INTERPRETABLE MACHINE LEARNING FOR PREDICTING SYNCHRONIZATION OF COUPLED OSCILLATORS

Agam Goyal
University of Wisconsin-Madison
agoyal25@wisc.edu

Bella Wu
University of Wisconsin-Madison
zwu363@wisc.edu

Binhao Chen
University of Wisconsin-Madison
binhao@cs.wisc.edu

Bryan Xu
University of Wisconsin-Madison
zxu455@wisc.edu

Hanbaek Lyu
University of Wisconsin-Madison
hlyu@math.wisc.edu

ABSTRACT

In this paper, we develop an interpretable machine learning-based approach for the prediction of the synchronization of coupled oscillators. More precisely, we touch on the problem of learning key latent features of large-scale networked dynamical systems that are explicable and responsible for the long-term synchronization behavior under the auspices of the dictionary learning method and its supervised variant. We also take an alternative approach to the prediction of synchronization, transforming the prior-studied binary classification problem into a regression problem that regresses the percentage of synchronizing/non-synchronizing cases out of a certain number of initial configurations for an arbitrary but fixed undirected graph. The numerically simulated results based on these two types of machine learning problems simultaneously corroborate that predicting the synchronization of a large-scale networked system solely with graph topology and/or basic graph statistics remains an intractable problem. By tweaking three deterministic models of discrete coupled oscillators - the (discretized) Kuramoto model, Firefly Cellular Automata, and Greenberg-Hastings model, we propose several justified directions that may be of interest in advancing the theoretical understanding of the discriminating conditions for global synchronization in systems of pulse-coupled oscillators with arbitrary initial configurations.

Keywords Coupled Oscillators · Synchronization · Non-negative Matrix Factorization · Dictionary Learning

1 Introduction

For decades, theoretically understanding the large-scale behavior of any complex systems that consisting of locally interacting dynamic agents has been an important subject of research in various areas of science and mathematics. In particular, one of the most widely studied of such collective behaviors should be the long-term global synchronization of coupled oscillators [1, 2](e.g., BZ chemical oscillators, blinking fireflies, circadian pacemakers). The analysis of the long-term synchronization behaviors in such kind of networked systems can be used to model many common but important phenomena that we would like to have an in-depth understanding: formation of public opinion, development of cancer cells, the outbreak of epidemics, trending topics on social media, and collective computation in distributed computing systems. This problem is further closely linked to that of clock synchronization, which is essential in establishing shared notions of time in distributed systems and has enjoyed fruitful applications in many areas including designing distributed algorithms, wildfire monitoring, electric power networks, robotic vehicle networks, large-scale information fusion, and wireless sensor networks [3–5].

Synchronization occurs when the oscillators spontaneously consensus and lock to a common frequency or phase of the coupled dynamical system. The problem of predicting whether a given system of coupled oscillators with an underlying arbitrary graph structure will eventually synchronize, although relevant, is analytically intractable in a variety of fields. In spite of several sufficient conditions for model parameters (e.g., large coupling strength) or initial configuration (e.g., phase concentration being an open semicircle) are known, it often seems elusive to obtain analytical or asymptotic solutions to prediction problems and, especially when the underlying graph is heterogeneous and the initial phase configuration is not confined in a small arc of the phase space. The design and analysis of a perfect algorithm for the synchronization prediction problem might be indirectly solved if any mathematically justified characterization of the sufficient and necessary conditions of the ultimate synchronization is proposed. This importance also elicits the motivation of our work.

Binhao: A paragraph that gives the reader a first taste of what the exact problem/question we will be dealing with in this paper, just like what the others did in LP2Sync

1.1 Related Works

The importance of synchronization problem in coupled oscillators and more broadly, the general dynamical systems, has been recognized and well studied in a variety of settings. Some recent works also take advantage of machine learning's black box-like powerful capability of pattern recognition and extraction of key information from the complex and large-scale datasets, to study the long-term global synchronization behavior of given system of coupled oscillators.

HL: Can also add other papers that use ML to dynamical systems: see the related works in the L2Psync paper

Binhao: Sure, will do a more comprehensive literature review when we are approaching a prerprint version of this paper.

A vital prior work (L2PSync). A recent work called *Learning to Predict Synchronization* (`L2PSync`) [6] views the synchronization prediction problem as a binary classification task, where each data point consisting of initial dynamics and/or the characteristic statistics of underlying graphs is classified into either the 'synchronizing' or 'non-synchronizing' bins, depending on whether a given system eventually synchronizes or converges to a non-synchronizing limit cycle. This framework shows standard binary classification algorithms trained on large enough datasets of initial dynamics can successfully predict the unseen future of a system on highly heterogeneous sets of unknown graphs with surprising accuracy. Although `L2PSync` [6] shows that the lack of a general analytical solution does not preclude the possibility of successful prediction of synchronization, the fundamental problems of theoretically characterizing conditions for synchronization and deriving global synchronization of phase oscillators from arbitrary initial configurations still remain challenging. One of the main themes of this work is to further understand how the standard classification algorithm deployed in the `L2PSync` [6] framework is so successful in predicting the synchronization of large-scale and complex systems. The result of the `L2PSync` [6] also highlights the fact that there is no meaningful difference between the performance of rudimentary binary classification algorithms (e.g., random forests) and advanced deep neural network architectures (e.g., Graph LRCN), where the latter type of algorithms are normally much more expensive in terms of computational efficiency. These homogeneous performances suggest that there may be some interpretable and fundamental patterns in graph topologies or their short-term dynamics that these machine learning algorithms recognize effectively. We therefore seek to develop a machine learning-based approach to synchronization prediction, similar to the method in `L2PSync` [6], but extracting interpretable features that the algorithm learns to distinguish between synchronized and non-synchronized examples.

Main contributions. In this work, we combine two main ingredients, namely, *cellular automata* and *supervised dictionary learning*, to investigate solutions to the problem of developing a machine learning method that can predict whether a given coupled oscillator system will eventually synchronize and also simultaneously extract discriminating features that are used for the classification of training examples. Although there a considerable amount of work that utilizes modern machine learning techniques to investigate problems on coupled oscillators has achieved superior performance, the interpretability of those machine learning-based frameworks has not been well-addressed yet, including the `L2PSync` [6]. We first show that predicting the synchronization of any arbitrary networked systems solely based on graph topology and/or fundamental graph statistics, even adopting several more complex graph representation algorithms (graph embedding) to increase the density of the underlying graph, remains an intractable problem. Reversely, this result also implicitly implies that the initial dynamics, even just one single configuration at any time t, carry out the decisive information that is responsible for determining whether the given networked system (in the form of an undirected connected graph with specific evolving rules) will eventually synchronize.

(A brief summary of the main conclusion we have for this work, will be filled up later)

Our work could be viewed as a multi-objective optimization problem, which is hoping to achieve a decent accuracy of the classification task while providing reasonable guarantees of the interpretability of features selected by the dictionary learning algorithm simultaneously. To the best of our knowledge, the numerical experiments and analysis undertaken in this paper are the first of their kind. We hope that this work contributes to, at least offers several justified directions that may be helpful in elucidating the precise characterization of the conditions one must make to guarantee the eventual global synchronization of the given system of coupled oscillators.

1.2 Roadmap

This paper is organized as follows. In §2, we define the problem statement and introduce the unified notations that are exclusively used throughout the paper. A general discussion on the method we used to run the numerical experiments is included in ğ3. The main results of the paper are presented in §4, with illustrative results and discussion given in §5 and §6. We remark the conclusion of this paper and point out interesting directions for future work in §7.

2 Problem Formulation and Preliminaries

2.1 Problem Statement

Consider a graph $G = (V, E)$, where V and E denotes its set of nodes and edges respectively. Let Ω denote the phase space of each node $v \in V$, which may be considered to be the circle $\mathbb{R}/2\pi\mathbb{Z}$ for continuous-state oscillators (e.g. Kuramoto model) or the color wheel $\mathbb{Z}/\kappa\mathbb{Z}, \kappa \in \mathbb{N}$ for discrete-state oscillators. Here we define the phase configuration as a map $X : V \to \Omega$, and say the corresponding coupled oscillators system is synchronized if it locks to a common frequency. In other words, the system is deemed as synchronized if it takes a same value for all nodes at a single time point, (i.e., $X(v) = c, c \in \mathbb{Z}_+$. for all $v \in V$). We also define the coupling as a function ϕ that maps each pair of graph and its initial configuration (i.e., (G, X_0)) to a deterministic trajectory $(X_t)_{t\geq 0}$ of phase configurations, where $X_t : V \to \Omega$. For instance, ϕ could be the time evolution rule for the coupled oscillators that we used for modeling networked systems (e.g., KM, FCA, or GHM). The main problem we investigate in this work is stated below:

Interpretable *L2PSync*. Given any connected graph $G = (V, E)$, coupling ϕ, and fixed parameters $n \in \mathbb{N}, T \gg r > 0$. Develop a machine learning method that can predict the following indicator function $\mathbb{1}\,(X_T$ is synchronized)[1] while also output discriminating features that are used for classification, based on the initial trajectory $(X_t)_{0\leq t\leq r}$ that are determined by graph topology and the coupling ϕ and optionally also with statistics of graph G.

Remark. In this setting, we assume that T tends to infinity as the indicator function ($\mathbb{1}\,(X_t$ is eventually synchronized) here

2.2 Our approach: Cellular Automata + Supervised Dictionary Learning.

Binhao: (A brief summary of the methods/algorithms/... we used, will be filled up later)

Models of Coupled Oscillators

Prior works on the study of oscillator and clock synchronization problems by using machine learning have seen little significant progress. The main barrier that slower this line of research is because most traditional oscillator and clock synchronization models assume that each oscillator is continuous-time and continuous-state, so the dynamics of the oscillators quickly become intractable on heterogeneous underlying graphs. This makes it difficult to analyze the complex interplay between network topology and the collective behavior of oscillators. To overcome this difficulty, we discretize the time and state of the model by using cellular automata instead of the usual continuous model. However, although Kuramoto (KM) [2] is a continuous-time system that models synchronization, describes $n \gg 1$ phase oscillators on a real line $\mathbb{R}$, it is worth taking deep research on it as Kuramoto (KM) [2] might be the most well-studied model for the behavior of a large set of coupled oscillators. We discretize the differential equations that defines the (KM) [2] model and analyze the synchronization behavior on it together with the Firefly Cellular Automata (FCA) [7], and the Greenberg-Hastings model (GHM) [8]

[1] Throughout this work, $\mathbb{1}(\cdot)$ denotes the indicator function.

Kuramoto Model (KM)

While there are many continuous-state oscillator models, perhaps one of the most well-studied over the years has been the Kuramoto Model [1, 2, 9, 10]. See Figure 1 for an example of Kuramoto dynamics simulation on a 2D-Lattice.

Consider a graph $G = (V, E)$, and a continuous phase space $\Omega = \mathbb{R}/2\pi\mathbb{Z}$. The evolution of the phase dynamics of the initial phase configuration $X_0 : V \to \Omega$ is determined by the following system of ordinary differential equations

$$\text{(KM)} \qquad \frac{d}{dt}X_t(v) = \omega_v + K \sum_{u \in \mathcal{N}(v)} \sin(X_t(u) - X_t(v)) \qquad \forall\, v \in V \tag{1}$$

where $\mathcal{N}$(v) represents the set of nodes neighboring v in G, ω_v denotes the intrinsic frequency of v, and K denotes the *coupling strength* of the model.

Like the authors in [6], we are interested in the dichotomy between 'synchronization' and 'non-synchronization', and thus will be assuming identical intrinsic frequencies, which can be assumed to be *zero* without loss of generality by using a rotating frame of dynamics. This assumption leads to synchronization being an *absorbing state*, which means that if the phase configuration X_s at an arbitrary time s is synchronized, then for all $t \geq s$, X_t is also synchronized. This closely relates to the concentration lemma (See Section 3.1). Lastly, for our simulation of the Kuramoto model that has been implemented in [11], we use a step size of $h = 0.01$ for each iteration of application of the differential equation (1).

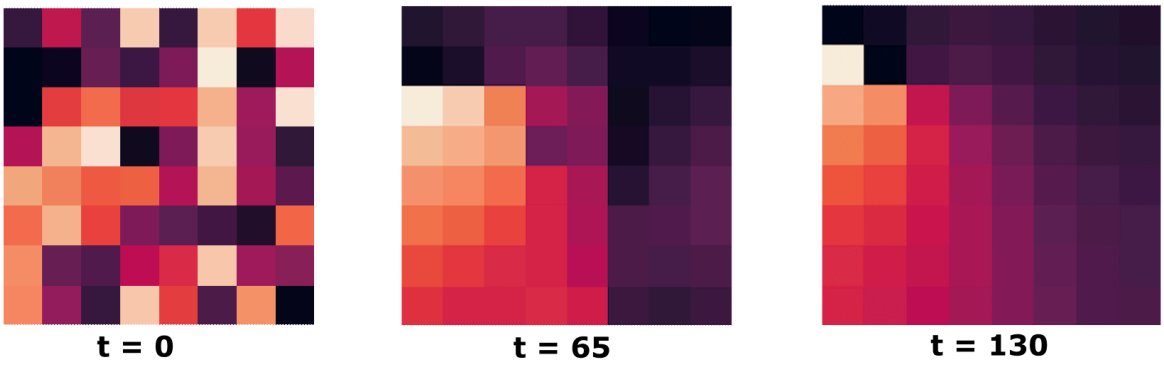


Figure 1: Simulation of the Kuramoto Model on an 8×8 2D-Grid graph (8-by-8 2D Lattice). Each heap map represents the phase configuration X_t of the system at the corresponding iteration t mentioned below the figure.

Firefly Cellular Automata (FCA)

The κ-color Firefly Cellular Automata (FCA) is introduced as a discrete model for inhibitory Pulse Coupled Oscillators (PCOs) [7]. Assume we are having a finite simple graph $G = (V, E)$. The number of nodes is denoted by n. The map X maps the set of vertices to a corresponding state, written as $X : V \to Z/nZ$, where the later is a cyclic group taking order $0 < 1 < \cdots < n - 1$. At any specific time t, the node v takes the state/coloring of $X_t(v)$. Define the neighbors of blinking states as a set $N(b)$, the blinking state as $b(n) = \lfloor \frac{n-1}{2} \rfloor$, then the transition rule of FCA writes:

$$\text{(FCA)} \qquad X_{t+1}(v) = \begin{cases} X_t(v) & X_t(v) \in N(b), X_t(v) > b(n) \\ X_t(v) + 1 & \text{otherwise} \end{cases} \tag{2}$$

Lyu [7] also proposes and proves some necessary and sufficient conditions that guarantees synchronization for every initial configuration when having $n \in \{3, 4, 5, 6\}$. And the key point is that each vertex should be able to blink infinitely times and not pulled from updating. Not surprisingly, the vertex that is being dragged is the one that has higher degree than n. Therefore, all specific nodes that has lower degrees can blink infinitely and not impeded. What naturally follows is that if the maximum degree of the tree is strictly less than n, then it synchronizes.

