Network Embedding via Motifs

PING SHAO, Zhejiang University, China
YANG YANG*, Zhejiang University, China
SHENGYAO XU, Finvolution Group Inc., China
CHUNPING WANG, Finvolution Group Inc., China

Network embedding has emerged as an effective way to deal with downstream tasks, such as node classification [17, 32, 43]. Most existing methods leverage multi-similarities between nodes such as *connectivity* which considers vertices that are closely connected to be similar and *structural similarity* which is measured by assessing their relations to neighbors; while these methods only focus on static graphs. In this work, we bridge connectivity and structural similarity in a uniform representation via motifs, and consequently present an algorithm for Learning Embeddings by leveraging Motifs Of Networks (LEMON), which aims to learn embeddings for vertices and various motifs. Moreover, LEMON is inherently capable of dealing with inductive learning tasks for dynamic graphs. To validate the effectiveness and efficiency, we conduct various experiments on two real-world datasets and five public datasets from diverse domains. Through comparison with state-of-the-art baseline models, we find that LEMON achieves significant improvements in downstream tasks. We release our code on Github at https://github.com/larry2020626/LEMON. [1]

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**.

Additional Key Words and Phrases: Motif, Network embedding, Motif super-vertex, Motif embedding

ACM Reference Format:
Ping Shao, Yang Yang*, Shengyao Xu, and Chunping Wang. 2021. Network Embedding via Motifs. *ACM Trans. Knowl. Discov. Data.* 1, 1, Article 1 (January 2021), 20 pages. https://doi.org/10.1145/3473911

1 INTRODUCTION

Graphs are an efficient and natural way to represent data from various domains such as social networks and academic networks. Graph structures can be various and irregular, as the size of graphs are variable and unordered vertices may have more or less neighbors [50]. Graph embedding, also known as network embedding, which aims to learn the representation of vertices, has emerged as an effective methodology for many downstream applications, such as node classification, link prediction, *etc* [17, 32, 43]. Many algorithms employ random walks to generate a corpus, which is then fed into the Skip-gram model [28] to learn latent embeddings. The basic concept involves automatically projecting vertices of a given network into a low-dimensional latent space in which *similar* vertices are close to each other. A crucial issue is how to define the vertex-similarity measure that is most appropriate to the downstream application. Previous works in the literature have taken different vertex-similarity measures into consideration such as, *connectivity* and *structural similarity*.

[1]Corresponding author: Yang Yang.

Authors' addresses: Ping Shao, pinshao006@gmail.com, Zhejiang University, Hangzhou, Zhejiang, China; Yang Yang*, Zhejiang University, Hangzhou, Zhejiang, China, yangya@zju.edu.cn; Shengyao Xu, xushengyao@xinye.com, Finvolution Group Inc. Shanghai, China; Chunping Wang, wangchunping02@xinye.com, Finvolution Group Inc. Shanghai, China.

1556-4681/2021/1-ART1 $15.00
https://doi.org/10.1145/3473911

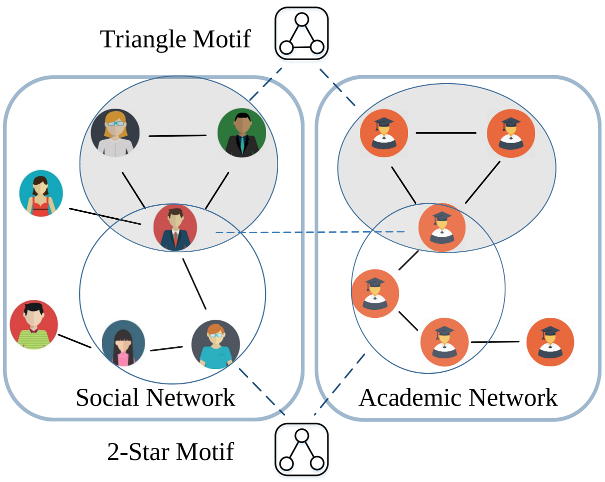


Fig. 1. Illustration of motifs in different networks, such as social network and academic network.

Connectivity considers vertices that are closely connected or have many common neighbors to be similar to each other [17, 32, 43]; on the other hand, structural similarity measures whether two vertices share similar local structures [35]. Some existing works have tried to take multiple similarities into consideration, for example, VERSE [46] propose three similarities: community structure, roles and structural equivalence. It is worth noting that community structure is substantially equal to connectivity (nodes connected or have many common neighbors) while structural similarity can covers roles and structural equivalence. For real-world datasets, researchers usually may encounter with dynamic scenes: in the telecommunication datasets, users call each other dynamically; in the academic networks, collaborations are formed and new papers are published continuously. However, to our best knowledge, most approaches mentioned above only learn representations for vertices from the perspective of one kind of similarity; moreover, most existing methods are unsuitable to generate embeddings for unseen vertices.

Some existing works have already focused on inductive learning tasks, such as GraphSAGE [18] which represents vertices by aggregating features from neighborhood. However, aggretating features from neighbors alone only consider connectivity might fail in tasks that value structural property. Take telecom fraud detection as an example, frauds behave differently from normal users which can be reflected by the structural patterns of anomalies [53]. As a result, aforehead mentioned method is inappropriate for datasets which value structural similarity and not capable enough of capturing comprehensive features of complex real-world datasets. Therefore, we propose to study a problem: *how can we simultaneously measure connectivity and structural similarity in a uniform representation and in the meantime be able to deal with inductive learning tasks*?

As fundamental building blocks of networks [29], motifs describe small subgraph patterns with specific connections among vertices, which represent the structures and contains the connection information of vertices inside. Motifs are effective and crucial in a range of domains, including bioinformatics, neuroscience, biology and social networks [55]. Figure 1 illustrates triangle motif in different networks; in the social network, three close friends form a triangle motif (△); in the academic network, three scholars who often collaborate together and publish papers form another one. Motifs contain rich information and can reveal semantic information of vertices; for example, in the social network, a vertex with a few ( ) represents a person who tends to introduce her friends to know each other, in contrast to vertices surrounded by ( ). A few existing works have tried to generate embeddings via motifs [23, 41, 59]. To the best of our knowledge, most of these methods only focused on node embedding for static graphs.

In light of the above, to address these issues, we propose a method that considers multi-similarities via *motifs* to learn representations. Apart from other motif-related methods, the major difference lies in that our proposed method explicitly proposes motifs as super-vertices, instead of implicitly utilizing motifs to exert influence when learning representations of vertices. Moreover, motifs allow us to implement an inductive learning method; in particular, we can effectively obtain the new vertex's representation by embedding vectors of its related motifs. In this paper, we propose a novel algorithm framework: Learning Embeddings based on Motifs Of the Network (LEMON). LEMON puts various motifs into the network as super-vertices and constructs a heterogeneous network consisting of relations between motif super-vertices and the original vertices in the network. Furthermore, in order to incorporate the heterogeneous network into the Skip-gram model [28], we propose a motif-step random walk strategy that ensures that vertices with high connectivity or high structural similarity will appear close together in the corpus. Thus, our framework is able to simultaneously measure both connectivity and structural similarity. The main contributions of our paper can be summarized as follows:

- We propose to study the problem of uniformly learning network embedding by measuring connectivity and structural similarity.
- We present a new framework, named LEMON, which leverages motif super-vertex to generate embeddings for both vertices and motifs and can be applied for inductive learning tasks.
- We employ two real-world datasets and five public datasets to evaluate the effectiveness and efficiency of our proposed framework on four different tasks. The experimental results indicate that LEMON achieves superior performances to **eleven** state-of-the-art baseline models.

Organization. The remainder of this paper is organized as follows. Section 2 provides a detailed description of the problem we study; consequently Section 3 illustrates the details of our proposed approach. In Section 4, all experimental settings and results are presented along with corresponding analysis. Section 5 summarizes the related works on network embedding and motifs. Finally, Section 6 concludes this paper.

2 PRELIMINARIES

2.1 Problem Definition

In this paper, we use lower-case letters to indicate scalar parameters and bold letters for vectors; moreover, the superscript i of the bold vector symbol represents the i-th dimension of the vector.

