Betweenness Approximation for Hypernetwork Dismantling with Hypergraph Neural Network

DENGCHENG YAN, School of Computer Science and Technology, Anhui University, China
WENXIN XIE, School of Computer Science and Technology, Anhui University, China
YIWEN ZHANG, School of Computer Science and Technology, Anhui University, China

The purpose of Network Dismantling (ND) is to find an optimal set of nodes and removing these nodes can greatly decrease the network connectivity. However, current dismantling methods are mainly focus on traditional simple network which only consider the pairwise interaction between two nodes, while the hypernetwork, which can model higher order groupwise relation among arbitrary nodes, is more suitable for modeling real world. Due to the structural difference between simple network and hypernetwork, current dismantling methods cannot be directly applied to hypernetwork dismantling. Although some centrality measures in hypernetwork can be used for hypernetwork dismantling, they face the problem of balancing effect and efficiency. Therefore, in this paper, we propose a novel HyperNetwork Dismantling methods based on hypergraph neural network, called HND. Specifically, our method first generates plenty of node ranking samples with the help of synthetic hypernetwork generator. Then, a node betweenness approximation model in hypernetwork is built based on hypergraph neural network. And this model is trained on those ranking samples generated in previous step until convergence. Finally, the well-trained model is utilized to approximate the nodes’ betweenness in real world hypernetworks and further used for dismantling. To confirm the effectiveness of our method, we conduct extensive experiments on five real world hypernetworks. The experimental results demonstrate that the HND outperforms various baselines.

CCS Concepts: • Computer systems organization → Embedded systems; Redundancy; Robotics; • Networks → Network reliability.

Additional Key Words and Phrases: datasets, neural networks, gaze detection, text tagging

ACM Reference Format:
Dengcheng Yan, Wenxin Xie, and Yiwen Zhang. 2018. Betweenness Approximation for Hypernetwork Dismantling with Hypergraph Neural Network. J. ACM 37, 4, Article 111 (August 2018), 16 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Network has the ability of depicting entities and their relations. So it is widely applied to model various real world complex systems such as social platform [24, 26, 41], power grid [25, 34] and so on [10, 30]. As a basic problem in network science, network dismantling (ND) [6] aims to find a set of nodes whose removal will greatly destroy the connectivity of network. Due to the network connectivity is highly related to spread efficiency of information or infectious disease, ND has a

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Association for Computing Machinery.
0004-5411/2018/8-ART111 $15.00
https://doi.org/XXXXXXX.XXXXXXX

J. ACM, Vol. 37, No. 4, Article 111. Publication date: August 2018.
widely application in corresponding fields [12, 20]. Many researchers are attracted and multiple dismantling methods are proposed to solve this problem. However, current methods [6, 14, 31] are mainly focus on the traditional simple network which only consider the pairwise relation between two nodes, while there are lots of higher-order groupwise relations among arbitrary number of nodes in real world and traditional network cannot model this kind of relation. Thus, the application of these methods is limited when faces the groupwise relation.

Fortunately, hypernetwork [3] have unique advantages in modeling such groupwise relation. As Figure 2 shown, the hyperedge in hypernetwork can naturally express the higher-order relation among arbitrary number of nodes. Thus, utilizing hypernetwork to model real world is more suitable than traditional network especially face groupwise relations. However, even the modeling problem is solved by hypernetwork, the existing dismantling methods still have limitations. On one hand, these methods cannot be directly applied due to the structural difference between hypernetwork and traditional network. On the other hand, although various centrality measures in hypernetwork can be used to dismantling just like in traditional network, both of them face the problem of balancing effect and efficiency. For example, betweenness centrality is very suitable for dismantling a network but its calculation has a huge complexity. Degree centrality is easy to calculated but it performs not well on dismantling.

Recently, with the surge of deep learning technology, researchers have attempted to use some deep model to approximate complex centrality measures such as betweenness [13], closeness [27] and so on. With the help of powerful expression ability of deep model, these measures can be approximated in a limited error with less computational complexity. Then, the approximation centrality values is naturally utilized to various tasks so as to take both effect and efficiency into consider.

So, in this paper, we propose a novel HyperNetwork Dismantling method based on deep learning technology, called HND. This method adopt the hypergraph neural network to approximate betweenness centrality in hypernetwork, and utilized the approxiamtion values to accomplish the hypernetwork dismantling task. Specifically, our proposed HND first generates many small scale synthetic hypernetworks and construct betweenness ranking samples according to them. Then, a hypergraph neural network ranking model is built and the samples generated in previous step are used to trained this ranking model. Finally, the well-trained model can be used to approximate the betweenness in real world hypernetworks, and the approximation value can be used to dismantling.

Our main contributions are summarized as follows:
• We design a betweenness approximation model based on hypergraph neural network. The model can be trained with lots of synthetic ranking samples and applied to real world hypernetworks. With the help of deep learning’s powerful representation ability, the well-trained model can well approximate betweenness with a lower computation complexity.
• We propose a novel hypernetwork dismantling method, called HND. HND utilizes the betweenness approximation model to calculate the approximation betweenness which is adopted to dismantle hypernetworks. Due to the approximation ability of model, HND can achieve the effect close to betweenness with much lower computational complexity.
• We conduct extensive experiments on five real world hypernetworks. The results show that the performance of our proposed method outperforms several baselines. Moreover, the experimental results also confirm that the betweenness ranking model can approximate betweenness with less time consumption.

The rest of paper is summarized as follows. Section 2 introduces the related works about this article. Section 3 gives some necessary definitions and symbols. In Section 4, we introduce the proposed method in detail. Section 5 describes the experiments and makes detailed analyses of experimental results. In Section 6, we summarize the whole paper and make a vision of our future work.

