DPPIN: A Biological Repository of Dynamic Protein-Protein Interaction Network Data

Dongqi Fu
University of Illinois at Urbana-Champaign
Illinois, USA
dongqif2@illinois.edu

Jingrui He
University of Illinois at Urbana-Champaign
Illinois, USA
jingrui@illinois.edu

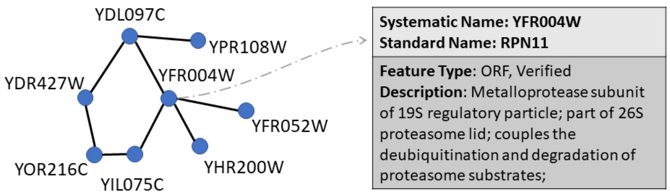
Abstract*—In the big data era, the relationship between entries becomes more and more complex. Many graph (or network) algorithms have already paid attention to dynamic networks, which are more suitable than static ones for fitting the complex real-world scenarios with evolving structures and features. To contribute to the dynamic network representation learning and mining research, we provide a new bunch of label-adequate, dynamics-meaningful, and attribute-sufficient dynamic networks from the health domain. To be specific, in our proposed repository DPPIN, we totally have 12 individual dynamic network datasets at different scales, and each dataset is a dynamic protein-protein interaction network describing protein-level interactions of yeast cells. We hope these domain-specific node features, structure evolution patterns, and node and graph labels could inspire the regularization techniques to increase the performance of graph machine learning algorithms in a more complex setting. Also, we link potential applications with our DPPIN by designing various dynamic graph experiments, where DPPIN could indicate future research opportunities for some tasks by presenting challenges on state-of-the-art baseline algorithms. Finally, we identify future directions to improve the utility of this repository and welcome constructive inputs from the community. All resources (e.g., data and code) of this work are deployed and publicly available at https://github.com/DongqiFu/DPPIN.

***Index Terms*—Biological Networks, Dynamic Networks, Network Datasets**

I. Introduction

Networks (or graphs)[1] are complex data structures containing comprehensive node-attribute and node-interaction information, which attracts many research interests in network representation learning algorithms [59] and network mining tasks [8] to serve for a wide range of real-world applications such as information retrieval [33], recommendation [64], fraud detection [32], and drug discovery [12]. To fit the real-world networks evolving attributes and typologies, many graph representation learning and mining algorithms have transferred from the static setting to the dynamic setting [2], [15]–[19], [27], [62], [63], where the network structure (i.e., graph topology) and the node features are evolving and dependent on time.

However, compared with the amount of publicly available static network datasets, the dynamic network datasets are not that sufficient. This phenomenon may hinder the related research progress. To contribute to the dynamic network representation learning and mining research community, we provide a new dynamic network repository from the biological domain, named DPPIN. To be specific, DPPIN has 12 individual dynamic network datasets at different scales describing dynamic protein-protein interactions of yeast cells, and each dynamic network dataset in DPPIN is label-adequate (i.e., high label rate of nodes), dynamics-meaningful (i.e., metabolic patterns of yeast cells), and attribute-sufficient (i.e., accessible node features and edge features). We hope these domain-specific node features, structure evolution patterns, and node and graph labels could inspire the next-generation graph machine learning algorithms in the dynamic setting.



Systematic Name: YFR004W
Standard Name: RPN11

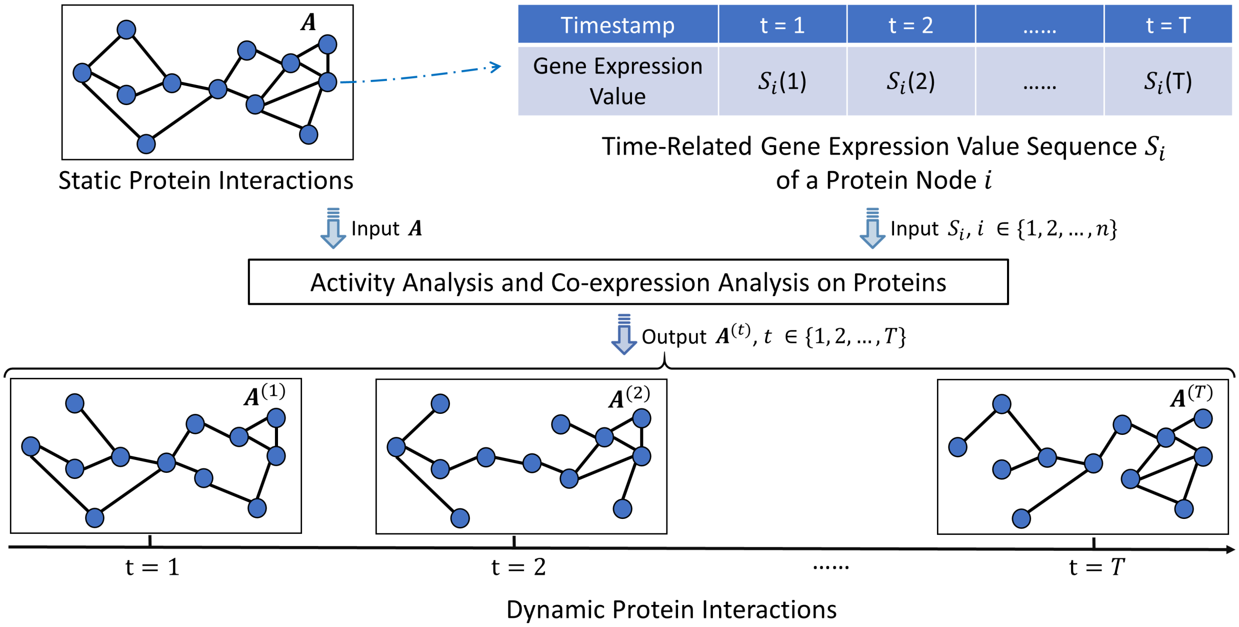
Feature Type: ORF, Verified
Description: Metalloprotease subunit of 19S regulatory particle; part of 26S proteasome lid; couples the deubiquitination and degradation of proteasome substrates;

Fig. 1: A Subgraph Extracted from the Static Protein-Protein Interaction Network of Yeast Cells [4]. Each node stands for a gene coding protein, and the description information of each protein node can be retrieved from the Saccharomyces Genome Database.[2]

In this paper, we first start by introducing the generation process of each dynamic network dataset in our DPPIN, by which a static protein-protein interaction network as shown in Figure 1 can be decomposed into different timestamps as shown in Figure 2. Moreover, during the generation process, the details about the gene expression value calculation, node feature determination, and node label retrieval are discussed in Section III with the generation statistics. The utility of DPPIN data repository are linked with the real-world problems by potential applications in Section IV. With different generated network datasets in DPPIN, we design extensive experiments such like dynamic spectral clustering and dynamic graph classification experiments in Section V, where several experimental results suggest that DPPIN presents challenges on state-of-the-art baseline algorithms and indicates future

[1] We use the term ”network” and ”graph” interchangeably throughout the paper.