Given an undirected graph $G = (V, E)$, where V denotes vertices and E represents edges respectively, $E \in (V \times V)$. Our goal is to map vertices and various motifs into a low dimensional space R^d, with $d \ll |V|$, capturing the connectivity and structural information of vertices via exploiting the frequently occurring substructures (motifs).

Definition 1.1. (Motifs): Given a graph $G = (V, E)$, motifs are subgraphs $G' = (V', E')$ that recur significantly in statistics, where $V' \subset V$; $E' \subset E$ and $|V'| \ll |V|$.

2-vertex motif is an alias of edge. 3-vertex and 4-vertex motifs are widely used in network embedding [11, 23, 39, 41, 51, 54] and graphlet counting [4, 6]. Thus, in this paper, we utilize 2, 3, 4-vertex motifs as illustrated in Figure 2 and indicated as M_0 to M_8.

Definition 1.2. (Motif Count Vector): For a vertex u in the network, we calculate motif count vector $\boldsymbol{c_u}$; the i-th dimension of $\boldsymbol{c_u}$ indicates the number of i-th type motif that contains vertex u.

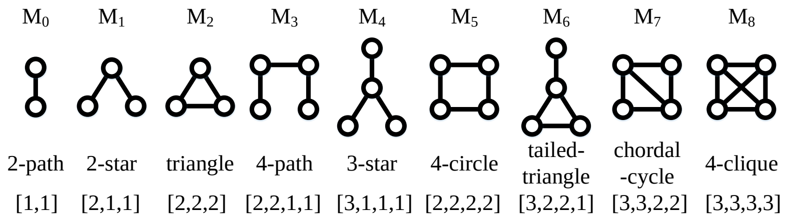


Fig. 2. Illustration of 2, 3 and 4-vertex undirected motifs, indicated as M_0 to M_8. In this paper, we only consider undirected graphs for simplicity. Our framework can be easily applied to directed graphs by adopting directed motifs.

2.2 Motif Extraction

Motif extraction has been a heated topic in data mining domain. Ribeiro et al. [36] divide motif extraction algorithms into exact subgraph counting algorithms [4, 20, 33] and approximate subgraph counting algorithms [10, 21]. Classical methods enumerate all connected motifs, result in enormous time complexity while analytic algorithms [4, 33] calculate motif count through math derivation. Compared to exact subgraph counting algorithms, approximate subgraph counting algorithms trade off accuracy for time complexity to some extent, which can be divided into random walk algorithms [21], and color-coding algorithms [10]. From another perspective, motif extraction algorithms can be divided into global motif counting algorithms and local motif counting algorithms [12, 20, 27, 49]. In this paper, we adopt an exact subgraph counting algorithm Orca [20], which calculates the motif count vector for each vertex; the time complexity of Orca is $O(k \cdot |E| + T_4)$ and more details of execution time can be found in Section 4.7.

3 OUR APPROACH

We propose a novel representation learning method named LEMON to bridge connectivity and structural similarity in a universal form and learn desirable representations for vertices and various motifs, which is suitable for inductive learning tasks.

3.1 LEMON: Learning Representations for Vertices and Motifs

The proposed framework LEMON is expected to do the following: 1) bridge connectivity and structural similarity; 2) learn embeddings for vertices and motifs; 3) support inductive learning scenarios.

We extract the structural information of vertices via motifs: consider two vertices to be structurally similar if their motif count vectors are similar. In our model, we put all motifs into the network as super-vertices, as illustrated in Algorithm 1. The structural edges between vertices and corresponding motif super-vertices are constructed; the weight of the structural edge between i-th motif super-vertex M_i and vertex u, denoted as $w(u, M_i)$, is proportional to c_u^i, which means the number of i-th type of motif containing vertex u, as shown below:

$$w_{(u,M_i)} = \frac{c_u^i}{\sum_{v \in V} c_v^i} \tag{1}$$

As can be seen from Figure 3, the weight of structural edge between the triangle motif super-vertex and vertex 6 ((5, 6, 7), (6, 8, 9)) is larger than the weight of the structural edge between the triangle motif super-vertex and vertex 7 (5, 6, 7); the weight of the structural edge between the 2-star motif super-vertex and vertex 4 ((1, 2, 4), (1, 3, 4), (2, 3, 4), (2, 4, 5), (3, 4, 5), (1, 4, 5)) is larger than the

Algorithm 1 LEMON Algorithm

```
Input:  graph $G = (V, E)$, motifs $M$
Output:  embedding vectors for vertices and motifs
 1: Extract Motifs
 2: Add all motifs into the network as super-vertices
 3: for each vertex $u \in V$ do
 4:     Construct structural edge between $u$ and each super-vertex
 5: end for
 6: for each vertex $u \in V$ do
 7:     W=Motif-Step RandomWalk($G$, $u$, $q$, $length$)
 8: end for
 9: $Skip\text{-}gram(W)$, get embedding vectors
10: ----------------------------------------------------------------
11: Motif-Step RandomWalk($G$, $u$, $q$, $length$)
12: Initialize walk $W$ as $[u]$.
13: while $|W| <$ length do
14:     $u_{cur} = W[-1]$
15:     Sample x $\sim Uniform[0, 1]$
16:     if $x < q$ and $u_{cur} \in V$ then
17:         According to edge weight, pick one motif super-vertex $M_i$
18:         Add motif super-vertex $M_i$ into $W$
19:     else
20:         neighs $s$ = GetNeighs($V$, $u_{cur}$)
21:         Randomly pick one vertex from $s$ and add into $W$
22:     end if
23: end while
```

weight of the structural edge between the 2-star motif super-vertex and vertex 2 ((1, 2, 4), (2, 3, 4), (2, 4, 5)).

Motif-Step Random Walk: Inspired by natural language processing models in which the embedding vectors of words are trained based on the corpus, DeepWalk [32] maps the word-context concept in a text corpus into the network by taking vertices as words and executing random walk to generate paths as "context". It then utilizes the Skip-gram model [28] to learn the representation of the vertices that facilitate the prediction of its context. Considering that we have two types of edges in the network: structural edges and normal edges, we propose a parameter q to determine the probability of random walking on structural edges or normal edges [2]. Clearly, the embedding vector of the vertex should be more similar to ones of the motif super-vertices that contain this vertex more frequently than others. In response, we improve the random walk strategy by proposing a new strategy named motif-step random walk; π_u represents the transition probability of the next step of the motif-step random walk for vertex u.

[2]We will refer to the vertices (edges) in the given network as normal vertices (edges) later in this paper to avoid ambiguity.

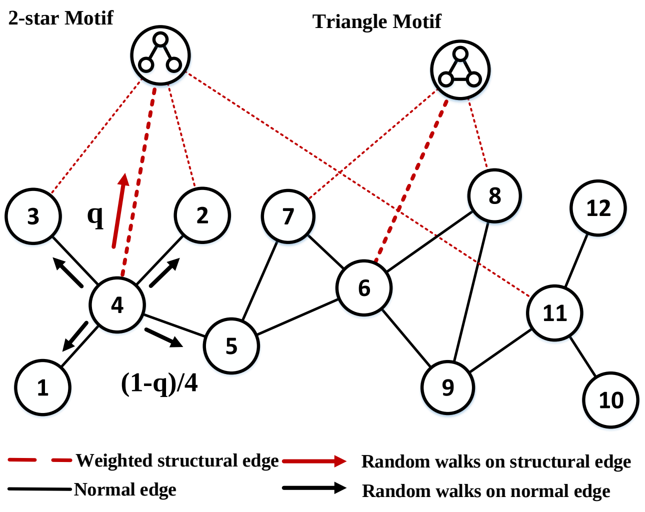


Fig. 3. An illustration of LEMON. LEMON puts motifs into network as super-vertices and constructs structural edges between vertices and motif super-vertices (for brevity, only a part of edges are drawn), where the weight is proportional to the number of certain type of motif containing this vertex. The model adopts $q \in [0, 1]$ to indicate the transition probability of traveling on different type of edges. If a walk travels to vertex 4, the next step probability distribution is q for the 2-star motif super-vertex and $\frac{1-q}{4}$ for vertex 1, 2, 3 and 5. It is worth noticing that structurally similar vertices such as vertex 4 and vertex 11 (both contains several 2-star motifs in their neighborhood) are bridged through motif super-vertex while these two vertices are far away from each other in original network.