2 RELATED WORK
2.1 Network Dismantling
Network dismantling [6] aims to find an optimal set of nodes whose deletion will greatly destroy the network connectivity. As a graph combinatorial optimization problem, ND is NP-hard and its exact solution is hard to solve. Thus, researchers attempted to find an approximate solution and propose various dismantling methods. Generally, current dismantling methods can be divided into three classes. The first class is based on centrality measures. In this kind of methods, nodes are selected greedily according to their centrality measures. However, these methods usually face the problem of balancing effect and efficiency. Local centrality measures (e.g., degree centrality) are easy to calculated but cannot achieve a well dismantling effect, while global centrality measures (e.g., betweenness centrality, closeness centrality) perform well on dismantling but have a large computation complexity. In order to take both the dismantling effect and efficiency into consider, some centrality measures between local and global are propsed. For example, the collective influence (CI) [28] proposed by Morone et al. can flexibly balance the globality and locality through tuning a hyperparameter, so as to consider both the dismantling effect and efficiency. The second class is heuristic methods designed by researchers. This kind of methods dismantle a network by multiple steps. For example, Braunstein et al. proposed a three step method MinSum [6] which dismantles a network by decycling, tree breaking and nodes reintroducing. Similar to MinSum, CoreHD [42] and BPD [29] are also adopt this framework but different in detail. Moreover, Ren et al. proposed the GND [31] algorithm to consider the case of weighted nodes. The third class methods are based on deep learning. Researchers hope to achieve better dismantling effect with the help of powerful ability of deep learning. Specifically, Fan et al. proposed the FINDER [14] framework based on deep reinforcement learning. This method trains an agent to do dismantling exercises on lots of small scale synthetic networks, and finally applied to real large scale networks. Besides, Grassia et al. proposed the GDM [16] method which trains a graph neural network ranking model by lots of groundtruth dismantling sequences and applied to real network dismantling. Recently, with the surge of hypernetwork, Yan et al. proposed a dismantling method suitable for hyernetworks based on deep reinforcement learning, called HITTER [40]. Generally, current works are mainly focus on
trainditional simple network but ignore the hypernetwork. Thus, in this paper, we attempt to solve the hypernetwork dismantling problem via deep learning technology.

2.2 Graph Neural Network

As a kind of network embedding method, graph neural network (GNN) can map nodes into low-rank vectors according to specific tasks. Based on the most basic neighbors information aggregation mechanism, various GNNs are proposed and applied to many fields. Among these GNNs, graph convolution network (GCN) [22] is usually seemed as a basic method. Through neighbor information aggregation, feature linear transformation and nonlinear activation, nodes’ embedding can be obtained and applied to various down-stream tasks such as node classification, line prediction and so on. Other GNNs following this framework are proposed successively. For example, graph attention network (GAT) [35] thought that different neighbors have different importance to target node in the step of neighbor aggregation, and it introduced the attention mechanism to make target node adaptively aggregation information. Hamilton et al. proposed inductive GraphSAGE [17] which can inference the embeddings of nodes unseen in training stage. Moreover, due to its powerful expression ability, GNNs also have been applied to various fields such as recommendation systems [8, 18, 37], user profile [9, 36] and so on. However, the core neighbors information aggregation mechanism in GNN relies on the pairwise interactions. Therefore, these GNNs cannot be directly applied to hypernetworks due to the structural difference with traditional simple network. In order to solve this problem, researchers proposed various methods to extend the traditional GNNs to hypernetworks. For example, Feng et al. proposed HGNN [15] to applied GCN to hypernetworks through transforming hypernetworks into simple networks according to clique expansion. Similarly, Yadati et al. [39] also transform hypernetworks into simple networks through breaking hyperedges into pairwise edges by specific rules, and directly applied GCN to it. Different from the above two methods, UniGNN [19] proposed by Huang et al. gives the neighbor information aggregation mechanism suitable for hypernetworks. Through the aggregation paths of nodes to hyperedges and hyperedges to nodes, various traditional GNNs such as GCN, GAT and GIN [38] can be transfered to hypernetworks. In this paper, we adopt hypergraph neural network to accomplish betweenness approximation task. Although some works have attempted to approximate betweenness by GNNs, they are not suitable for hypernetworks. Therefore, a hypergraph neural network for hypernetwork betweenness approximation is necessary.

3 PRELIMINARIES

In this section, we briefly introduce some related concepts.

**Definition 1 (Hypernetwork [3]).** A hypernetwork is defined as \( G = (V, E) \). The \( V = \{v_1, v_2, \ldots, v_N\} \) denotes nodes set in hypernetwork and \( N = |V| \) is the number of nodes. The \( E = \{e_1, e_2, \ldots, e_M\} \) is the set of hyperedges and \( M = |E| \) denotes the number of hyperedges. Each hyperedge \( e \) is defined as \( e \subseteq V \) and \( e \neq \emptyset \).

For each hyperedge \( e \) in hypernetwork, the size of \( e \) denotes the number of nodes contained in it. Obviously, due to the size of hyperedges is flexible, hypernetwork can model both pairwise and groupwise interactions. When the size of all hyperedges in a hypernetwork equals to 2, this hypernetwork also a traditional simple network. Therefore, traditional simple network can be seemed as a special form of hypernetwork.
Betweenness Approximation for Hypernetwork Dismantling with Hypergraph Neural Network

Definition 2 (Incidence Matrix [7]). Incidence matrix of a hypernetwork is a matrix $H \in \mathbb{R}^{N \times M}$. Each element of $H$ is given as follows:

$$H_{v_i, e_j} = \begin{cases} 1, & v_i \in e_j \\ 0, & \text{otherwise} \end{cases}$$

(1)

Similar to adjacent matrix in traditional simple network, incidence matrix can be used to express relations between nodes and hyperedges in hypernetwork. Figure 2(a) shows an example hypernetwork and its incidence matrix.

