * \text{colmax}(H)$

by normalizing each factor as we fit it, holding each column of $C1, C2, A$ at most 1 and maintaining the group “strength” in an overall scaling vector $s$. Prior work has shown that conducting similar re-normalization at every step of a factor-by-factor CPD yields beneficial convergence properties[92].

We measure overall convergence via changes in the normalized reconstruction error. We use a convergence threshold of $10^{-7}$ unless stated otherwise. Relaxing this threshold or bounding the number of iterations yields significant speed-up at minimal quality cost.

Complexity. The steps of the main optimization procedure are summarized in Alg. 3. We repeat the updates from Step 2 - Step 7 until convergence. The dominant part of the complexity comes from FACTOR calls (Alg. 4), therefore, we focus on the complexity of this procedure. In Algo. 4, the major complexity is from step 5, 6 and the while-loop as other steps involve linear processes or sparse matrix multiplications. Step 5 uses Lower Cholesky decomposition which has complexity $O(K^3)$. In step 6, we apply the MTTKRP operation [62] which exploits sparsity in the matrix when available and has a complexity of $O(mK)$, where $m$ is the number of nonzero elements in $F$. In the while-loop, step 8 consumes the most running time, which involves an inversion of a quadratic matrix with a complexity of $O(K^3)$ in general. When $L$ is sparse, this reduces to $O(mS_L)$, where $S_L$ is the number of non-zero elements in $L$. The overall complexity is $O(K^3 + mK + t_1 mS_L)$, where
$t_1$ is the number of iterations of the while-loop. This occurs thrice in the outer loop, along
with a single iteration of ShapeApprox().

| Dataset          | Statistics $|V|$ | $T$ | $K$ | DIV | Time (s) | DIV | Time (s) | DIV | Time (s) | DIV | Time (s) |
|------------------|-----------------|-----|-----|-----|-------|-----|-------|-----|-------|-----|-------|
| Reddit[55]       | 35196×35196     | 2880| 5   | 0.5208| 148.9| 0.5919| 171.3| 0.8914| 149.7| 0.9431| 8.0   |
| Delicious[53]    | 10k×10k         | 1430| 10  | 0.4803| 307   | 0.6644| 72.7 | 0.7630| 288.3| 0.8191| 19.5  |
| Flickr[93]       | 3478×100k       | 705 | 6   | 0.5847| 98.6  | 0.7552| 43.7 | 0.9453| 68.7 | 0.9412| 1.4   |
| Github Repos     | 8294×8294       | 2183| 5   | 0.8638| 45.5  | 0.8943| 31.5 | 0.9006| 146.0| 0.9108| 7.4   |
| Github Topics    | 9020×137819     | 2184| 5   | 0.584 | 188.5 | 0.6081| 57.9 | 0.5982| 99.5 | 0.6432| 22.6  |

Table 3.2: Real Dataset statistics and results. (* Denotes the dimension of the ground truth mode)

3.5.5 Datasets.

**Synthetic Data:** To demonstrate performance of our method on data with known behavior, we generate a set of small synthetic datasets representing communities within an unweighted dynamic graph. We create two overlapping 30-node internally interacting groups, which allows for controlled variation in experimental parameters, along with overlaying a background activity level. Edges are generated from two groups of thirty nodes, with 10 nodes participating in both groups, according to group-level activity traces. These traces are generated from an NHP with known parameters ($\bar{\alpha} = 0.1, \alpha_0 = 0, \gamma = 0.5, \beta = 0.45$), with one group being offset by 10 timepoints from the other to retain a difficult separability, and contain three small bursts (see Fig. 3.4(e)). Edges are produced by randomly drawing the number of edges matching the activity of the group from the pairs of nodes within it. We overlay global behavior by producing a third group containing all 100 nodes with non-bursty behavior, where the number of edges in a given time period is drawn from a Poisson distribution with a constant or varying intensity depending on the experiment.

**Real-world Datasets:** We also evaluate our methods on several real-world datasets presented in Table 3.2.

**Reddit:** This dataset is obtained from a full dump of Reddit comments collected in 2015 [55]. We combine multiple subreddits (each a ground truth group), with responses to comments or a top-level post recorded as bi-directional edges between the posting users. The final dataset contains "programming", "gaming", "politics", "geek", and "lost" for the first four months of 2009. Data is aggregated hourly.

**Flickr:** This is a benchmark image captioning dataset for sentence-based image de-
Figure 3.3: Performance of MYRON with varying activity volume, burstiness, and global behavior.

The dataset used in this chapter is obtained by selecting the six most frequent tags from [93]. Tensor modes represent *user-image-date*, where date is at daily granularity.

**Delicious:**[93] This dataset was obtained by crawling Del.icio.us portals during 2006 and 2007. We use a set of frequently-used tags as groups, with the most-tagged pages and most active users within those tags as nodes (weighted by participation frequency).

**Github:** We present two Github-derived datasets using hourly data from the first three months of 2016. Github Repos consists of users interacting with a set of six repos (those ranked 100-105 by activity within those three months, to avoid bot-heavy and single-user repos), with a ground truth of activity-weighted user-repo interaction. Github Topics consists of users who interact with repos tagged with five language-based keywords java, javascript, python, go, and C. We use repo events within these topics as our ground truth.

### 3.5.6 Experimental setup.

To demonstrate the benefits of incorporating proper temporal dependence and removal of the global trend, we compare two versions of our method, MYRON-NHP and MYRON-Wav, against two alternative methods. The first is direct non-negative tensor factorization.
Figure 3.4: Scalability on different dimensions (a-c) and convergence behavior (d) of all methods. Panel (e) compares detection of three pronounced bursts (outlined), which MYRON-NHP fits in period and “shape” via CPD (TF), which imposes no additional structure on the network. The second baseline, LARC [62], does not consider global behavior explicitly and assumes a piecewise-constant temporal behavior on groups.

For all experiments we use group number $K$ equal to the ground truth number of groups, which is obtained through the construction of the datasets. In general, $K$ is an unknown parameter, but as our metrics are based on alignment with ground truth it is more difficult to measure the quality of a fit with a different number of communities. In fact, effectively estimating $K$ is a difficult problem and an active area of research. [49, 94]. Results are presented as averages of five runs, with initializations identical across methods for each run.

The primary comparison metric presented is Jensen-Shannon divergence (DIV), which measures the difference between two distributions. We treat the weights on the node factors as a distribution over nodes. Lower DIV values correspond to a better agreement with ground truth distributions of node weights within a community. Clusters are matched to ground truth in a greedy best-match fashion.

### 3.5.7 Detection of ground truth groups.

MYRON-NHP and MYRON-Wav effectively detect ground truth groups with a variety of behaviors. To determine sensitivity to variation in group and global activity levels, figures 3.3(a) and 3.3(b) vary the total energy (i.e. number of edges) within the group or within the global behavior respectively, while keeping the other fixed. In fig 3.3(a), low-activity groups lead to poor performance for all methods, however MYRON gains accuracy much more rapidly with increasing burst energy, even as early as only 20% of global energy within each group. Conversely, increasing global activity effectively serves as noise, masking in-
group relationship between nodes. Low noise presents little trouble for any method save TF, but LARC quickly begins to add noise to the groups; MYRON, on the other hand, is all but insensitive to significant degrees of non-group activity.

Figure 3.3(c) shows the importance of modeling bursty behavior in particular by varying the $\gamma$ parameter in the generated NHP groups. Low values of $\gamma$ yield mostly static traces that rise neither far nor frequently, whereas larger values yield strong, more frequent bursts; total group activity is normalized to be equal across all settings. We observe that all methods perform poorly at low burstiness but MYRON quickly improves. High burstiness groups are more pronounced and easier to detect for all methods.

Figure 3.3(d) demonstrates that despite relying on mean centering, MYRON retains performance even with non-constant global behavior. The global group behavior is defined as $pB1 + (1-p)B2$, where $B1$ is a linearly increasing trace representing a common setting of overall network growth over time whereas $B2$ is a constant intensity global behavior. None of the methods are noticeably affected by variations in the global behavior, but the relatively high volume of this behavior (equivalent to 0.75 in figure 3.3(b)) obscures the groups to LARC and TF.