[2] https://www.yeastgenome.org/


$\boldsymbol{A}$
Static Protein Interactions
Time-Related Gene Expression Value Sequence S_i of a Protein Node i
Input $\boldsymbol{A}$
Input $S_i, i \in \{1, 2, \ldots, n\}$
Activity Analysis and Co-expression Analysis on Proteins
Output $\boldsymbol{A}^{(t)}, t \in \{1, 2, \ldots, T\}$
$\boldsymbol{A}^{(1)}$
$\boldsymbol{A}^{(2)}$
$\boldsymbol{A}^{(T)}$
t = 1
t = 2
......
t = T
Dynamic Protein Interactions

Timestamp	t = 1	t = 2	......	t = T
Gene Expression Value	$S_i(1)$	$S_i(2)$	......	$S_i(\text{T})$

Fig. 2: The Dynamic Network Structure Generation Process for each Dataset in DPPIN.

research opportunities. The data source, generation process program, and other necessary information of our DPPIN repository are well documented and released[3].

Our main contributions are summarized as follows.

- **Datasets**: Regarding the biological domain, beyond many static networks[4], we provide a new repository of 12 dynamic protein-protein interaction network datasets to contribute to the dynamic network machine learning research community.
- **Experiments**: To demonstrate the utility of DPPIN, we design extensive experiments and discuss how networks of DPPIN can inspire baseline algorithms to get improved.
- **Future Work**: We also indicate the future work directions to improve our DPPIN repository for better services.

II. Preliminary

We use lowercase letters (e.g., α) for scalars, capital letters (e.g., V) for sets, bold lowercase letters for column vectors (e.g., $\boldsymbol{p}$), bold capital letters for matrices (e.g., $\boldsymbol{A}$), and parenthesized superscripts to denote the timestamp of matrices (e.g., $\boldsymbol{A}^{(t)}$).

A protein-protein interaction network (shot for PPIN in the rest) is denoted as an undirected graph $G = (V, E)$, where each node represents a gene coding protein (shot for protein in the rest), each edge represents a protein-protein interaction, and matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix encoding interactions of all nodes [38]. Moreover, protein-protein interactions in a cell are usually dynamic and change over time [9], [23], [25], [40], [41], [45]. Many different efforts have been devoted to the dynamic PPIN construction [52], [53], [60], [61], where a dynamic PPIN is represented as a sequence of observed snapshots, i.e., $\tilde{G} = \{G^{(t)}\}$ with $t \in \{1, 2, \ldots, T\}$. Each snapshot is represented as $G^{(t)} = (V^{(t)}, E^{(t)})$ plus optional node and edge features, and $V^{(t)}$ and $E^{(t)}$ denote the nodes and edges that existed at time t, respectively. Note that the number of nodes at different timestamps is different. For the clear notation, we denote $|V^{(t)}| = |V|$ in the paper, which means a node without any connections at time t can just be regarded as a dangling node at that time.

To generate dynamic networks for our DPPIN, we apply the dynamic protein-protein interaction network construction method [61], which could use both gene expression variance analysis and co-expression correlations analysis to establish dynamic PPINs. The generation process shown in Figure 2 can be described as follows. Given a static PPIN with the adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ and time-aware gene expression value sequences S_i for each node i at each timestamp $t \in \{1, \ldots, T\}$, *i.e.,* $S_i(t)$ *denotes the gene expression value of protein node* i *at timestamp* t, we sequentially build the dynamic PPIN $\tilde{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, which is a sequence of snapshots $G^{(t)}$ at each observed timestamp t.

III. Generated Data in DPPIN

In this section, we first introduce the dynamic structure generation process. Then, we introduce the feature and label extraction the with the corresponding biological meaning. The statistics of each dynamic network dataset in DPPIN are summarized in Table II.

A. Dynamic Structure Generation Process

The generation process requires two inputs, as shown in Figure 2, which are (1) a static PPIN and (2) time-aware gene expression values of each protein in that static PPIN. Then the generation process is divided into three sequential steps: (1) determine active proteins at each desired timestamp, (2)

[3] https://github.com/DongqiFu/DPPIN

[4] https://www.inetbio.org/yeastnet/downloadnetwork.php

determine co-expressed protein pairs at that timestamp, and (3) preserve both active and co-expressed protein connections for that timestamp snapshot.

Determine Active Proteins. If the gene expression value of a protein is higher than a pre-defined threshold at a certain timestamp, then that protein is considered to be active for that timestamp. In [52], the active probability of proteins is expressed as follows.

$$p^{(t)}(i) = \begin{cases} 0.99, & \text{if } S_i(t) \geq Th_3(i) \\ 0.95, & \text{if } Th_3(i) > S_i(t) \geq Th_2(i) \\ 0.68, & \text{if } Th_2(i) > S_i(t) \geq Th_1(i) \\ 0.0, & \text{if } S_i(t) < Th_1(i) \end{cases} \tag{1}$$

where S_i is the sequential gene expression data of i-th protein over a period of time, and $S_i(t)$ denotes the expression value of the i-th protein at time t. $\boldsymbol{p}^{(t)} \in \mathbb{R}^{n \times 1}$ denotes the probability vector at time t, which is a column vector encoding the active probability of each protein at time t. Scalars Th_3, Th_2, and Th_1 are three thresholds, which are expressed as follows.

$$Th_k(i) = \mu(S_i) + k\sigma(S_i)(1 - \frac{1}{1+\sigma^2(S_i)}),\ k \in \{1, 2, 3\} \tag{2}$$

where $\mu(S_i)$ stands for the mean of the entire sequence S_i, and $\sigma(S_i)$ denotes the standard deviation of S_i.

With the computed probability vector $\boldsymbol{p}^{(t)}$ at time t, we can obtain the activity matrix $\boldsymbol{A}_{act}^{(t)} \in \mathbb{R}^{n \times n}$ at time t as follows.

$$\boldsymbol{A}_{act}^{(t)} = \boldsymbol{p}^{(t)} \boldsymbol{p}^{\top(t)} \tag{3}$$

where $\boldsymbol{p}^{\top(t)}$ denotes the transpose of the column vector $\boldsymbol{p}^{(t)}$.