Definition 3 (Hyperdegree [7] and Degree [3]). For each node $v_i$ in hypernetwork $G$, the hyperdegree of $v_i$ is defined the number of hyperedges which contain node $v_i$:

$$hdeg(v_i) = \sum_{e_j \in E} H_{v_i, e_j}$$

(2)

And the degree denotes the number of nodes adjacent to it:

$$deg(v_i) = \sum_{v_j \in V} (HH^T)_{v_i, v_j} - hdeg(v_i)$$

(3)

Due to the dismantling problem is highly related to network connectivity, so we define the connectivity of hypernetwork as follows:

Definition 4 (Hypernetwork Connectivity). The connected component contains the most number of hyperedges in hypernetwork $G$ is called the giant connected component (GCC). And the connectivity of $G$ is defined as the ratio of node number in GCC to the node number of $G$:

$$\text{connectivity}(G) = \frac{|V_{GCC}|}{|V_G|}$$

(4)

Different to traditional simple network, we define the connectivity of hypernetwork is related the connections between hyperedges. According to Berge [5], what a connected hypernetwork
relies on is relations among hyperedges, not nodes. Moreover, it seems that selecting the connected component which contains the most number of nodes as the GCC is also reasonable. However, the connectivity defined in this way will be influenced by those huge hyperedges. As shown in Figure 2(b), selecting the left connected component cannot reflect the property of hypernetwork connectivity, and it also violates the purpose of dismantling.

**Definition 5** (2-section network [7]). 2-section network is a traditional simple network transformed by a hypernetwork. For each hyperedge \( e \) in a hypernetwork \( G \), through linking each two nodes in \( e \), the groupwise inreaction can be transform into multiple pairwise interactions, and the 2-section network is obtained when all hyperedges are transformed over.

As shown in Figure 2(c), each hypernetwork can transform into simple network. In this way, various methods designed for simple network can be applied to hypernetwork.

### 4 Proposed Method: HND

#### 4.1 Overall framework

In this section, we will introduce our proposed method in detail. Generally, our method can be divided into three steps: First, to supply training samples for subsequent model, we adopt hypernetwork generator to generate lots of small scale synthetic hypernetworks, and calculate each node’s betweenness value in them. According to the groundtruth betweenness values, training samples space are constructed (Section 4.2). Then, we build a node betweenness approximation model in hypernetwork based on hypergraph neural network. This model can be applied to approximate node’s betweenness in hypernetwork according to the network structure and return nodes’ approximation betweenness values (Section 4.3). Finally, combining nodes ranking samples and approximation betweenness values, the pairwise ranking loss are built to optimize parameters in ranking model. After multiple iterations, the model can be used to approximate nodes’ betweenness in real world hypernetworks and further applied to dismantling (Section 4.4).

#### 4.2 Training Samples Generation

In this step, lots of node pair samples are generated according to their betweenness values. Thus, a synthetic hypernetwork generator is needed to generate small scale synthetic hypernetworks. There are two recent proposed synthetic hypernetwork generator, i.e., HyperPA [11] and HyperFF [23]. The main difference between them are that HyperPA generates hypernetwork according to predefined distribution from real world hypernetwork, while HyperFF adopts two hyper-parameters (i.e., burning probability \( p \) and expanding probability \( q \)) to tune the density of synthetic hypernetworks. Comparing with HyperPA, HyperFF is more flexible and hypernetworks generated by it are more generalized. So HyperFF is adopted to accomplish this task.

Once the generator is determined, lots of small scale synthetic hypernetworks can be generated. For each synthetic hypernetwork \( G \), we transform it into its 2-section network, and calculate the corresponding betweenness values \( B \) (where \( B_{v_i} \) denotes the betweenness value of node \( v_i \) ). Then, multiple node pairs are sampled to construct ranking instances. For two random nodes \( v_i \) and \( v_j \), instance \((v_i, v_j, l_{v_i,v_j})\) can be constructed as follows:

\[
l_{v_i,v_j} = \begin{cases} 
1, & B_{v_i} > B_{v_j} \\
0, & B_{v_i} < B_{v_j}
\end{cases}
\]  

It is worth noting that ranking instances cannot be constructed when two nodes’ betweenness values are equal. Therefore, this case should be avoid when sampling node pairs.
Through the method described above, each synthetic hypernetwork can generate one corresponding ranking samples set, and the training samples space \( \Omega \) is also obtained:

\[
S_G = \{(v_0^G, v_1^G, r_0^G), (v_1^G, v_2^G, r_0^G), \ldots\}
\]

\( \Omega = \{SC_1, SC_2, \ldots \} \)

4.3 Betweenness Approximation Model

To approximate node’s betweenness in hypernetwork, we build a ranking model based on hypergraph neural network. For each ranking samples set \( S_G \), model first embeds nodes in \( G \). Then, nodes embedding is feed into a full-connected neural network to obtain the approximation betweenness value. The detail process are introduced as follow.

For given hypernetwork \( G \), it is needed to map the nodes in it into low rank vectors and hypergraph neural network is a well choice. However, due to the ranking model is trained on lots of synthetic hypernetworks and applied to real world hypernetworks which are unseen in training stage. Therefore, current transductive hypergraph neural networks such as HGNN and HyperGCN are not suitable in this case. To solve this problem, we select the inductive hypergraph neural network, HyperSAGE [40], to embed nodes in hypernetworks. Generally, the HyperSAGE contains two level of information aggregation, i.e., hyperedge level aggregation and node level aggregation.