Results from real-world datasets are presented in Table 3.2. Overall, MYRON-NHP outperforms competitors despite occasionally being marginally slower. MYRON-Wav also does well to varying degrees likely dependent on how well the wavelet basis describes the underlying data structure. However, MYRON-Wav is notably faster on most datasets. We present performance on bipartite graphs as well, in particular through the Delicious, Flickr, and Github Topics datasets, demonstrating that MYRON is not confined to community detection alone.

3.5.8 Scalability.

Figure 3.4 investigates the scaling of competing techniques with respect to the number of entities (Fig. 3.4(a)), $T$ (Fig. 3.4(b)) and the fraction of non-zero entries (Fig. 3.4(c)). Figure 3.4(d) provides more detail on the convergence properties of each method, for a $10,000 \times 100 \times 750$ tensor. Note that DIV is not the convergence measure in any method. Both versions of MYRON yield good performance almost immediately, and in fact converge to near-optimality in DIV only marginally slower than non-negative TF, reinforcing that
early stoppage (or looser convergence bounds) still yields good, fast results - we present a slightly loosened bound compared to LARC and TF in other figures.

Fig. 3.4(a) shows qualitatively similar polynomial growth across all methods with larger $N_1$, corresponding to similarity in the fitting process for node-mode factors - slowdown is largely due to increased convergence complexity. MYRON performs noticeably faster than LARC with only a slight relaxation in tolerance. Increasing $T$ again shows similar behavior with method-specific overhead, though the benefit of MYRON-Wav is clearer: NHP is more complex with time (which also accounts for the initial gap at small $N_1$ and $T$. Running times are minimally affected by tensor density, at least in the regime tested. Real interaction data is likely to be very sparse - constant communication within large cliques is a different setting altogether.

3.5.9 Burst detection.

In Fig. 3.4(e), we present the estimated bursts for competing methods in synthetic data with a ground truth bursty profile (top). MYRON-NHP closely reconstructs the bursty profile of the ground truth behavior. Since LARC aims to obtain a piece-wise on/off behavior without too many switches it “breaks down” bursts into short spikes and thus fails to detect the presence of longitudinal bursts. TF doesn’t impose any shape regularization on the temporal information but simply minimizes the reconstruction error. While it manages to partially recover bursts, it detects too much non-burst activity leading to noisy detected groups and higher DIV for group detection.

3.5.10 Estimating K

3.5.11 Case study: analysis of a Flickr dataset.

The Flickr dataset was crawled by Gorlitz et al. [95] and spans the period from January 2004 to December 2005. From the raw data, which consists of (User, Image, Tag, Time) tuples, we extract (User, Image, Time) tuples corresponding to six frequently used tags: Party, Family, Roma, Japan, and Friends; the tags are then used as our ground truth groups.

The experimental analysis of our results in Flickr is shown in Fig 3.5, where the x-axis is the time (day), and the y-axis is the factor value corresponding to each reconstructed
The bursts of the curves can be explained by different tagging intensity within a particular group. For example, users in group 4 (Japan), do not have much activity during the data collection period (2004-2005) but between September 5-20, 2005, following a general election in Japan where the Liberal Democratic Party with its coalition partner New Komeito won the majority of the House of Representatives and at same time Seiji Maehara defeated Naoto Kan for the presidency of the Democratic Party of Japan. MYRON-Wav elucidates a temporal pattern corresponding to this political activity as shown in Figure 3.5.

Another interesting tag/group, Roma (group 3), is active throughout the year. The images tagged by this group are mostly historic places like La Fontana dei Quattro Fiumi (Fountain of Four Rivers), Piazza Navona, Tiber and St. Peter church etc. As we do not have any demographic information of users, we mapped other groups i.e. Party, Family and Friends, to USA events; results show bursts that are consistent with holiday seasons. We do not observe any sudden bursts in group 5 (Travel), but this tag is active during holidays like Christmas, New year etc., and may simply have a higher overall level of activity that is less distinctly captured.

3.6 Discussion

**Raw interaction detection:** Our experiments demonstrate that MYRONis able to detect ground truth communities containing bursts, solely using interaction data, more accurately than close baselines which do not explicitly account for bursty activity. Hence our methodology is especially applicable and necessary for social media and online forum discussion datasets whose activity is inherently self-exciting. The community burstiness that we leverage can be in the form of particular events to discuss, release-centered activity in a github repository within a topic ”group”, or topic-level events or salient moments as in the case
study above.

**Burst modeling:** The distinction between MYRON-NHP and MYRON-Wav deserves special mention. Modeling bursty behavior via any means is beneficial, as both forms of MYRON outperform baselines. However, in most of our datasets, the NHP version performs better quality-wise, bolstered by a more general modeling of bursty processes. This quality comes at a higher computational cost, particularly for longer time scales. In addition to group detection, NHP also provides a more general, faithful, and interpretable description of the “type” of burstiness of a given group in terms of the NHP process parameters. These parameters have physical equivalents in standard Hawkes parameters and can be employed to simulate more group interaction with “faithful” temporal properties.

On the other hand, wavelets essentially pinpoint burst locations without describing the entire temporal behavior. The Wav formulation also offers trivial migration to other wavelet forms for temporal group behavior which may fit particular data better. Many standard wavelets are less bursty in shape, but the MYRON framework is able to accommodate these regardless.

### 3.7 Conclusion

In this chapter we introduced a general and robust framework for bursty group detection from interaction data based on tensor factorization, called MYRON. We incorporated burstiness via two alternative models: i) an interpretable non-homogeneous Poisson model which generalizes classic Hawkes process models for individual events and and ii) a lightweight alternative employing Daubechies wavelet decomposition. We performed extensive evaluation of our framework on synthetic and real-world datasets spanning different types of online interaction data. Our evaluation demonstrated the advantage of MYRON for group and temporal burst detection in comparison to recent state-of-the-art baselines. Our methodology enabled improvement of quality for group detection of up to 30% in synthetic and 40% in real data. In addition, MYRON was able to detect interpretable bursty behavior, which we linked to real-world events, when employed to mine the user-photo interactions in the Flickr dataset.
3.8 Extending to other behaviors

In this chapter we have so far examined self-exciting behavior as a reasonable model for a variety of networks. Other behaviors are also reasonable, however. The framework developed herein, consisting of a goodness-of-fit metric via tensor factorization reconstruction error and a set of shape regularization terms is easily applicable to a variety of temporal behaviors. Periodicity, for example, is common in settings ranging from daily work patterns to seasonal animal migration. We can fit periodic communities within this framework [73] by utilizing dictionary learning, fitting “periodic” shapes to the temporal aspect alongside adding additional regularization for the fitted temporal factors.

More specifically, said work fits the objective:

\[
\arg\min_{C,A,Y,O} \frac{1}{2} \|W - [U,A]\|_F^2 + \lambda_0 \|A - \Phi Y - O\|_F^2 \\
+ \lambda_1 \|HY\|_1 + \lambda_2 \|O\|_1,
\]

where \(\Phi\) is a matrix encoding prototypical periodic behaviors and \(H\) assigns increasing cost to larger periods (by more highly weighting later columns in \(\Phi\)). In this formulation we specifically target outlier behavior, \(O\), which deviates from the overall periodic trends in the communities. The primary modeling of temporal behavior, in this case, is through the matrix \(\Phi\), which is composed of columns representing various representations of different period lengths. Any community’s temporal activity, can be represented via combinations of these periodicities, alongside any outlying time points captured within \(O\).