$$\pi_u = \begin{cases} \frac{q w_{(u,M_i)}}{\sum_{j=0}^{8} w_{(u,M_j)}} & \text{motif super-vertex } M_i \\ \frac{1-q}{|N(u)|} & \text{nerighbor vertex} \end{cases} \tag{2}$$

If walks travel from normal vertices, they have a probability of $q \in [0, 1]$ of traveling on structural edges and a probability of $1-q$ of traveling on normal edges; the weights of normal edges are treated as the same, therefore the probability of traveling on normal edges will be equally distributed to neighbors. Taking vertex 4 in Figure 3 as an example, if a walk travels to vertex 4, the probability distribution of next step for vertex 4 is q for the 2-star motif super-vertex and $\frac{1-q}{4}$ for vertex 1, 2, 3 and 5 respectively. With motif-step random walk strategy, vertices and their highly related motif super-vertices as well as two vertices that are surrounded by the same type of motif will be close to each other in the corpus, and their corresponding embedding vectors will be driven close to each other.

After obtaining the corpus, we then feed the collected corpus into the Skip-gram model [28] to map vertices into the low-dimensional embedding vectors. For simplicity, object function can be described as an optimization problem:

$$\begin{aligned} min_{\Phi(v_i),\Phi(M_j)} - (log\,Pr(\{v_{i-w}, ..., v_{i+w}\} \backslash v_i | \Phi(v_i), \Phi(M_j)) \\ + log\,Pr(\{v_{j-w}, ..., v_{j+w}\} \backslash v_{M_j} | \Phi(v_i), \Phi(M_j))) \end{aligned} \tag{3}$$

where v_i, $\Phi(v_i)$ denotes vertex i and the representation vector of vertex i while M_j, $\Phi(M_j)$ denotes motif super-vertex j and the representation vector of motif super-vertex j, and w denotes windows size. For different networks, we can adjust the coefficient q according to specific prior knowledge. The larger the coefficient q is, the more likely that the random walks travel on structural edges; the model pays more attention to the structural similarity than it does to connectivity in the network.

Algorithm 2 Inductive learning of LEMON

```
Input: $G = (V, E)$, set of unseen nodes $\tilde{U}$
Output: embeddings of unseen nodes $v_{\tilde{u} \in \tilde{U}}$
 1: for each node $\tilde{u} \in V'$ do
 2:     Initialize $\boldsymbol{c_{\tilde{u}}}$ as $[0, ..., 0]$
 3:     Get 3-vertex and 4-vertex motif count vector of unseen node and concate together $\boldsymbol{c_{\tilde{u}}}$
 4:     Calculate structural part embeddings based on motif representations: $\alpha \sum_{i=0}^{8} \frac{c_{\tilde{u}}^i}{\sum_{v \in V} c_v^i} \boldsymbol{v_{M_i}}$
 5:     Aggregate the embeddings of neighbors: $(1 - \alpha) \sum_{x \in N(\tilde{u})} \boldsymbol{v_x}$
 6:     Calcualte embeddings of unseen nodes $\boldsymbol{v_{\tilde{u}}}$
 7: end for
 8: ______________________________________________
 9: Get n-vertex motif count vector of $\tilde{u}$ (node $\tilde{u}$, number of vertex in the motif:n, repeat times T)
10: for each t=1 to T do do
11:     Initialize $W$ as $[\tilde{u}]$
12:     while $|W| <$ n do
13:         $u_{cur} = W[-1]$, randomly select one neighbor of $u_{cur}$ and add into $W$
14:     end while
15:     Sort degree vector of collected subgraph and compared to degree vector of different motifs
16:     Add 1 to corresponding dimension of $\boldsymbol{c_{\tilde{u}}}$
17: end for
```

3.2 Inductive Learning for Unseen Vertices

In dynamic scenarios, vertices arrive continuously in the network in a stream. As illustrated above, LEMON leverages the learned motif embeddings and vertex embeddings to efficiently generate embeddings for unseen vertices.

LEMON adopts attention mechanism that assigns importance to different motifs around the unseen vertex. Naturally, the easier it is for random walks to obtain what kind of motif, the more important this pattern is. In particular, if nodes collected by random walk can often constitute a certain motif, it shows the importance of this frequent pattern; the importance of each motif may be assigned via the this frequency. [3] We may repeat the process of adopting random walk to obtain subgraph, $\boldsymbol{c_{\tilde{u}}} = \{c_{\tilde{u}}^0, c_{\tilde{u}}^1, ..., c_{\tilde{u}}^8\}$ for the newly arrived vertex $\tilde{u}$. Specifically, we adopt random walk from vertex $\tilde{u}$ to collect a subgraph and then calculate its degree vector. After sorting the degree vector in descending order and compare it with degree vectors of different motifs listed in Figure 2. The time complexity of this process can be represented as $O(k \cdot nlog \cdot n)$, where k represents the times of repetition and n indicates the number of nodes in certain motif, in this case, n equals 3 or 4.

[3]If an amount of unseen vertices arrive, we may adopt Orca to get the motif count vector for new vertices and indicate the importance of each motif for simplicity.

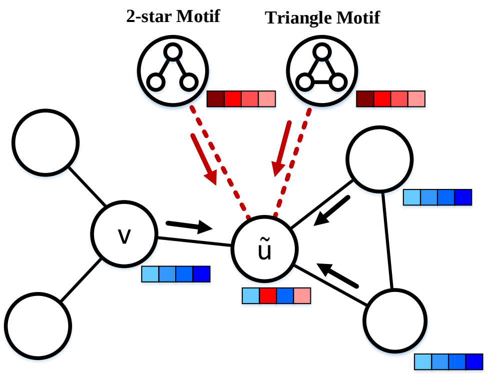


Fig. 4. Illustration of inductive learning for unseen vertex $\tilde{u}$.

The embedding vector for the unseen vertex $\tilde{u}$ can be calculated as follows:

$$\boldsymbol{v}_{\tilde{\boldsymbol{u}}} = \alpha \sum_{i=0}^{8} \frac{c_{\tilde{u}}^{i}}{\sum_{v \in V} c_{v}^{i}} \boldsymbol{v}_{\boldsymbol{M}_i} + (1-\alpha) \sum_{x \in N(\tilde{u})} \boldsymbol{v}_{\boldsymbol{x}} \tag{4}$$

where $\boldsymbol{v}_{\boldsymbol{x}}$, $\boldsymbol{v}_{\boldsymbol{M}_i}$ indicates embedding vector for vertex x, i-th type motif super-vertex respectively, c_u^i means the number of i-th type motif containing vertex $\tilde{u}$ and $N(\tilde{u})$ represents the neighbor vertices of $\tilde{u}$.

As shown in Figure 4, for the unseen vertex $\tilde{u}$ in the network, we first calculate its motif count vector, and then calculate the representation vector of the vertex $\tilde{u}$ via the above formula. Specifically, the representation vector of vertex $\tilde{u}$ consists of two parts: (a). connectivity, which is represented by the features of the neighbors of vertex $\tilde{u}$ such as vertex v; (b). structural similarity, which is calculated via the representation of the motifs.

4 EXPERIMENTS

4.1 Datasets

We utilize following datasets to perform experiments:

- Mobile [60]: This dataset is provided by China Telecom, one of China's major mobile service providers. Mobile users are considered as vertices, while edges indicate that one user has called another. The network contains call records for one week, with 5,000 vertices (4105 negative and 895 positive), 228,110 edges.
- Loan [60]: This dataset is provided by Finvolution Group. Vertices represent registered users of Finvolution Group, while edges represent mobile phone calls between users from January 2018 to January 2019. Users who fail to repay money in time are considered positive vertices, while others are negative. We use random walk to collect the largest connected subgraph as our dataset, which contains 1104 vertices (123 positive and 981 negative) and 1719 edges.
- Wiki [26]: This network contains 2,405 vertices (17 labels) and 17,981 edges. The vertices indicate words, while edges represent the co-occurrence relationship between words. The labels are given based on the Part-of-Speech tags.