Determine Co-expressed Protein Pairs. The co-expression correlation coefficient is a strong indicator of protein functional associations [61] that is used to indicate whether co-expressed genes have the same expression variance patterns across different conditions [52]. To investigate whether two protein nodes are co-expressed at time t, the co-expression matrix $\boldsymbol{A}_{coe}^{(t)} \in \mathbb{R}^{n \times n}$ of a protein network is expressed as follows with the coefficient.

$$A_{coe}^{(t)}(i,j) = \begin{cases} |PCC^{(t)}(i,j)|, & \text{if } |PCC^{(t)}(i,j)| \geq Th_{PCC} \\ 0, & \text{if } |PCC^{(t)}(i,j)| < Th_{PCC} \end{cases} \tag{4}$$

where $PCC^{(t)}$ is the Pearson Correlation Coefficient function at time t that takes $\{S_i(t-1), S_i(t), S_i(t+1)\}$ and $\{S_j(t-1), S_j(t), S_j(t+1)\}$ from two protein nodes i and j as the input. Th_{PCC} is the pre-defined threshold for Pearson Correlation Coefficient, which is set to be 0.5 through preliminary experiments [61].

Determine Active and Co-expressed Protein Interactions. With the computed activity matrix $\boldsymbol{A}_{act}^{(t)} \in \mathbb{R}^{n \times n}$ and co-expression matrix $\boldsymbol{A}_{coe}^{(t)} \in \mathbb{R}^{n \times n}$ at time t, we can now indicate active and co-expressed protein interactions to form the weighted adjacency matrix at time t, denoted as $\boldsymbol{A}^{(t)}$.

$$\boldsymbol{A}^{(t)} = \boldsymbol{A}_{act}^{(t)} \odot \boldsymbol{A}_{coe}^{(t)} \odot \boldsymbol{A} \tag{5}$$

where $\odot$ stands for the element-wise multiply operation, and $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ denotes the adjacency matrix of the input static PPIN.

To obtain two necessary inputs of the above-mentioned methods to generate dynamic PPIN structures, we select 12 static networks from *High-Throughput Protein-Protein Interactions of yeast cells*[5] for the matrix $\boldsymbol{A}$ in Eq. 5, and the *GSE3431 gene expression data* [48][6] for the time-aware sequence $S_i(t)$ in Eq. 1 and Eq. 2. In GSE3431, each gene coding protein has 36 observed gene expression values at 36 timestamps. To be specific, those 36 timestamps consist of 3 successive metabolic cycles of yeast cells, where each cycle occupies 12 timestamped intervals, and each time interval occupies 25 minutes in the real world. Thus, we could totally have 36 timestamps for each generated dynamic network in DPPIN.

B. Feature and Label Extraction

According to Eq. 5, generated dynamic network structures in DPPIN are undirected and weighted. Hence, each edge is represented as (i, j, t, w), where i and j are two nodes, t denotes the timestamp, and w denotes the weight computed through Eq. 5 and represents the active and co-expressed probability as edge features. Moreover, the node feature of each node i can be directed obtained from gene expression sequences, i.e., $S_i(t)$. The label of each protein node can be accessed through the Saccharomyces Genome Database[7]. For example, the protein node YNR066C is an uncharacterized protein, YLR366W is a dubious protein, and YGR062C is a verified protein. Now, the construction of each dataset in DPPIN is completed with evolving typologies, node features, edge features, and node labels.

The entire generation is summarized in Algorithm 1, where Steps 2–3 are responsible for building the protein activity matrix, then Step 4 is responsible for building the protein co-expression matrix, and Step 5 is responsible for constructing the weighted adjacency matrix at that time t. For the feature matrix, Step 6 concatenates the protein expression values from the first time to the current time and stores this concatenation as the node feature vector. Note that the length of this node feature vector is dependent on time.

C. Generation Statistics

With the static structures, gene sequence, protein feature and label information discussed in above two subsections, we are ready to call Algorithm 1 to generate dynamic networks with time-aware node features and labels. The selected static protein networks are summarized in Table I. For example, in Krogan (LCMS) [29], the authors identified proteins by the liquid chromatography-tandem mass spectrometry (i.e., LCMS). Correspondingly, the dynamic version of Krogan (LCMS) is generated through Algorithm 1, i.e., DPPIN-Krogan (LCMS)

[5] https://www.inetbio.org/yeastnet/downloadnetwork.php

[6] https://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE3431

[7] https://www.yeastgenome.org/

TABLE I: Selected Static PPINs for Generating Dynamic PPINs

Static HT PPINs	Description of Protein Identification Methods
Uetz [49]	Genome-scale yeast-two-hybrid (Y2H) screen
Ito [26]	Genome-scale Y2H with pooled library
Ho [24]	Genome-scale affinity purification followed by mass spectrometry analysis of co-purified proteins (APMS)
Gavin [20]	Genome-scale APMS
Krogan (LCMS) [29]	Genome-scale APMS with liquid chromatography tandem mass spectrometry (LCMS)
Krogan (MALDI) [29]	Genome-scale APMS with matrix-assisted laser desorption/ionization (MALDI)
Yu [56]	Genome-scale Y2H
Breitkreutz [7]	Large-scale APMS for kinase and phosphatase interactions
Babu [4]	Large-scale APMS for membrane proteins
Lambert [30]	Protein interactions for chromatin-related proteins
Tarassov [44]	Genome-wide protein-protein interactions
Hazbun [22]	Interactome for 100 essential genes

TABLE II: Generated Dynamic Network Datasets in DPPIN

Generated Dynamic PPINs	#Nodes	#Edges	Node Features	Edge Features	Node Label Rate	#Timestamps
DPPIN-Uetz	922	2,159	✓	✓	921/922 (99.89%)	36
DPPIN-Ito	2,856	8,638	✓	✓	2854/2856 (99.93%)	36
DPPIN-Ho	1,548	42,220	✓	✓	1547/1548 (99.93%)	36
DPPIN-Gavin	2,541	140,040	✓	✓	2538/2541 (99.88%)	36
DPPIN-Krogan (LCMS)	2,211	85,133	✓	✓	2208/2211 (99.86%)	36
DPPIN-Krogan (MALDI)	2,099	78,297	✓	✓	2097/2099 (99.90%)	36
DPPIN-Yu	1,163	3,602	✓	✓	1160/1163 (99.74%)	36
DPPIN-Breitkreutz	869	39,250	✓	✓	869/869 (100.00%)	36
DPPIN-Babu	5,003	111,466	✓	✓	4997/5003 (99.88%)	36
DPPIN-Lambert	697	6,654	✓	✓	697/697 (100.00%)	36
DPPIN-Tarassov	1,053	4,826	✓	✓	1051/1053 (99.81%)	36
DPPIN-Hazbun	143	1,959	✓	✓	143/143 (100.00%)	36

Algorithm 1 Dynamic Protein Interaction Networks Construction

```
Input:
    static adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, time-aware gene expression data $S_i(t)$ $i \in \{1, 2, \dots n\}$, $t \in \{1, 2, \dots T\}$
Output:
    weighted adjacency matrix $\boldsymbol{A}^{(t)} \in \mathbb{R}^{n \times n}$, node feature matrix $\boldsymbol{X}^{(t)} \in \mathbb{R}^{n \times t}$, $t \in \{1, 2, \dots T\}$
for $t = 1 : T$ do
    /*Determine Active Proteins*/
    Determine the active probability $p^{(t)}(i)$ based on $S_i(t)$ with Eq. 1 and Eq. 2 for each protein node $i = 1, \dots, n$
    Construct the activity matrix $\boldsymbol{A}_{act}^{(t)}$ based on Eq. 3
    /*Determine Co-expressed Protein Pairs*/
    Construct the co-expression matrix $\boldsymbol{A}_{coe}^{(t)}$ based on $S_i(t)$ with Eq. 4
    /*Preserve Active and Co-expressed Protein Pairs*/
    Construct the weight adjacency matrix $\boldsymbol{A}^t$ at time $t$ based on Eq. 5
    /*Extend Node Features via Value Concatenation*/
    $X^{(t)}(i, :) = S_i(1) \mid\mid \dots \mid\mid S_i(t)$
end for
```

is summarized in Table II with other generated dynamic network data.