More generally, combining various templates for temporal behavior as regularizers on the time factor with the framework in the previous two chapters can yield numerous models that are tailorable to the data of interest. Further regularization of the factors aids in interpretability and accurate reconstruction of “true” behaviors.
4.1 Introduction

Modern smartphones have an extensive array of capabilities which typically rely on a client-server communication paradigm and continuous access to the Internet. There are, however, various non-centralized application settings including communications in rural infrastructure-challenged areas [96], coordination during disaster response upon disruption of the default infrastructure [97], information dissemination and coordination for displaced populations [98] and social movements [99]. The above have led to proposals of various wireless ad hoc networks (WANETs) and corresponding routing protocols that enable end-to-end delay-tolerant connectivity [100]. The focus in the majority of the research on WANETs is on the network layer, since routing in a constantly evolving ad hoc topology poses significant challenges. Ensuring connectivity at the data link layer in WANETs, however, poses an equally important challenge, especially for modern energy-hungry smartphone devices which apart from communication are utilized in multiple other power-hungry applications such as navigation, photography, gaming and augmented reality.

Ad hoc connectivity at the data link layer among personal smartphone devices (peers) hinges on adaptive modeling of human mobility in the context of constrained power resources. Specifically, devices have to be equipped with predictive decision making for when and where to attempt to connect to peers, since continuous searching for peers is infeasible in terms of battery drain [101]. Past research employing mobile phone traces has demonstrated that human mobility, although variable, is largely predictable [102, 103], thus, opening the door for model-driven data link connectivity. The mobility patterns in rural areas, during social unrest or disaster response and other application scenarios for WANETs, may not be fully aligned to stationary urban mobility typically considered in the above studies. For example, recent analysis of human mobility during natural disasters showed deviations in mobility from that during non-disaster periods [104] limiting the predictive power of models assuming stationary mobility. Hence the main research question we focus on in this chapter is: How
Figure 4.1: An example summary of daily trajectories (thicker lines represent more traveled routes) for three people in a rural area where cellular or broadband connectivity is lacking or intermittent. Person 1 (red) and 2 (blue) travel to the same office and back to their homes and occasionally make stops to a cafe, a store and the post office. Person 3 (green) travels to a farm the store, cafe and the post office. The “typical” trajectories are spatio-temporal, i.e. while all three go to the store, they might be going at different times.

To maximize the number of successful peer-to-peer (p2p) connections in WANETs under battery constraints by adaptively modeling individual and population mobility in a distributed fashion?

Consider, for example, emergency information exchange among residents of a rural community in which broadband and cellular data connectivity is either missing or limited (Fig 4.1). Given a finite battery (smartphones typically require at least one daily recharge in typical usage scenarios [105]), deciding when to enable wireless interfaces and attempt to connect to peers for exchange is critical in terms of both minimizing battery footprint and maximizing successful p2p connections, and thus, emergency information exchange. For the example in Fig 4.1, assume that only person 3 (green) has access to the Internet at their home, but not when they are away from home. In this scenario, person 3 will receive up-to-date emergency information on their device every day when they are at home. It will be critical to learn when and where their typical trajectories intersect with those of other people (in this case the cafe, store and post office) so that the information can be pushed to other devices at times they are co-located, albeit offline. While we will stay with rural information exchange as our example application in the paper, it is important to note that optimizing mobility-informed data link connectivity is equally important in other WANET applications such as p2p connectivity in refugee camps, during disaster-caused network disruptions and as part of social movements coordination.

There are three key challenges in maximizing the p2p connectivity based on predictive mobility modeling. First, the optimization and decision making needs to be distributed, i.e.,
assuming partial information at each communication device. This requirement in WANETs is by design as the premise is that no global connectivity is possible and individual devices need to “learn” about others’ mobility based on information sharing. Second, the limited smartphone battery imposes energy constraints for peer discovery and p2p connection and exchange. Finally, while prior research suggests that human mobility is predictable, over a long time horizon and upon perturbations (such as natural disasters), there may be changes to typical trajectories. Hence, the optimization needs to be adaptive to individual and global changes in user mobility.

To address the above challenges, we propose CORE (Connectivity Optimization via REinforcement learning): a data link p2p connectivity protocol for WANETs that is distributed, adaptive to changes in device mobility and energy-aware. To enable distributed decisions, we learn a local (specific to a given device) and global (summary of all other devices) trajectory mobility models, where the latter is being updated based on mobility information exchange upon successful connections with peers. We employ a reinforcement learning (RL) strategy for connection attempt decisions, which places a higher confidence in recent mobility observations and is thus capable of adapting to changes in the mobility patterns. Our protocol is energy aware as it learns to optimally distribute in time a fixed energy budget for p2p connectivity. Using both synthetic and real-world mobility traces, we demonstrate that within our protocols agents are able to materialize 95% of the maximum possible connection opportunities, where the maximum is based on global knowledge of users’ locations. Agents are also able to quickly adapt to changes in the underlying mobility patterns, i.e., our model does not assume stationary of daily mobility for individual agents.

The contributions of this chapter are as follows:

• We develop an adaptive ad hoc protocol for device-to-device communication that minimizes power use while maximizing the communication opportunity of user encounters.

• We devise realistic power consumption models for Android smartphones communicating over WiFi Direct.

• We evaluate our protocol employing a combination of synthetic and real-world mobility traces and demonstrate that agents are able to materialize 95% of all possible connections within 20% of the battery capacity.
• We show that CORE is able to rapidly adapt to changes in the underlying mobility patterns while minimizing the necessary past horizon (and corresponding data) that has to be exchanged among users.

4.2 Related Work

**WANETs:** Research on delay tolerant communication in WANETs dates back to the early PRNET project [106] over 5 decades ago. This rich research area encompasses a wide variety of applications: emergency response and coordination [107, 97, 108, 109, 110, 111], data dissemination [112, 113, 114], defense [115], sensing [111, 116], distributed mobile computation [117] and others. Our work on improving the data link layers in WANETs is complementary to the above systems and deployment evaluations, as they predominantly focus on multi-hop routing. Particularly, if employed in applications with highly mobile nodes and limited battery resources, our protocol can improve connectivity and reduce energy footprint.

The long line of research on WANETs also addresses an array of challenges at different layers of the protocol stack: media access control [118], multi-casting [119] and a significant focus on multi-hop opportunistic routing [120, 121, 100, 122]. The data link layer, or the means to establish direct peer-to-peer connection has received less attention. For example, protocols for routing typically assume a short-term “fixed” topology snapshot in which information needs to be routed without global knowledge of the topology. Our work on battery-efficient distributed link formation is complementary to protocols for routing as it focuses on predictive modeling of peer connection opportunities.

**Modeling and mining human mobility.** Longitudinal studies have demonstrated that human mobility in urban settings exhibits recurrent patterns and can be successfully predicted to a fairly high accuracy[102, 103]. While the above human-centric analysis has wide implications for urban transportation, planning and resource management, to the best of our knowledge such models have not been previously employed to optimize WANET connectivity. In the fields of geographical information systems and data mining, human mobility observations are typically modeled as geo-spatial trajectories [123]. A number of analytics methods and corresponding applications have been proposed for trajectory data including hotspot detection [124, 125], trajectory mining [126, 127], trajectory clustering [128, 2].
**Reinforcement learning.** Our setting lends itself well to a reinforcement learning approach. For one, agents are exposed to minimal information and must obtain it as the system runs, updating their behavior in light of new data. Additionally, the presence of a budget means that a limited number of connection attempts can be made, and therefore the agent must balance maximizing connections based on current information (exploitation) and increasing known information (exploration), a classical tradeoff in reinforcement learning. While various components of our form of model have been explored, their combination makes for a novel problem. Our system has parallels to multi-armed bandits [129, 130], particularly budgeted versions. Budgeted reinforcement learning in general is an existing field [131, 132]. However, the dependencies between different arms (in this case, decisions at each time point that prevent simultaneous choice of all arms) as well as the distribution of budget across these dependent options introduce significant complexity to the decision process. The spatiotemporal nature of our problem has been investigated in the reinforcement learning context as well, although existing approaches address other problems, frequently in regards to resource allocation including node or sensor placement [133, 134] or bike [135] or taxi shares [136]. Our multi-agent setting is also unique; literature exists within adjacent fields addressing interference or channel selection issues among groups of agents, for instance [137, 138]. An additional concern with many of the state-of-the-art approaches is their reliance on deep learning models, which are unsuited to our data- and computational-power-limited setting.