In the step of hyperedge level aggregation, features of hyperedges are first obtained according to nodes’ features as follow:

\[
A^l = \text{softmax}(H^T \odot (X^{l}W_1)^T)
\]

\[
Y^l = A^lX^l
\]

where \( X^l \in \mathbb{R}^{N \times D_l} \) and \( Y^l \in \mathbb{R}^{M \times D_l} \) are nodes’ and hyperedges’ embedding in the \( l \)-th layer, respectively (\( D_l \) denotes embedding dimension in the \( l \)-th layer). \( W_1 \in \mathbb{R}^{D_l \times 1} \) is the trainable parameters. \( \odot \) denote the operation of element-wise product. Function softmax is used to normalize the aggregation weight. It is worth stating that due to the synthetic hypernetworks lack initial nodes feature \( X^0 \), we specify the \( X^0 = 1 \). Based on the view of information propagation, nodes with high betweenness values as hubs of multiple shortest paths, have more powerful information propagation ability than those with low betweenness values. So, the initial nodes features can be regarded as the initial information and each layer of hypergraph neural network is a kind of information propagation. After multiple layers, nodes’ embedding can reflect their ability of information propagation, and be further applied to betweenness approximation.

Once the hyperedges embedding is obtained, the hyperedge level aggregation can be performed as follow:

\[
Y^{l+1} = f([Y^{l}W_2][H^THY^{l}W_3])[W_4]
\]

where \( W_2, W_3 \in \mathbb{R}^{D_l \times D_{l+1}} \) and \( W_4 \in \mathbb{R}^{2D_{l+1} \times D_{l+1}} \) are all trainable parameters. \( \| \) denote the operation of matrix concatenation. Non-linear activation function \( f \) is specified as ReLU.

The next step if node level aggregation. In this step, nodes aggregate information from their adjacent hyperedges as follow:

\[
X^{l+1} = f([X^{l}W_5][HY^{l+1}W_6])[W_7]
\]

where \( W_5 \in \mathbb{R}^{D_l \times D_{l+1}}, W_6 \in \mathbb{R}^{D_{l+1} \times D_{l+1}} \) and \( W_4 \in \mathbb{R}^{2D_{l+1} \times D_{l+1}} \) are trainable parameters.

After multiple layers of HyperSAGE, nodes’s final embedding \( X^L \) is obtained (\( L \) denotes the layer number of HyperSAGE). Through feeding them into a full-connected neural network, nodes’ approximation betweenness values can be calculated as follow:

\[
\hat{B} = f(X^LW_8 + b)
\]

where \( W_8 \in \mathbb{R}^{D_L \times 1} \) and \( b \in \mathbb{R} \) are trainable parameters.
Through the steps above, nodes’ approximation betweenness values are calculated. According to the values, a hypernetwork can be dismantled by greedily removing nodes with the highest approximation values. Once batch nodes are removed, it is necessary to re-calculated the residual nodes’ approximation values due to the changes of hypernetwork structure. So, values approximation and nodes removal are conducted repeated until the scale of GCC in hypernetwork decreased to a threshold value or all nodes are removed.

4.4 Optimization

In order to optimize the approximation model, node ranking samples generated in Section 4.2 are utilized to update all trainable parameters in model. Thus, we build the pairwise ranking loss as follow.

For each samples set $S_G$, the BPR loss of each instance $(v^G_i, v^G_j, l^G_{v_i v_j})$ are calculated:

\[
\hat{l}^G_{v_i v_j} = \text{sigmoid}(\hat{B}^G_{v_i} - \hat{B}^G_{v_j})
\]

\[
loss^G_{v_i v_j} = -l^G_{v_i v_j} \log(\hat{l}^G_{v_i v_j}) - (1 - l^G_{v_i v_j}) \log(\hat{l} - \hat{l}^G_{v_i v_j})
\]

where $\hat{B}^G_{v_i}$ denotes the approximation betweenness value of node $v_i$ in hypernetwork $G$. According to the pairwise loss of sinle instance, the total loss of whole samples space $\Omega$ are calculated as follow:

\[
loss = \frac{1}{\Omega} \sum_{S_G \in \Omega} \frac{1}{|S_G|} \sum_{(v^G_i, v^G_j, l^G_{v_i v_j}) \in S_G} loss^G_{v_i v_j}
\]

After got the total loss of whole samples space $\Omega$, gradient decrease [21] can be utilized to update all trainable parameters in model. Through multiple iterations of updating, the loss of model will be convergence. Then, the model can be used to approximate nodes’ betweenness in real world hypernetworks. Algorithm 1 shows the whole training process of HND.

4.5 Time Complexity

In the process of inference, the time complexity of HND is mainly from hypernetwork embedding (i.e., HyperSAGE) and the approximation function (i.e., the full-connected neural network). For the former part, there are two steps in each layer of HyperSAGE: In the step of hyperedge level aggregation, the time complexity of attention mechanism and information propagation are both $O(MD)$. In the step of node level aggregation, the time complexity of information propatgation is $O(ND)$. Therefore, the time complexity of HyperSAGE is $O((N + 2M)LD)$. For the latter part, the time complexity of full-connected layer is $O(ND)$. So, the inference complexity of HND is $O((N + 2M)LD) + ND$. For betweenness centrality, its time complexity for unweight network is $O(NM)$. Thus, comparing with betweenness, our proposed HND has a linear time complexity and is more suitable for application.

5 EXPERIMENTS

In this section, we conduct extensive experiments to verify the effectiveness of our proposed method. Detail experimental setup and results are shown as follow.

5.1 Experimental Datasets and Setup

5.1.1 Experimental Datasets. Five real world hypernetworks are collected to evaluate our proposed method. Brief introductions about these datasets are listed below.