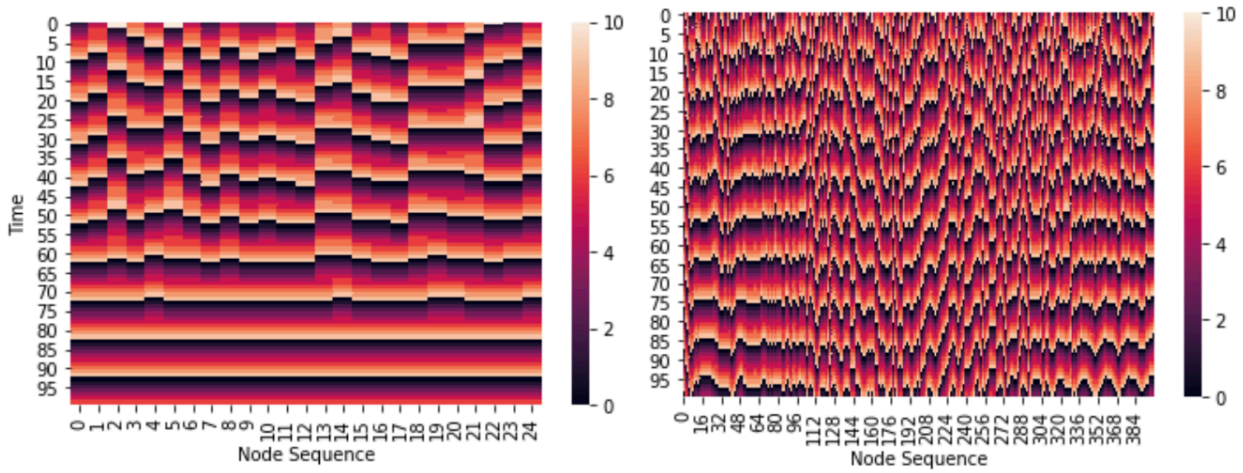


Figure 2: Illustrative example of FCA($\kappa = 10$). Left: A synchronizing example of FCA model on 5×5 2d grid. Right: A non-synchronizing example of FCA model on 20×20 2d grid.

.

Greenberg-Hastings Model (GHM)

Greenberg Hasting Model is a typical cellular automata model that emulates an excitable media. The cells of the system march forward in states with a wave propagating fashion following a refractory period. The detailed updating rules for the states is given as follow:

$$\text{(GHM)} \qquad X_{t+1}(v) = \begin{cases} 0 & \text{if } X_t(v) = 0 \quad \& \quad X_t(u) \neq 1 \, \forall u \in N(v) \\ 1 & \text{if } X_t(v) = 0 \quad \& \quad \exists \, u \in N(v) \implies X_t(u) = 1 \\ X_t(v) + 1 & \text{otherwise} \end{cases} \tag{3}$$

where v is a node in the embedded graph and $N(v)$ the set of its neighbors. To add a refractory period for the system, the state space should be a finite group settled to be $\Omega = \mathbb{Z}/5\mathbb{Z}$ hereafter. Figure 3 illustrates a pair of synchronizing and non-synchronizing examples on a 20-nodes Watts-Strogatz graph constructed by rewiring 4 nearest neighbors with 0.65 probability. Notice that the GHM model has a very clear distinction between synchronizing and non-synchronizing behaviors such that it either synchronizes very fast or get trapped in a wave refractory dynamics forever, leading to non-synchronization.

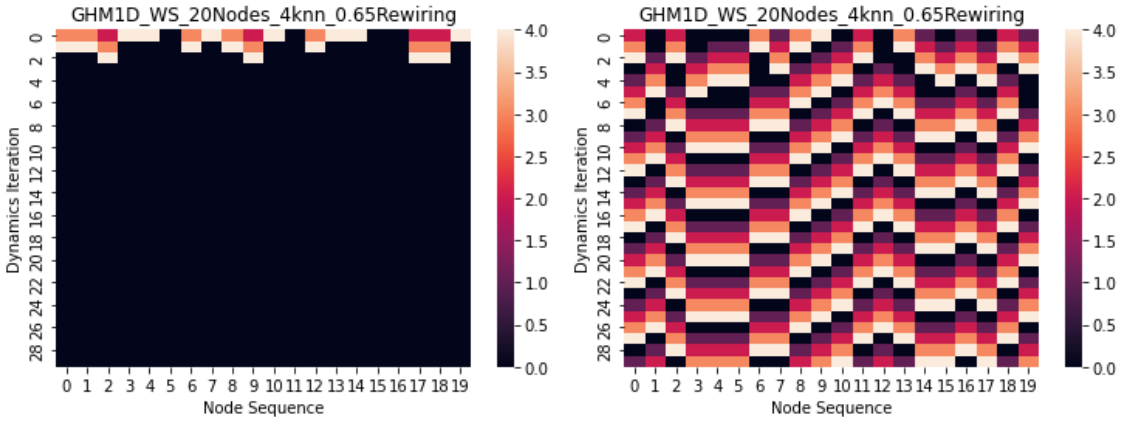


Figure 3: Simulation of the GHM synchronizing and non-synchronizing cases on 20-node Watts Strogatz graph sof 4 nearest neighbors with 0.65 rewiring probability.

Graph Embedding Algorithms

Graph embedding generally refers to the transformation of any given property graphs to a vector, or sometimes a set of several vectors. This kind of embedding algorithms aim to capture the major characteristics of the graph, including graph topology, vertex-to-vertex relationship, and also other relevant information of the graphs, subgraphs, and vertices. More rigorously, we consider any undirected graph $G = (V, E)$, where the cardinality of V, i.e., the number of nodes of this graph is denoted by $|V| = n$, with adjacency matrix denoted by A. We let $D = \mathrm{diag}(d)$ with $d = A\mathbf{1}$ represent the diagonal matrix of node degrees. Every graph we consider in this paper is assumed to be connected, otherwise, we can take a look at each of them separately. We aim at representing the graph in some vector space of low dimension than the original dimension it requires, say $\mathbb{R}^r$ with r much lower than n, while maximally preserving properties like the topological structure and information of the original graph. We say the embedding method maps from the graph to the feature representation of dimension r. More specifically, each node $i \in V$ is represented by some vector $X_i \in \mathbb{R}^r$. We use X to denote a matrix of dimension $n \times r$ whose i-th row X_i corresponds to the embedding of node i. The structure of the graph must be encoded in its representation X, since the two "close" nodes i, j in the graph should correspond to the two "close" vectors X_i, X_j in embedding space.

In this work, we give attempts at predicting the synchronization problem of the coupled oscillator system exclusively using graph features, and no information about the dynamics of each node is included. We hope to see whether these advanced graph embedding algorithms would yield a surprising classification accuracy or not, which might be of interest in verifying that dynamics is a crucial part of the information to ensure a decent classification accuracy. We provide a brief discussion on the main embedding algorithms and techniques , including the spectral embedding [12], node2vec [13], graph2vec [14], for the self-contained purpose.

Spectral Embedding

Spectral embedding [12] is an approach to calculating a nonlinear embedding by using nonlinear dimensionality reduction. Normally this algorithm implements Laplacian Eigenmaps, which finds a low dimensional representation of the data using a spectral decomposition of the graph Laplacian. It forms an affinity matrix given by the specified function and applies spectral decomposition to the corresponding graph laplacian.

The resulting transformation is given by the value of the eigenvectors for each data point. The graph generated can be considered as a discrete approximation of the low dimensional manifold in the high dimensional space. Minimization of a cost function based on the graph ensures that points close to each other on the manifold are mapped close to each other in the low dimensional space, preserving local distances.

Node2Vec

Node2Vec is a an algorithmic framework for representational learning on graphs developed by Aditya Grover and Jure Leskovec [13]. This framework is 2nd order (biased) random walk-based node embedding method, that learns continuous feature representations for nodes in networks. This framework allows the user to map nodes in a graph G to an embedding space of specified dimension r. Generally, the embedding space is of much lower dimensions than the number of nodes in the original graph G. The algorithm tries to preserve the initial structure within the original graph. In other words, it learns a mapping of nodes to a low dimensional feature space that maximizes the likelihood of preserving network neighborhoods of nodes. The notion of a network neighborhood is flexible by using a biased random walk procedure using different sampling techniques like BFS and DFS. Essentially, the nodes which are similar within the graph will yield similar embeddings in the embedding space. These embedding spaces are essentially a vector corresponding to each node in the network.

The process of Node2Vec is fairly simple, it first inputs a graph and extracts a set of random walks from the input graph. The walk can then be represented as a directed sequence of words, where each node represents a word (based on word2vec [15]). The generated random walk is then passed to the skip-gram model. The skip-gram model works on both words and sentences, each node in a random walk can be represented as a word, and the whole walk can be represented as a sentence. The result of the skip-gram model produces an embedding for each node.

Graph2Vec

Graph2Vec [14], an unsupervised framework for learning distributed representations of arbitrary sized graphs, is a modification to the Node2Vec variant. Graph2Vec essentially learns to embed a graphs subgraphs without the need of class labels primarily for graph classification and clustering. Graph2Vec samples nonlinear substructures in form of rooted subgraphs, which helps in yielding similar embeddings for structurally similar graphs.

Dictionary Learning

Dictionary Learning is a machine learning technique that is used to learn interpretable latent structures of complex data sets in order to realize what features of the data the model considers to be the most relevant ones for its task. It consists of two main tasks: (1) Sampling a large number of structured subsets (usually square patches) of a data set, and (2) Applying *non-negative matrix factorization* as described below, to find a set of basis elements that form our *dictionary*. Each element of this learned dictionary describes a latent shape in the image.

Non-negative Matrix Factorization

Non-negative matrix factorization (NMF) [16] is an algorithm devised to learn part-based representations of whole objects rather than focusing on a holistic figure as in Principal Component Analysis (PCA) or Vector Quantization (VQ). So in a task of learning facial objects - while the basis images for PCA would be "eigenfaces" which represent distorted versions of the entire face, for NMF the basis images would be "parts of the whole face" that are combined together in an *additive* fashion to obtain the full image. Sparseness in both basis and encoding of NMF is crucial for part-based representation. NMF can also be applied to semantic analysis of text documents, since this method is able to group semantically-related words into semantic features. However, NMF of single level is not specialized in learning parts from any database with great complexity compared with models with multiple levels of hidden variables.

In NMF, if we have a data matrix $\mathbf{X} \in \mathbb{R}^{d\times n}$ and a rank $r \in \mathbb{N}$, we define a *dictionary* $\mathbf{W} \in \mathbb{R}^{d\times r}$ and an *encoding* $\mathbf{H} \in \mathbb{R}^{r\times n}$, and solve the following optimization problem under the L2-norm (Frobenius) or a more generalized approach under the KL divergence loss

$$\inf_{\mathbf{W}\in\mathbb{R}^{d\times r}_{\geq 0},\mathbf{H}\in\mathbb{R}^{r\times n}_{\geq 0}} \|\mathbf{X}-\mathbf{W}\mathbf{H}\|^2 \tag{4}$$

with non-negativity constraints on both $\mathbf{W}$ and $\mathbf{H}$.

This problem is then optimized using *Alternating Least Squares* (ALS). In our work, we will use this as a technique for supervised dictionary learning to be able to perform the classification task and understand the relevant features that our model is learning, especially because of its high interpretability.

Supervised Dictionary Learning

Supervised Dictionary Learning (SDL) aims to also include the task of classification using these learned dictionaries along with class-labels in a manner that the dictionaries learned not just perform their original task of *reconstruction*, but also perform a *discrimination* task that aids classification.

For our work, since we're looking to predict 'synchronization' or 'non-synchronization' apart from just learning what latent representations the ML algorithms are learning we will be using an SDL task. Let us consider n labeled signals (x_i,y_i) for $i = 1,\cdots,n$, where $x_i \in \mathrm{R}^p$ is a $p-$dimensional signal and $y_i \in \{0,1\}$ is the binary label we're looking to classify our graphs into. Since dictionary learning and the classification task may not necessarily have aligned objectives, we look to optimize the SDL task using a trade-off parameter ξ to control what task is the most important to us, best representation learning or best classification accuracy.

Dictionary Learning by Tensor Factorization

Binhao: Will be filled up if we later choose to use this novel representation in SDL.

3 Methods

3.1 Concentration Principle and Baseline Predictor

A fundamental observation in the literature on coupled oscillator is the *concentration principle*, which is widely used in clock synchronization literature [17–21] as well as multi-agent consensus problems [22–24]. This principle follows from the fact that the couplings in the statement monotonically contract given any initial phase configuration under the half-circle condition toward synchronization.

Concentration Principle [6]: *Let G be an arbitrary connected graph. For the Kuramoto Model (KM) with identical intrinsic frequency and for Firefly Cellular Automata (FCA), the given dynamics on G synchronize if all phases at any*

given time are confined in an open half-circle[2] *in the phase space* Ω*. Furthermore, if all states used in the configuration* X_t *are confined in an open half-circle for any* $t \geq 1$*, then the trajectory on* G *eventually synchronizes.*

This principle follows from the fact that the couplings in the statement monotonically contract given any initial phase configuration under the half-circle condition toward synchronization.

For the Greenberg-Hastings Model, the concentration principle doesn't hold, so we define a phase X_t to be *concentrated* if X_t is synchronized.

Further, we define a *Baseline Predictor*, which helps in evaluating various algorithms applied to oscillator dynamics in this work compared to a "baseline" which either predicts the truth given enough information, and if not, gives a completely uninformed prediction.

Baseline Predictor: *Given the set of dynamics* $(X_t)_{0 \leq t \leq r}$ *and* $T \geq r$*, predict "synchronization" of* X_T *if* X_t *is "concentrated" for any* $1 \leq t \leq r$*. Otherwise, flip a fair coin to choose between the two labels of "synchronization" or "non-synchronization".*

4 Experiment Design

Kuramoto Model (KM)

We have various experimental setups to learn more about the Kuramoto dynamics, and also to apply Non-negative Matrix Factorization (NMF) and Supervised Dictionary Learning (SDL) to learn the dictionary "atoms" and analyze the results obtained by each of these methods. A key step before we could run our experiments and simulations was to generate the underlying graphs on which the dynamics would be run.

Generating the Dynamic Datasets

We generate datasets by sampling sub-graphs from three types of connected networks. The `UCLA` and `Caltech` networks are part of the `FACEBOOK100` dataset [25], where the nodes represent users in the Facebook network of Caltech or UCLA on one day in Fall of 2005, and the edges encode Facebook friendships between these accounts. Furthermore, for our third large network, we generate a large connected graph using the Newman–Watts–Strogatz (NWS) [26] small-world graph generation model with 20000 nodes, 1000 nearest neighbor for each of those nodes and a shortcut edge probability of 0.67. We made the choice to study real-world networks since random graph generation algorithms often fail to capture clusters and communities that the real-world exhibits. For details about each of these networks, see Table 1.

We employ the *motif-sampling* [27] technique for sampling sub-graphs from these large networks, by considering $k-$node sub-graphs on sampling from these networks. We then use one of the Markov-chain Monte Carlo (MCMC) algorithms for motif sampling described in [28] to efficiently and uniformly randomly sample a k-walk from a sparse network-path, and then choose only those that are $k-$paths ($k-$walks with k distinct nodes).

Table 1: Basic Graph Statistics of the three large-scale Graph Networks we use for sampling Subgraphs

Networks	`UCLA`	`Caltech`	`NWS`
Number of Nodes	20467	769	20000
Number of Edges	747613	16656	16702185
Edge Density	0.0036	0.0564	0.0835
Average Clustering Coefficient	0.2149	0.4092	0.3092

Effect of Graph Size on Synchronization

In this subsection, we discuss the effect of increasing node sizes of subgraphs on the synchronization of the Kuramoto dynamics on the Graphs. First, we generated $k-$paths from each of the three networks in Table 1 for $10 \leq k \leq 55$, and then we sampled 100 sub-graphs for each of these $k-$paths obtained, and ran the Kuramoto dynamics on each of them, to compute the ratio of graph configurations on which the Kuramoto dynamics synchronizes. Figure 4 shows the trends

[2] Refers to any arc of length $< \pi$ for the continuous phase space $\Omega = \mathbb{R}/2\pi\mathbb{Z}$ and any interval of $< \kappa/2$ consecutive integers (mod κ) for the discrete phase space $\Omega = \mathbb{Z}/\kappa\mathbb{Z}$. This confinement in an open half-circle is what we define as being "concentrated".

for synchronization as the number of nodes increases for samples drawn from each of the three networks. In [29], the authors observed that multi-cluster synchronization occurs in Kuramoto dynamics applied on an underlying graph, and to understand this better, we also compute the *Average Edge Density* and *Average Clustering Coefficient* [30] of the 100 graphs sampled in each iteration for different number of nodes, which are plotted along with the synchronization ratio in Figure 4. A detailed discussion of our findings and results is in the Results section (See section 5.1).