IV. Potential Applications

In this section, we discuss graph-based applications and tasks that could leverage network data of our DPPIN repository. For example, the datasets of DPPIN could be used for but are not limited to dense community detection, graph querying, node similarity retrieval, node property prediction, and graph property prediction in the dynamic setting.

A. Dense Community Detection

Detecting densely connected communities (i.e., node clustering or graph partitioning) is a fundamental research problem in the graph mining research community [3], [43]. However, in the dynamic setting, when the graph topology updates, the previously identified clusters are outdated. To fast capture new clusters, many algorithms are proposed such like [19], [35].

To be specific, detecting densely connected clusters can be realized in a global view or a local view with different compactness objectives. For example, the dynamic spectral clustering [35] can be described as follows.

Problem 1: Dynamic Spectral Clustering

Input: (i) a dynamic graph $\tilde{G} = \{G^{(1)}, G^{(2)}, \dots, G^{(T)}\}$, and (ii) the desired number of disjoint clusters q.

Output: q disjoint clusters $\{C_1^{(t)}, C_2^{(t)}, \dots, C_q^{(t)}\}$ minimizing the normalized cut, and $G^{(t)} = \bigcup_{i=1}^{q} C_i^{(t)}$ at each timestamp $t \in \{1, 2, \dots, T\}$.

Instead of exploring the entire graph, local clustering algorithms try to identify a dense local cluster near the user-specified seed node. For example, the problem of dynamic local clustering [19] can be generalized as follows.

Problem 2: Dynamic Local Clustering

Input: (i) a dynamic graph $\tilde{G} = \{G^{(1)}, G^{(2)}, \dots, G^{(T)}\}$, (ii) a seed node u, and (iii) the conductance upper bound ϕ.

Output: a local cluster $C^{(t)}$ near the seed node u such that the conductance score of $C^{(t)} \leq \phi$ at each timestamp $t \in \{1, 2, \ldots, T\}$.

With the dynamic network data from our DPPIN, dynamic clustering algorithms can be tested and further improved for faster and denser solutions.

B. Graph Querying

Searching whether a smaller specific query graph exists in a larger data graph has a very wide range of applications such as intrusion detection, VLSI reverse engineering, and chemical compound detection [46], [58]. To save the computational complexity for the query process, many dynamic subgraph matching methods are proposed to leverage the historical information to fast produce the *exact matching* results [28], [31], i.e., the exact positions of the query graph in the data graph. Generally speaking, the dynamic subgraph exact matching can be described as follows.

Problem 3: Dynamic Subgraph Exact Matching

Input: (i) a dynamic data graph $\tilde{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, and (ii) a query graph Q.

Output: the positions (indexed by nodes) where each node and each edge of the query Q is matched in the data graph $G^{(t)}$ at each timestamp $t \in \{1, 2, \ldots, T\}$.

To obtain the exact matching solutions on dynamic data graphs, our DPPIN repository plays a fundamental role by providing time-evolving node interactions.

C. Node Similarity Retrieval

Measuring the proximity (or similarity) score between nodes in a graph is very important for many graph mining research problems and paves the way for many real-world applications, such as ranking and recommendation [47]. Due to the intrinsic dynamics of real-world networks, retrieving the similarity ranking list for every node is time-consuming. Many efforts [37], [55], [57] have contributed to fast and accurately tracking the node proximity in the dynamic setting, where the topology is frequently evolving over time.

Formally, the proximity tracking problem in dynamic graphs [47] can be described as follows.

Problem 4: Proximity Tracking

Input: (i) a dynamic graph $\tilde{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, and (ii) a set Q of interest nodes.

Output: the top-k most related objects of each interest node in Q and their proximity (or similarity) scores at each timestamp $t \in \{1, 2, \ldots, T\}$.

Dynamic networks in our DPPIN repository enable the innovation and improvement of similarity retrieval algorithms and further link prediction applications by bringing the dynamic connections in the form of protein-protein interactions.

D. Node/Graph Property Prediction

Based on predicted properties, accurately identifying (or classifying) nodes or graphs has great importance in many domains such as drug discovery [12], [39], molecular property prediction [14], [21], and epidemic infectious pattern analysis [13], [36]. Recently, many efforts have tried to investigate the role of temporal dependencies in identifying the node property [54] and graph property [5], [36] for achieving higher classification accuracy in the dynamic setting.

In general, the node-level or graph-level classification problem based on property predictions can be instanced as follows.

Problem 5: Dynamic Node Classification

Input: (i) a dynamic graph $\tilde{G} = \{G^{(1)}, G^{(2)}, \ldots, G^{(T)}\}$, and (ii) a node label set Y

Output: a representation model that could output appropriate node label predictions for the acceptable classification accuracy against the label set Y at each timestamp $t \in \{1, 2, \ldots, T\}$.

Problem 6: Dynamic Graph Classification

Input: (i) a dynamic graph set $\{\tilde{G}_1, \tilde{G}_2, \ldots \tilde{G}_N\}$, and (ii) a graph label set Y

Output: a representation model that could output appropriate graph label predictions for the acceptable classification accuracy against the label set Y.

Our DPPIN provides 12 different classes of dynamic networks, where each network has meaningful time-dependent node connections. Therefore, DPPIN can be utilized by dynamic node classification and graph classification algorithms to verify whether they could capture temporal representations to improve the performance of static baseline algorithms.

V. Experiments

In this section, we design extensive experiments to show that network datasets from our DPPIN repository could pave the way for many graph machine learning algorithms. e.g., node-level unsupervised learning and graph-level supervised learning.