- Cora, Citeseer [37]: Cora contains 2708 vertices (7 labels, such as Genetic Algorithms and Theory) and 5278 edges, while Citeseer contains 3264 vertices (6 labels) and 4536 edges. Each vertex indicates a scientific paper, while edges represent academic citations.
- PPI [9]: This network contains 3890 vertices and 37845 edges. This is a subgraph of the Protein-Protein Interactions network for Homo Sapiens.
- AMiner [44]: AMiner network contains 2,092,356 academic papers and 8,024,869 citations between papers. We select 20 vertices from top 200 vertices with largest degree and conduct breadth first search to sample a subgraph which contains 81725 nodes indicating authors, and 485175 edges indicating collaborations; the label of vertex is defined by whether the author's H-index is in the top 50% of all selected vertices or not.

4.2 Experimental Settings

To elaborate on the advantages of our proposed model, we construct four different experiments on seven above datasets and answer the following questions:

- Q_1: Is using motif an effective way to exploit network's information for subsequent tasks, such as node classification?
- Q_2: Is LEMON able to uniformly learn both connectivity and structural similarity spontaneously?
- Q_3: Is the structural information of various motifs well represented by the learned embedding vectors?

We compare LEMON with following baselines. For fairness, baselines are mainly unsupervised learning methods. And the details of baseline models can be found in Table 1.

- struc2vec [35]: Struc2vec learns latent embedding vectors through the structural identity (symmetry) of vertices. The parameters are set to recommended value: dimensions 128, walk length 80, num walks 10.
- GraphWave [14]: GraphWave represents vertices via low-dimensional vectors by utilizing heat wavelet diffusion patterns in an unsupervised way. Sample number 50, heat coefficient 1000, under exact mechanism.
- NetMF [34]: NetMF unifies word2vec based embedding models into the matrix factorization framework, including DeepWalk, node2vec, LINE and PTE. We adopt default parameters: 256 eigenpairs used to approximate normalized graph laplacian to get 128 dimension of embeddings.
- VERSE [46]: VERSE takes three similarities into consideration: community structure, roles and structural equivalence, and learns embeddings through neural network. We set α 0.85 and learning rate 0.0025 as default settings.
- Motifwalk [30]: Motifwalk leverages higher-order organization via motifs by exploiting a biased random walk strategy to generate contexts in the motif-aware graph. We adopt walk number 10, negative samples 15, walk length 15 as default settings.
- ProNE [58]: ProNE propose a scalable and effective model which adopts sparse matrix factorization and propagating in the spectral space. We adopt default parameter values, *i.e.* $\theta = 0.5$, $\mu = 0.2$, step of recursion 10.
- Meta-GNN [42]: Meta-GNN proposes a novel metagraph convolution operation to capture metagraph-structured neighborhoods' information.
- NEU [52]: NEU proposes to apply approximating higher-order proximities to network embedding methods to enhance the performance on downstream tasks. We utilize NetMF as basic embedding model.

Table 1. Implementation details of baseline models.

Baseline	Code
struc2vec	https://github.com/leoribeiro/struc2vec
NetMF	https://github.com/xptree/NetMF
VERSE	https://github.com/xgfs/verse
Motifwalk	We implemented the algorithm ourselves at https://github.com/larry2020626/motifwalk
ProNE	https://github.com/THUDM/ProNE
GraphWave	https://github.com/benedekrozemberczki/GraphWaveMachine
Meta-GNN	https://github.com/aravindsankar28/Meta-GNN
NEU, GLEE, BoostNE, Role2Vec	https://github.com/benedekrozemberczki/karateclub

- GLEE [45]: GLEE proposes to construct graph embeddings based on the geometric properties of the Laplacian matrix.
- BoostNE [24]: BoostNE proposes to learn the embeddings from multi-level without the requirement of low-rank assumption.
- Role2Vec [5]: Role2Vec adopts attributed random walks as a basis for existing random walk based method and enhances the performance of downstream tasks.

All experiments are conducted under the environment of 256G memory, 2 CPUs of 2.20GHz and the Ubuntu 16.04.3 system. LEMON and baselines learn embedding for vertices taking only the graph structure as input.[4] For node classification experiments, the embedding vectors of vertices are randomly divided into training, validation and testing set according to the ratio of 7:1:2. For link prediction experiments, we concatenate two vertices' embedding vectors together as the representation of edges. We randomly select 80% of the edges as the training and validation set. The remaining 20% edges are used as positive cases in testing set and we manually fabricate the same number of edges as negative cases. We adopt LightGBM [22] as the classifier. The Loss function is based on cross-entropy, as shown below:

$$L = -\sum_{c=1}^{m} y_c log(p_c), y_c \in \{0, 1\} \tag{5}$$

where m indicates the number of classes; if class label c is the correct classification for prediction, y_c is 1 and 0 otherwise; p_c indicates the probability of class c.

Hyperparameter settings are introduced as follows to facilitate better reproductivity: 1) for random walks settings, by default, there are 10 walks starting at each vertex, with a length of 80; 2) for embedding part, we set window size for optimization is 10, the dimension of output vectors is 128 for both vertices and motif super-vertices and adopt stochastic gradient descent as optimization strategy; 3) for classification, we utilize LightGBM model with 1000 boosting iterations with early stopping technique to avoid overfitting or underfitting.

4.3 Experimental Results

[4]This paper focuses on graph representation without node attributes. Although our proposed method is unable to achieve SOTA effects compared with models which consider node attributes, LEMON still exceeds SOTA baselines which also focus on non-attribute graph representation. From this perspective, LEMON has the superiority to baselines to a certain extent.

Table 2. Experimental results of anomaly detection. Symbol / suggests execution time is larger than 1 day.

Methods	Mobile			Loan		
	Prec.	Recall	Micro-F1	Prec.	Recall	Micro-F1
struc2vec	/	/	/	0.167±.090	0.200±.026	0.189±.013
GraphWave	0.756±.071	0.455±.047	0.568±.047	0.276±.016	0.133±.011	0.180±.004
Motifwalk	0.279±.084	0.248±.031	0.262±.054	0.027±.002	0.152±.014	0.045±.001
NetMF	0.151±.001	0.183±.001	0.166±.001	0.130±.001	0.167±.001	0.146±.001
VERSE	0.161±.012	0.225±.012	0.188±.012	0.043±.001	0.143±.017	0.067±.002
ProNE	0.646±.040	0.417±.008	0.507±.018	0.275±.041	0.141±.022	0.186±.028
Meta-GNN	0.110±.001	0.201±.001	0.142±.001	0.124±.001	0.164±.001	0.141±.001
NEU	0.721±.001	0.408±.001	0.521±.001	**0.304±.001***	0.175±.001	0.222±.001
GLEE	0.733±.001	0.445±.001	0.554±.001	0.174±.001	0.174±.001	0.174±.001
BoostNE	**0.802±.001***	0.445±.001	0.573±.001	0.261±.001	0.115±.001	0.160±.001
Role2Vec	0.700±.038	0.453±.017	0.549±.024	0.044±.001	0.098±.025	0.060±.005
LEMON	**0.765 ± .042**	**0.521 ± .003***	**0.620 ± .010***	**0.208 ± .036**	**0.263 ± .004***	**0.233 ± .023***

Anomaly detection. On the whole, LEMON outperforms baselines for both datasets. As shown in Table 2, LEMON exceeds the highest recall and f1 score for Mobile dataset. More specifically, for Mobile dataset, LEMON achieves an improvement of 9.16% in terms of F1 score relative to the best baseline model, GraphWave; in the mean time, LEMON also achieves the highest recall and relatively superior precision to baselines, only second to BoostNE.