### 4.3 Problem Formulation

The input to our problem is spatio-temporal observations of device locations. We assume that we have full knowledge of the locations for the current device on which our protocol is executing and partial knowledge about locations of other devices which has been propagated when successful connections have been made. Since our protocol is fully distributed, we present it from the perspective of a single device (agent), however, our goal is to maximize the average number of connections among all participating devices. Before we formally define the problem we introduce some necessary notation summarized in Table 4.1. We assume a set of $N$ participating agents (devices) who track their private spatio-temporal locations over time. This set may vary over time, however, we assume a finite $N$ for presen-
tation purposes. Over the course of a day, each agent $a_i$, $i \in [1, N]$ records their locations $l_t^i = \{\text{lat}, \text{lon}\}$ at regular time points $t = 0 \ldots T$. While devices may not track locations at regular intervals we assume that one can interpolate likely locations at $T$ regular time points at which our protocol will make decisions about peer discovery and connections. We use the span of the day to model mobility, however, other intervals can be considered. E.g., one can model users over a week to capture differences in mobility, although such a design will require longitudinal observations to estimate typical behaviors. Shorter periods than a day will likely be harder to predict, while predictive accuracy gains from a longer periods (e.g., a week) may be quickly outweighed by the additional memory requirements.

Since smartphones are used for multiple applications, CORE cannot spend a significant fraction of the battery on peer discovery and connections as this will diminish the utility. Hence, we assume that each device has a finite budget $B$ dedicated to connection attempts across all time points $T$. To derive our model, we initially assume that this budget is measured in terms of number of discovery/connection attempts throughout the day. Then in Sec. 4.4.4 we redefine the budget in terms of realistic battery drain measured on Android devices establishing p2p connections over Wi-Fi Direct [139].

When a device establishes a connection with a peer, it shares its location history and that of other devices it has interacted with for a predefined past interval of $M$ days. As a result, we assume that at any point in time, an agent $a_i$ has access to its own location history

| $N$ | Number of agents |
| $T$ | Number of time points within a day |
| $L_t^i = \{l_t^i\}$ | Location history of agent $a_i$, $l_t^i$ is location at time $t$ |
| $B$ | Daily budget in i) number of connections or ii) % battery |
| $b_t^i$ | Current available budget for agent $i$ at time $t$ |
| $M$ | Agent’s memory for past interactions, in #days |
| $D_i^g$ | Agent’s model for global behavior |
| $D_i^l$ | Agent’s model for its own behavior |
| $\alpha$ | Confidence interval for expected reward comparison |
| $\mu_{k,t}$ | Mean position of trajectory $k$ at time $t$ |
| $\Sigma_{k,t}$ | Covariance matrix for trajectory $k$ at $t$ |
| $sp_{k,t}$ | Trajectory aggregate conditional probability |
| $w_k$ | Trajectory weight |
| $P_H(l, t)$ | Total probability density at location $l$ and time $t$, by model $D$ |
| $\beta$ | Model persistence/decay coefficient |

Table 4.1: Notations used throughout the paper.
and the partial location histories of a subset of other devices \( \{L_j^M\}, j \neq i, j \in [1, N] \). Note that successful exchanges lead to more information about other devices’ locations and even more successful exchanges in turn.

Based on the above input and definitions we are ready to define our connectivity optimization problem:

**Definition 2 Connectivity Optimization for WANETs:** Given the location history \( L_i^M \) of agent \( a_i \), the partial location histories of peers \( L_j^M \), and a remaining budget \( b_i^t \) at time \( t \) in terms of number of remaining attempts for the day, design a decision function \( d(i, t) \) for whether to connect at time \( t \) such that the total number of successful connections for agent \( a_i \) is maximized over a given future time interval.

The above problem definition optimizes the number of connections of a specific agent \( a_i \), however, since we assume a collaborative system of \( N \) symmetric agents, in effect, the goal is to maximize the overall number of connections among peers. Also implicit is a coordination problem, as each connection is a two-way exchange. Note that if the actual future locations for all agents during the day is known, the problem of scheduling when to attempt connections can be trivially solved by simply computing the future time instances in which the agent is within range with peers and allocating the budgets to those times. In our setting of human mobility, however, the agent does not know i) its future locations and ii) that of peers, and hence we need to rely on predictive models for those. In addition, the daily mobility may not be stationary, and thus, we need to design adaptive predictive models for future locations.

### 4.4 CORE: Reinforcement Learning Optimization

The goal of each agent is to maximize successful connection attempts while utilizing as much of the available budget as possible. Thus, our connection attempt decision function \( d(i, t) \) should be able to weigh the costs (risk of budget spent without successful connections) and benefits (successful connection) of a decision at the current and any future time within the day. Since future locations are unknown, the agent should be able to predict its own future locations and that of others to assess trade-offs for a decision at time \( t \).
tackle these challenges, we propose a reinforcement learning approach, named CORE, for the connection decision problem, which incrementally learns i) mobility predictors for the current agent and peers and ii) optimal decision functions for connectivity maximization. CORE does not expect stationary mobility, and can adapt to changes in the daily trajectories of agents.

4.4.1 Modeling and predicting agent mobility

At any point of time an agent has access to its full location history $L_t^i$ and the partial history of peers $L_t^j, j \leq N$. These histories are composed of full and partial daily location histories, derived by splitting these histories along day boundaries.

To model overall mobility $D$, we employ a Gaussian Mixture (GM) to represent trajectories with each component in the mixture representing a cluster of observed histories. A trajectory cluster-based formulation has a number of advantages: 1) it unifies the representation of global and agent-level behavior in a single model, 2) it allows representation of spatio-temporal density in a way that is temporally consistent due to trajectory-level weightings, and 3) it allows for updates both at the level of single point observations and complete trajectories.

We store each of the $k = \{1...K\}$ clusters as a pair of time series $\mu_k = \{\mu_{(k,1)}...\mu_{(k,T)}\}$ representing mean location at each time point and $\Sigma_k = \{\Sigma_{(k,1)}...\Sigma_{(k,T)}\}$ representing covariance in space. Additionally, this structure contains a weighting vector $w \in [0, 1]^K, \sum w = 1$ with a value for each cluster — intuitively, this is the portion of the population that follows each cluster in a given day. Alternatively, this structure can be viewed as a time series of Gaussian mixtures where the weight vector is conserved across time steps. Note that this GM is maintained at the agent level as global model $D^g_i$, though we omit the subscript when unnecessary for clarity.

Each agent also maintains an agent-specific mixture of their own trajectories, $D^l_i$, of the same form. This allows for private deviation from global behavior, or for more precise representation of trajectories that are strong at the private level but uncommon at the global level.

These models can be used to predict density at location $l$ and time $t$ interpreted as
either a normalized agent count in the case of $D^g$ or a probability of presence from the local model. This density is given by $P^D(l, t) = \sum_k w_k P^D(l, t|k)$, where $D$ is $D^l$ or $D^g$ and $P(l, t|k)$ is the Gaussian density according to the mean and variance for cluster $k$ at time $t$, $P(x|k) = (2\pi|\Sigma|)^{-0.5}\exp(0.5(x - \mu)\Sigma^{-1}(x - \mu))$. A single expected location can also be calculated, if necessary.

Since these models represent trajectories through space-time, our prediction of agent locations can be more precise. Specifically, an agent can calculate the anticipated current trajectory from available history within the current day and subsequently project this to future time. The best-match trajectory $k$ is given as arg max$_k \prod_{\tau<t} P^D(l^i_{\tau}, \tau|k)$.

### 4.4.2 Mobility model updates.

As we predict the mobility in an online manner, a key challenge is to be able to update the models upon receipt of new information as well as changes of the underlying mobility patterns. In particular we need to devise updates to cluster parameters from new observations and describe the merger of clusters. We next present how these updates are performed.