A. Dynamic Spectral Clustering (Motif-Aware)

In brief, spectral clustering aims to partition the input graph into disjoint dense clusters, and different spectral clustering algorithms choose different metrics to measure the compactness of the resulting clusters. Motif-aware spectral clustering algorithm (MSC) [6] analyzes the eigenvalue and eigenvector of the normalized motif Laplacian matrix of the input graph and then produces clusters under the constraint of the motif conductance. A motif is a high-order connected subgraph structure, and the order of a motif stands for the number of nodes in that subgraph. For example, a third-order motif can be a triangle or a three-node line. The low motif conductance score indicates that few motifs are broken during the graph partitioning, and the resulting clusters are dense. Although motif clustering has broad application scopes [6], the motif-aware spectral clustering problem on dynamic graphs remains opening. To this end, we design a novel and simple incremental solution, called MSC+T, and evaluate its performance based on different datasets of DPPIN. To be specific, MSC+T is realized based on MSC [6] and an eigenpairs (i.e., an eigenvalue and its corresponding eigenvector) tracking method [10].

TABLE III: Performance of Spectral Clustering Algorithms

Methods	DPPIN-Krogan (LCMS)	
	Motif Conductance	Time Consumption
MSC	0.0000	2.0276s
MSC+T	0.0000	0.0997s

Methods	DPPIN-Ho	
	Motif Conductance	Time Consumption
MSC	0.0000	0.7354s
MSC+T	0.7500	0.0469s

In this experiment, we select DPPIN-Krogan (LCMS) and DPPIN-Ho datasets and set the third-order motif (i.e., triangle) being preserved during graph cuts. The static spectral clustering algorithm (i.e., MSC) is directly executed on DPPIN-Krogan (LCMS) and DPPIN-Ho at timestamp $t = 29$ to report the performance, and the dynamic spectral clustering method (i.e., MSC+T) receives the clustering result at $t = 28$ then tracks the clustering for $t = 29$. We use the motif conductance to measure the compactness performance of resulting clusters, and the lower motif conductance score dictates the fewer motif structures are broken due to the graph cuts. We partition the input graph into two clusters, and the performance of MSC and MSC+T is shown in Table III.

Research Opportunities. We can observe that MSC achieves the better clustering compactness but consumes a larger amount of time. Although MSC+T outputs the solution in a fast manner, the compactness of clusters is not always ideal. An intuitive explanation is that the graph structure changes (between two consecutive timestamps) beyond the representation ability of MSC+T in terms of tracking the exact eigenvalues and eigenvectors. Using the dynamic networks from DPPIN identifies the future research direction that designing more dynamics-informative and accurate dynamic spectral clustering methods is necessary.

B. Dynamic Local Clustering (Conductance-Guided)

Unlike spectral clustering algorithms standing in a global optimization view, local clustering only cares about the local cluster compactness. PageRank-Nibble [3] is a traditional local clustering algorithm designed for static graphs, which uses the stationary distribution of random walks to search an edge-preserving dense local cluster near the seed node, guaranteeing a lower bound of graph conductance. TPPR [37] could track that stationary distribution of PageRank-Nibble and fast update the local cluster when a new graph structure arrives.

Research Opportunities. Setting the same seed node (i.e., YOL033W in DPPIN-Krogan (MALDI) and YER052C in DPPIN-Ho), we report the local clustering compactness performance (i.e., conductance) of PageRank-Nibble (static result at $t = 28$) and TPPR (tracking result at $t = 29$ from $t = 28$) on two dynamic network datasets in Table IV. Similar to the performance comparison in Table III, the dynamic local clustering method (i.e., TPPR) also fast provides the solution but does not always guarantee optimal. In this viewpoint, dynamic networks of DPPIN provide the foundation for future advanced dynamic local clustering algorithms.

TABLE IV: Performance of Local Clustering Algorithms

Methods	DPPIN-Krogan (MALDI)	
	Conductance	Time Consumption
PageRank-Nibble	0.3775	4.9119s
TPPR	0.7109	4.2645s

Methods	DPPIN-Ho	
	Conductance	Time Consumption
PageRank-Nibble	0.3364	1.2128s
TPPR	0.6486	0.9415s

C. Dynamic Subgraph Matching (Clique-Based)

Cliques are complete connected subgraphs, which means each pair of nodes is connected. A k-clique means that the clique structure has k nodes, and every two nodes are connected. Examples of k-cliques (i.e., $k = \{3, 4, 5\}$) are shown in Figure 3.

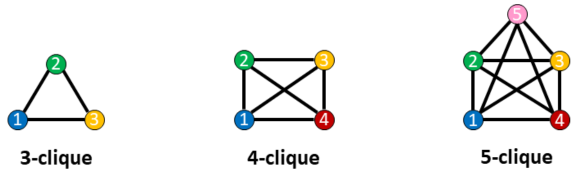


Fig. 3: Examples of k-cliques.

Research Opportunities. Determining whether a data graph contains a specific clique and listing all matches has potential research value in many domains, such as identifying protein complexes and discovering new groups of functionally associated proteins [1]. Aiming at 3-cliques and 4-cliques, we use G-Finder [31] to enumerate all matched subgraphs in all dynamic network datasets of DPPIN at $t = 28$ and $t = 29$, respectively. The number of matched subgraphs is reported in Table V. We can observe that different network datasets of DPPIN have very different distributions of the number of cliques. Moreover, even some network dataset has a considerably large number of nodes, but it is sparse and corresponding the number of cliques is small.

TABLE V: Number of Exact Matched Subgraphs in DPPIN

Dynamic PPINs	Number of 3-cliques		Number of 4-cliques	
	$t = 28$	$t = 29$	$t = 28$	$t = 29$
DPPIN-Uetz	0	1	0	0
DPPIN-Ito	3	1	0	0
DPPIN-Ho	173	60	79	9
DPPIN-Gavin	285	589	184	503
DPPIN-Krogan (LCMS)	129	202	46	79
DPPIN-Krogan (MALDI)	557	371	490	171
DPPIN-Yu	0	0	0	0
DPPIN-Breitkreutz	396	422	233	244
DPPIN-Babu	111	269	26	109
DPPIN-Lambert	6	1	1	0
DPPIN-Tarassov	3	8	0	2
DPPIN-Hazbun	7	32	1	20

D. Proximity Tracking (PageRank-Based)

The goal of proximity tracking is to monitor important nodes towards the selected interest node at each observed timestamp and how the ranking of these important nodes changes as time evolves.

Here, we use the PageRank-based proximity tracking method [47] and report the top-10 most similar proteins to the query protein YDR377W at two timestamps, $t = 25$ and $t = 35$, in DPPIN-Babu dynamic network dataset. As shown in Table VI, the most similar protein to YDR377W is itself, and the ranking of following similar proteins changes at different timestamps for evolving structures.