Non-negative Matrix Factorization (NMF)

In this section, we discuss our findings on applying Non-negative Matrix factorization (See Section 2.2) to the adjacency matrices of graphs separated by their label of whether they host synchronizing Kuramoto dynamics or non-synchronizing Kuramoto dynamics.

For each of the three networks in Table 1, we sample 1600 $50-$node subgraphs and simulate the Kuramoto dynamics on each of these graphs. For our representation matrix (the original feature space) we construct a $N \times k^2$ matrix where there are N rows (here 1600) each containing the "flattened-out" adjacency matrix in the row-major fashion. The dimensions of this flattened-out adjacency matrix is k^2 for a $k-$node subgraph, which leads to a feature space of dimension 1600×2500. We use the transpose of these matrices for the purpose of performing Non-negative Matrix Factorization on them so that the features are the columns of the matrix.

Next, we separate out the cases of synchronization and non-synchronization into two separate matrices while ensuring that the split between the two is mildly imbalanced (20-40% for the minority class) at worst. We perform non-negative matrix factorization on these matrices containing feature spaces for synchronized and non-synchronized cases separately, learning 9 dictionary atoms for each case. In other words we choose to learn the 9 latent basis elements that would best reconstruct the original feature matrix for each case of the synchronizing and non-synchronizing, under the conditions of NMF. A detailed discussion of our findings and results is in the Results section (See section 5.1).

4.1 GHM

4.1.1 GHM Synchronizing Behavior

Now the updating scheme of GHM model has been elaborated, a natural question to ask in this case is that if the synchronizing frequency is closely related to any graph features. Here we perform an experiment on four colleges' real world facebook networks - UCLA, Caltech, Wisconsin, and Harvard. The goal is to obtain a quantitative visualization of the relationship between GHM synchronization and node number of the underlying graphs, as well as their transitivity. We span a node range from 5 to 30 and sample 25 subgraphs for each of those node numbers from the four real world networks. The ratio of synchronizing cases and the average transitivity out of 25 samples for each node are recorded. Detailed results for this experiment will be elaborated on in section 5.3.1 for GHM.

4.1.2 Generating the Dynamic Dataset

In our machine learning prediction experiments, 1-D graphs will be mainly focused on. The networks utilized will be traditional Newman-Watts-Strogatz graphs and the four real world networks mentioned previously. We then carry out NMF and supervised learning techniques on each of those datasets respectively. Table 2 will contain basic information of the generated data on each network types. Those includes number of nodes, average and standard deviation of edge numbers and diameters of the graphs, and (non-)synchronizing numbers.

Datasets	**NWS**	**UCLA26**	**Caltech36**	**Harvard1**	**Wisconsin87**
(k)# nodes	30	30	30	30	30
Avg of # edges	99.06	42.41	98.7	47.72	41.41
Std of # edges	3.73	7.21	16.35	7.97	7.37
Avg diameter	3.36	12.34	4.26	9.11	12.25
Std of diameter	0.48	4.37	0.74	3.02	4.96
Sync.	198	1536	843	1596	1430
Nonsync.	9802	8464	9157	8404	8570

Table 2: Dynamics datasets generated for FCA on 20-node sub-graphs of a large NWS graph and UCLA26. In each dataset, all graphs are connected.

4.1.3 Interpretable Non-Negative Matrix Factorization for GHM

Before performing the classification tasks over the generated datasets, we firstly examine the underlying structures of the datasets via unsupervised dictionary learning on synchronizing and non-synchronizing cases separately. This can be conducted through matrix factorization, specifically with the constraint of non-negative entries in dictionaries and coding matrices for easier interpretation and analysis. For the factorization, it is for convenience to only pick up the adjacency matrix of the graphs as features. So a preliminary data matrix $X_{Adj} \in R^{N\times k^2}$ could be filtered out for factorization. In this case, N is the number of samples in each case, where the $k \times k$ adjacency matrix is flattened out into $1 \times k^2$. A general formulation of the NMF is written as:

$$X^{d\times N} = W^{d\times r} \cdot H^{r\times N} \tag{5}$$

In our case:

$$(X_{Adj}.T)^{k^2\times N} = W^{k^2\times r} \cdot H^{r\times N} \tag{6}$$

We sample 16 most important latent factors that best reconstruct the data matrix and visualize them in $k \times k$ squares along with their graphs. Results will be discussed in section 5.3.2

4.1.4 Interpretable Supervised Learning for GHM

Next, we predict synchronization using the datasets with different combinations of features using several supervised learning approaches and compare their performances. The methods we are testing here includes SVM, Random Forest, SNMF, and BCD. Although dictionary learning and classification tasks are two distinct objectives and need a trade-off, SNMF still provides very nice interpretability due to the NMF part. The BCD model will be considered seprately for feature based and filter based. The distinction lies in whether to use h_i extracted from the coding matrix H (feature based) or filtered input $\mathbf{W}^T\mathbf{x}_i$ in the classification process. For the supervised learning, we mixed the synchronizing and non-synchronizing cases together in the training process. The features combinations are basic graph features only, adjacency matrix only, basic graph features together with adjacency matrix, dynamics only, and dynamics with all other features. Detailed training performance will be discussed in section 5.3.3. Further exploration would include color encoded adjacency matrix, which is essentially a 3-D tensor. For each node in the adjacency matrix, it represents an edge in the actual graph and thus we pile up color difference of the nodes connected by the edges to form the thrid dimension. In this case, a tensor factorization based supervised dictionary learning will be conducted for classification.

5 Results

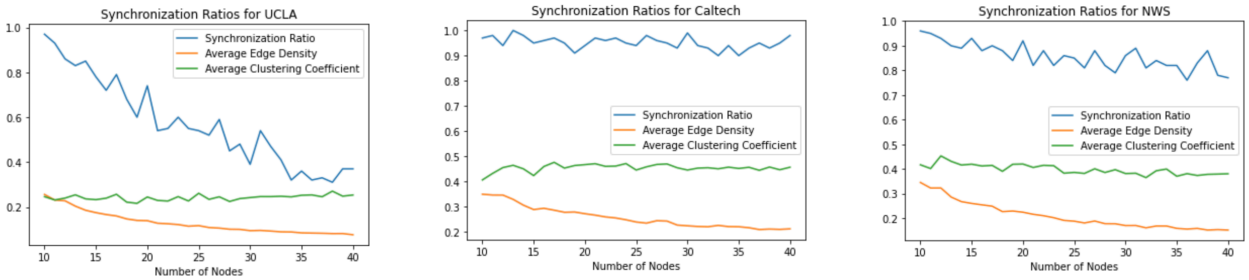
5.1 Kuramoto

Effect of Sub-Graph Size on the Likelihood of Synchronization

The experiment designed in 4 to see the effect of graph sizes on synchronization helped us better understand the three networks and their sampled sub-graphs that we deal with in the case of Kuramoto dynamics. There are two common trends that we notice in all three plots from figure 4 - (1) The Average Clustering Coefficient for all sub-graph sizes in the three networks remains roughly constant, and just a little above the average clustering coefficient of the parent networks as described in Table 1, and (2) The Average Edge Density for sub-graphs sampled from each network shows an asymptotically decreasing trend as the number of nodes in the sub-graphs increase.

In Figure 4a, we notice that the as the number of nodes increases ,the percentage of sampled sub-graphs on which dynamics synchronize tends to decrease. The `UCLA` network is quite sparse compares to the other two networks (See Edge Density in Table 1) with roughly a 0.35% edge density compared to a complete graph with the same number of nodes. Furthermore, even the Average Clustering Coefficient remains low around 0.25, which could be a factor playing a role in this trend. It is known that Kuramoto tends towards a non-synchronizing pattern on sparse networks and this is in accordance with what we see in this plot.

In Figure 4b, we notice that the as the number of nodes increases ,the percentage of sampled sub-graphs on which dynamics synchronize tends to remain roughly the same (above 90%). This can be explained by the fact that the `Caltech` network is pretty dense with an Edge Density of 5.5% and also has a much higher average clustering coefficient compared to the other two networks. The trend seen for sub-graphs sampled from the `NWS` network roughly lies somewhere between `UCLA` and `Caltech`, with a decreasing synchronization trend, which however seems to level out as the number of nodes increases. The synchronization ratios are not as high as in `Caltech` primarily because even though it has a higher Edge Density (See Table 1), it has a lower Average Clustering Coefficient, which would lead to less local synchronization, perhaps hindering a very high level of global synchronization.



(a) `UCLA` Synchronization Ratio (b) `Caltech` Synchronization Ratio (c) `NWS` Synchronization Ratio

Figure 4: Plots showing the *Average Synchronization Ratio* for the 100 subgraphs sampled from (4a) `UCLA`, (4b) `Caltech` and (4c) `NWS` networks with the number of nodes ranging from 10 to 40. Also superimposed on the plot is the *Average Edge Density* and the *Average Clustering Coefficient* [30] for the 100 graphs sampled at each new iteration.

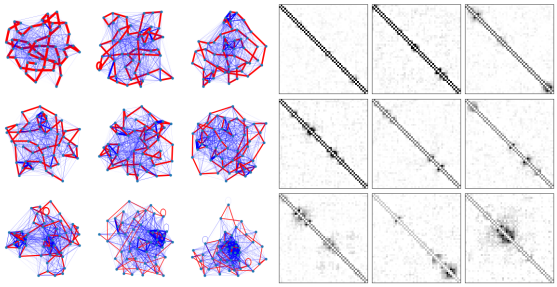
Non-negative Matrix Factorization (NMF)

In this section, we dive deep into our observations from the setup for Non-negative Matrix Factorization experiments described in 4.

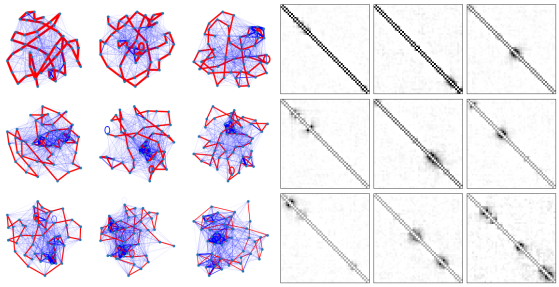
Figures 5, 6, and 7 show the dictionaries learned from our sampling of the `UCLA` network using NMF, for each of the two cases of 'synchronizing' and 'non-synchronizing' as well as the corresponding graphical representation of these dictionary elements. Furthermore, to analytically understand the differences in the dictionary atoms for the synchronized and non-synchronized cases better, we use a box plot that denotes the distribution of the L_1 matrix norms [31] of each of these atoms in an easy to distinguish way.

We now summarize our key observations from the NMF dictionaries for Kuramoto dynamics on subgraphs for the three networks.

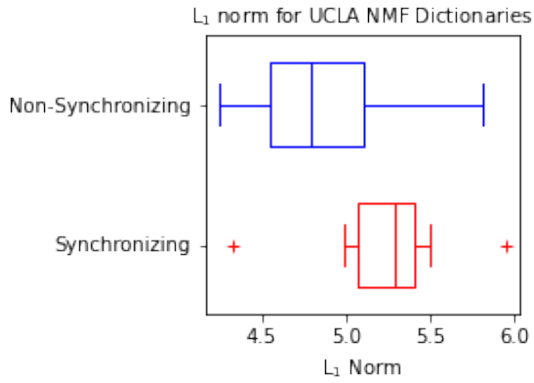
1. `UCLA` Dictionaries
   - The dictionaries for both the synchronizing and non-synchronizing cases seem to have "hubs" lying along the Hamiltonian path. Moreover, overall the dictionary atoms indicate that the feature space contains sparse graphs, which is in accordance with our understanding of the `UCLA` network from Table 1.
   - From Figures 5a and 5b, we notice that outside the Hamiltonian path, most dictionaries in the synchronizing cases have a higher edge density compared to those in the non-synchronizing cases. This can also be observed from the graph plots of these dictionaries on careful observation (the graphs for synchronizing cases are denser).
   - To confirm this observation, we compute the L_1 matrix norm [31] of the synchronizing dictionaries and non-synchronizing dictionaries and summarize the statistics in the box plot (See Figure 5c). Clearly, the median L_1 norm for synchronizing cases lies higher than not just the median but even the 75^{th} norm for the non-synchronizing cases, which strengthens our claim.
2. `Caltech` Dictionaries
   - The dictionaries for this network don't seem to possess any significant hubs like seen in the case of `UCLA`. However, both the synchronizing and non-synchronizing cases seem to have one or more "pivot nodes" lying along the Hamiltonian path, which have a connection with a lot of other nodes in the network (See figures 6a and 6b). Further, we notice that the dictionary atoms indicate that the feature space contains denser graphs compared to those that we saw in `UCLA`, which is again in accordance with our understanding of the `Caltech` network from Table 1.
   - From Figures 6a and 6b, unlike in the case of `UCLA`, it is harder to notice a significant difference in the dictionary atoms or even their graphical representation. Both cases produce dense dictionaries, and contain the pivot nodes, which however arguably seem more noticeable in the case of the Synchronizing cases in 6a and slightly more dense, which we confirm analytically next.
   - To confirm whether the graphs for the Synchronizing cases really are denser, we again use a similar box plot that summarizes the L_1 matrix norms [31] of the synchronizing and non-synchronizing dictionaries. (See Figure 6c). Once again, the median L_1 norm for synchronizing cases lies higher than not just the



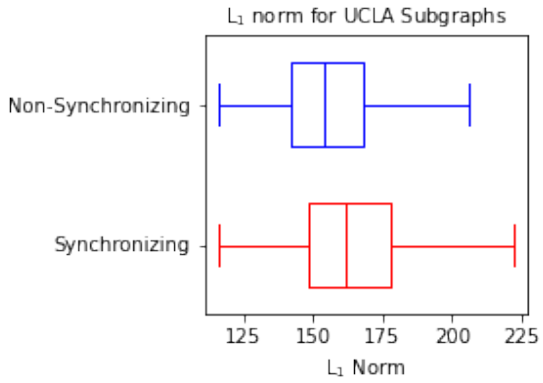
(a) UCLA Synchronization Dictionaries



(b) UCLA Non-Synchronization Dictionaries



(c) UCLA Dictionaries L_1 norm Box Plot



(d) UCLA Subgraphs L_1 norm Box Plot

Figure 5: Figures 5a and 5b show the Dictionary atoms learned using Non-negative Matrix Factorization and their Graphical Representations for UCLA network. Figure 5c shows a box plot comparing the L_1 matrix norm of the dictionaries learned in each of these cases.

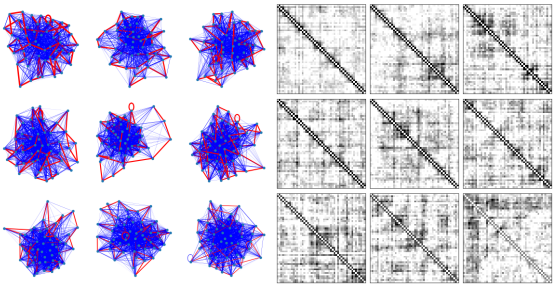
median but even the 75th norm for the non-synchronizing cases, which strengthens our claim, and is almost as high as the maximum norm for the non-synchronizing cases.