Algorithm 1: Training process of HND

**Input:** Synthetic hypernetworks scale range $(N_{min}, N_{max})$, burning propability range $(p_{min}, p_{max})$ in HyperFF, expanding propability range $(q_{min}, q_{max})$ in HyperFF, number $J$, max iteration number $I$, HyperSAGE layer number $L$, embedding dimension $D$, node pairs sample ratio $r$

**Output:** Approximation model parameters $\Theta$

1. Initialize model parameters $\Theta$ randomly
2. Initialize samples space $\Omega = \{\}$
3. for $n = 1$ to $J$ do
4.   take random values $p, q, N$ in range $[p_{min}, p_{max}], [q_{min}, q_{max}], [N_{min}, N_{max}]$, respectively
5.   generate a hypernetwork $G$ which has $N$ nodes by $F$ with hyperparameters $p$ and $q$
6.   calculate the betweenness value $\hat{B}^G$ of $G$’s corresponding 2-section network
7.   sample $\lceil rN \rceil$ node pairs randomly
8.   generate ranking samples $S_G$ according to Equations (5) and (6)
9.   insert $S_G$ into samples space $\Omega$
10. end for
11. for $i = 1$ to $I$ do
12.   for $S_G \in \Omega$ do
13.     embed nodes in hypernetwork $G$ according to Equations (8) - (11)
14.     calculate nodes’ approximation betweenness values according to Equation (12)
15.     calculate pairwise loss according to Equation (14)
16.     update model parameters $\Theta$ through gradient decreased
17.   end for
18. end for
19. return Model parameters $\Theta$

- **Cora-co-authorship** [39] This dataset contains some papers released in the field of machine learning. Authors of each paper are co-authors. We construct hypernetwork through regarding authors as nodes and co-author relations as hyperedges.
- **Citeseer** [39] This dataset contains papers in six fields and their citation relations. To construct hypernetwork, papers and citation relations are map into nodes and hyperedges, respectively.
- **MAG** [33] This dataset contains paper from the history field of Microsoft Academic Graph and their authors. Similar to Cora-co-authorship, papers are considered as nodes and co-author relations are maped into hyperedges.
- **Pubmed** [39] This dataset contains paper from the field of diabetes and their citation relations. Due to the form of Pubmed is same as Citeseer, the way of constructing hypernetworks in Citeseer is also suitable for Pubmed.
- **NDC** [4] This dataset contains many drugs and each drug is consisted of multiple substances. Through considering substances and drugs as nodes and hyperedges, respectively, hypernetwork can be constructed.

For these datasets, we conducted some pre-process steps. Due to the dismantling problem concentrates on the connectivity of network, so a disconnected network will disturb experimental results. Therefore, for each original hypernetwork, we only select their GCCs as the initial hypernetwork waiting to be dismantled. Some statistics of hypernetworks after pre-process are summarized in Table 1.
Table 1. The statistics of datasets

| Datasets             | Cora-co-authorship | Citeseer | MAG    | NDC    | Pubmed  |
|----------------------|--------------------|----------|--------|--------|---------|
| Number of Nodes      | 1676               | 1019     | 1669   | 3065   | 3825    |
| Number of Hyperedges | 463                | 626      | 784    | 4533   | 5432    |
| Average Node Hyperdegree | 1.66        | 2.23     | 1.59   | 13.57  | 7.45    |
| Average Hyperedge Size | 6.00           | 3.63     | 3.38   | 9.17   | 5.25    |

5.1.2 Baselines. To evaluate the dismantling performance of our method, various baselines are selected and their introduction are listed below:

- **Highest Degree Adaptive (HDA)** This method dismantles a network according to degree. In each removal step, node with the highest degree will be removed. After each removal step, nodes’ degree will be re-calculated due to the changes of network structure.

- **Highest Hyperdegree Adaptive (HHDA)** This method dismantles hypernetwork according to hyperdegree. Similar to HDA, node with the highest hyperdegree will be removed in each step, and nodes’ hyperdegree will be updated after removal.

- **Collective Influence (CI)** [28] This method removes nodes according to their CI values, and the way of calculating CI value for each node is:

\[
CI_{v_i} = (\text{deg}(v_i) - 1) \sum_{v_j \in \text{Nei}_k(v_i)} (\text{deg}(v_j) - 1)
\]

where $\text{Nei}_k(v_i)$ denotes the $k$-hop neighbors of node $v_i$ and in this paper we set the $k$ is 2. Node with the highest CI value will be removed in each step, and the CI values of residual nodes will also be re-calculated.

- **GND** [31] GND first computes the network’s weighted laplacian matrix which is utilized to obtain nodes’ eigenvector through spectrum approximation. Then, nodes are split into two groups according to their eigenvector. Finally, weighted vertex cover algorithm is used to select nodes to be removed.

- **FINDER** [14] This method is based on deep reinforcement learning. It builds an agent and make it do dismantling exercises on lots of small scale synthetic networks to optimize the agent’s dismantling strategy. Once its strategy achieves convergence, the agent can be used to dismantle real world networks.

- **SubTSSH** [1] This method is designed for the problem of selecting target nodes set in hypernetworks. Through iteratively conduct nodes removal and influence propagation, it can be used for hypernetwork dismantling.

- **HITTER** [40] This method is specifically designed for hypernetwork dismantling based on deep reinforcement learning. It first builds an agent to do trial-and-errors on many small scale synthetic hypernetworks. After the agent’s dismantling strategy is well optimized, it can be utilized to dismantle real world hypernetworks.

In these above baselines, HDA, HHDA, CI and SubTSSH are methods based on centrality measure, GND is the recently proposed heuristic methods, while the FINDER and HITTER are methods based on deep learning. Generally, these baselines cover several commonly used dismantling methods. Moreover, some of these baselines (HDA, CI, GND and FINDER) are not suitable for hypernetworks.
Table 2. The Overall Performance

| Datasets          | HDA | CI     | GND   | FINDER | HHDA | SubTSSH | HITTER | HND    |
|-------------------|-----|--------|-------|--------|------|---------|--------|--------|
| Cora-co-authorship| 0.1564 | 0.1181 | 0.1111 | 0.4068 | 0.0977 | 0.1267 | 0.0792 | **0.0713** |
| Citeseer          | 0.0930 | 0.0915 | 0.2528 | 0.1109 | 0.0788 | 0.0965 | 0.0607 | **0.0592** |
| MAG               | 0.0261 | 0.0238 | 0.0335 | 0.0410 | 0.0195 | 0.0226 | 0.0130 | 0.0161  |
| Pubmed            | 0.3933 | 0.3930 | 0.3606 | 0.4809 | 0.3831 | 0.4038 | 0.3529 | **0.3344** |
| NDC               | 0.2608 | 0.2623 | 0.4372 | 0.4804 | 0.2374 | 0.2666 | 0.2209 | **0.2144** |

Therefore, we transform hypernetworks into their corresponding 2-section network when applying these methods.