Recall that $D$ captures overall mobility as GMM-like trajectories, which allows for updates from single observations to incorporate new information on the fly. This consists of updating the weight and parameters of each cluster according to the likelihood of the observations belonging to it. Updating at each time point is equivalent to maintaining the mean and covariance of a weighted stream, for which we use a modified (for two dimensions and weights) Welford’s algorithm [140],

\[
\begin{align*}
sp_{k,t} &= sp_{k,t} + P(k|l_t) \\
\omega &= \frac{P(k|l)}{sp_{k,t}} \\
\delta l &= l - \mu_{k,t} \\
\mu_{k,t} &= \mu_{k,t} + \omega \delta l \\
\Sigma_{k,t} &= (1 - \omega) \Sigma_{k,t} + \omega (\delta l)(l - \mu_{k,t})
\end{align*}
\]

Here, $P(k|l) = \frac{P(l|k)w_k}{\sum_k P(l|k)w_k}$ via Bayes’ theorem. $sp_{k,t}$ is the aggregate weight for each path cluster $k$ as observed at time $t$; an overall weights $w_k$ can be derived from the sum of these over all $t$. Explicit updates depend on connection success:
1) Update after failed connection: the agent observes that there are no others seeking a connection at the current location, i.e., a connection failure. Nearby clusters are weighted down based on a single observation ‘elsewhere’. Specifically, for clusters whose Mahalanobis distance $(l - \mu)'\Sigma^{-1}(l - \mu)$ exceeds the a selected $(\gamma_{-})$ threshold on the $\chi^2_2$ distribution [141], we update aggregate weights $sp_{k,t} = sp_{k,t} - P(k|l)$, and consequently $w_k$. The mean and covariance parameters are not updated based on this type of observation, as it simply represents additional mass somewhere else as opposed to at a particular location. Any local aggregate weight is also bounded below by 0 to maintain interpretability and consistency. We set closeness parameter $\gamma_{-}$ to 0.05%; other values performed similarly.

2) Mobility update under successful connections: Upon connection, agents exchange unknown information as [agent, time, location] triples. Each of these new point observations is incorporated into $D_g$ as described in Eq.4.1. Transferred information includes an agent’s location history alongside partial location histories of any contacted individuals, up until the memory bound $M$. New information is maintained and exchanged further. In addition to point-level data, information about the encountered agent’s private model is incorporated by adding the clusters contained therein to the receiving agent’s global model. This enriches the global cluster model with additional possible trajectories, if those differ from known behaviors, and gives more complete information about the encountered agent where possible. This exchange is done prior to the incorporation of point data, to make sure that information corresponding to a “new” cluster is properly incorporated.

There are two related updates in this scenario: i) new cluster creation when no existing clusters match a point and ii) cluster merging when a new cluster well matches an existing cluster.

2.1) New cluster creation. A poorly informed model, either through novelty of the agent or changes in true behavior, may not contain any clusters that fit a new data point. In the case of no nearby clusters (determined by $\gamma_{-}$ defined above), a new trajectory is created representing entities remaining in the current location at all times, with some default $\Sigma$.

2.2) Cluster merging. To maintain a sparse and informative model, clusters may need to be combined if i) clusters exist that are sufficiently close or ii) the model grows too large, forcing merger of ”closest” clusters (we set the maximum cluster count to 20). Cluster proximity is measured via time-aggregated Bhattacharyya distance
<table>
<thead>
<tr>
<th>Function</th>
<th>Parameters</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CORE</td>
<td>α</td>
<td>Confidence bound on future RE (Alg. 5, line 3)</td>
</tr>
<tr>
<td>CORE-Now</td>
<td>α, ϵ</td>
<td>Early random exploration (Alg. 5, lines 8-9)</td>
</tr>
<tr>
<td>CORE-Later</td>
<td>α, ϵ</td>
<td>Early random wait (Alg. 5, lines 11-12)</td>
</tr>
</tbody>
</table>

Table 4.2: Summary of alternative decision functions.

\[ D_B = \sum t \frac{1}{8}(\mu_{1,t} - \mu_{2,t})'\Sigma_t^{-1}(\mu_{1,t} - \mu_{2,t}) - \frac{1}{2} \ln(\frac{\text{det} \Sigma_t}{\text{det} \Sigma_{1,t} \text{det} \Sigma_{2,t}}), \] with \( \Sigma_t = 0.5(\Sigma_{1,t} + \Sigma_{2,t}) \). If \( D_B \) is below a pre-determined threshold (or if too many clusters are present, the smallest distance is used), two clusters are merged. Underlying clusters are assumed to be Gaussian, so we create a new Gaussian by merging two existing ones. Mean/variance updates follow:

\[
\begin{align*}
\mu^* &= w_1\mu_1 + w_2\mu_2 \\
\Sigma^* &= w_1\Sigma_1 + w_2\Sigma_2 + w_1\mu_1'\mu_1 + w_2\mu_2'\mu_2 - \mu^*'\mu^*
\end{align*}
\]

(4.2)

Where \( w_1 \) and \( w_2 \) are the weights of the two clusters normalized on their sum, \( w_1 + w_2 = 1 \). Other quantities, including aggregate observed weights \( sp \), are also combined.

3) Daily updates for \( D_i \). The agent’s daily trajectory provides additional information that can be used to update the entire \( D_i \), similar to the single-observation updates described above. Here, however, we iterate through the entire day’s history and update \( \mu_{k,t} \) and \( \Sigma_{k,t} \) for each time point. If the observed trajectory is a poor fit to any of the pre-existing clusters, as measured by the sum of the Mahalanobis distance compared to the appropriate \( x^2_{2*T} \) statistic and closeness \( \gamma_- \), the agent creates a new trajectory with \( \mu_{K+1} \) given by the observed path and \( \Sigma_{K+1} \) set to a default value.

4.4.3 Reinforcement learning for connection decisions

Incomplete information about the past, uncertainty about the future, and limited resources present difficulties for traditional machine learning methods. Agents are unable to collect full information on peer activity because each attempt to do so comes with the risk of potential failure and unjustified battery drain. Additionally, while the goal of a system like CORE is to be responsive to changes, collecting sufficient information for adequate training after every behavioral shift is time-intensive. This tradeoff between information collection (exploration) and target optimization (exploitation), is the purview of Reinforcement Learning, which aims to optimize utility through balancing these two considerations.
Algorithm 5 CORE

Require: $t, l, b_t, D^g$, weighted own future location $P_{D^g}(l, \tau)$
1: $RE_{\text{future}} \leftarrow \text{empty array of length } T - t$
2: for $\tau = 1$ to $(T - t)$ do
3: \hspace{1em} Value[\tau] $\leftarrow \sum_t P_{D^g}(l, t + \tau)D^g(l, t + \tau) + CI(\alpha) \ast \sigma(t + \tau)$
4: end for
5: Sort $RE_{\text{future}}$, descending
6: $RE_{\text{now}} \leftarrow D^g(l, t)$
7: DECISION $\leftarrow RE_{\text{now}} \geq RE_{\text{future}}[b_t]$
8: if CORE-Now and $rand < \epsilon$ then
9: \hspace{1em} DECISION $\leftarrow True$
10: end if
11: if CORE-Later and $rand < \epsilon$ then
12: \hspace{1em} DECISION $\leftarrow False$
13: end if
14: if DECISION then
15: \hspace{1em} Attempt Connection
16: \hspace{2em} if Successful Connection then
17: \hspace{3em} Absorb $D^g$ from partner into $D^g$, if new partner
18: \hspace{3em} Merge clusters in $D^g$, if necessary (IV.B.2.2)
19: \hspace{3em} Update $D^g$ with new data from partner (IV.B.2)
20: \hspace{1em} end if
21: \hspace{1em} Do a ”failed” update at the current location (IV.B.1)
22: end if
23: end if

The overall goal for an agent is to optimize total connections, but the immediate reward for a decision is the success of the connection attempt. Maintaining a function for the expected reward is a key component of a viable RL model; in our case, the mobility model described above yields a usable approximation to the expected reward through the probability of encountering other agents $P_{D^g}(l, t)$. The impact of the agents action (the decision to connect) is both on the available budget, and on the total reward, as it reduces the budget for possible future connections. An effective decision function must be able to balance this opportunity cost with current reward. The decision is contextual, with context inferred from both the current model state and the agent’s current location.