An interesting observation is that VERSE, which also considers multi-similarities, did not perform as well as LEMON. One possible reason for this phenomenon is that in Mobile dataset, the positive cases are Fraudsters who get the contact information of people from all walks of life in the society through various channels and commit frauds via telephone calls. The people contacted by the fraudsters are not acquaintances to each other; as a result, most motifs around them are unclosed, such as ( ) and ( ). In order to verify this analysis, we calculate the distribution of the number of motifs around fraudsters and normal users, and the results are shown in Figure 5.

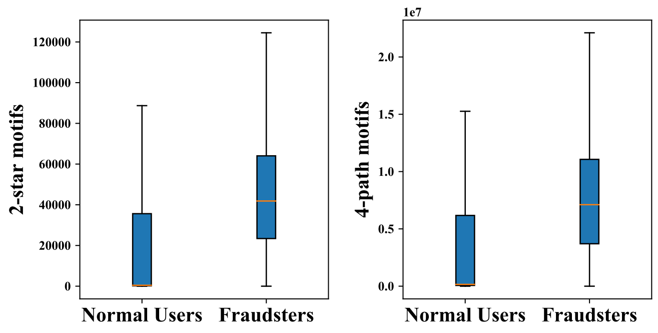


Fig. 5. Motif distributions around fraudsters and normal users.

It can be seen that compared with normal users, fraudsters are more likely to contact two other users who do not know each other. As a result, more unclosed motifs appear around fraudster users. On the contrary, two people called by a normal user are relatively more likely to be acquainted to each other and also have call records. For example, a company employee calls his/her two

Table 3. Experimental results of node classification. The results of Motifwalk on Cora and Citeseer are quoted from [30], which does not provide standard deviations.

Methods	Wiki	Cora	Citeseer	AMiner
	Micro-F1	Micro-F1	Micro-F1	Acc.
struc2vec	0.209±0.011	0.408±0.005	0.354±0.011	/
GraphWave	0.291±0.016	0.594±0.001	0.541±0.001	0.712±0.001
Motifwalk	0.328±0.031	0.680	0.457	0.527±0.001
NetMF	0.628±0.001	0.282±0.001	0.279±0.001	0.527±0.001
VERSE	0.626±0.007	0.772±0.024	0.642±0.015	0.716±0.003
ProNE	0.682±0.004	0.845±0.004	0.700±0.001	0.734±0.003
Meta-GNN	0.216±0.001	0.295±0.001	0.342±0.001	0.553±0.001
NEU	0.672±0.001	0.849±0.001	0.695±0.001	0.696±0.001
GLEE	0.470±0.001	0.762±0.001	0.626±0.001	0.769±0.001
BoostNE	0.520±0.001	0.710±0.001	0.585±0.001	0.744±0.001
Role2Vec	0.597±0.018	0.707±0.013	0.522±0.026	0.798±0.001
LEMON	**0.716 ± 0.005***	**0.858 ± 0.005***	**0.704 ± 0.005***	**0.830 ± 0.008***

colleagues, and these two colleagues may be likely to have phone calls between them. Therefore, there will be relatively more (△) around normal users. This key structural information can be precisely captured by motifs. Moreover, NetMF tends to preserve the information of the first-order network, which is unsuitable to deal with tasks values structure information. The experimental results in Table 2 verifies the analysis. NEU also values high-order structure information and it performs better than NetMF while it is not as effective as LEMON which indicates that LEMON can extract high-order structure information well.

For Loan dataset, NEU achieves the highest precision, while LEMON obtains the best performances in terms of recall, f1 and the second highest precision. It is worth noting that we pay more attention to identifying users who may not be able to pay back loans or scam users. Thus, we care more about positive samples by paying more attention to recall rather than precision particularly in this case. In the case of unbalanced categories, f1 score is the most balanced indicator; from this perspective, LEMON is still superior to NEU and other baselines.

Both Mobile and Loan datasets value structural information; moreover, LEMON demonstrates its ability to deal with structural similarity based datasets by surpassing methods which values structure information such as struc2vec and NEU. Based on these experimental results, we can conclude an answer to Q_1 that motifs contain valuable information and adopting motifs is an effective way to exploit structural information.

Node classification. For balanced node classification experiment, we utilize Wikipedia, Cora, Citeseer and AMiner datasets. As shown in Table 3, LEMON achieves the best result for all four datasets. In terms of Micro-F1 score, LEMON outperforms 4.99%, 1.06% and 0.57% than the most competitive baseline model for Wikipedia, Cora and Citeseer respectively; moreover, it also achieves an improvement of 4.01% in terms of accuracy for AMiner.

For Wikipedia, Cora, Citeseer and AMiner datasets, the ratio of edges connecting the same category vertices among all edges is 71.2%, 81.00%, 73.8% and 68.5%, suggesting that these datasets pay more attention to connectivity similarity than structural similarity; these experiments prove that LEMON can also perform well in connectivity based datasets.

Table 4. Experimental results of link prediction.

Methods	Mobile			PPI		
	Prec.	Recall	F1	Prec.	Recall	F1
struc2vec	/	/	/	0.695±.016	0.699±.016	0.697±.006
GraphWave	0.955±.001	0.887±.002	0.920±.002	**0.790± .005***	0.703±.001	0.744±.002
Motifwalk	0.948±.003	0.870±.002	0.907±.002	0.672±.007	0.713±.009	0.692±.008
NetMF	0.942±.001	0.860±.001	0.899±.001	0.693±.001	0.714±.001	0.703±.001
VERSE	0.955±.002	0.879±.002	0.915±.001	0.729±.006	0.720 ±.004	0.724±.002
ProNE	**0.970± .001***	0.923±.003	0.946±.002	0.759±.009	0.776±.006	0.767±.007
NEU	0.969 ±.001	0.924±.001	0.946±.001	0.770±.001	0.742±.001	0.756±.001
GLEE	0.969±.001	0.924±.001	0.946±.001	0.706±.001	0.750±.001	0.727±.001
BoostNE	0.968±.001	0.925±.001	0.946±.001	0.739±.001	0.719±.001	0.729±.001
Role2Vec	0.963±.001	0.910±.002	0.936±.002	0.768±.003	0.713±.003	0.740±.003
LEMON	**0.967 ± .001**	**0.927 ± .002***	**0.947 ± .001***	**0.759 ± .007**	**0.777 ± .001***	**0.768 ± .003***

Link prediction. For link prediction experiments, we utilize Mobile and PPI datasets. LEMON outperforms all baselines in recall and F1 score for both datasets; ProNE achieves the highest precision for Mobile while GraphWave obtains the highest precision for Loan.

These experiments indicate that LEMON generally generates superior performances to the state-of-the-art baselines for various downstream tasks which proves that LEMON is an effective method to leverage information of graphs. Moreover, the standard deviation of LEMON stays low in different experiments, indicating that LEMON is stable, which is another advantage of our proposed model.

4.4 Parameter Analysis

Our model has one important hyperparameter: q, which controls the probability of traveling on structural edges or normal edges. The effect of different q reflects on the results of different datasets is illustrated in Figure 6. As the value of q varies, the performance also changes. For datasets which values connectivity, LEMON achieves the best results when parameter q is relatively small (0.05 for Cora and Citeseer, 0.10 for Wikipedia). For Loan, which places more weight on structural similarity, the optimal q for Loan is relatively large; when $q < 0.3$, precision, recall and F1 score increase as q becomes larger; when $q > 0.3$, all metrics decrease as q increases.

For Q_2, LEMON achieves relatively satisfying results on both the connectivity based datasets and structural similarity based datasets, suggesting that our proposed model can simultaneously and uniformly leverage connectivity and structural similarity information by adjusting parameter q. In conclusion, LEMON performs well in various downstream tasks.

Generally, for datasets that value connectivity, q ought to be set relatively small and optimized in a small range; while we ought to set a larger q for datasets that pay more attention to structural similarity. In real-world scenarios, an empirical formula of determining the value parameter of q is required. As illustrated in above analysis, some datasets values connectivity while others pay attention to structural similarity. First, we use p to denote the ratio of edges connecting the same

[4]As Meta-GNN trains a GNN through semi-supervised learning instead of expressing embeddings through unsupervised learning. The framework of applying Meta-GNN method to link prediction task is quite different from other modes; hence, for fairness, it is not utilized as the baseline of link prediction task.