3. NWS Dictionaries

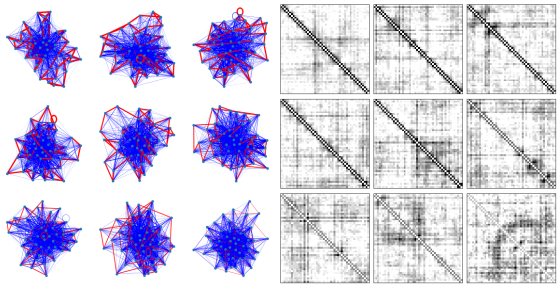
- Similar to the case of UCLA, the dictionaries for NWS in both the synchronizing and non-synchronizing cases seem to have "hubs" lying along the Hamiltonian path. Overall, the dictionary atoms indicate that the feature space contains denser graphs than UCLA, but slightly sparser compared to Caltech, which once again is in accordance with our understanding of these networks from Table 1.

- From Figures 7a and 7b, we notice that outside the Hamiltonian path, most dictionaries in both the synchronizing and non-synchronizing cases have seem to have similar edge density. Moreover, the hubs in the synchronization case as well as the non-synchronizing case seem just as dense as each other. This can be observed from the graph plots of these dictionaries on careful observation. We thus turn to an analytical exploration of this to understand it better.

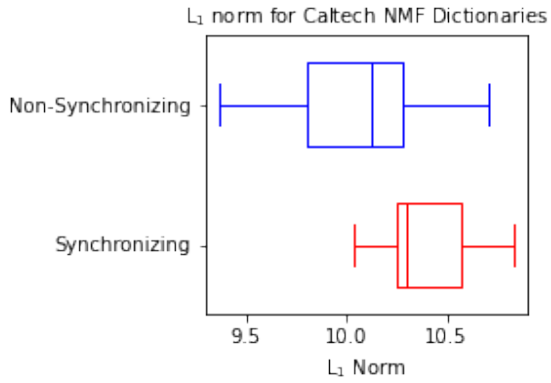
- To confirm this observation, we compute the L_1 matrix norm [31] of the synchronizing dictionaries and non-synchronizing dictionaries and summarize the statistics in the box plot (See Figure 7c). Even after a couple of outliers being neglected for each case, the median L_1 norm for synchronizing cases lies marginally higher than not just the median but even the 75th norm for the non-synchronizing cases, which strengthens our claim.



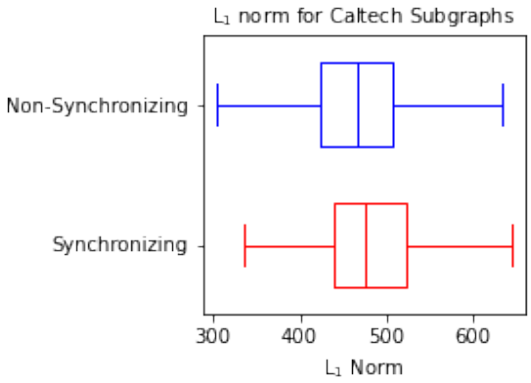
(a) `Caltech` Synchronization Dictionaries



(b) `Caltech` Non-Synchronization Dictionaries



(c) `Caltech` Dictionaries L_1 norm Boxplot



(d) `Caltech` Subgraphs L_1 norm Box Plot

Figure 6: Figures 6a and 6b show the Dictionary atoms learned using Non-negative Matrix Factorization and their Graphical Representations for `Caltech` network. Figure 6c shows a box plot comparing the L_1 matrix norm of the dictionaries learned in each of these cases.

5.2 FCA

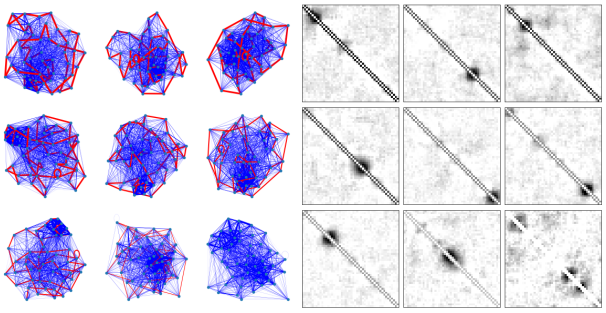
5.2.1 Experimental Design

Generating the dynamics datasets

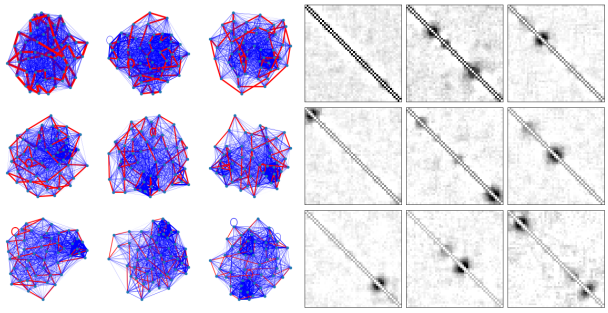
Our team generate datasets of FCA trajectories with synchronization labels on various initial data of graph structure and coloring for the task of applying classification algorithm to build synchronization predictor as well as extracting features and learning interpretable classification protocols. More specifically, two different datasets are generated where each data point is given an underlying connected graph and a set of randomly simulated initial coloring to each node. Moreover, all data points are labeled as $\mathbb{1}(X_t$ synchronizes) under FCA($\kappa = 8$) and consist of different features based on different settings.

In both datasets, we generate all underlying graphs by sampling a Hamiltonian path of length 20 from a large network as mentioned in section ???? so that we could obtain a fixed set of node ordering for each graph. This helps us better encode the adjacency matrix in a deterministic way as the feature input to understand how the graph structure is influencing long-term dynamics. For one of the datasets, all the underlying graphs are sampled from a large Newman-Watts-Strogatz (NWS) graph [26] of 20000 nodes with nearest neighbors of 1000 and shortcut edge probability of 0.7. Since the random graph generator might not be able to capture all potential behaviors observed in real-life networks such as hub nodes or clustering, it would be interesting to compare the synchronizing behaviors of `NWS` sub-networks with some other real-life sub-networks. Therefore, the underlying graphs in the other dataset are sampled from `UCLA26` network (see description in section ????).

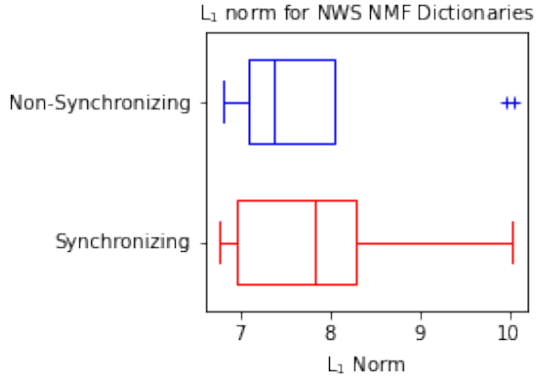
In comparison, sub-networks from `NWS` have higher density and greater edge variance than sub-networks from `UCLA26` according to Table 3. As for the largest shortest path distance, sub-networks from `UCLA26` have a larger average and variance of diameter than sub-networks from `NWS`. We sampled 10000 underlying graph structures for each dataset, and since FCA tends to synchronize on any graph, around 70% of sampled sub-networks are labeled as synchronizing as shown in Table 3.



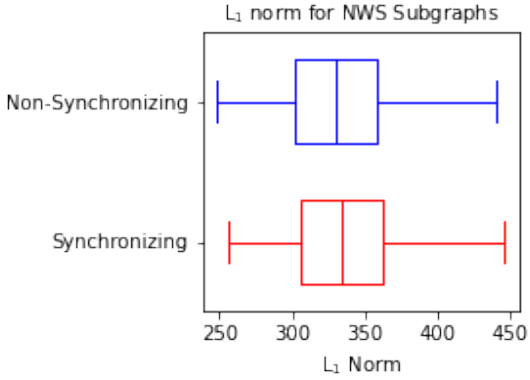
(a) NWS Synchronization Dictionaries



(b) NWS Non-Synchronization Dictionaries



(c) NWS Dictionaries L_1 norm Box Plot



(d) NWS Subgraphs L_1 norm Box Plot

Figure 7: Figures 7a and 7b show the Dictionary atoms learned using Non-negative Matrix Factorization and their Graphical Representations for NWS network. Figure 7c shows a box plot comparing the L_1 matrix norm of the dictionaries learned in each of these cases.

Datasets	NWS	UCLA26
# nodes	20	20
Avg of # edges	41.02	25.93
Std of # edges	9.53	4.75
Avg diameter	5.28	11.18
Std of diameter	1.22	3.47
r (training iter)	50	50
T (prediction iter)	200	200
# Sync.	7548	7553
# Nonsync.	2452	2447

Table 3: Dynamics datasets generated for FCA on 20-node sub-graphs of a large NWS graph and UCLA26. In each dataset, all graphs are connected.

Black-box model on predicting synchronization

As a black-box model, random forest algorithm (RF) is employed to solve the problem of predicting whether a given graph and initial coloring for FCA will tend to a synchronizing or non-synchronizing trajectory. We are interested to see what types of information help the algorithm achieve the best performance by feeding in different feature inputs, compared with the baseline model (See Section 3.1). In total, our team set up 8 different feature inputs listed as below:

1. Dynamics: Define the feature matrix as $D^{m\times nr}$, where m is the total number of underlying graphs, n is the number of nodes of the underlying graph, and r is the number of training iteration. The i^{th} row of D corresponds to the set of coloring of the i^{th} underlying graph from time 1 to time r. The j^{th} column of D corresponds to the coloring of $j \mod n$ node at time $\lceil \frac{j}{n} \rceil$.

2. Width of dynamics: Let the feature matrix be $D^{m\times r}_{width}$. The i^{th} row of D_{width} corresponds to the coloring width of the i^{th} underlying graph from time 1 to time r. The j^{th} column of D_{width} corresponds to the coloring width at time j.
3. Shifted dynamics: $D_{shifted}$ TODO

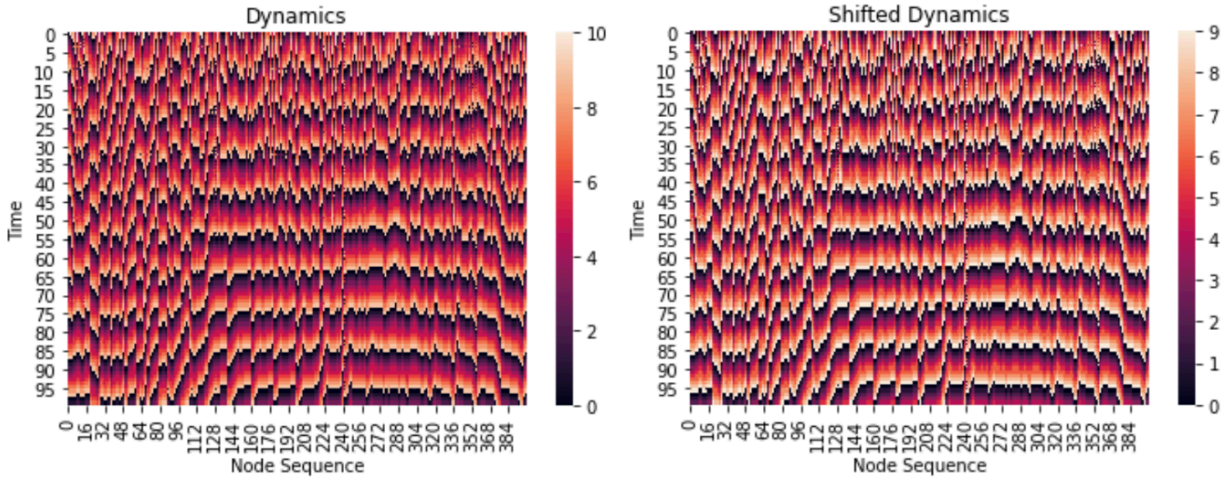


Figure 8

.

4. Dynamics + Adjacency matrix: Let the feature matrix $\begin{bmatrix} D^{m\times nr} & A^{m\times n^2}_{adj} \end{bmatrix}$ be of size $m \times (nr + n^2)$. In the i^{th} row of the feature matrix, the first nr elements correspond to the set of coloring of the i^{th} underlying graph from time 1 to time r while the remaining n^2 elements correspond to the vectorized $n \times n$ adjacency matrix of the i^{th} underlying graph. The vectorization of the adjacency matrix is done by flattening the matrix in a row-major order.
5. Dynamics + Graph features: Let the feature matrix $\begin{bmatrix} D^{m\times nr} & A^{m\times(9+6n)} \end{bmatrix}$ be of size $m\times(nr+(9+6n))$. In the i^{th} row of the feature matrix, the first nr elements correspond to the set of coloring of the i^{th} underlying graph from time 1 to time r while the remaining $9 + 6n$ elements correspond to the 9 different graph-level features and 6 different node-level features of each node of the i^{th} underlying graph as indicated in Table 4.
6. Dynamics + node2vec: Let the feature matrix $\begin{bmatrix} D^{m\times nr} & A^{m\times 24n}_{node2vec} \end{bmatrix}$ be of size $m \times (nr + 24n)$. In the i^{th} row of the feature matrix, the first nr elements correspond to the set of coloring of the i^{th} underlying graph from time 1 to time r while the remaining $24n$ elements correspond to the node features of dimension 24 extracted by node2vec of the i^{th} underlying graph.
7. Dynamics + graph2vec: Let the feature matrix $\begin{bmatrix} D^{m\times nr} & A^{m\times 16}_{graph2vec} \end{bmatrix}$ be of size $m \times (nr + 16)$. In the i^{th} row of the feature matrix, the first nr elements correspond to the set of coloring of the i^{th} underlying graph from time 1 to time r while the remaining 16 elements correspond to the graph features of dimension 16 extracted by graph2vec of the i^{th} underlying graph.
8. Dynamics + spectral embedding TODO

	Features
Graph-level	# edges, # nodes, min degree, max degree, diameter, degree assortativity coefficient, # cliques, average clustering coefficient, density
Node-level	Degree centrality, eigenvector centrality, betweenness centrality, closeness centrality, clustering coefficient, degree

Table 4: Graph features calculated for sub-graphs of a large `NWS` graph and `UCLA26`. The graph-level feature is applied to each underlying graph. The node-level feature is applied to each node of each underlying graph.

Since our datasets have unbalanced classes, we apply undersampling to both datasets before training RF model under each setting, avoiding overfitting in the synchronizing class. We use accuracy score and confusion matrix to evaluate RF models under different settings to identify the most informative feature space about FCA synchronization.

Nonnegative matrix factorization to compare synchronizing and nonsynchronizing pairs

Nonnegative matrix factorization (NMF) is used in this paper to draw insights about key features on the network structures and dynamics of synchronizing and non-synchronizing pairs of our initial data. Here, we apply NMF separately on feature matrices of synchronizing and non-synchronizing pairs, which are generated according to the fifth setting in section 5.2.1. For each feature matrix, the algorithm learns a set of four latent basis elements.

Supervised Dictionary Learning on predicting synchronization

We also apply Supervised Dictionary Learning (SDL) to learn interpretible classification protocols. Again, aiming to extract the most distinctive features for synchronization, we experiment with different feature matrices on SDL, including dynamics (the second setting in section 5.2.1), shifted dynamics (the fourth setting in section 5.2.1), dynamics + adjacency matrix (the fifth setting in section 5.2.1).