5.1.3 Metrics. Accumulated normalized connectivity (ANC) [32] is adopted to evaluate performance of hypernetwork dismantling. The calculation of ANC is:

\[
\text{ANC}(\kappa) = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{connectivity}(G\{v_1, v_2, \ldots, v_k\})}{\text{connectivity}(G)}
\]

where \(\kappa = \{v_1, v_2, \ldots, v_K\}\) is the nodes removal sequence obtained by various dismantling methods. \(G\{v_1, v_2, \ldots, v_k\}\) denotes the residual hypernetwork after removing nodes \(\{v_1, v_2, \ldots, v_k\}\) from \(G\). Intuitively, given a nodes removal sequence \(\kappa\), a low ANC value means this sequence can dismantle network effectively.

5.1.4 Experimental Setup. In the step of samples generation, the minimal scale of synthetic hypernetwork \(N_{min}\) is fixed to 100 and the maximal scale \(N_{max}\) is set to 150. The burning propabality range \((p_{min}, p_{max})\) and expanding propabability range \((q_{min}, q_{max})\) are all fixed to \((0.1, 0.4)\). The number of synthetic hypernetworks \(J\) is set to 1000 and for each hypernetwork, we randomly sample \([0.9N]\) node pairs as training samples. In the betweenness approximation model, the layer of HyperSAGE is fixed to 4 and the embedding dimension is set to 32. In training process, the learning rate of model is set to 0.005. And 50 synthetic hypernetworks are also generated as validation dataset to verify the training of model. Early stopping mechanism is adopted to avoid the over-fitting of model, and its patience is fixed to 100.

5.2 Experimental Results and Analyses

5.2.1 Overall Experimental Results. The proposed HND and baselines are applied to these real world hypernetworks for dismantling, and their performance are summarized in Table 2. According to the results, we can easily conclude that our proposed HND has a significant dismantling performance comparing with baselines in the majority of datasets. This indicates that with the help of deep learning technology, HND can cost a limitation time consumption to well approximate the betweenness centrality which has a high computation complexity. Figure 3 also shows the ANC curves of dismantling these hypernetworks via HND and baselines. It is obvious that our HND can break a hypernetwork with removing less nodes when comparing with other methods, which further confirm the effectiveness of HND.

Although the HND can achieve a better performance than the majority of baselines, it is weakly inferior to HITTER in the MAG dataset. To explore the reason of this case, we dismantle the MAG

J. ACM, Vol. 37, No. 4, Article 111. Publication date: August 2018.
through nodes’ groundtruth betweenness centrality and the curve of dismantling is shown in Figure 4. It is easily to find that the dismantling results of HITTER is very close to the performance of betweenness centrality. Therefore, in the case of existing approximation error, HND cannot achieve a better effect than HITTER. According to the statistics of MAG and its dismantling curves, we can find that the MAG is pretty sparse, which cause that it has no resistance to dismantling. So, the majority of methods can destroy this hypernetwork through removing less than 10 percent of nodes. This is the core reason why the performance in it is hard to increase.

Another phenomenon worth noting is that performances of these methods suitable for traditional simple network (i.e., HDA, CI, GND and FINDER) are weak than those suitable for hypernetwork (i.e., HHDA, SubTSSH and HITTER). We think the reason is that the transformation of hypernetwork
to 2-section network will introduce noise which disturb dismantling. For example, two people which attend a party are not certainly know each other. Through modeling this party by a hyperedge, these two people are regarded as nodes. However, after the 2-section transformation, there is a link between these two people, which is obvious a noise relation. So, applying these methods through this way will influence dismantling performance. Moreover, the FINDER, which is the SOTA method for traditional network as our best of knowledge, performs bad even than other traditional methods. We think that the BA model [2] used in its training process cannot model the structure of hypernetwork. Therefore, the training direction of FINDER is wrong, which leads to the worst performance of it.

5.2.2 Analyses of Model Efficiency. The motivation of proposing HND is to solve the high computation complexity of betweenness. Moreover, adopting the mode of supervise learning also should make the HND more efficient than HITTER which employs reinforcement learning. So, we conduct experiments to analyse the efficiency of HND, HITTER and betweenness.

To show the inference efficiency of HND, we utilize the HND, HITTER and betweenness centrality to dismantle different scale of synthetic hypernetworks. The time consumption of each calculation in these methods is shown in Figure 5. It is obviously that HND and HITTER are more efficient than betweenness centrality. With the increasement of hypernetwork scale, the time consumption of HND and HITTER increase linearly, while the betweenness’ cost grows exponentially. The main reason is that the deep graph learning technology adopted by HND and HITTER has powerful representation ability, and they can fit necessary values in less computation complexity. Moreover, we can easily find that the calculation time of betweenness increases exponentially with the change of network scale, while the calculation time of our method keeps growing with a linear rate. This phenomenon confirm that our proposed HND has a linear time complexity.