Decision Functions. We present our task as a decision problem solved by each agent at each time point $t$. The agent weighs the value of an immediate connection attempt against the values of future attempts. For a budget of $b_t$ remaining connections, an attempt at time $t$ is optimal in terms of reward if the current value $D^g(l, t)$ exceeds the value of the $b_t^{th}$ best future time point. Computing the value of connection at any place and time requires an estimate of the spatio-temporal distribution of other agents; computing future values additionally requires knowledge of one’s own future location.
This approach incorporates a confidence interval $\alpha$, which accounts for anticipated variance in expected future rewards $RE_{\text{future}}$. This variance, $\sigma(t + \tau)$, can be computed via the statistical variance of $D^g(l, t + \tau)$ weighted by $P^D(l, t + \tau)$ (as a significant portion of variance comes from uncertainty in future location). Coefficient $CI(\alpha)$ is the confidence bound determined using a standard normal distribution for simplicity.

An approach using $\alpha = 0.5$ would eliminate prediction uncertainty, effectively resulting in a greedy counterpart. Alternatively, employing an $\epsilon$-greedy RL approach would incorporate $\epsilon$, controlling the probability of taking an action regardless of state. We can prioritize either early attempts (connect **Now** with probability $\epsilon$) or later attempts (connect **Later** with probability $\epsilon$). Different values of $\alpha$ can be combined with $\epsilon$ for CORE-Now or CORE-Later. The three families of decision functions are listed in Tbl. 4.2.

**The CORE Algorithm.** The CORE steps are detailed in Alg. 5. In every time point, an agent computes the current expected reward $RE$ based on immediate location as well as predicted future rewards based on anticipated locations and the current model. If this decision indicates a connection attempt should occur, budget is allocated; either a failed or a successful connection yields updates as described in the previous sections. An additional update is done at the end of each day on local model $D^l$.

**Model Update Considerations.** Apart from the immediate reward from a connection attempt, each time step yields useful information to the agent. Location information is collected for free, regardless of the agent’s decision, whereas any attempted connection results in information about the presence or absence of others at $(l_t, t)$, which can be used to update the trajectory model described above.

A frequent consideration in RL is the learning rate for updating the learned reward function with new data. For CORE, this equates to two memory-based parameters. Increasing agent memory $M$ gives greater information at each connection and therefore more updates to an agent’s global model $D^g$. Model persistence, on the other hand, is maintained through a decay parameter $\beta$, which is multiplied by all cluster $sp_{k,t}$ at the end of the day, preserving cluster relative weights $w_k$. This parameter is inversely related to the traditional learning rate - a lower $\beta$ means less weight on an existing model and therefore more on newly arriving information.
Table 4.3: Percentage of battery drain (as well as power in mW and latency) required to *connect* to a peer and *send* 1 Byte of data. Energy for receiving data is negligible (omitted).

<table>
<thead>
<tr>
<th>Event</th>
<th>Power (mW)</th>
<th>STD</th>
<th>∆T (h)</th>
<th>STD</th>
<th>% Battery</th>
<th>STD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connect</td>
<td>1.1330e+3</td>
<td>58.01</td>
<td>0.0022</td>
<td>1.842e-4</td>
<td>0.0215</td>
<td>0.0011</td>
</tr>
<tr>
<td>Send</td>
<td>2.3406e+3</td>
<td>104.95</td>
<td>2.056e-5</td>
<td>3.020e-7</td>
<td>8.30e-10</td>
<td>4.02e-11</td>
</tr>
</tbody>
</table>

4.4.4 Realistic budget based on battery drain

In the algorithm and models described above we made the simplifying assumption that the budget is specified in terms of number of connection attempts, and that each connection attempt and the corresponding exchange has a fixed battery cost. However, this assumption does not hold for real-world exchanges. Furthermore, the allowable battery expenditure should be exposed as a configurable setting to CORE users. Hence, we next discuss how to augment our protocol to allow for measurement-based realistic cost in terms of battery drain.

We first measure the connection and exchange cost in terms of percentage of battery drain when two Android devices establish a connection and exchange information using Wi-Fi Direct. Then we discuss the changes to our protocol to account for budget in terms of battery cost. Note that our measurements for Android devices and Wi-Fi Direct are only for demonstration and evaluation purposes with user-friendly budget definition (see Evaluation Fig. 4.5). In real deployments, one can estimate device-specific costs and also consider alternative (or even multiple per device) radio connectivity such as Bluetooth and multipeer connectivity for Apple devices.

To estimate connection and exchange costs we use one Motorola G(6) and one Samsung Galaxy 5 Duos smartphone with a simple Wi-Fi Direct application installed. The Samsung is connected to a power monitor [142], bypassing the battery, so we can measure the energy consumption of the device. We measure the required power to i) establish a Wi-Fi Direct connection (*Connect*) and ii) *Send* variable amounts of data to a peer (Table 4.3). Note that the *Connect* cost includes power required to scan for peers and establish a successful Wi-Fi Direct connection. The average power \( P_{avg} \) (mW) and duration \( \Delta T \) (s) is measured over multiple exchange instances and transformed into percentage of battery discharge per byte of data transmitted. To find this percentage, we calculate the required milliwatt-hours \( (mWh) \) to transmit 1 byte of data by \( P_{avg} \times \Delta T \) (normalized to hours) and then divide by the number of bytes transmitted. This will give us the \( mWh \) per byte. The percentage of
battery discharge per byte is reached by dividing the $mWh$ per byte by the battery capacity, which is 11550 $mWh$ for the Samsung device.

Modifying CORE to admit variable connection costs consists of two components. First, we are able to compute actual exchange cost and therefore can decrease the budget by exact non-integer amounts, instead of simply in units, after any successful connection. Second, we need to adjust the anticipated number of remaining connections at any decision point by diving available power by an expected cost. This expected cost can be computed using the same streaming mean computation as the trajectory cluster means described in the previous section. Each agent stores this expected connection cost $C_{exp}$ locally, and updates it with the computed cost of every connection attempt. We can then modify the total daily budget $B$ to be a battery percentage, and the remaining number of connections becomes $b_t = \frac{B}{C_{exp}}$.

4.5 Experimental Evaluation

We evaluate the ability of CORE to optimize the number of successful connections among peers in both synthetic mobility traces and a real-world human mobility trace.

4.5.1 Datasets

Synthetic data. We generate synthetic data based on a trajectory model in which we can control for agent co-location over time. The overall intuition is to model agents as subscribing to a subset of random global trajectories from which they draw their daily location histories. Figure 4.2 illustrates the sampling approach for individual agents (location is 1D only for illustrative purposes). We generate a set of global proto-trajectories (or clusters) characterized by spatio-temporal mean locations and covariance matrices, and corresponding to large-scale mobility patterns (Fig. 4.2-left). Individual agents randomly draw from two global trajectories (weekday and weekend patterns) and then sample their daily location histories from these draws as demonstrated in Fig. 4.2-right. The agent’s locations for two days in this example (red and green traces) are randomly sampled from the agent-level proto-trajectory depicted as a dashed black line with user-specific variance added which we control by a parameter $\sigma_{self}$.

Our generated synthetic traces are based on four trajectories, each representing what
Figure 4.2: An illustration of the synthetic data generation. An agent “subscribes” to a subset of global proto-trajectories (left panel). Right panel: Daily location histories (red and green lines) are sampled from a proto-trajectory (dashed line), with user specific variance (grey envelope) applied.

A typical global daily behavior - an early-day static location with large variance representing multiple “home” locations, a mid-day location with small variance (“work place”), and a return home while passing through a set of locations with mid-to-large variance during the latter third of the day. There is some overlap at various stages of the day for different global trajectories. Agents are randomly distributed between two daily paths (one path per day), which could represent weekday vs. weekend behavior. Overlap between agents can come from either a shared global path or from intersections between paths, the latter being a better source of global information for a given agent. Unless otherwise specified, data is “hourly” (i.e. $T = 24$), 30 agents are present in a $100 \times 100$ grid, and $\sigma_{self} = 0$.

Since our goal is to simulate both cold start learning and adaptation to significant changes, we also introduce a shift in the underlying mobility in our synthetic traces. Unless otherwise specified, this shift appears on day 30 with the “work place” location in two of the proto-trajectories moving along with the timing of the workday. This affects the locations of agents which subscribe to these trajectories.