Furthermore, we define a coloring adjacency matrix at time t (ψ_t) as below:

Given $G = (V, E)$

$$\text{(i j -th entry)}\Psi_t(i,j) = \begin{cases} 0 & \text{if } v_i, v_j \in V \text{ and } (v_i, v_j) \notin E \\ X_t(v_i) - X_t(v_j) & X_t(v_i) \geq X_t(v_j) \\ X_t(v_i) - X_t(v_j) + \kappa & X_t(v_i) < X_t(v_j) \end{cases} \tag{7}$$

The coloring adjacency matrix retains information about the graph structure as well as how the configuration propogates over time by encoding the color difference between the two vertices of each edge on an adjacency matrix. To be fed into SDL, we denote the feature matrix as $A_{coloring}^{m \times n^2(r+1)}$, where the first n^2 columns are the vectorized actual adjacency matrices of the underlying graphs, and the remaining $n^2 r$ columns are the vectorized coloring adjacency matrix from time 1 to time r.

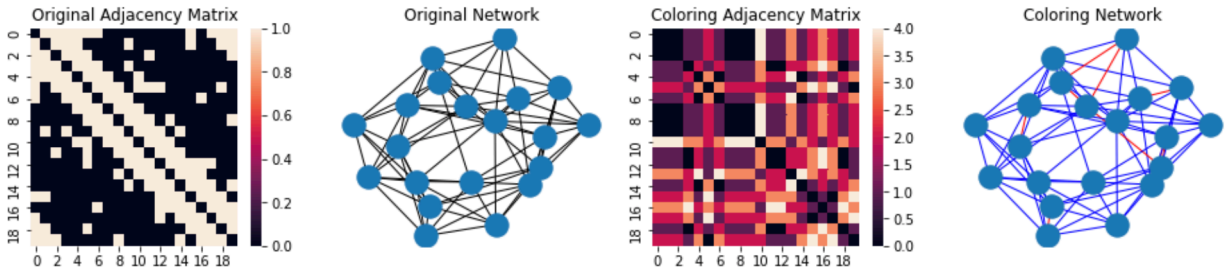


Figure 9

.

SDL is trained on the four different settings as mentioned above to learn eight different dictionary elements, and the model performance like accuracy score is compared with the baseline model and the black-box model for further evaluation.

5.2.2 Results

By feeding in different feature matrices to RF models, we observe variation in classification accuracy. In Figure 11 and 10, it is not hard to see that the task of predicting synchronization based on sub-graphs of `UCLA26` is a slightly more difficult task than that based on sub-graphs of a large `NWS` graph, because `NWS` has higher accuracy than `UCLA26` when using different feature inputs. What's more, `NWS` even has a higher baseline accuracy of 8% than `UCLA26`, indicating that initial FCA dynamics of `NWS` sub-graphs is easier to concentrate This matches our expectation as FCA is known to synchronize on dense graphs, which is the case of `NWS`, while less prone to synchronize on sparse networks, such as sub-graphs sampled from `UCLA26`.

Since the baseline model is based upon half-circle concentration lemma, improvement of the prediction accuracy with different dynamics feature matrices demonstrates that dynamics provides more information about synchronization

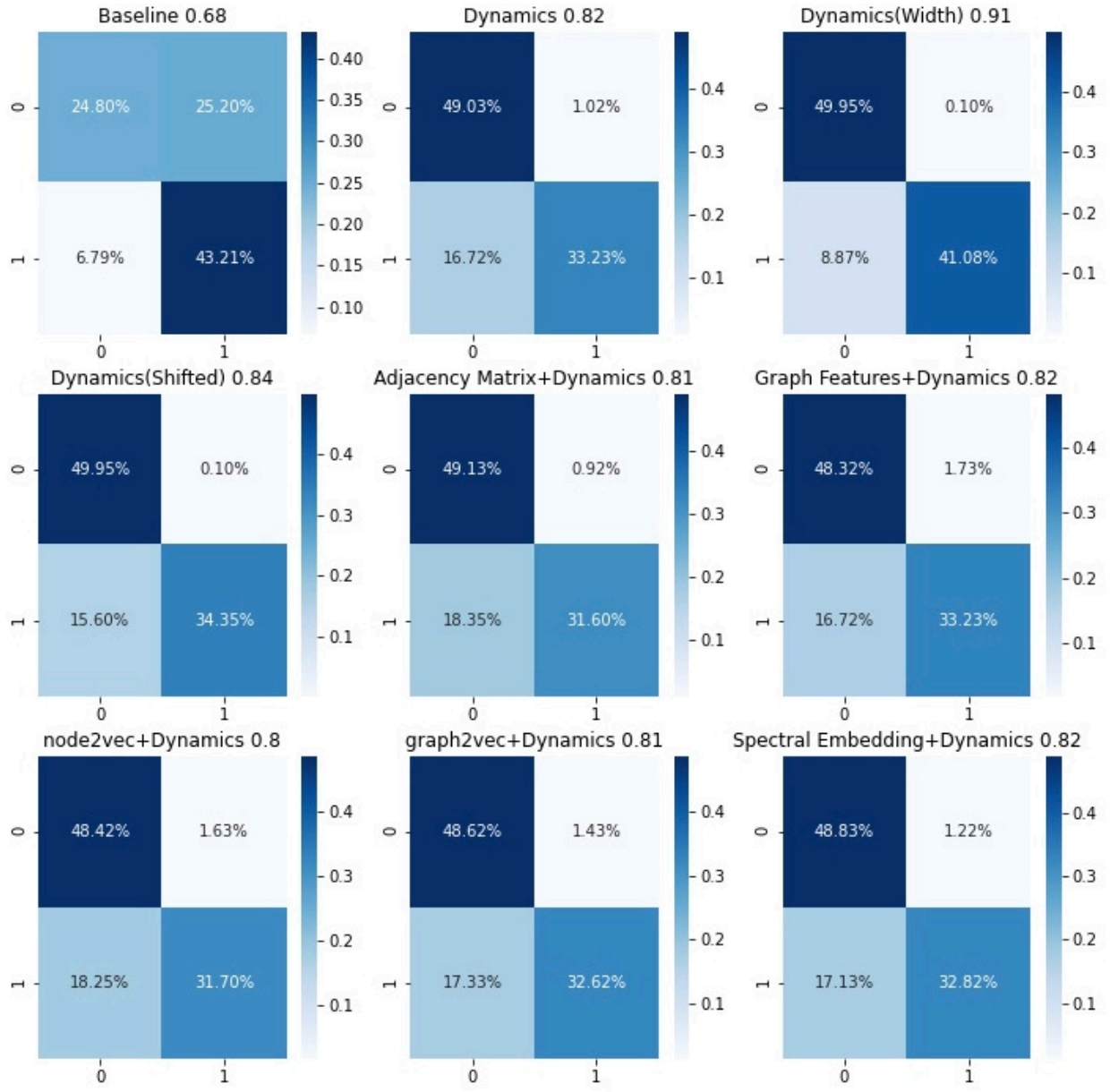


Figure 10: RF performance on different feature inputs for sub-graphs of a large NWS network. The number in the title is the accuracy score. For each confusion matrix, the x-axis is the predicted value and the y-axis is the actual value. If it is labeled as 0, then it is non-synchronizing. If it is labeled as 1, then it is synchronizing. The baseline model is defined in Section 3.1

.

than the half-circle concentration lemma. By incorporating dynamics into RF models, the accuracy scores increases by more than 10% in both `UCLA26` and `NWS` datasets. Moreover, the accuracy of using width of dynamics is 9% higher than just using the dynamics dataset as shown in Figure 10, implying that the RF model could not directly infer the width of dynamics from the dynamics dataset. Meanwhile, the accuracy of using shifted dynamics dataset is 5% higher than just using the dynamics dataset as shown in Figure 11. Similar phenomenon could also be discovered in Figure 10, indicating that the RF model could handle incremental transitions but having troubles wrapping around, and as a result, inferring the updating rules of FCA is a difficult task for RF.

There is barely any improvement on RF performance after combining graph structure information with dynamics. In Figure 10 and in Figure 11, applying adjacency matrix, graph features, node2vec embedding or graph2vec embedding

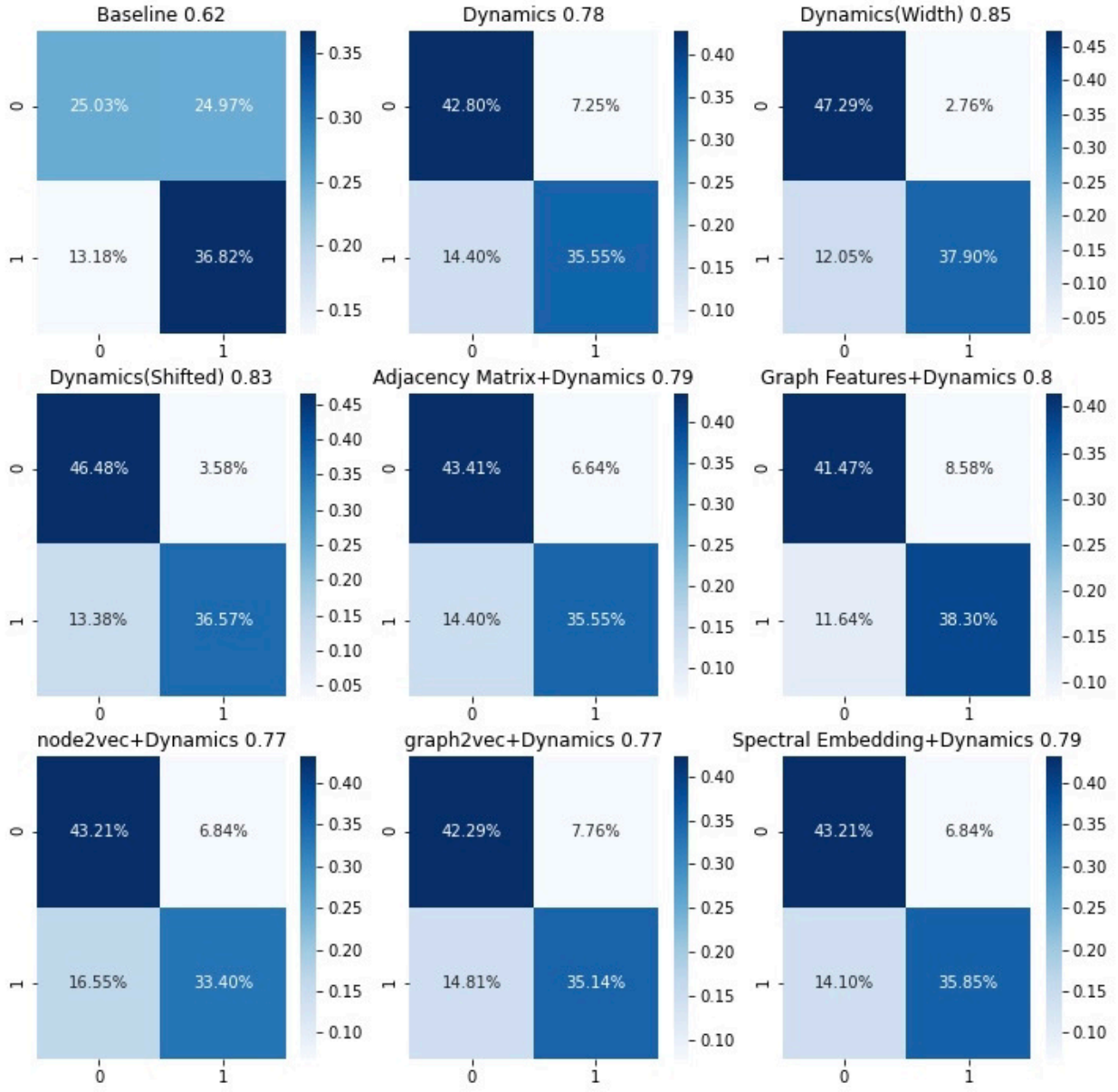


Figure 11: RF performance on different feature inputs for sub-graphs of UCLA26. The number in the title is the accuracy score. For each confusion matrix, the x-axis is the predicted value and the y-axis is the actual value. If it is labeled as 0, then it is non-synchronizing. If it is labeled as 1, then it is synchronizing. The baseline model is defined in 3.1

affects the accuracy score within solely 2% using the accuracy score training upon the dynamics dataset as a reference. It is highly likely that RF model is mainly utilizing dynamics information for classification instead of learning from graph topology. Changing the graph embedding methods has little effect on model performances.

Figure 12 and Figure 13 allow us to compare synchronizing and non-synchronizing dictionary elements from NMF learned on dynamics and adjacency matrix. For the synchronizing class, dynamics appear to be stable and there is no apparent disturbance as time propagates. More specifically, in the beginning of the training iteration, the colors in the heatmap seems to be blanded together especially in the fourth dicionary element learned from `NWS` sub-networks. This might result from the attempt of the dynamical system to synchronizing at the early stage, where dominating nodes constantly pulling its nearby neighbors. At the end of the training iteration, the coloring for each node synchronizes with the same color in Figure 12. However, not all colorings of nodes synchronize in Figure 13, but still the majority

of node coloring are the same, following the half-circle concentration lemma. In contrast, in the non-synchronizing class, there is a wave-like pattern in the dynamics heatmap, and especially in Figure 13, dictionary elements appear to be wiggling across all nodes, illustrating that there exists variation in node coloring for non-synchronizing cases and each node cannot affect its neighbors' colors as easily as in the synchronizing class. It is unexpected to see a uniform node coloring at the bottom of the first and fourth non-synchronizing heatmap in Figure 12.

Taking a look at the network structures and degree distributions from the dictionary elements help us understand the relationship between graph structure and synchronization. In Figure 12, we can observe density difference between synchronizing pairs and non-synchronizing pairs, particularly from the first two synchronizing elements and the second and thrid non-synchronizing elements. According to the degree distribution in Figure 12, the graph topology with lower degree variance and more hub nodes is easier to synchronize. However, all patterns mentioned above is not as obvious in dictionary elements learned from the `UCLA26` dataset, which is probably due to the insignificant density variance in `UCLA26` dataset according to Table 3. All network visualizations seem to be a sparse path with no apparent hub nodes or clustering.

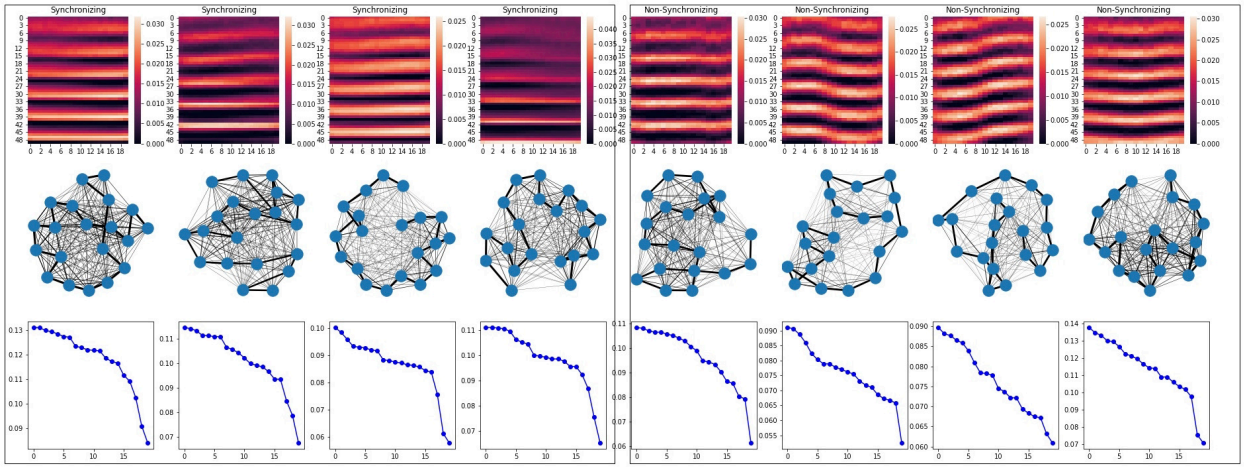


Figure 12: The dictionary elements learned from NMF for synchronizing pairs and non-synchronizing pairs of subgraphs from a large NWS network. The left four dictionary elements are from the synchronizing pairs, and the right four dictionary elements are from the non-synchronizing pairs. Each column is a dictionary element. The heatmaps are the dictionary elements learned from dynamics, where x-axis is the ordering of node and the y-axis is time. The networks are the dictionary elements learned from adjacency matrices, where the weights of edges are set to be the values in the dictionary elements. The degree distribution corresponds to the network above it.