To show the training efficiency of HND, we train the HND and HITTER repeatedly and record their training time. The results is shown in Table 3. It is easily find that the HITTER usually needs tens of thousands iterations to achieve convergence, while the HND only cost hundreds of iterations. The training time of HND is also obviously less than HITTER. The core reason which leads to this case is the difference of training mode. The mode of reinforcement learning adopted by HITTER needs to cost huge time to explore the excellent action in dismantling, while the HND employs

---

**Fig. 5. Changes of Computation Time with The Increase of Hypernetwork Scale**

---

J. ACM, Vol. 37, No. 4, Article 111. Publication date: August 2018.
Table 3. Efficiency Comparison between HITTER and HND

| Method | Iterations Number until Model Convergence | Time of Training Process (s) |
|--------|------------------------------------------|-------------------------------|
| HITTER | 84766(±17470)                            | 32505.51(±2808.82)            |
| HND    | 101(±45)                                 | 11235.07(±4866.57)           |

5.2.3 Impact of Embedding Layer Number for Dismantling. HND relies on node embeddings obtained by multiple layer aggregation to approximate the betweenness centrality. Therefore, the layer number $L$ has a vital influence to dismantling performance. To show the impact of layer number, we increase the $L$ from 1 to 8 and training the HND to observe the dismantling performance. The influence of layer number to dismantling in each dataset is shown in Figure 6. According to the figure, a moderate value of $L$ can achieve better effect, while a too high or too low value will all lead to bad results. We think this case is mainly caused by three reasons: In one hand, setting a too low layer number will limit the propagation of information, which makes node embeddings not aware of global structure. In other hand, a too high layer number will lead to information propagating repeatedly and introduce noise to node embeddings. Moreover, GNNs will face the problem of over-smoothing when the layer number is too high, which causes node embeddings cannot be distinguished and further lead to the approximation error. In addition to the efficiency, it also will bring extra time computation when setting a high layer number. Thus, taking both the effect and efficiency into consider, it is necessary to choose a proper value in real application.

6 CONCLUSION AND FUTURE WORK

In this paper, we proposed a novel hypernetwork dismantling method, called HND. A node betweenness approximation model is built based on hypergraph neural network. Through training the mode of supervise learning and avoid so complex action explorations. So, HND can obviously decrease the training time comparing with HITTER.
this model on lots of synthetic node ranking samples, the well-trained model can be utilized to approximate the nodes' betweenness in real world hypernetworks and further used for dismantling. With the help of powerful representation ability of deep learning technology, our model can achieve a better dismantling effect. Extensive experiments are conducted and the experimental results have confirmed the effectiveness of our proposed model. Moreover, the experiments and analyses of model also prove that our HND also have the advantages of efficiency.

In the future, we will consider to introduce the factor of time to hypernetwork and study the problem of dynamic hypernetwork dismantling. Although the hypernetwork is suitable to model real world than traditional network, the factor of time has not been considered. There is no doubt that the dynamic hypernetwork is more suitable for modeling the real groupwise inreations in some cases such as epidemic control and so on. So, exploring the method to dismantle dynamic hypernetwork is meaningful.

REFERENCES

[1] Alessia Antelmi, Gennaro Cordasco, Carmine Spagnoolu, and Przemyslaw Szufel. 2021. Social Influence Maximization in Hypergraphs. Entropy 23, 7 (2021), 796.
[2] Albert-László Barabási and Réka Albert. 1999. Emergence of Scaling in Random Networks. Science 286, 5439 (1999), 509–512.
[3] Federico Battiston, Giulia Cencetti, Iacopo Iacopini, Vito Latora, Maxime Lucas, Alice Patania, Jean-Gabriel Young, and Giovanni Petri. 2020. Networks beyond pairwise interactions: Structure and dynamics. Physics Reports 874 (2020), 1–92.
[4] Austin R. Benson, Rediet Abebe, Michael T. Schaub, Ali Jadbabaie, and Jon Kleinberg. 2018. Simplicial closure and higher-order link prediction. Proceedings of the National Academy of Sciences 115, 48 (2018), E11221–E11230.
[5] Claude Berge. 1989. Hypergraphs: combinatorics of finite sets. Elsevier, North Holland.
[6] Alfredo Braunstein, Luca Dall’Asta, Guilhem Semerjian, and Lenka Zdeborová. 2016. Network dismantling. Proceedings of the National Academy of Sciences 113, 44 (2016), 12368–12373.
[7] Alain Bretto. 2013. Hypergraph Theory: An Introduction. Springer International Publishing.
[8] Tianwen Chen and Raymond Chi-Wing Wong. 2020. Handling Information Loss of Graph Neural Networks for Session-Based Recommendation. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Virtual Event, CA, USA, 1172–1180.
[9] Weijian Chen, Yulong Gu, Zhaochun Ren, Xiangnan He, Hongtao Xie, Tong Guo, Dawei Yin, and Yongdong Zhang. 2019. Semi-supervised User Profiling with Heterogeneous Graph Attention Networks. In Proceedings of the 28th International Joint Conference on Artificial Intelligence. 2116–2122.
[10] Zhongpu Chen, Bin Yao, Zhi-Jie Wang, Xiaofeng Gao, Shuo Shang, Shuai Ma, and Minyi Guo. 2021. Flexible Aggregate Nearest Neighbor Queries and its Keyword-Aware Variant on Road Networks. IEEE Transactions on Knowledge and Data Engineering 33, 12 (2021), 3701–3715.
[11] Manh Tuan Do, Se-Eun Yoon, Bryan Hooi, and Kijung Shin. 2020. Structural Patterns and Generative Models of Real-World Hypergraphs. In Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. Virtual Event, CA, USA, 176–186.
[12] Mohammadreza Doostmohammadian, Hamid R. Rabiee, and Usman A. Khan. 2020. Centrality-based epidemic control in complex social networks. Social Network Analysis and Mining 10, 1 (2020), 32:1–32:11.
[13] Changjun Fan, Li Zeng, Yuhui Ding, Muhao Chen, Yizhou Sun, and Zhong Liu. 2019. Learning to Identify High Betweenness Centrality Nodes from Scratch: A Novel Graph Neural Network Approach. In Proceedings of the 28th ACM International Conference on Information and Knowledge Management. New York, USA, 559–568.
[14] Changjun Fan, Li Zeng, Yizhou Sun, and Yang-Yu Liu. 2020. Finding key players in complex networks through deep reinforcement learning. Nature Machine Intelligence 2 (2020), 317–324.
[15] Yifan Feng, Haoxuan You, Zhihao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph Neural Networks. In Proceedings of the AAAI Conference on Artificial Intelligence. Honolulu, Hawaii, USA, 3558–3565.
[16] Marco Grassia, Manlio De Domenico, and Giuseppe Mangioni. 2021. Machine learning dismantling and early-warning signals of disintegration in complex systems. Nature Communications 12, 1 (2021), 5190.
[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive Representation Learning on Large Graphs. In Advances in Neural Information Processing Systems. Long Beach, CA, USA, 1024–1034.
[18] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, YongDong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and Powering Graph Convolution Network for Recommendation. In Proceedings of the 43rd International ACM SIGIR
[19] Jing Huang and Jie Yang. 2021. UniGNN: a Unified Framework for Graph and Hypergraph Neural Networks. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*. 2563–2569.