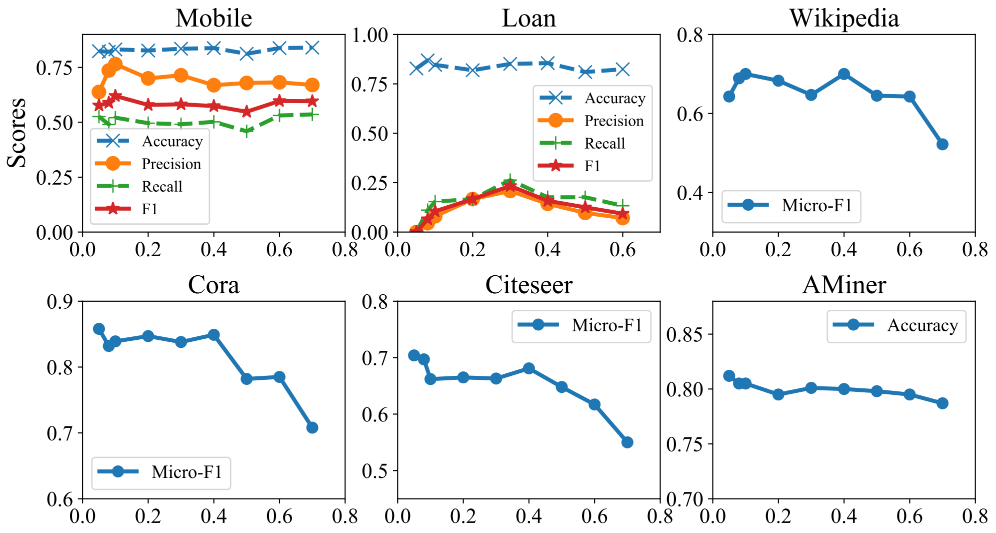


Fig. 6. Parameter sensitivity analysis for different datasets (x axis indicates q).

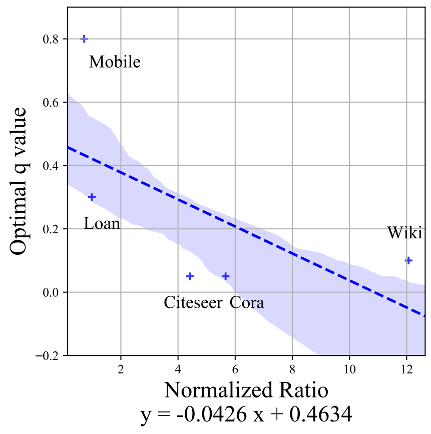


Fig. 7. Empirical formula of the optimal value of parameter q.

category vertices among all edges, then propose normalized $\bar{p} = \frac{p}{p_r}$, where p_r indicates the ratio p under random circumstances. The larger p_r, the more dataset values connectivity instead of structural similarity. We learned an empirical formula:

$$q = -0.0426\bar{p} + 0.4634 \tag{6}$$

as the regression line shown in Figure 7, which represent the linear relationship between the normalized ratio $\bar{p}$ and optimal q value. To validate the effectiveness of this formula, we repeat experiments on Cora, Citeseer, Wiki, Mobile and Loan with q that calculated based on above formula. Table 5 represents the comparison results of LEMON with optimal q and empirically calculated q. Overall, LEMON(emp) achieves at least comparable performance to the most competitive baseline. Through this empirical formula, we can quickly determine the value of q.

Table 5. Experimental result comparison of LEMON. LEMON(opt) and LEMON(emp) indicates LEMON with optimal q and q calculated based on formula 6. Baseline* denotes the most competitive baseline.

Methods	Cora	Citeseer	Wiki	Loan	Mobile
	Acc.	Acc.	Acc.	Micro-F1	Micro-F1
Baseline*	0.849±0.001	0.700±0.001	0.682±0.004	0.222±0.001	0.798±0.001
LEMON(opt)	0.858±0.005	0.704±0.005	0.716±0.005	0.233±0.023	0.620±0.010
LEMON(emp)	0.847±0.004	0.663±0.002	0.643±0.021	0.167±0.004	0.587±0.001

4.5 Case Study

Considering Q_3, how do we evaluate the quality of learned motif embeddings? In the Word2Vec framework [28], the quality of embedding vectors for vertices can be explained using the example that the vector of the word *big* is similar to the vector of the word *bigger*, in the same sense that the vector of the word *small* is to the vector of the word *smaller* [28]. In other words, $\boldsymbol{v_{smaller}} - \boldsymbol{v_{small}} = \boldsymbol{v_{bigger}} - \boldsymbol{v_{big}}$, where $\boldsymbol{v_{smaller}}, \boldsymbol{v_{small}}, \boldsymbol{v_{bigger}}, \boldsymbol{v_{big}}$ indicates representation vector of word *smaller*, *small*, *bigger* and *big* respectively.

Transferring this idea to motifs, and taking motifs as words, we calculate $\boldsymbol{X} = \boldsymbol{v_{M_2}} - \boldsymbol{v_{M_1}} + \boldsymbol{v_{M_4}}$ where M_1 ( ), M_2( ), M_4( ), M_6( ) are four types of motifs. We then search in the vector space and find the closest embedding vector of motifs by Euclidean distance, Cosine similarity and Pearson similarity respectively.

Delightfully, we find that the embedding vector $\boldsymbol{v_{M_6}}$ for ( ) is the closest embedding vector to X, which fit the commonsense interpretation (adding one edge from the center vertex from M_1 we get M_4 and adding one edge from the center vertex from M_2 we get M_6; both M_2 and M_6 contain a closed triangle pattern, while M_1 and M_4 contain open 2-star pattern). The embedding vectors of other structurally similar motif pairs are also close in hidden space.

To verify whether the embedding vectors of structurally similar motifs are also close together in latent space, we also calculate the closest motifs for each motif; results are shown in Table 6. From this, we can perceive some interesting phenomena: 1) The closest motif to the most complex motif M_8 ( ) is M_7 ( ), with only one edge missing from M_8. 2) The closest motif to the simplest motif M_0 ( ) is M_1 ( ), with only one more edge from M_0 under Euclidean distance, Cosine similarity and Pearson similarity respectively. 3) For the 2-star motif M_1 ( ), the three closest motifs are M_3 ( ), M_4 ( ) and M_0 ( ). From the structural perspective, M_0, M_3 and M_4 are very close to M_1: adding an edge at the non-centered vertex of ( ), we get ( ); adding an edge at the centered vertex of ( ), we get ( ); removing an edge of ( ), we get ( ). These above analyses suggest that the learned embedding vectors of various motifs encode structural information into low-dimensional vectors.

4.6 Inductive Learning Experiment

We adopt Mobile dataset, a dynamic network, to perform the inductive learning experiment. We consider the vertices and edges that appear in the first 80% of time to form a known network that contains 4716 vertices, 227625 edges, while the follow-up unseen data contains 283 vertices and 485 edges.

Table 6. Closest motifs in the latent space under different similarity calculation method.

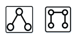
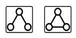
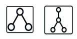
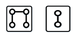
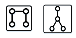
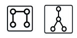
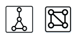
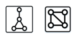
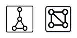
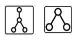
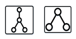
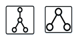
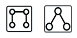
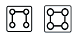
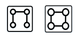
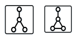
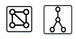
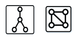
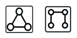
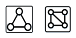
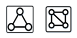
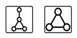
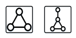
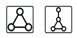
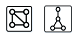
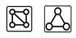
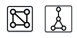
Motifs	Eucli-dis	Cosine-sim	Pearson-sim
			
			
			
			
			
			
			
			
			

Table 7. Experimental results of inductive learning task.