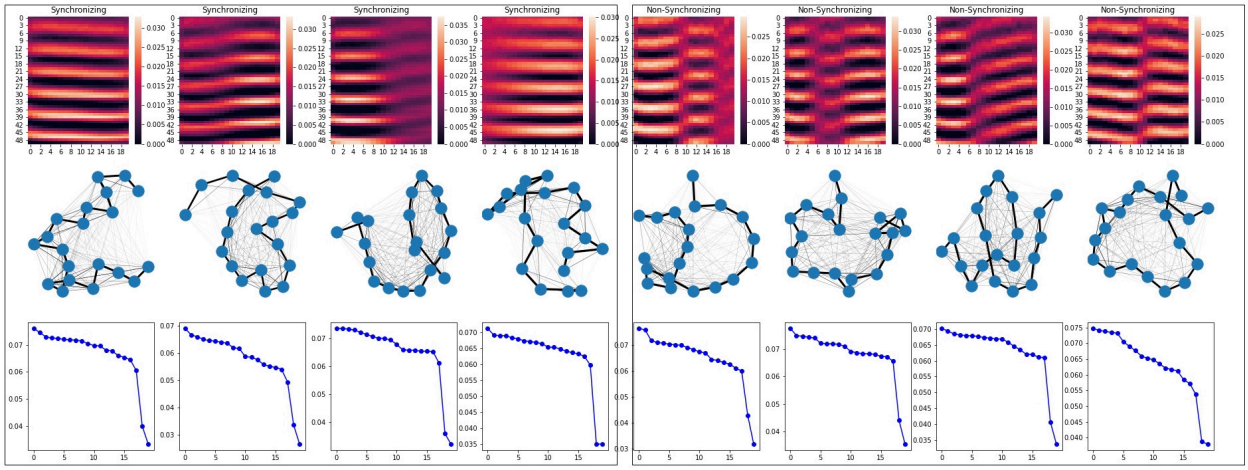


Figure 13: The dictionary elements learned from NMF for synchronizing pairs and non-synchronizing pairs of subgraphs from UCLA26.

Our team move on to examine SDL classification accuracy on different feature inputs, in comparison to the baseline model and RF in Table 5 and Table 6. Since the baseline model solely depends on the half-circle concentration lemma, its accuracy score remains the same regardless of the feature matrices. As mentioned earlier, incorporating graph topology barely affects RF performance, and thus it is not surprising that using coloring adjacency matrix on RF gives similar accuracy score from feeding (shifted) dynamics and dynamics + adjacency matrix into RF. The performance of SDL is similar to the baseline model, but much worse than the black-box model if it is trained on dynamics information, according to Table 5 and Table 6. This justified that the dictionary elements learn from SDL are unable to differentiate synchronizing and non-synchronizing behaviors solely based on dynamics dataset, but the black box model has the ability to distinguish the two classes. The SDL model accuracy even decrease by around 18% if additional graph information is added to the dynamics dataset in Table 6, while dropping by around 7% in Table 5, demonstrating that adding more graph structure information is unable to improve SDL performance on dynamics dataset, but sometimes even worsening its performance. Nevertheless, in both Table 5 and Table 6, using coloring adjacency matrix could outperform the baseline model by more than 13%, showing that SDL is able to learn distinguished features from the coloring adjacency matrix. It is expected that SDL is unable to achieve higher accuracy score than the random forest model as SDL has to compromise its classification performance for dictionary learning. Even though feeding coloring adjacency matrix into RF gives similar output from the dynamics dataset, SDL is able to learn the most distinguishing basis elements from coloring adjacency matrix and achieve the highest accuracy score among the four different settings.

Feature	**Baseline**	**RF**	**SDL**
Dynamics	0.68	0.82	0.68
Shifted dynamics	0.68	0.84	0.68
Dynamics + adjacency matrix	0.68	0.81	0.61
Coloring adjacency matrix	0.68	0.83	0.80

Table 5: The classification accuracy of the baseline model, RF model, and SDL model based upon different feature inputs from the dataset of NWS

Feature	**Baseline**	**RF**	**SDL**
Dynamics	0.61	0.78	0.63
Shifted dynamics	0.61	0.83	0.68
Dynamics + adjacency matrix	0.61	0.79	0.50
Coloring adjacency matrix	0.61	0.82	0.74

Table 6: The classification accuracy of the baseline model, RF model, and SDL model based upon different feature inputs from the dataset of UCLA26

Given the outstanding performance of SDL on coloring adjacency matrix, it is worth examining the dictionary elements learned for insights. In Figure 14, the two dictionary elements with highest regression coefficients appear to be densely connected, with no isolated nodes or any loosely connected nodes, while in contrast, the dictionary element with the lowest regression coefficient appears to a sparse network. What's more, for dictionary elements with positive regression coefficients, the networks gradually progress to synchronize with time, and almost all nodes synchronize at the end of the training iteration. However, for dictionary elements with negative regression coefficients, the synchronizing part of the network increase only from time 0 to time 10 but stopped progressing later on, and at the end of the training iteration, only a small fraction of the network is synchronized. The dictionary elements in the middle lie between completely synchronizing cases and completely non-synchronizing cases, and they all contain some loosely connected nodes and a large cluster of nodes. With decreasing regression coefficients, the density of clusters in dictionary elements decrease simultaneously. This observation suggests that having loosely connected nodes in the network decreases the likelihood of the network to synchronize, but having a heavily connected cluster in the network increases the synchronizing likelihood.

Similar to the dictionary learned from `NWS` samples, in Figure 15, there is significant density different between the elements with the highest and the lowest regression coefficients. The fifth to the seventh elements all have a big cluster and some loosely connected nodes, where the size of the cluster decays as the regression coefficient diminished, corresponding to the findings in `NWS` samples. Nevertheless, the dictionary elements learned from samples in `UCLA26` are different from those in `NWS`, in a way that the density of each dictionary element is not decreasing with decreasing regression coefficients. This is probably due to the small density variance and sparsity of the samples in `UCLA26`,

resulting in a much harder classification task than NWS samples. More specifically, the second dictionary element seems to be denser than the fourth element, but sparser than the third element, even though the second has a higher regression coefficient. The fourth dictionary element is a network with great sparsity, and there seems to be two synchronizing components in the network structure. At the end of the training iteration, the synchronizing part in the upper left corner shrinks in size compared with time 40. INTERPRET. For the element with lowest regression coefficient, synchronizing part initially increases from time 0 to time 30 but later decreases from time 30 to time 50.

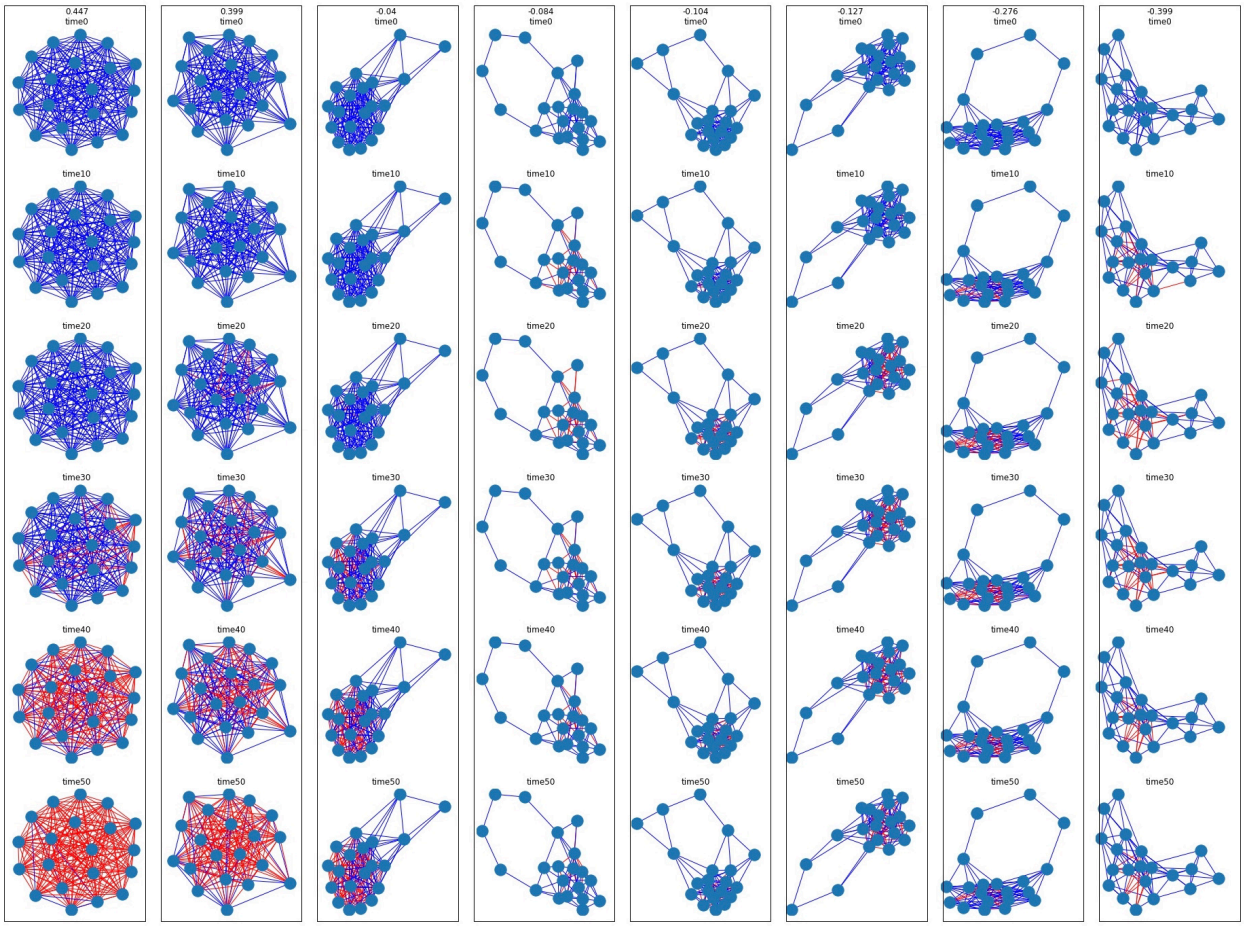


Figure 14: The dictionary elements learned from SDL on the coloring adjacency matrix from dataset of NWS. Each column is a dictionary element and the number on the top of the column is the regression coefficient of SDL. Each row represents the dictionary element at different time. The network structure is from the dictionary values of the adjacency matrix. The coloring of edges is based on the dictionary values of the coloring adjacency matrix. If an edge is colored red, then the two vertices of the edge have the same color. If an edge is colored blue, then the two vertices of the edge have different colors. The weights of all edges are the same.

5.3 GHM

5.3.1 GHM Synchronizing Behavior

By carrying out the implementation discussed in the experiment design section, we generate Figure 16. One can infer a general trend that for GHM model, the synchronizing probability decreases with increasing node. This is not mind-boggling since it has been proved that GHM synchronizes on a path definitely [6] and with additional cycles and complexity in the graph topology, it witnesses more non–synchronizing cases. Here we introduce a concept of transitivity to indicate the clustering in the graph, and the transitivity is defined to be the number of closed triangles in a graph divided by possible maximum number of triangles with that node number. The average transitivity of the 25 subgraphs are determined and the denser Caltech network involves noticeably higher closed triangles for their subgraphs. The Caltech graphs also has more rapid decreasing synchronizing rate than all other sparser networks.

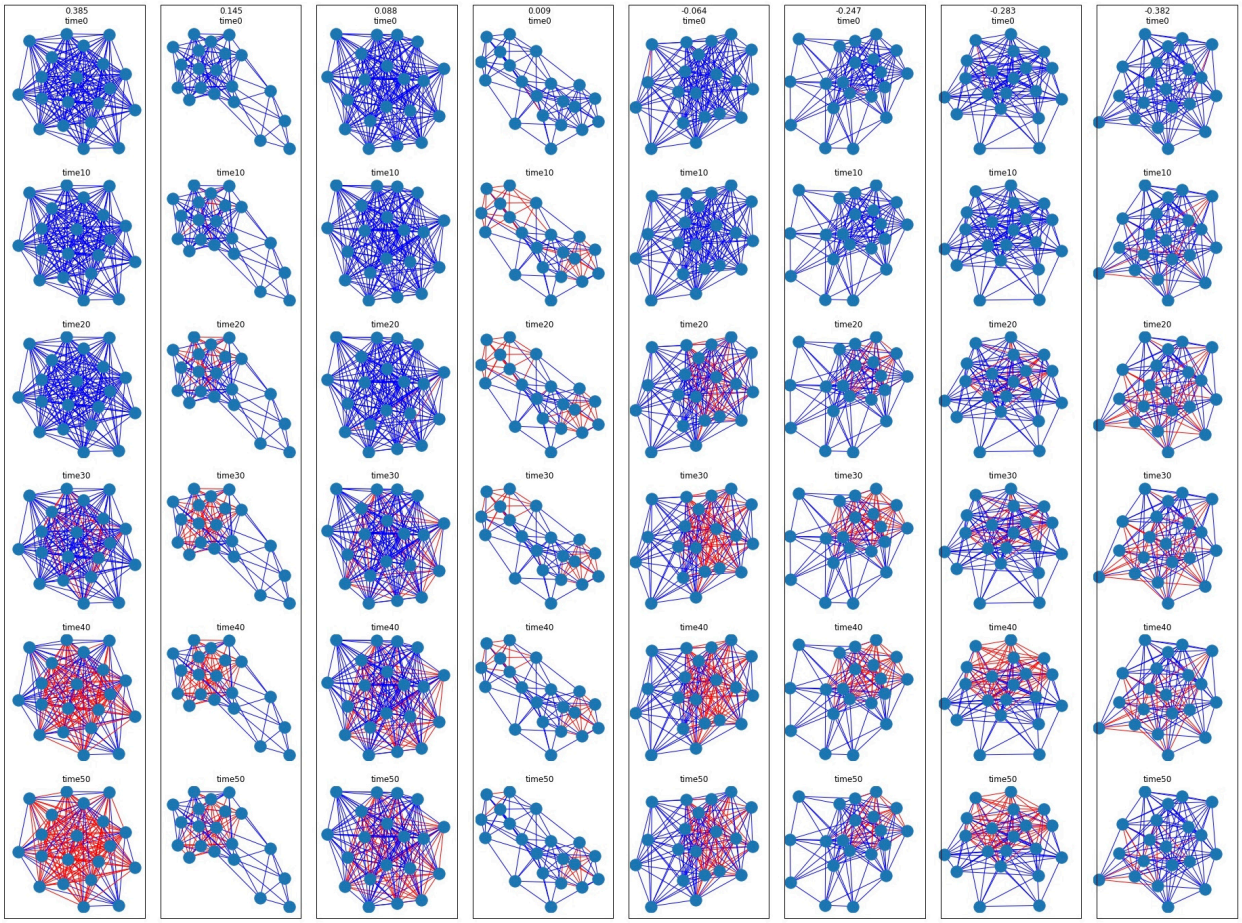


Figure 15: The dictionary elements learned from SDL on the coloring adjacency matrix from dataset of UCLA26. Each column is a dictionary element and the number on the top of the column is the regression coefficient of SDL. Each row represents the dictionary element at different time. The network structure is from the dictionary values of the adjacency matrix. The coloring of edges is based on the dictionary values of the coloring adjacency matrix. If an edge is colored red, then the two vertices of the edge have the same color. If an edge is colored blue, then the two vertices of the edge have different colors. The weights of all edges are the same.