Real-world data. For our real-world experiments we utilize the Yonsei Lifemap dataset [143] from CRAWDAD, which contains location observations at Yonsei University in Korea. The dataset contains nine users whose location histories overlap over 63 days in 2011. To create a uniformly sampled location histories, we impute location every hour based on the temporally closest available data point for each user. We remove days in which users’ do not collocate. In order to enable longer temporal window for learning, we randomly re-sample and “replay” full daily histories for all users, thus creating a 250-day long trace for evaluation of our method.
Figure 4.3: (a) Comparison of CORE’s quality (FC) for different choice and Random and Preset baselines. All compared methods have a budget of 10 connection attempts each day.

4.5.2 Evaluation metrics

The primary metric presented for our analysis is the Fraction of ideal Connections (FC) per agent, defined as the number of successful connections divided by all possible connections (based on actual co-locations with other agents and capped by the available budget). This metric is the larger of precision (true positives over all selections) and recall (selected positives over all positives) due to the budget bound on the denominator. From RL perspective, the quantity \(1 - FC\) is also a daily measure of the regret, i.e. the difference from an optimal agent. The maximum attained value of FC is denoted by \(FC_{\text{max}}\). Due to the variability of this statistic, we present a five-period moving average maximum in all experiments.

The evolution of FC over time is a measure of learning rate \(T_{\text{pct}}\), defined as the number of days needed to achieve a fixed percentage \(pct\) of \(FC_{\text{max}}\). Our synthetic traces all feature a midpoint shift in the underlying mobility. To quantify the quality of CORE to learn from a “cold start” and after a shift, we present \(T_{\text{pct}}\) and \(FC_{\text{max}}\) for the period before the shift \(\text{Pre-shift}\) and after it \(\text{Post-shift}\).
4.5.3 Effect of the decision function, memory and budget

In what follows, we first evaluate CORE’s quality when varying essential parameters in separate experiments. Unless stated otherwise, we employ CORE with memory parameters \( \beta = 0.8 \) and \( M = 2 \) and a budget of 10 connections. Variances and averages in the figures are computed based on five runs.

Effect of decision function and comparison to baselines. We begin by examining the effects of various decision functions. In Fig. 4.3 we present the quality of CORE’s variants with decision functions from section IV.C in comparison to two baselines: Random, in which the budget is spent at random times by each agent; and Preset in which daily connection attempts are performed every two hours starting at hour 4. All methods have a budget of 10 daily connections.

Differences in quality among variants of CORE are subtle but informative. In the basic CORE decision function (no exploration), confidence bound of \( \alpha = 75\% \) seems to be optimal compared to a simpler mean-comparison approach (\( \alpha = 0.5 \)) or lower confidence (\( \alpha = 0.25 \)) which performs close to random. Increasing the confidence to \( \alpha = 0.9 \) becomes too ”conservative” setting and leads to slower learning. Introducing exploration as opposed to greedy decisions, akin to an \( \epsilon \)-greedy in reinforcement learning, improves performance noticeably. In particular, CORE-Now which prefers early exploration outperforms all alternatives, while CORE-Later which has a preference for conservative (delayed) decisions to connect performs on par with the best basic approach CORE, \( \alpha = 0.5 \). The baselines Random and Preset perform significantly worse than the best versions of CORE.

Another notable observation from this experiment is that CORE-Now reaches 90% of its peak \( FC_{\text{max}} \) in as few as 6 days both before (Pre) and (After) the shift mobility at day 30 While cold-start performance is largely similar for all \( \beta \) in Fig. 4.4(a) due to a lack of model to persist, the inertia of a bad model means higher \( \beta \) takes longer to recover. On the other hand, too low a \( \beta \) means minimal learning is retained and overall performance suffers. (Rows \( T_{90} \) in Fig. 4.3). Its performance peaks at over 74% (Pre) and 77% (Post) of the maximal possible connections given sufficient time to learn the underlying mobility model. This performance is aided by CORE-Now’s randomness, as a fixed method can be locked in to suboptimal choices when information about alternatives is limited, particularly in our approach where there exists an implicit preference for attempts later in the day if
initial early attempts fail. Random and Preset’s quality in comparison is limited as they do not model and learn the underlying mobility.

Figure 4.4: Effect of model decay $\beta$ (a) and memory $M$ (b) on the successful connections. Panels (c) and (d) present effects of both parameters on learning rates at the beginning of the experiment and after a mid-stream behavioral shift. Panels (e) and (d)

**Connection budget.** Another important question is: What is the battery footprint for attaining high levels of FC? A greater budget allows for more information gain via exchanges among agents and our goal in the next experiment is quantify the relationship between battery percentage dedicated to connections and exchanges and the attained FC. For this experiment we estimate the actual battery drain for each exchange (both connection and data transfer) based on our Android-based cost model from Sec. 4.4.4. Figure 4.5 demonstrates that higher budget enables not only higher connectivity, but also a greater learning rate, i.e.
Figure 4.5: (a) Quality of CORE ($\alpha = 0.5$) over variable budget in battery % and (b) corresponding metrics. Daily time points $T = 144$

faster convergence to $FC_{max}$. In particular a budget of 20% allows for success in almost all connection opportunities, while 10% budget peaks at $FC_{max} = 0.67$. In settings of limited co-location of agents (fewer opportunities to connect) it makes sense to increase the budget initially to enable some exchanges to happen and bootstrap learning. Additionally, higher budgets allow for consistent performance while lower ones results in larger variance. CORE is able to quickly recover from the mid-point shift at day thirty across battery budgets, but more so with higher budget allocations.

**Memory and model decay.** An important aspect of CORE is its ability to adjust to evolving mobility, which is controlled by the amount of observations used for training (or memory $M$) and the amount of "inertia" or weight given to a trained model (decay parameter $\beta$). Fig. 4.4 shows the impact of these parameters, particularly given the shift in agent mobility at day 30. While cold-start performance is largely similar for all $\beta$ in Fig. 4.4(a) due to a lack of model to persist, the inertia of a bad model means higher $\beta$ takes longer to recover. On the other hand, too low a $\beta$ means minimal learning is retained and overall performance suffers. Larger memory $M$, however, is consistently beneficial according to Fig. 4.4(b). Based on $T_{90}$, a longer memory $M$ (i.e. more information exchanged) enables faster learning through more model updates. The effect of model decay rate $\beta$ is less clear for “cold start” learning (4.4(c)), however, a lower inertia is clearly preferable when responding
Figure 4.6: Effect of (a) variability in personal trajectory, (b) number of agents, and (c) spatio-temporal intersection of global trajectories.

to a large-scale shift (4.4(d)), as it enables faster recovery across different memory lengths. Examination of maximal performance in Figs. 4.4(e) and 4.4(f) yields results consistent with the timecourse figures - an optimum exists at medium $\beta$, while more memory is beneficial only to a point.

### 4.5.4 Effect of the underlying mobility and population size

We next evaluate CORE’s performance for varying agent mobility dynamics and population size. Intuitively, more stable individual trajectories mean more reliable information from exchanges. In Fig. 4.6(a), we see that increasing the variance of single-agent behavior leads to paths that are harder to learn and require longer time frames to reach a high FC. The data shift at day 30 in our synthetic examples doesn’t have an observable effect for larger variances, as the lower consistency of mobility leads is comparable to changes incurred by the shift.

In Fig. 4.6(b) we consider the impact of the number of agents $N$ on the performance of CORE. With higher $N$ the availability of partners, and thus, “mixing”, increases. Additionally, as demonstrated before, more data availability leads to faster learning, which results in lower $T_{90}$ for larger agent pools. Learning occurs among as few as 10 agents.

The interactions of global trajectories play a role in agent effectiveness. More intersection between clusters means more opportunity for agents between clusters to encounter each other, and therefore more learning can occur. Fig. 4.6(c) demonstrates this by varying the overlap, in number of identical spatiotemporal locations, between pairs of global trajectories. Due to generator considerations, these global trajectories are more chaotic. Less of a pattern in behavior leads to less clear results, however learning is still present albeit quick: $T_{90} = 1$ for all approaches. Variability in learning on this parameter indicates that agents do not
only connect with those on the same global trajectory (the number of those is, on average, fixed across settings).