TABLE VI: Top-10 Proximity Tracking Result of YDR377W

Query Protein	Target Network	Similarity Rankings	
		$t = 25$	$t = 35$
YDR377W	DPPIN-Babu	YDR377W	YDR377W
		YBL099W	YMR001C
		YPL271W	YDR291W
		YNL225C	YJR090C
		YDR126W	YER093C
		YIL026C	YOR032C
		YNR006W	YOR307C
		YHR155W	YDR408C
		YGL263W	YNL147W
		YGR206W	YHR166C

E. Dynamic Node Classification (Attention-Aware)

To identify the node property (e.g., class labels), inspired by Graph Attention Network (GAT) [51], Temporal Graph Attention Network (TGAT) [54] is a recently proposed graph neural network model for aggregating the temporal-topological neighborhood information into the node representation vector (through attention mechanisms) for improving the node classification accuracy in the dynamic setting.

TABLE VII: Performance of Graph Neural Networks w.r.t. Nodel-level Classification Accuracy

Methods	DPPIN-Ito	
	Training Acc.	Testing Acc.
GAT	0.9426 ± 0.0004	0.9435 ± 0.0078
TGAT	0.9478 ± 0.0005	0.9558 ± 0.0096

Here, we want to investigate whether TGAT could achieve higher node classification than GAT in our DPPIN-Ito dataset by capturing the temporal metabolic evolution information. Based on the chronological order, we split the first 70% temporal edges as the training set, the middle 15% temporal edges as the validation set, and the last 15% temporal edges as the testing set. Then, we report the average node classification accuracy and the standard deviation of GAT and TGAT with the varying number of test samples in Table VII, where GAT and TGAT are converged under the same condition (e.g., the same number of layers, same dropout probability, same learning rate, and same training epochs). Moreover, to make the static method GAT could take the dynamic network DPPIN-Ito as input, we use Reduced Graph Representation [36] to map temporal graphs into dynamics-preserving static graphs. In Table VII, we can observe that TGAT achieves the higher training accuracy and testing accuracy in terms of the node classification, which suggests that temporal metabolic patterns are helpful in regularizing the representation learning process to help determine the category of nodes, TGAT could capture those temporal metabolic patterns in DPPIN-Ito to help identify protein properties.

F. Dynamic Graph Classification (Few-Shot Learning)

To a large extent, accurately classifying different class dynamic graphs and saving labeling workload requires corresponding graph embedding algorithms to encode the class-distinctive graph property (i.e., structure and/or feature evolution pattern) appropriately in the learning process.

Here, we conduct the dynamic graph classification experiment on state-of-the-art baseline algorithms and see if datasets in DPPIN could help indicate some latent research opportunities. According to [5], tdGraphEmbed is the first graph-level dynamic graph representation learning algorithm, which could capture the representation of each observed snapshot. We use tdGraphEmbed and call the sum pooling function to aggregate snapshot representations for the entire dynamic graph representation vector for further classification. We also involve two static graph-level representation learning algorithms, Graph2Vec [34] and GL2Vec [11]. Furthermore, to make static algorithms deal with dynamic inputs, we use Reduced Graph Representation [36] to map evolving graphs into dynamics-preserving static graphs. In our DPPIN, there are 12 graph classes (indicated by network names) in total, and each class has one large dynamic graph. In each class, we take subgraphs as samples. To be specific, for each graph, we take a subgraph by extracting a length of 5 timestamps with every 3 timestamps elapsed. For example, t_1, t_2, t_3, t_4, t_5 together compose the first dynamic subgraph, and t_4, t_5, t_6, t_7, t_8 together compose the second dynamic subgraph. The extracted subgraph shares the class label with its original entire graph, and we have 11 dynamic subgraphs per class. Therefore, we can sample subgraphs and form the N-way k-shot classification setting (i.e., N classes and k samples per class during meta-training and no shared class labels between the meta-training and meta-testing stages) with two few-shot classifiers: ProtoNet and its special cases kNN [42]. We split 8 classes for the meta-training and 4 classes for the meta-testing, and we randomly shuffle this split 4 times to report the testing accuracy of the meta-testing stage in Table VIII.

TABLE VIII: Graph-level Classification Accuracy during Meta-testing with Varying Few-shot Settings

Methods	Few-Shot Setting	
	3 way - 5 shot	3 way - 3 shot
Graph2Vec + kNN[8]	–	–
GL2Vec + kNN	0.0717 ± 0.0900	0.0917 ± 0.0793
tdGraphEmbed + kNN	0.2167 ± 0.1736	0.1056 ± 0.0814
Graph2Vec + ProtoNet	0.3792 ± 0.0459	0.3958 ± 0.0731
GL2Vec + ProtoNet	0.7100 ± 0.0361	0.6625 ± 0.0407
tdGraphEmbed + ProtoNet	0.6562 ± 0.1882	0.6791 ± 0.1141

[8] Cannot get the results within 48 hours on a Linux machine with a single NVIDIA Tesla V100 32GB GPU

Research Opportunities. As shown in Table VIII, tdGraphEmbed+ProtoNet achieves the best performance in both few-shot settings. An intuitive explanation is that, compared just simply calling Reduced Graph Representation [36] for static algorithms, the dynamics representation learning process of tdGraphEmbed is more suitable for capturing the bioinformatic evolution patterns. However, tdGraphEmbed takes each snapshot individually, and we just call sum pooling to aggregate all of them. But there is still an opening question about how to aggregate each snapshot representation more reasonably. For instance, if there are some snapshots globally shared by different class dynamic graphs (i.e., different class graphs share several same or similar snapshots), then the simply summarized graph-level representation may not be class-distinctive. Based on this observation, we know that capturing the lifelong evolution pattern may be an essential factor in designing the next-generation dynamic graph-level representation learning algorithms. Some hints may be found in applying the attention mechanism [50] of current node-level representation learning methods [54].

VI. Future Work

Although every gene coding protein that appeared in our DPPIN repository has observed time-aware expression values based on the GSE3431 gene expression data [48], some protein nodes are currently lacking meaningful labels, as shown in the Saccharomyces Genome Database[9]. Consequently, some dynamic network datasets in our DPPIN repository have no 100.00% label rate, which can be seen in Table II. This defect may hinder executing supervised learning algorithms on the datasets of DPPIN, such as some node classification tasks. In the future, we will continually follow related research of unlabeled proteins of our DPPIN repository and incrementally update the label set of DPPIN.

Acknowledgement

This work is supported by National Science Foundation under Award No. IIS-1947203, IIS-2117902, and IIS-2137468. The views and conclusions are those of the authors and should not be interpreted as representing the official policies of the funding agencies or the government.

VII. Conclusion

In this paper, we first release a new repository of dynamic network datasets from the bioinformatics domain, called DPPIN. To be specific, each network of DPPIN consists of dynamic protein-protein interactions generated by analyzing active and co-expressed protein pairs at corresponding timestamps. Then, we connect our DPPIN repository with the real-world applications that could take advantage of it through corresponding experiments, such as dense community detection, graph querying, node similarity retrieval, and node/graph property prediction in the dynamic setting. Finally, we indicate the future direction to continually improve the utility of our DPPIN data repository. We hope the various evolution patterns in each network dataset of the DPPIN repository could provide guidance or constraint for developing more effective dynamic graph machine learning and mining algorithms.