Methods	F1	Time(sec)
Incremental SVD	0.172±0.001	0.245
GraphSAGE	0.219±0.016	1.099
LEMON	**0.319 ± 0.030***	0.900

The known network is fed into our proposed model and baseline models (GraphSAGE and Incremental SVD [8]), and vertices embeddings are obtained. We then split these embeddings vectors as training and validation set to train the classifier. Follow-up vertices and edges are added into the network; the model generates embeddings for these unseen vertices, which are fed into classifier as the testing set. From Table 7, we can see that in the inductive learning scenario, LEMON outperforms Incremental SVD [8] and GraphSAGE effectively, as our model achieves an improvement of 45.7% and 85.5% in terms of F1 score relative to GraphSAGE and Incremental SVD. GraphSAGE obtains representations for unseen vertices by aggregating features from their neighbors. Thus, it is not suitable for detecting anomalies. The results of the inductive learning experiment also suggest that the learned embedding vectors of motifs contain rich network information.

4.7 Complexity Analysis

Given a network G, We firstly extract motifs for vertices in G and then adopt motif-step random walk to collect corpus in order to learn embedding vectors for vertices and motifs. The running time for both parts are listed in Figure 8. The time complexity of Orca is $O(k \cdot |E| + T_4)$, where k denote the maximum node degree and T_4 denotes the time needed to enumerate ( ). LEMON adopts hierarchical softmax based on Huffman coding, therefore the time complexity for learning embedding part is $O(|V| \cdot log|V| + n \cdot l)$, where $|E|$, $|V|$ indicates the number of edges and vertices respectively, n indicates the number of random walks and l indicates the length of random walks.

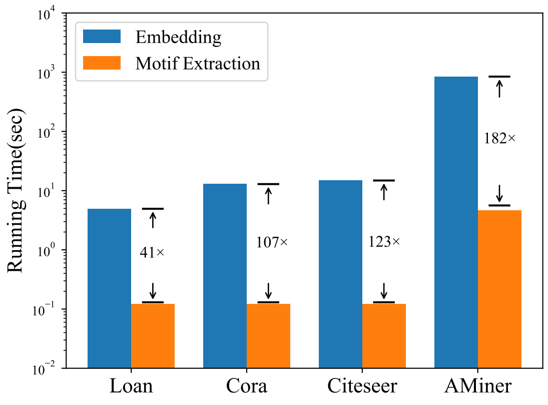


Fig. 8. Execution time analysis (seconds) for counting motif extraction and learning embedding vectors for vertices and motifs of different networks.

5 RELATED WORK

In the interests of clarity, related work of this paper is mainly divided into two parts: representation learning and motif related works.

Representation learning. Network embedding is a crucial area that has attracted significant research attention over the past couple of decades. Early works can be traced back to these traditional models based on the matrix factorization perspective [3, 31, 48]. However, apart from sparse matrix factorization methods [58], due to the large computational cost of decomposing a large-scale matrix, along with its statistical performance drawback [17], these models inevitably encounter limitations in both efficiency and effectiveness.

In the literature of network embedding studies, random walk based models play an important role, such as DeepWalk [32] preserve the connectivity information between a vertex and its neighbors. Node2vec [17] further develops this idea by designing a biased random walk strategy to capture diverse neighborhood patterns; LINE [43] captures local and global structures through first-order and second-order proximity; DPWalker [15] learns representations for scale-free network. Moreover, GLEE [45] and BoostNE [24] also learn representation of nodes based on connectivity properties of nodes. One of the representative structural similarity based works: struc2vec [35], uses the degree of vertices as the basis measure of structural vertex-similarity. In addition, metapath2vec [13] and hin2vec [16] use meta-path random walk to generate contexts for heterogeneous networks. There are some existing works that take multi-similarities into consideration, such as VERSE [46] (community structure, roles and structural equivalence); RUM [57] (local triads, neighborhood relationships and global community proximity). RolX [19] discovers structural roles for vertices and use non-negative matrix factorization to generate embedding. Meanwhile, Graphwave [14] treats the spectral graph wavelet signatures of structurally similar vertices as probability distribution. Moreover, NEU [52] enhance the performance of network embedding methods via approximating higher-order proximities.

Motif related network embedding. There are some pioneering works that learn embeddings based on motifs are outlined below: MCN [23] proposes a motif-based attention model which leverages higher-order neighborhoods using multi-hop motif adjacency matrices. Motif-CNN [41] employs an attention model to effectively capture high-order structural pattern information; [11, 38, 59] adopt motif-based adjacency matrices, while Nest [51] utilizes motif filtering and convolutional neural networks to capture structures. OFFER [56] improves the effectiveness of graph representation learning via refining the adjacency matrix and transition probability matrix based on motif degree of nodes and motif degree of edges respectively. There are some works utilize motif to capture higher-order structures in the network and conduct link prediction tasks effectively, *i.e.* [1, 47]. HONEM [40] incorporates non-Markovian higher-order dependencies during learning embeddings of the networks besides pairwise interactions. SNS [25] combines neighbor information and local-subgraphs similarity together to enhance the quality of embeddings. Sub2Vec [2] utilizes random walk and word2vec framework proposes a scalable embedding method based on two intuitive properties of subgraphs. Subrank [7] introduces a subgraph to subgraph proximity measure and computing subgraph embeddings based on personalized PageRank.

6 CONCLUSION

In this paper, we propose a novel representation learning framework, LEMON, bridging connectivity and structural similarity in a universal form via motifs and LEMON is able to effectively learn representations for both vertices and motifs, which enables LEMON to deal with inductive tasks. Experimental results show that LEMON outperforms state-of-the-art algorithms on two real-world datasets and five public datasets for four different tasks.

Acknowledgments. This work is supported by the National Key Research and Development Project of China (2018AAA0101900) and a research funding from Tongdun Technology.

REFERENCES

[1] Ghadeer Abuoda, Gianmarco De Francisci Morales, and Ashraf Aboulnaga. 2019. Link Prediction via Higher-Order Motif Features. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2019*, Vol. 11906. Springer, 412–429.

[2] Bijaya Adhikari, Yao Zhang, Naren Ramakrishnan, and B Aditya Prakash. 2018. Sub2vec: Feature learning for subgraphs. In *PAKDD*, Vol. 10938. Springer, 170–182.

[3] Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. 2013. Distributed large-scale natural graph factorization. In *WWW*. ACM, 37–48.

[4] Nesreen K Ahmed, Jennifer Neville, Ryan A Rossi, and Nick Duffield. 2015. Efficient graphlet counting for large networks. In *ICDM*. IEEE, 1–10.

[5] Nesreen K Ahmed, Ryan A Rossi, John Boaz Lee, Theodore L Willke, Rong Zhou, Xiangnan Kong, and Hoda Eldardiry. 2019. role2vec: Role-based network embeddings. In *Proc. DLG KDD*. 1–7.

[6] Nesreen K Ahmed, Theodore L Willke, and Ryan A Rossi. 2016. Estimation of local subgraph counts. In *ICBD*. IEEE, 586–595.

[7] Oana Balalau and Sagar Goyal. 2020. SubRank: Subgraph Embeddings via a Subgraph Proximity Measure. In *PAKDD*, Vol. 12084. Springer, 487–498.

[8] Matthew Brand. 2006. Fast low-rank modifications of the thin singular value decomposition. *Linear Algebra and Its Applications* (2006).

[9] Bobby-Joe Breitkreutz, Chris Stark, Teresa Reguly, Lorrie Boucher, Ashton Breitkreutz, Michael Livstone, Rose Oughtred, Daniel H Lackner, Jürg Bähler, Valerie Wood, et al. 2007. The BioGRID interaction database: 2008 update. *Nucleic acids research* 36, suppl_1 (2007), 637–640.

[10] Marco Bressan, Flavio Chierichetti, Ravi Kumar, Stefano Leucci, and Alessandro Panconesi. 2018. Motif counting beyond five nodes. *TKDD* 12, 4 (2018), 48:1–48:25.

[11] Manoj Reddy Dareddy, Mahashweta Das, and Hao Yang. 2019. motif2vec: Motif aware node representation learning for heterogeneous networks. In *ICBD*. IEEE, 1052–1059.

[12] Vachik S Dave, Nesreen K Ahmed, and Mohammad Al Hasan. 2017. E-CLoG: counting edge-centric local graphlets. In *ICBD*. IEEE, 586–595.

[13] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. ACM, 135–144.