[20] David Kempe, Jon Kleinberg, and Eva Tardos. 2003. Maximizing the Spread of Influence through a Social Network. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Washington, D.C., USA, 137–146.

[21] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*. San Diego, CA, USA.

[22] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. In *5th International Conference on Learning Representations*. Toulon, France.

[23] Yunbum Kook, Jihoon Ko, and Kijung Shin. 2020. Evolution of Real-world Hypergraphs: Patterns and Models without Oracles. In *IEEE International Conference on Data Mining*. 272–281.

[24] Dong Li and Jiming Liu. 2021. Modeling Influence Diffusion over Signed Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 33, 2 (2021), 613–625.

[25] Dong Liu, Chi K. Tse, and Xi Zhang. 2019. Robustness Assessment and Enhancement of Power Grids From a Complex Network’s Perspective Using Decision Trees. *IEEE Transactions on Circuits and Systems II: Express Briefs* 66, 5 (2019), 833–837.

[26] Wenjian Luo, Daofu Zhang, Li Ni, and Nannan Lu. 2021. Multiscale Local Community Detection in Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 33, 3 (2021), 1102–1112.

[27] Sunil Kumar Maurya, Xin Liu, and Tsuyoshi Murata. 2021. Graph Neural Networks for Fast Node Ranking Approximation. *ACM Trans. Knowl. Discov. Data* 15, 5 (2021), 1556–4681.

[28] Flaviano Morone and Hernán A. Makse. 2015. Influence maximization in complex networks through optimal percolation. *Nature* 524, (2015), 65–68.

[29] Salomon Mugisha and Hai-Jun Zhou. 2016. Identifying optimal targets of network attack by belief propagation. *Physical Review E* 94, 1 (2016), 012305.

[30] Huaxin Pei, Yi Zhang, Qinghua Tao, Shuo Feng, and Li Li. 2021. Distributed Cooperative Driving in Multi-Intersection Road Networks. *IEEE Transactions on Vehicular Technology* 70, 6 (2021), 5390–5403.

[31] Xiao-Long Ren, Niels Gleinig, Dirk Helbing, and Nino Antulov-Fantulin. 2019. Generalized network dismantling. *Proceedings of the National Academy of Sciences* 116, 14 (2019), 6554–6559.

[32] Christian M. Schneider, André A. Moreira, José S. Andrade, Shlomo Havlin, and Hans J. Herrmann. 2011. Mitigation of malicious attacks on networks. *Proceedings of the National Academy of Sciences* 108, 10 (2011), 3838–3841.

[33] Arnab Sinha, Zhihong Shen, Yang Song, Hao Ma, Darrin Eide, Bo-June (Paul) Hsu, and Kuansan Wang. 2015. An Overview of Microsoft Academic Service (MAS) and Applications. In *Proceedings of the 24th International Conference on World Wide Web*. Florence, Italy, 243–246.

[34] Haicheng Tu, Yongxiang Xia, Herbert Ho-Ching Fu, and Xi Chen. 2019. Optimal Robustness in Power Grids From a Network Science Perspective. *IEEE Transactions on Circuits and Systems II: Express Briefs* 66, 1 (2019), 126–130.

[35] Petar Veličkovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. In *6th International Conference on Learning Representations*. Vancouver, BC, Canada.

[36] Dongjie Wang, Pengyang Wang, Kunpeng Liu, Yuanchun Zhou, Charles E Hughes, and Yanjie Fu. 2021. Reinforced Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Neural Networks for Fast Node Ranking Approximation. *ACM Trans. Knowl. Discov. Data* 15, 5 (2021), 1556–4681.

[37] Xiang Wang, Xiangnan He, Meng Fang, and Tat-Seng Chua. 2019. Neural Graph Collaborative Filtering. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Paris, France, 165–174.

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks?. In *7th International Conference on Learning Representations*. New Orleans, LA, USA.

[39] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2019. HyperGCN: A New Method For Training Graph Convolutional Networks on Hypergraphs. In *Advances in Neural Information Processing Systems*. Vancouver, BC, Canada, 1511–1522.

[40] Dengcheng Yan, Wenyin Xie, and Yiwen Zhang. 2021. Hypermnetwork Dismantling via Deep Reinforcement Learning. *CoRR* abs/2104.14332 (2021).

[41] Xiaoyan Yin, Xiao Hu, Yanjiao Chen, Xu Yuan, and Baocun Li. 2021. Signed-PageRank: An Efficient Influence Maximization Framework for Signed Social Networks. *IEEE Transactions on Knowledge and Data Engineering* 33, 5 (2021), 2208–2222.

[42] Lenka Zdeborová, Pan Zhang, and Hai-Jun Zhou. 2016. Fast and simple decycling and dismantling of networks. *Scientific reports* 6, 1 (2016), 37954.

J. ACM, Vol. 37, No. 4, Article 111. Publication date: August 2018.