[9] https://www.yeastgenome.org/

References

[1] Balázs Adamcsek, Gergely Palla, Illés J. Farkas, Imre Derényi, and Tamás Vicsek. CFinder: locating cliques and overlapping modules in biological networks. *Bioinformatics*, 2006.

[2] Charu C. Aggarwal and Karthik Subbian. Evolutionary network analysis: A survey. *ACM Comput. Surv.*, 2014.

[3] Reid Andersen, Fan R. K. Chung, and Kevin J. Lang. Local graph partitioning using pagerank vectors. In *FOCS*, 2006.

[4] Mohan Babu, James Vlasblom, Shuye Pu, Xinghua Guo, Chris Graham, Björn DM Bean, Helen E Burston, Franco J Vizeacoumar, Jamie Snider, Sadhna Phanse, et al. Interaction landscape of membrane-protein complexes in saccharomyces cerevisiae. *Nature*, 2012.

[5] Moran Beladev, Lior Rokach, Gilad Katz, Ido Guy, and Kira Radinsky. tdgraphembed: Temporal dynamic graph-level embedding. In *CIKM*, 2020.

[6] Austin R. Benson, David F. Gleich, and Jure Leskovec. Higher-order organization of complex networks. *Science*, 2016.

[7] Ashton Breitkreutz, Hyungwon Choi, Jeffrey R Sharom, Lorrie Boucher, Victor Neduva, Brett Larsen, Zhen-Yuan Lin, Bobby-Joe Breitkreutz, Chris Stark, Guomin Liu, et al. A global protein kinase and phosphatase interaction network in yeast. *Science*, 2010.

[8] Deepayan Chakrabarti and Christos Faloutsos. Graph mining: Laws, generators, and algorithms. *ACM Comput. Surv.*, 2006.

[9] Bolin Chen, Weiwei Fan, Juan Liu, and Fang-Xiang Wu. Identifying protein complexes and functional modules—from static ppi networks to dynamic ppi networks. *Briefings in Bioinformatics*, 2014.

[10] Chen Chen and Hanghang Tong. Fast eigen-functions tracking on dynamic graphs. In *SDM*, 2015.

[11] Hong Chen and Hisashi Koga. Gl2vec: Graph embedding enriched by line graphs with edge features. In *ICONIP*, 2019.

[12] Hanjun Dai, Bo Dai, and Le Song. Discriminative embeddings of latent variable models for structured data. In *ICML*, 2016.

[13] Tyler Derr, Yao Ma, Wenqi Fan, Xiaorui Liu, Charu C. Aggarwal, and Jiliang Tang. Epidemic graph convolutional network. In *WSDM*, 2020.

[14] David Duvenaud, Dougal Maclaurin, Jorge Aguilera-Iparraguirre, Rafael Gómez-Bombarelli, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P. Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NeurIPS*, 2015.

[15] Dongqi Fu, Yikun Ban, Hanghang Tong, Ross Maciejewski, and Jingrui He. DISCO: comprehensive and explainable disinformation detection. In *CIKM*, 2022.

[16] Dongqi Fu, Liri Fang, Ross Maciejewski, Vetle I. Torvik, and Jingrui He. Meta-learned metrics over multi-evolution temporal graphs. In *KDD*, 2022.

[17] Dongqi Fu and Jingrui He. Sdg: A simplified and dynamic graph neural network. In *SIGIR*, 2021.

[18] Dongqi Fu, Zhe Xu, Bo Li, Hanghang Tong, and Jingrui He. A view-adversarial framework for multi-view network embedding. In *CIKM*, 2020.

[19] Dongqi Fu, Dawei Zhou, and Jingrui He. Local motif clustering on time-evolving graphs. In *KDD*, 2020.

[20] Anne-Claude Gavin, Patrick Aloy, Paola Grandi, Roland Krause, Markus Boesche, Martina Marzioch, Christina Rau, Lars Juhl Jensen, Sonja Bastuck, Birgit Dümpelfeld, et al. Proteome survey reveals modularity of the yeast cell machinery. *Nature*, 2006.

[21] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.

[22] Tony R Hazbun, Lars Malmström, Scott Anderson, Beth J Graczyk, Bethany Fox, Michael Riffle, Bryan A Sundin, J Derringer Aranda, W Hayes McDonald, Chun-Hwei Chiu, et al. Assigning function to yeast proteins by integration of technologies. *Molecular Cell*, 2003.

[23] Shubhada R Hegde, Palanisamy Manimaran, and Shekhar C Mande. Dynamic changes in protein functional linkage networks revealed by integration with gene expression data. *PLoS Computational Biology*, 2008.

[24] Yuen Ho, Albrecht Gruhler, Adrian Heilbut, Gary D Bader, Lynda Moore, Sally-Lin Adams, Anna Millar, Paul Taylor, Keiryn Bennett, Kelly Boutilier, et al. Systematic identification of protein complexes in saccharomyces cerevisiae by mass spectrometry. *Nature*, 2002.

[25] Lun Hu, Xiaojuan Wang, Yu-An Huang, Pengwei Hu, and Zhu-Hong You. A survey on computational models for predicting protein–protein interactions. *Briefings in Bioinformatics*, 2021.

[26] Takashi Ito, Tomoko Chiba, Ritsuko Ozawa, Mikio Yoshida, Masahira Hattori, and Yoshiyuki Sakaki. A comprehensive two-hybrid analysis to explore the yeast protein interactome. *PNAS*, 2001.

[27] Seyed Mehran Kazemi, Rishab Goel, Kshitij Jain, Ivan Kobyzev, Akshay Sethi, Peter Forsyth, and Pascal Poupart. Representation learning for dynamic graphs: A survey. *J. Mach. Learn. Res.*, 2020.

[28] Kyoungmin Kim, In Seo, Wook-Shin Han, Jeong-Hoon Lee, Sungpack Hong, Hassan Chafi, Hyungyu Shin, and Geonhwa Jeong. Turboflux: A fast continuous subgraph matching system for streaming graph data. In *SIGMOD*, 2018.

[29] Nevan J Krogan, Gerard Cagney, Haiyuan Yu, Gouqing Zhong, Xinghua Guo, Alexandr Ignatchenko, Joyce Li, Shuye Pu, Nira Datta, Aaron P Tikuisis, et al. Global landscape of protein complexes in the yeast saccharomyces cerevisiae. *Nature*, 2006.

[30] Jean-Philippe Lambert, Jeffrey Fillingham, Mojgan Siahbazi, Jack Greenblatt, Kristin Baetz, and Daniel Figeys. Defining the budding yeast chromatin-associated interactome. *Molecular Systems Biology*, 2010.