While on a fully connected 2D grid, the graph topology is much more complicated than the 1D counterparts. Thus GHM rarely synchronizes using normal transition rules, and they fall into a periodic phase cycle afterwards, where $t = 8$ & $t = 20$ in Figure 17 are perfect illustration for the periodic behavior. Figure 17 is an example of time series for GHM on a $(70, 70)$ grid. To make them synchronize easier on 2D graphs, we innovatively introduce stochasticity to first two rules in GHM's updating scheme. However, in order to determine which part of the updating rule plays a decisive role, we add stochasticity to the second part of the rule solely and to both first two for comparison. Now we randomly generate a variable $P \in [0, 1]$ and fix a threshold H. The two new transition rules (one being the control group) read:

$$X_{t+1}(v) = \begin{cases} 0 & \text{if } X_t(v) = 0 \quad \& \quad X_t(u) \neq 1 \, \forall u \in N(v) \\ 1 & \text{if } X_t(v) = 0 \quad \& \quad X_t(u) \neq 1 \, \forall u \in N(v) \\ 0 & \text{if } X_t(v) = 0 \quad \& \quad \exists \, u \in N(v) \implies X_t(u) = 1 \quad \& \quad P > H \\ 1 & \text{if } X_t(v) = 0 \quad \& \quad \exists \, u \in N(v) \implies X_t(u) = 1 \quad \& \quad P \leq H \\ X_t(v) + 1 & \text{otherwise} \end{cases} \tag{8}$$

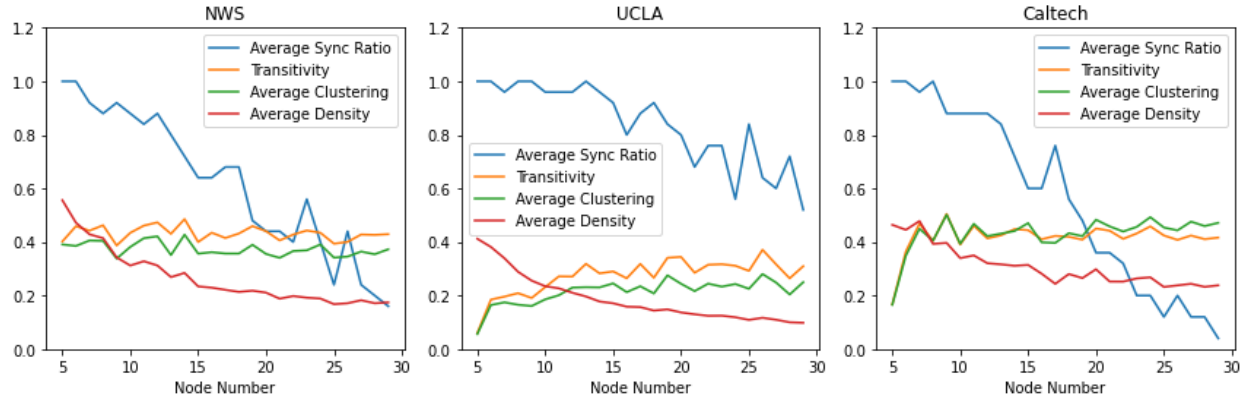


Figure 16: Comparison of synchronizing behavior of GHM on subgraphs of four real world networks HL: Maybe combine the first and the second rows into one

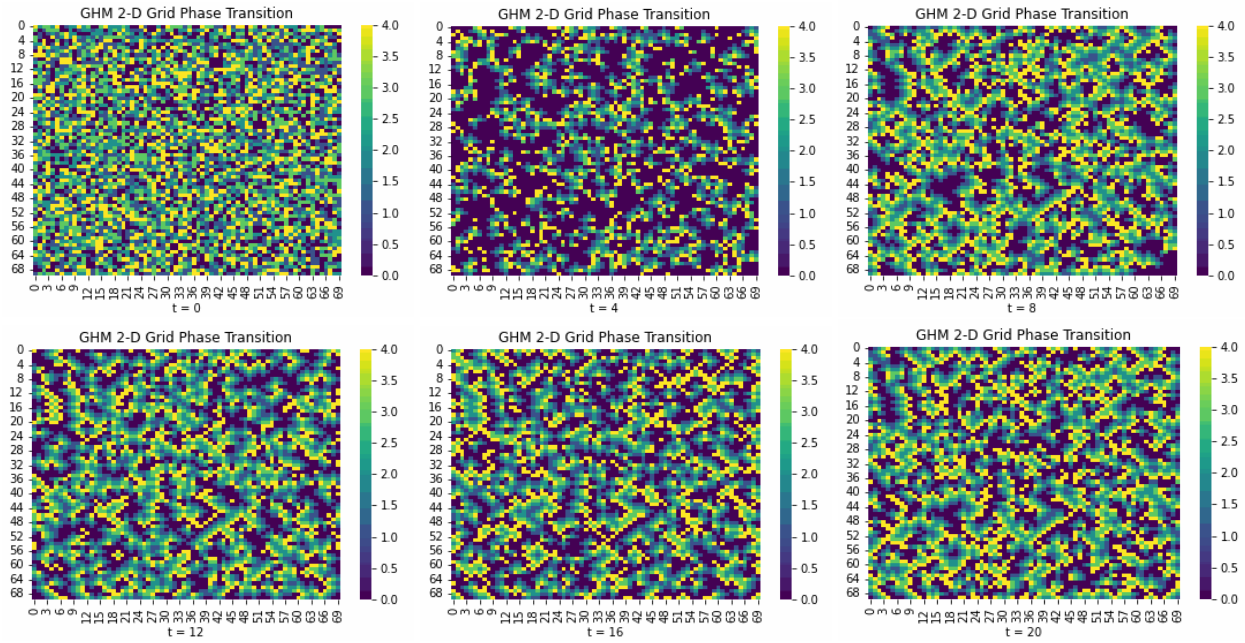


Figure 17: GHM dynamics on 70-by-70 grid graph

$$X_{t+1}(v) = \begin{cases} 0 & \text{if } X_t(v) = 0 \quad \& \quad X_t(u) \neq 1 \, \forall u \in N(v) \quad \& \quad P \leq H \\ 1 & \text{if } X_t(v) = 0 \quad \& \quad X_t(u) \neq 1 \, \forall u \in N(v) \quad \& \quad P > H \\ 0 & \text{if } X_t(v) = 0 \quad \& \quad \exists \, u \in N(v) \implies X_t(u) = 1 \quad \& \quad P > H \\ 1 & \text{if } X_t(v) = 0 \quad \& \quad \exists \, u \in N(v) \implies X_t(u) = 1 \quad \& \quad P \leq H \\ X_t(v) + 1 & \text{otherwise} \end{cases} \tag{9}$$

We refer the first set Equation 8 to be "static" stochasticity and second modification Equation 9 to be "static and excitation" stochasticity in the Figure 23. As one can infer from the graphs that the determining line in the transition rule for synchronizing is whether to lift the rested state (0) to excited state(1) when it is suppose to stay rested in the original rule. The general pattern tells that the less nodes that is stuck at rested state, the easier for the system to break a periodic behavior and to synchronize.

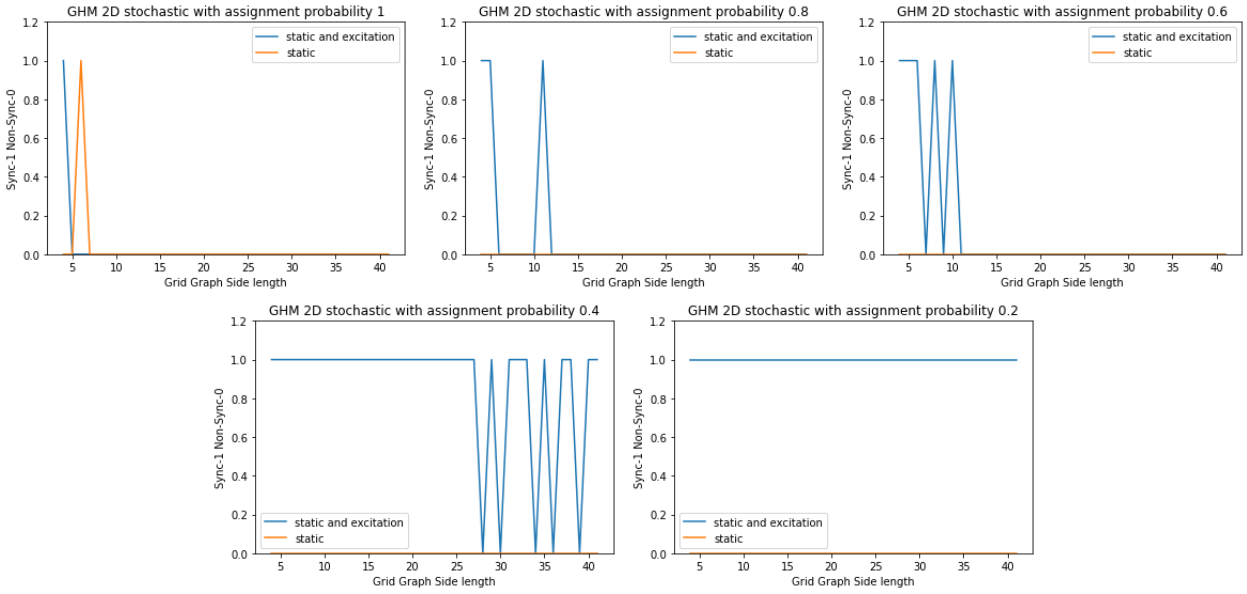


Figure 18: GHM dynamics on 2-D grid of different nodes under different threshold probability

5.3.2 Interpretable Non-Negative Matrix Factorization for GHM

In this section, we will be examining the dictionaries spit out by the Non-Negative Matrix Factorization for NWS graphs and subgraphs of the four real world networks. Take Figure 19 as an example, for each graph topology, we visualize the dictionary elements in square blocks as well as their graph representations. The L_1 norm is given as box plots to distinguish synchronizing and non-synchronizing cases.

1. `NWS` Dictionaries
   - The pattern observed in NWS graphs here might not seem to be a very clean cut since both synchronizing and non-synchronizing graphs host a lot of side edges from the Hamiltonian path. Neither side of the graphs contains noticeable hub or clustering. The observation is hard from the view of the dictionary or graph plot. Therefore, we quantitatively calculate the L_1 norm of the dictionary elements. It is obvious that the densest and sparsest dictionary graphs in the non-synchronizing cases are denser than their counterparts in synchronizing cases, so is the median and third quartile. yet the synchronizing graphs win the first quartile.
2. `UCLA` Dictionaries
   - In UCLA case, the graphs are all very sparse and there is no obvious distinctions between synchronizing and non-synchronizing cases. The pattern of the L_1 norm resembles that of the NWS graphs with slight variations on the median position.
3. `Wisconsin` Dictionaries
   - Similar to the network of UCLA, Wisconsin networks are also rather sparse, yet the synchronizing graphs are more noticeably sparser than non-synchronizing ones both from the dictionary plots (if scrutinized carefully) and L_1 norm graph.
4. `Harvard` Dictionaries
   - Harvard experiences same pattern as those of the other large university networks like Wisconsin and UCLA.
5. `Caltech` Dictionaries
   - Caltech is a more interesting case with much denser networks than all the other universities. The difference between the synchronizing and non-synchronizing graphs is more outstanding such that the synchronizing ones has significant less density than the non-synchronizing ones from the L_1 norm graph.

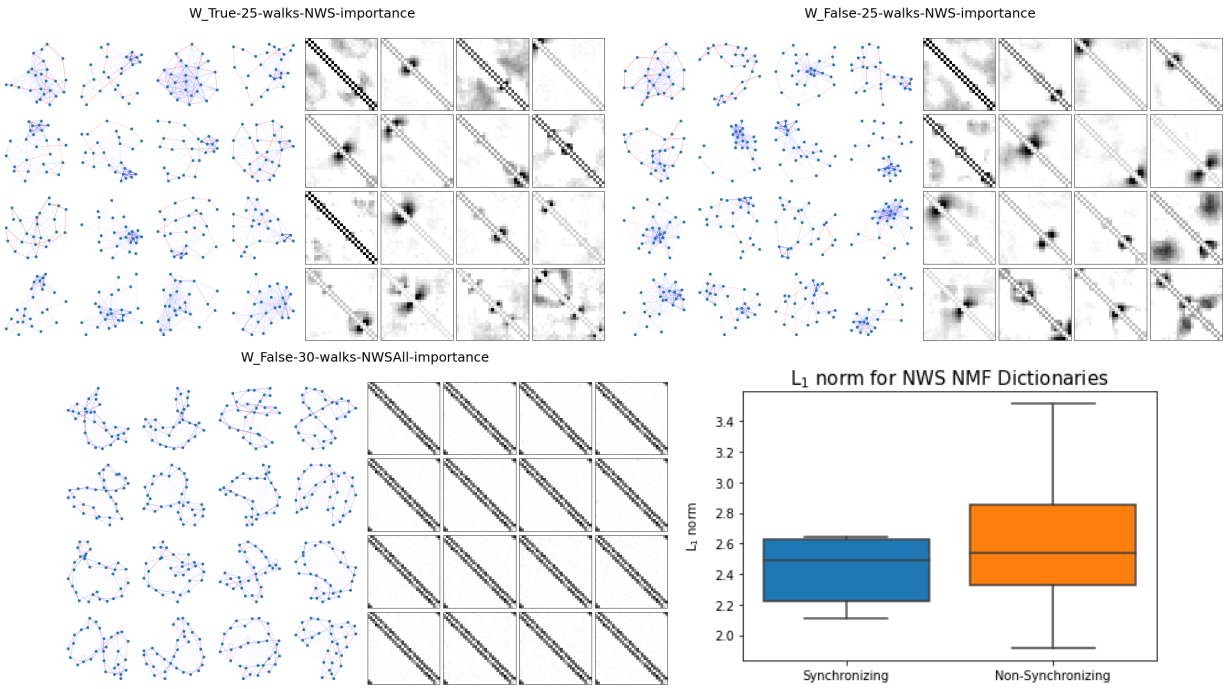


Figure 19: GHM 30 Node NWS NMF

5.3.3 Interpretable Supervised Learning for GHM

6 Discussion

6.1 Kuramoto

6.2 FCA

There is no apparent difference in dictionary elements learned from dynamics as time changes, which might explain the poor performance of sdl on dynamics dataset