[14] Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. 2018. Learning structural node embeddings via diffusion wavelets. In *SIGKDD*. ACM, 1320–1329.

[15] Rui Feng, Yang Yang, Wenjie Hu, Fei Wu, and Yueting Zhang. 2018. Representation learning for scale-free networks. In *AAAI*, Vol. 32.

[16] Tao-yang Fu, Wang-Chien Lee, and Zhen Lei. 2017. Hin2vec: Explore meta-paths in heterogeneous information networks for representation learning. In *CIKM*. ACM, 1797–1806.

[17] Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *SIGKDD*. ACM, 855–864.

[18] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *NeurIPS*. 1024–1034.

[19] Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. 2012. Rolx: structural role extraction & mining in large graphs. In *SIGKDD*. ACM, 1231–1239.

[20] Tomaž Hočevar and Janez Demšar. 2014. A combinatorial approach to graphlet counting. *Bioinformatics* 30, 4 (2014), 559–565.

[21] Nadav Kashtan, Shalev Itzkovitz, Ron Milo, and Uri Alon. 2004. Efficient sampling algorithm for estimating subgraph concentrations and detecting network motifs. *Bioinformatics* (2004), 1746–1758.

[22] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *NeurIPS*.

[23] John Boaz Lee, Ryan A Rossi, Xiangnan Kong, Sungchul Kim, Eunyee Koh, and Anup Rao. 2019. Graph convolutional networks with motif-based attention. In *CIKM*. ACM, 499–508.

[24] Jundong Li, Liang Wu, Ruocheng Guo, Chenghao Liu, and Huan Liu. 2019. Multi-level network embedding with boosted low-rank matrix approximation. In *ASONAM*. 49–56.

[25] Tianshu Lyu, Yuan Zhang, and Yan Zhang. 2017. Enhancing the network embedding quality with structural similarity. In *CIKM*. ACM, 147–156.

[26] Matt Mahoney. 2011. Large text compression benchmark. *URL: http://www.mattmahoney.net/dc/textdata* (2011).

[27] Dror Marcus and Yuval Shavitt. 2012. Rage–a rapid graphlet enumerator for large networks. *Computer Networks* (2012), 810–819.

[28] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. In *ICLR*.

[29] Ron Milo, Shai Shen-Orr, Shalev Itzkovitz, Nadav Kashtan, Dmitri Chklovskii, and Uri Alon. 2002. Network motifs: Simple building blocks of complex networks. *Science* (2002).

[30] Hoang Nguyen and Tsuyoshi Murata. 2017. Motif-aware graph embeddings. In *IJCAI*.

[31] Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. 2016. Asymmetric transitivity preserving graph embedding. In *SIGKDD*. ACM, 1105–1114.

[32] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *SIGKDD*. ACM, 701–710.

[33] Ali Pinar, C Seshadhri, and Vaidyanathan Vishal. 2017. Escape: Efficiently counting all 5-vertex subgraphs. In *WWW*. 1431–1440.

[34] Jiezhong Qiu, Yuxiao Dong, Hao Ma, Jian Li, Kuansan Wang, and Jie Tang. 2018. Network embedding as matrix factorization: Unifying deepwalk, line, pte, and node2vec. In *WSDM*. ACM, 459–467.

[35] Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. 2017. struc2vec: Learning node representations from structural identity. In *SIGKDD*. ACM, 385–394.

[36] Pedro Ribeiro, Pedro Paredes, Miguel EP Silva, David Aparicio, and Fernando Silva. 2021. A Survey on Subgraph Counting: Concepts, Algorithms, and Applications to Network Motifs and Graphlets. *ACM Computing Surveys (CSUR)* 54, 2 (2021), 1–36.

[37] Ryan Rossi and Nesreen Ahmed. 2015. The network data repository with interactive graph analytics and visualization. In *AAAI*. 4292–4293.

[38] Ryan A Rossi, Nesreen K Ahmed, and Eunyee Koh. 2018. Higher-order network representation learning. In *WWW*. 3–4.

[39] Ryan A Rossi, Nesreen K Ahmed, Eunyee Koh, Sungchul Kim, Anup Rao, and Yasin Abbasi Yadkori. 2018. HONE: higher-order network embeddings. *arXiv preprint arXiv:1801.09303*.

[40] Mandana Saebi, Giovanni Luca Ciampaglia, Lance M Kaplan, and Nitesh V Chawla. 2020. HONEM: learning embedding for higher order networks. *Big Data* 8, 4 (2020), 255–269.

[41] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 2017. Motif-based convolutional neural network on graphs. *arXiv preprint arXiv:1711.05697* (2017).

[42] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. 2019. Meta-GNN: Metagraph Neural Network for Semi-supervised learning in Attributed Heterogeneous Information Networks. *ASONAM. IEEE* (2019), 137–144.

[43] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. 2015. Line: Large-scale information network embedding. In *WWW*. 1067–1077.

[44] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. 2008. Arnetminer: extraction and mining of academic social networks. In *SIGKDD*. ACM, 990–998.

[45] Leo Torres, Kevin S Chan, and Tina Eliassi-Rad. 2020. GLEE: Geometric Laplacian Eigenmap Embedding. *Journal of Complex Networks* 8, 2 (2020).

[46] Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. 2018. Verse: Versatile graph embeddings from similarity measures. In *WWW*. 539–548.

[47] Lei Wang, Jing Ren, Bo Xu, Jianxin Li, Wei Luo, and Feng Xia. 2020. MODEL: Motif-Based Deep Feature Learning for Link Prediction. *IEEE Transactions on Computational Social Systems* 7, 2 (2020), 503–516.

[48] Xiao Wang, Peng Cui, Jing Wang, Jian Pei, Wenwu Zhu, and Shiqiang Yang. 2017. Community preserving network embedding. In *AAAI*. 203–209.

[49] Sebastian Wernicke and Florian Rasche. 2006. FANMOD: A tool for fast network motif detection. *Bioinformatics* (2006), 1152–1153.

[50] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. 2020. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems* 32, 1 (2020), 4–24.

[51] Carl Yang, Mengxiong Liu, Vincent W Zheng, and Jiawei Han. 2018. Node, motif and subgraph: Leveraging network functional blocks through structural convolution. In *ASONAM*. IEEE, 47–52.

[52] Cheng Yang, Maosong Sun, Zhiyuan Liu, and Cunchao Tu. 2017. Fast network embedding enhancement via high order proximity approximation.. In *IJCAI*. 3894–3900.

[53] Yang Yang, Yuhong Xu, Chunping Wang, Yizhou Sun, Fei Wu, Yueting Zhuang, and Ming Gu. 2019. Understanding Default Behavior in Online Lending. In *CIKM*. 2043–2052.

[54] Hao Yin, Austin R Benson, and Jure Leskovec. 2018. Higher-order clustering in networks. *Physical Review E* 97, 5 (2018), 052306.

[55] Hao Yin, Austin R. Benson, Jure Leskovec, and David F. Gleich. 2017. Local Higher-Order Graph Clustering. In *SIGKDD*. ACM, 555–564.

[56] Shuo Yu, Feng Xia, Jin Xu, Zhikui Chen, and Ivan Lee. 2020. OFFER: A Motif Dimensional Framework for Network Representation Learning. In *CIKM*. 3349–3352.

[57] Yanlei Yu, Zhiwu Lu, Jiajun Liu, Guoping Zhao, and Ji-rong Wen. 2019. Rum: Network representation learning using motifs. In *ICDE*. IEEE, 1382–1393.

[58] Jie Zhang, Yuxiao Dong, Yan Wang, Jie Tang, and Ming Ding. 2019. ProNE: Fast and Scalable Network Representation Learning. In *IJCAI*, Vol. 19. 4278–4284.

[59] Huan Zhao, Yingqi Zhou, Yangqiu Song, and Dik Lun Lee. 2019. Motif enhanced recommendation over heterogeneous information network. In *CIKM*.

[60] Lekui Zhou, Yang Yang, Xiang Ren, Fei Wu, and Yueting Zhuang. 2018. Dynamic network embedding by modeling triadic closure process. In *AAAI*. 571–578.