[31] Lihui Liu, Boxin Du, Jiejun Xu, and Hanghang Tong. G-finder: Approximate attributed subgraph matching. In *IEEE BigData*, 2019.

[32] Zhiwei Liu, Yingtong Dou, Philip S. Yu, Yutong Deng, and Hao Peng. Alleviating the inconsistency problem of applying graph neural network to fraud detection. In *SIGIR*, 2020.

[33] Kelong Mao, Xi Xiao, Jieming Zhu, Biao Lu, Ruiming Tang, and Xiuqiang He. Item tagging for information retrieval: A tripartite graph neural network based approach. In *SIGIR*, 2020.

[34] Annamalai Narayanan, Mahinthan Chandramohan, Rajasekar Venkatesan, Lihui Chen, Yang Liu, and Shantanu Jaiswal. graph2vec: Learning distributed representations of graphs. *CoRR*, 2017.

[35] Huazhong Ning, Wei Xu, Yun Chi, Yihong Gong, and Thomas S. Huang. Incremental spectral clustering with application to monitoring of evolving blog communities. In *SDM*, 2007.

[36] Lutz Oettershagen, Nils M. Kriege, Christopher Morris, and Petra Mutzel. Temporal graph kernels for classifying dissemination processes. In *SDM*, 2020.

[37] Naoto Ohsaka, Takanori Maehara, and Ken-ichi Kawarabayashi. Efficient pagerank tracking in evolving networks. In *KDD*, 2015.

[38] R Ranjani Rani, D Ramyachitra, and A Brindhadevi. Detection of dynamic protein complexes through markov clustering based on elephant herd optimization approach. *Scientific Reports*, 2019.

[39] Bernhard Schölkopf, Koji Tsuda, and Jean-Philippe Vert. *Kernel Methods in Computational Biology*. MIT press, 2004.

[40] Xianjun Shen, Li Yi, Xingpeng Jiang, Tingting He, Xiaohua Hu, and Jincai Yang. Mining temporal protein complex based on the dynamic pin weighted with connected affinity and gene co-expression. *PloS one*, 2016.

[41] Xianjun Shen, Li Yi, Xingpeng Jiang, Tingting He, Jincai Yang, Wei Xie, Po Hu, and Xiaohua Hu. Identifying protein complex by integrating characteristic of core-attachment into dynamic ppi network. *PloS one*, 2017.

[42] Jake Snell, Kevin Swersky, and Richard S. Zemel. Prototypical networks for few-shot learning. In *NeurIPS*, 2017.

[43] Daniel A. Spielman and Shang-Hua Teng. Nearly-linear time algorithms for graph partitioning, graph sparsification, and solving linear systems. In *STOC*, 2004.

[46] Hanghang Tong, Christos Faloutsos, Brian Gallagher, and Tina Eliassi-Rad. Fast best-effort pattern matching in large attributed graphs. In *KDD*, 2007.

[44] Kirill Tarassov, Vincent Messier, Christian R Landry, Stevo Radinovic, Mercedes M Serna Molina, Igor Shames, Yelena Malitskaya, Jackie Vogel, Howard Bussey, and Stephen W Michnick. An in vivo map of the yeast protein interactome. *Science*, 2008.

[45] Ratana Thanasomboon, Saowalak Kalapanulak, Supatcharee Netrphan, and Treenut Saithong. Exploring dynamic protein-protein interactions in cassava through the integrative interactome network. *Scientific Reports*, 2020.

[47] Hanghang Tong, Spiros Papadimitriou, Philip S. Yu, and Christos Faloutsos. Proximity tracking on time-evolving bipartite graphs. In *SDM*, 2008.

[48] Benjamin P Tu, Andrzej Kudlicki, Maga Rowicka, and Steven L McKnight. Logic of the yeast metabolic cycle: temporal compartmentalization of cellular processes. *Science*, 2005.

[49] Peter Uetz, Loic Giot, Gerard Cagney, Traci A Mansfield, Richard S Judson, James R Knight, Daniel Lockshon, Vaibhav Narayan, Maithreyan Srinivasan, Pascale Pochart, et al. A comprehensive analysis of protein–protein interactions in saccharomyces cerevisiae. *Nature*, 2000.

[50] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

[51] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2018.

[52] Jianxin Wang, Xiaoqing Peng, Min Li, and Yi Pan. Construction and application of dynamic protein interaction network based on time course gene expression data. *Proteomics*, 2013.

[53] Jianxin Wang, Xiaoqing Peng, Wei Peng, and Fang-Xiang Wu. Dynamic protein interaction network construction and applications. *Proteomics*, 2014.

[54] Da Xu, Chuanwei Ruan, Evren Körpeoglu, Sushant Kumar, and Kannan Achan. Inductive representation learning on temporal graphs. In *ICLR*, 2020.

[55] Minji Yoon, Woojeong Jin, and U Kang. Fast and accurate random walk with restart on dynamic graphs with guarantees. In *WWW*, 2018.

[56] Haiyuan Yu, Pascal Braun, Muhammed A Yıldırım, Irma Lemmens, Kavitha Venkatesan, Julie Sahalie, Tomoko Hirozane-Kishikawa, Fana Gebreab, Na Li, Nicolas Simonis, et al. High-quality binary protein interaction map of the yeast interactome network. *Science*, 2008.

[57] Hongyang Zhang, Peter Lofgren, and Ashish Goel. Approximate personalized pagerank on dynamic graphs. In *KDD*, 2016.

[58] Shijie Zhang, Shirong Li, and Jiong Yang. GADDI: distance index based subgraph matching in biological networks. In *EDBT*, 2009.

[59] Si Zhang, Hanghang Tong, Jiejun Xu, and Ross Maciejewski. Graph convolutional networks: a comprehensive review. *Computational Social Networks*, 2019.

[60] Yijia Zhang, Hongfei Lin, Zhihao Yang, and Jian Wang. Construction of dynamic probabilistic protein interaction networks for protein complex identification. *BMC Bioinformatics*, 2016.

[61] Yijia Zhang, Hongfei Lin, Zhihao Yang, Jian Wang, Yiwei Liu, and Shengtian Sang. A method for predicting protein complex in dynamic ppi networks. *BMC Bioinformatics*, 2016.

[62] Dawei Zhou, Lecheng Zheng, Dongqi Fu, Jiawei Han, and Jingrui He. Mentorgnn: Deriving curriculum for pre-training gnns. In *CIKM*, 2022.

[63] Dawei Zhou, Lecheng Zheng, Jiawei Han, and Jingrui He. A data-driven graph generative model for temporal interaction networks. In *KDD*, 2020.

[64] Yao Zhou, Jianpeng Xu, Jun Wu, Zeinab Taghavi Nasrabadi, Evren Körpeoglu, Kannan Achan, and Jingrui He. PURE: positive-unlabeled recommendation with generative adversarial network. In *KDD*, 2021.