6.3 GHM

7 Conclusion

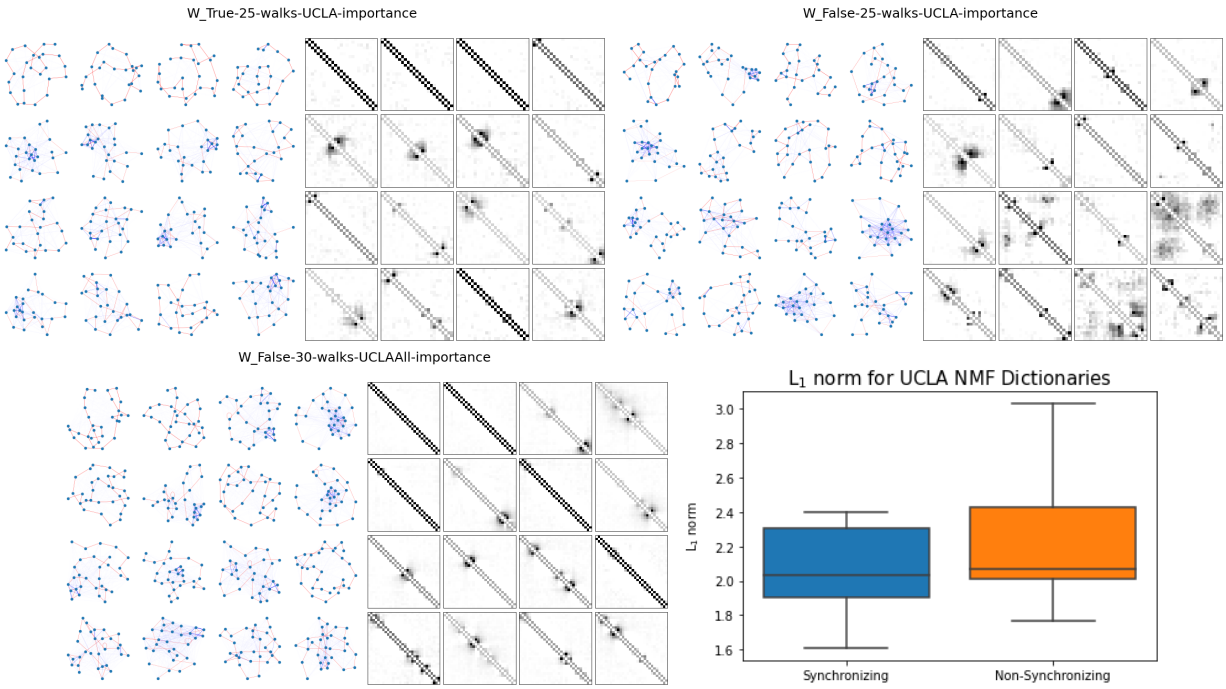


Figure 20: GHM dynamics on 2-D grid of different nodes under different threshold probability

References

[1] Steven H Strogatz. From kuramoto to crawford: exploring the onset of synchronization in populations of coupled oscillators. *Physica D: Nonlinear Phenomena*, 143(1-4):1–20, 2000. 1, 4

[2] Juan A Acebrón, Luis L Bonilla, Conrad J Pérez Vicente, Félix Ritort, and Renato Spigler. The kuramoto model: A simple paradigm for synchronization phenomena. *Reviews of modern physics*, 77(1):137, 2005. 1, 3, 4

[3] Sujit Nair and Naomi Ehrich Leonard. Stable synchronization of rigid body networks. *Networks & Heterogeneous Media*, 2(4):597, 2007. 1

[4] Roberto Pagliari and Anna Scaglione. Scalable network synchronization with pulse-coupled oscillators. *IEEE Transactions on Mobile Computing*, 10(3):392–405, 2010. 1

[5] Florian Dorfler and Francesco Bullo. Synchronization and transient stability in power networks and nonuniform kuramoto oscillators. *SIAM Journal on Control and Optimization*, 50(3):1616–1642, 2012. 1

[6] Hardeep Bassi, Richard Yim, Rohith Kodukula, Joshua Vendrow, Cherlin Zhu, and Hanbaek Lyu. Learning to predict synchronization of coupled oscillators on heterogeneous graphs. *arXiv preprint arXiv:2012.14048*, 2020. 2, 4, 7, 21

[7] Hanbaek Lyu. Synchronization of finite-state pulse-coupled oscillators. *Physica D: Nonlinear Phenomena*, 303:28–38, 2015. 3, 4

[8] James M Greenberg and Stuart P Hastings. Spatial patterns for discrete models of diffusion in excitable media. *SIAM Journal on Applied Mathematics*, 34(3):515–523, 1978. 3

[9] Yoshiki Kuramoto. Chemical turbulence. In *Chemical oscillations, waves, and turbulence*, pages 111–140. Springer, 1984. 4

[10] Fabrizio Damicelli. Python implementation of the kuramoto model. https://github.com/fabridamicelli/kuramoto, 2019. 4

[11] Agam Goyal, Bella Wu, Binhao Chen, Bryan Xu, and Hanbaek Lyu. Interpretable L2PSync. https://github.com/AGoyal0512/reu2022, 8 2022. 4

[12] Andrew Ng, Michael Jordan, and Yair Weiss. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 14, 2001. 6

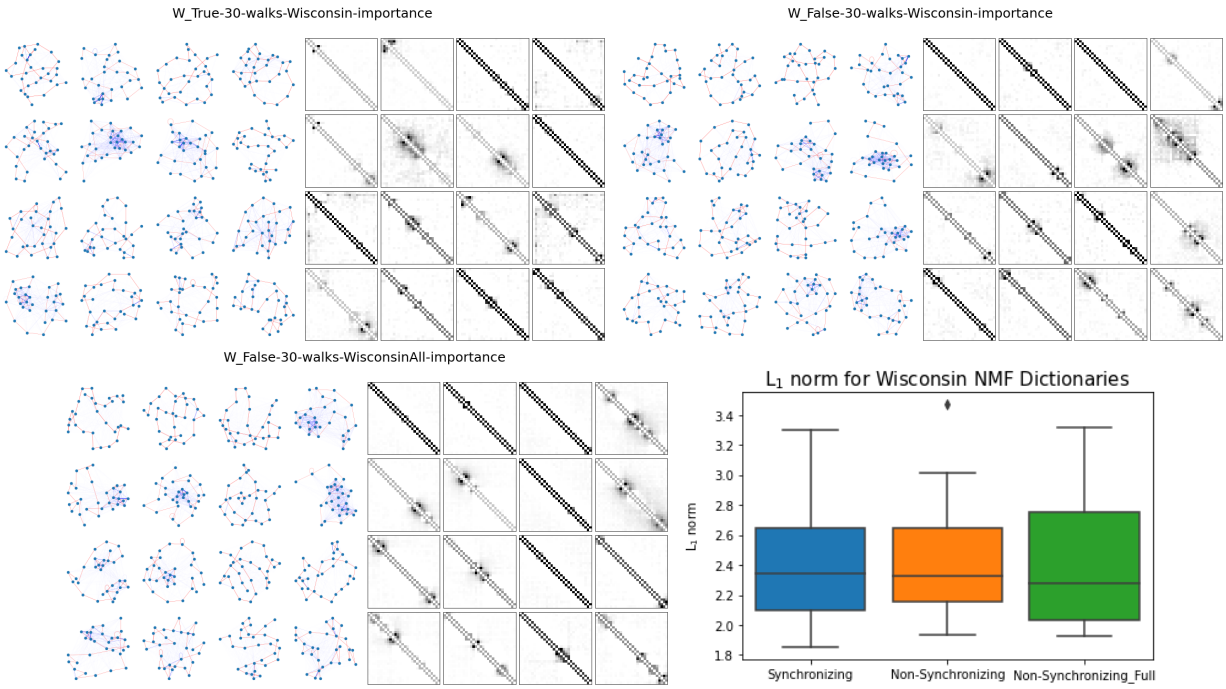


Figure 21: GHM dynamics on 2-D grid of different nodes under different threshold probability

[13] Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016. 6

[14] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017. 6

[15] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 6

[16] Daniel D Lee and H Sebastian Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791, 1999. 7

[17] Hanbaek Lyu. Global synchronization of pulse-coupled oscillators on trees. *SIAM Journal on Applied Dynamical Systems*, 17(2):1521–1559, 2018. 7

[18] Joel Nishimura and Eric J Friedman. Robust convergence in pulse-coupled oscillators with delays. *Physical review letters*, 106(19):194101, 2011. 7

[19] Johannes Klinglmayr, Christoph Kirst, Christian Bettstetter, and Marc Timme. Guaranteeing global synchronization in networks with stochastic interactions. *New Journal of Physics*, 14(7):073031, 2012. 7

[20] Anton V Proskurnikov and Ming Cao. Synchronization of pulse-coupled oscillators and clocks under minimal connectivity assumptions. *IEEE Transactions on Automatic Control*, 62(11):5873–5879, 2016. 7

[21] Felipe Nunez, Yongqiang Wang, and Francis J Doyle. Synchronization of pulse-coupled oscillators on (strongly) connected graphs. *IEEE Transactions on Automatic Control*, 60(6):1710–1715, 2014. 7

[22] Luc Moreau. Stability of multiagent systems with time-dependent communication links. *IEEE Transactions on automatic control*, 50(2):169–182, 2005. 7

[23] Antonis Papachristodoulou, Ali Jadbabaie, and Ulrich Münz. Effects of delay in multi-agent consensus and oscillator synchronization. *IEEE transactions on automatic control*, 55(6):1471–1477, 2010. 7

[24] Bernard Chazelle. The total s-energy of a multiagent system. *SIAM Journal on Control and Optimization*, 49(4):1680–1706, 2011. 7

[25] Amanda L Traud, Peter J Mucha, and Mason A Porter. Social structure of facebook networks. *Physica A: Statistical Mechanics and its Applications*, 391(16):4165–4180, 2012. 8

[26] Mark EJ Newman and Duncan J Watts. Renormalization group analysis of the small-world network model. *Physics Letters A*, 263(4-6):341–346, 1999. 8, 13

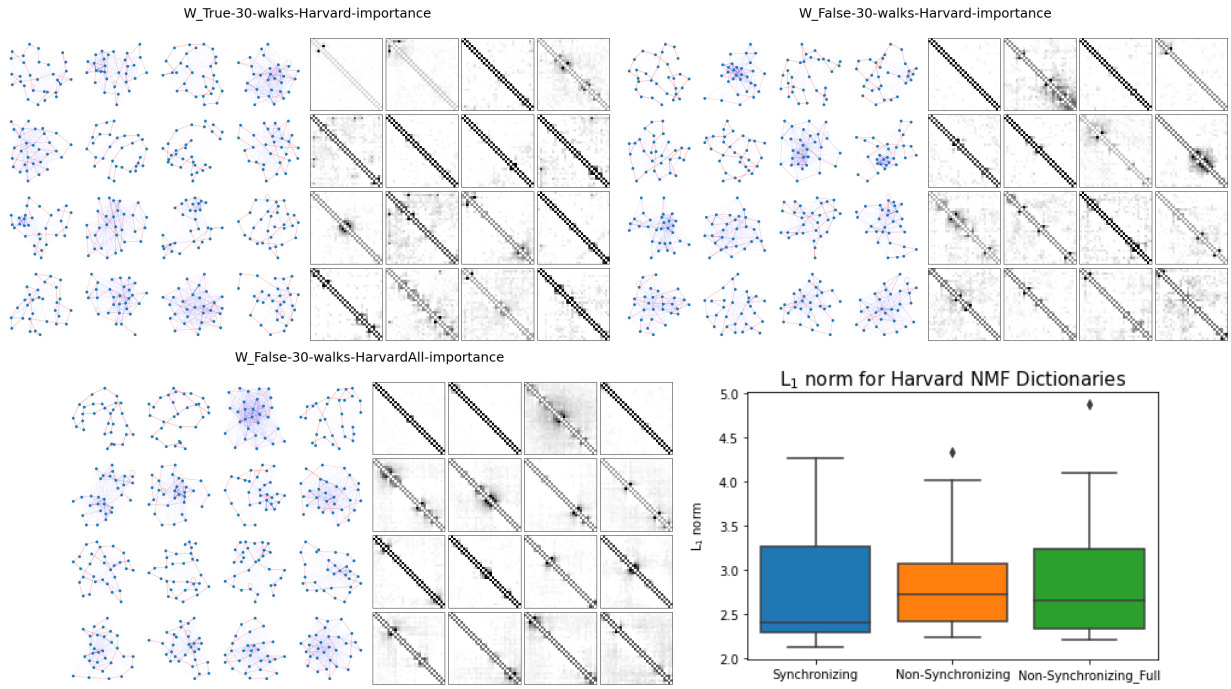


Figure 22: GHM dynamics on 2-D grid of different nodes under different threshold probability

[27] Hanbaek Lyu, Yacoub H Kureh, Joshua Vendrow, and Mason A Porter. Learning low-rank latent mesoscale structures in networks. *arXiv preprint arXiv:2102.06984*, 2021. 8

[28] Hanbaek Lyu, Facundo Memoli, and David Sivakoff. Sampling random graph homomorphisms and applications to network data analysis. *arXiv preprint arXiv:1910.09483*, 2019. 8

[29] Kazuha Itabashi, Quoc Hoan Tran, and Yoshihiko Hasegawa. Evaluating the phase dynamics of coupled oscillators via time-variant topological features. *Physical Review E*, 103(3):032207, 2021. 9

[30] Duncan J Watts and Steven H Strogatz. Collective dynamics of small-worldnetworks. *nature*, 393(6684):440–442, 1998. 9, 11

[31] Eric W Weisstein. Matrix norm. *https://mathworld. wolfram. com/*, 2002. 11, 12

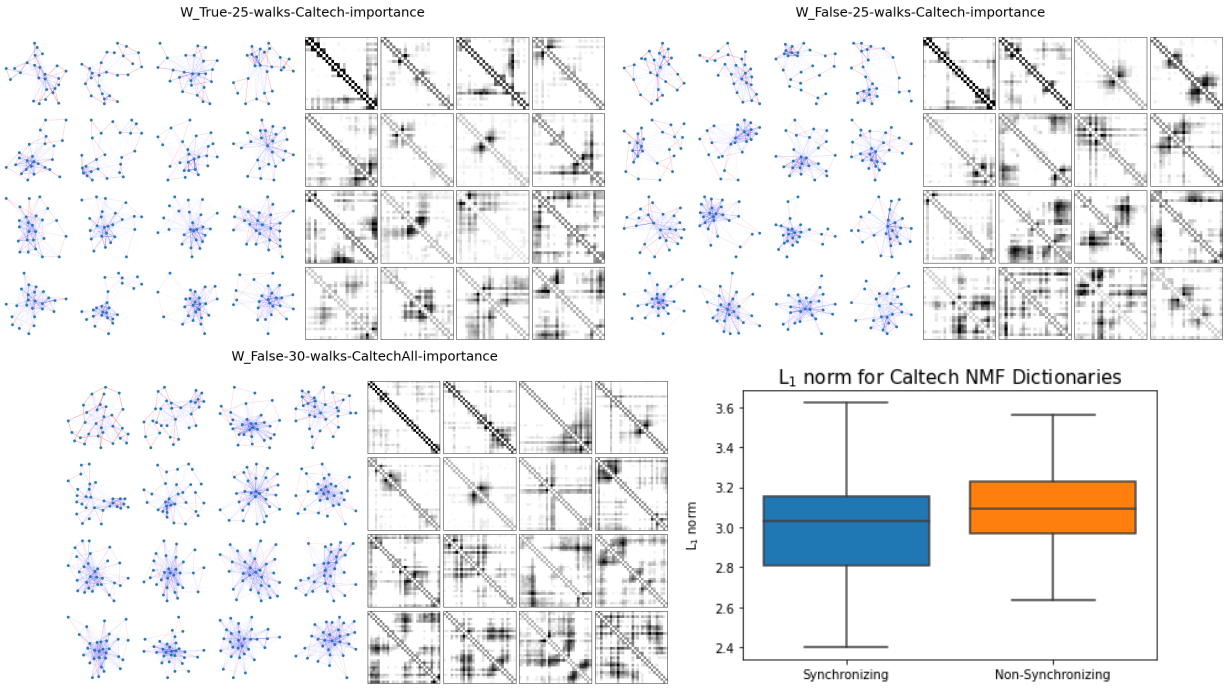


Figure 23: GHM dynamics on 2-D grid of different nodes under different threshold probability