4.5.5 Real Data Results

Next we evaluate CORE’s behavior on the Yonsei mobility trace. As we observed in synthetic traces, effective learning hinges on mobility consistency and large populations for mixing. The Yonsei trace with only 9 users and large variance in individual mobility presents a particularly challenging case even after re-sampling of full days. Nevertheless, CORE exhibits promising behavior in this trace.

Fig. 4.7 shows the performance of six different methods on the Yonsei dataset, with an available budget of 10 connections (out of 24 time points). An immediate takeaway is the importance of coordination: Random connection decisions perform particularly poorly, even compared to attempts that are timed randomly but synchronous across agents (Preset Random). Pre-selecting the first 10 time points on the middle of the work day (Preset 1-10) which we know covers most of the opportunities for connection due to co-location of the users performs particularly well. However, without this “oracle” knowledge, inappropriately hard-coded preset times (Preset 15-24) can perform arbitrarily bad (even worse than Random).

The worse performer of the two variants of CORE, is the basic decision function: CORE, $\alpha = 0.6$. By default, CORE is conservative with its agent budget. Limited opportunities for connections, combined with the high variance of individual agents, makes conservative learning without exploration (basic CORE) quite challenging. However, CORE-Now, $\epsilon = 0.1$ performs very well, and even outperforming the Preset 1-10 "oracle" by an average
FC of 0.04. Random exploration early allows better agent learning, and the information gained thereby allows agents to target times that don’t match the early-day pattern.

Further refinement of the decision function as well as modeling parameters are likely to allow for further improvement, though a detailed examination on this particular dataset is outside the scope of this work. As a simple direction, we can modify the randomness of the CORE-Now approach with time, reducing the $\epsilon$ parameter to target exploitation of learned patterns after early exploration has yielded meaningful information (i.e. time-evolving exploration).

4.6 Conclusion

We introduced an adaptive and distributed protocol, named CORE, with the objective to maximize peer-to-peer connectivity in WANETs based on reinforcement learning. Our protocol addresses an important challenge in ad hoc networks of smartphones, namely, a predictive and energy-aware data link layer based on modeling human mobility using partial knowledge. CORE models individual and global mobility based on partial historical location observations and incorporates predictions of these models into a reinforcement learning framework capable of on-the-fly decision making regarding when and where to scan for peers. We evaluated our model in both simulated and real-world mobility traces and demonstrated that agents are able to materialize 95% of the maximum possible connection opportunities using at most 20% of the phone battery for discovery and exchange. CORE can serve as a foundation for a plethora of distributed applications such as disaster response, activist coordination and information access in disconnected areas.
CHAPTER 5
Conclusions and Continuations

5.1 Summary and Conclusion

Within this thesis I have examined the role of subgroups within dynamic networks. In particular, I have focused on a principled modeling of temporal behavior for these subgroups, and how this affects algorithmic performance. In the first chapter I formulated a general framework for learning temporally-regularized, overlapping communities within dynamic networks under a simple, but effective, assumption on temporal behavior. The second chapter expanded this temporal behavior to more complex (and generally applicable) forms, particularly focusing on bursty or self-reinforcing temporal patterns. The potential for other patterns, for example periodicity, was also examined. The final chapter focused on the application of this concept to a spatiotemporal setting where the connections used as a basis for the preceding work were not guaranteed, and where information was limited and received in an online fashion. Therein I presented a reinforcement learning framework where modeling global spatiotemporal trajectories as clusters of similar, overlapping potential paths resulted in meaningful, learnable representations.

More generally, I have shown that properly considering time as an independently-structured aspect of dynamic data is key to understanding the overall system. An explicit model for temporal activity grounded in interpretable behaviors aids significantly in reconstructing community membership as well as reconstructing true community behaviors. Information contained within the temporal dimension is critical for unpacking the underlying structure of these systems. Likewise, modeling subgroups as drivers for overall behavior creates effective and usable models of the systems themselves, as I’ve demonstrated in mining and prediction tasks.
5.2 Potential Future Directions

5.2.1 A more comprehensive model of temporal behavior

The general framework presented in chapters 2 and 3 allows for a broader concept of temporal behavior within groups than has examined in this thesis. While we can easily enough model a variety of specific behaviors (on/off, bursty, periodic) by varying the regularizers, a more complete model should more thoroughly represent various temporal behaviors.

One potential approach is detecting behavioral changes at the subgroup or community level and applying different submodels when appropriate. A group might vary between bursty and periodic behaviors depending on external factors, for instance. In order to construct a suitable model, there needs to be a way to measure the fitness of a particular temporal assumption on a subset of data. Switching between regimes, likely at the subgroup level, could then be done in a manner similar to choosing a model among a set of candidates. A potential approach is one similar to that presented in Chapter 3 of this thesis, where we can measure distance to a candidate approximate “shape”, but such shapes need to be tailored specifically to the task at hand. Of course, this approach also brings with it concerns of appropriate windowing as well as computational complexity.

Windowing, aggregation, and their associated complexity are an additional aspect of temporal data that deserves further attention. Within this thesis time was considered to be discrete and at a fixed resolution. In Chapter 2 I discussed the concept of oversampling and how aggregation is insufficient to alleviate the associated problems if the correct (and consistent) level of aggregation is not known. Generally, this temporal “truth” is in fact unavailable. One alternative is to treat the full scale of data, though as we move towards continuous time computational complexity increases significantly and the discrete models presented herein are no longer directly valid. Alternatively, we can consider ways to aggregate (or even disaggregate) time in a data-aware manner. What levels of resolution provide sufficient information without introducing too many gaps or too much noise? For example, periods of quiescence between community bursts may be good candidates for “shrinking” into a small number of time points, but we would retain full resolution on information-dense bursty regions in time. This approach again requires a consistent model of time that is able to adapt to a varying distance between adjacent data points.
More generally, it may be possible to develop a universal representation for temporal behavior that can be more directly applied. An additional challenge here would be to retain interpretability for the behaviors derived — it is certainly possible to apply existing methods, such as a Fourier transform or a wavelet decomposition, but this risks losing the meaning behind "bursts" or "periods of inactivity" that more direct modeling grants.

5.2.2 Analysis and applications of derived temporal behavior

Large-scale temporal data is still a relatively new addition to the field, and as such it is difficult to come by datasets with suitable temporal ground-truth. The methods developed in this thesis output interpretable temporal traces for individual communities (time factors in chapters 2 and 3, the overall global and local models in chapter 4). Properly evaluating these, however, is a complex task that should become more tractable as more labeled data becomes available. Adding to the difficulty of this task is a lack of standard definitions for the subgroups themselves, particularly within dynamic networks and systems, which further obscures the “true” temporal activity patterns.

Even with limited ability for direct validation, however, there is room for deeper analysis of the temporal traces themselves. Can we utilize bursty, periodic, or otherwise consistent patterns to make predictions about the future, particularly since we have information on which subgroups follow which behaviors? This has direct applications to load prediction, advertising within social networks, disaster preparedness, and a host of other tasks. Being able to more accurately track (and predict) subgroups within a larger setting allows us to tailor cellular or computational nodes, advertisements, or critical information to where/whom and more importantly when they are most necessary.

Another interesting question is how we can utilize the temporal behaviors we learn directly. What can we infer from particular on/off patterns, either in relation to known events or in and of themselves? These behaviors may be usable as information as well. For example, if we learn the activation patterns of critical neuronal subnetworks within the brain, these patterns may differ between normal and diseased states, serving as biomarkers. A similar application may be possible with botnets or other technological settings, with differences between normal and compromised communications traffic informing us which subgroups of nodes are worth further investigation.
BIBLIOGRAPHY


76


[55] : Reddit comments crawl https://www.reddit.com/r/datasets/comments/3bxlg7/i_have_every_publicly_available_reddit_comment/


