HyperTTC: Hypergraph-Empowered Tactic-Specific Traffic Clustering for Atomized APT Detection

Wenhui Du*, Yuanhang He†, Gaolei Li*, Xiao Yang*, Jianhua Li*,Ge Ren*, Kai Zhou‡
*School of Electronic Information and Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China
†National Key Laboratory of Security Communication, China
‡Department of Computing, The Hong Kong Polytechnic University, Hong Kong SAR, China
{halloworld,gaolei_li,youngshall,lijh888,lanceren}@sjtu.edu.cn*, heyh2477@cetcsc.com†,kaizhou@polyu.edu.hk‡

***Abstract*—Cybersecurity professionals often encounter a significant semantic gap between low-level traffic audits and high-level system behaviors, which hinders effective tactic recognition of advanced persistent threats (APTs). Existing provenance graph-based traffic clustering methods uses binary relation to detect attack actions, which can not adequately capture complex interactions among multiple entities. To identify attack tactics from fragmented traffic audits on multiple entities, in this paper we propose a novel hypergraph-empowered tactic-specific traffic clustering (HyperTTC) scheme, which leverages the transformative potential of hypergraphs to aggregate entities that carry the same APT attack tactic together. Different from existing methods, HyperTTC is capable of achieving atomized APT detection, where each attack tactic can be identified with the combination of multi-dimension relations. By constructing a hypergraph structure of fragmented traffic audits, HyperTTC provides an exhaustive representation of APT behaviors, thereby enhancing detection precision and bolstering resilience against sophisticated attack strategies. Extensive experiments on real-world datasets validate the effectiveness of HyperTTC for the F1 score is 12.5% higher than the state of the art method.**

***Index Terms*—Intrusion detection , Hypergraph learning , Supervised learning.**

I. Introduction

The evolving battle between defensive mechanisms and malicious attacks in computing systems remains dynamic and challenging [1]. Despite widespread adoption of Cyber Threat Detection Platforms (CTDP) and Intrusion Detection Systems (IDS), Advanced Persistent Threats (APTs) persist as a significant concern, exploiting semantic gaps between low-level traffic audits and high-level system behaviors. To address this issue, provenance graph-based defense systems have been introduced [2] [3] [4]. These systems transform system data into graph representations, capturing temporal and causal dependencies among entities like processes and files. Advanced graph operations, such as traversal, enable these systems to detect APTs and perform root cause analyses. However, their reliance on binary relations limits their capacity to model complex interdependencies, impacting detection accuracy, runtime efficiency, and granularity in complex environments.

Hypergraph-based approaches offer a more expressive alternative [5]. Unlike conventional graphs, hypergraphs capture higher-order relationships among entities, enhancing the detection and clustering of malicious activities. They also achieve superior runtime efficiency by compactly encoding complex interactions, facilitating faster processing of large-scale datasets. Building on these advantages, we propose the Hypergraph-Empowered Tactic-Specific Traffic Clustering (**HyperTTC**) scheme for atomized APT detection. HyperTTC constructs a hypergraph using the UNSW dataset, representing network entities and their interactions with ten types of hyperedges. By integrating Graph Neural Networks (GNNs) with hypergraphs, HyperTTC models higher-order relationships, significantly improving clustering accuracy and detection capabilities by capturing intricate interdependencies within network traffic.

Our contributions can be summarized as follows:

- We introduce HyperTTC, an innovative hypergraph-based tactic-specific traffic clustering scheme designed for detecting and classifying network traffic in the context of APT attacks. Compared to existing approaches, HyperTTC demonstrates superior accuracy in identifying atomized APT threats. To the best of our knowledge, this work is the first to utilize hypergraphs for clustering network traffic into distinct attack tactics.
- We propose a qualitative representation of network traffic behavior, enabling the clustering of behaviors with similar tactics and the identification of patterns characteristic of APTs. By integrating hypergraph structures with qualitative behavior analysis, HyperTTC provides a more effective solution for atomized APT detection.
- We conducted a systematic evaluation of real-world malicious behavior alongside benchmark datasets. Comparative experiments across four datasets and eight methods reveal that HyperTTC achieves higher F1 and Jaccard scores, demonstrating its effectiveness in capturing nuanced information and detecting atomized APT behaviors.

II. Preliminary

We begin with a detailed exposition on the fundamental definitions and computational formulas relevant to hypergraphs, as outlined in Section II-A. Subsequently, Section II-B offers an example that elucidates the procedural complexities involved in executing an APT attack.

979-8-3315-2096-0/25/$31.00 ©2025 IEEE

A. Hypergraph Structure

In this section, we present key definitions related to hypergraphs. Notably, certain definitions also apply to traditional graphs, as hypergraphs extend graph theory by allowing edges to connect multiple vertices, thereby capturing higher-order interactions among nodes.

Edges in a hypergraph are not restricted to connecting only two vertices; they can link multiple vertices simultaneously. Formally, a hypergraph can be defined as a pair $H = (V, E, X)$, where $V = \{v_1, v_1, ..., v_N\}$ is the set of vertices with size $N = |V|$, $E = \{E_1, E_1, ..., E_M\}$ is the set of hyperedges with size $M = |E|$, a hyperedge $e \in E$ is a subset of vertices in V, X is the set of node attributes features where $x_i \in R^d$. For an arbitrary set S, we let $|S|$ stand for the cardinality of the set, and use $\delta(e) = |e|$ to denote the size of a hyperedge. The hyperedge is called a uniform hyperedge if for all $e \in E$, $\delta(e)$ equals a constant number.

This study explores the node classification problem in hypergraphs, emphasizing the importance of label consistency across the hypergraph structure. This challenge can be effectively framed as a regularization problem, as outlined by Zhou [5]:

$$argmin_f \{\mathcal{R}_{emp}(f) + \Omega(f)\}. \tag{1}$$

Here, $\Omega(f)$ acts as a regularization term on the hypergraph, while $\mathcal{R}_{emp}(f)$ represents the supervised empirical loss. Function $f(\cdot)$ corresponds to a classification function. The regularization $\Omega(f)$ is articulated as

$$\Omega(f) = \frac{1}{2} \sum_{e \in E} \sum_{\{u,v\} \in V} \frac{\omega(e) h(u,e) h(v,e)}{\delta(e)} \phi(u,v), \tag{2}$$

$$\phi(u,v) = \left(\frac{f(u)}{\sqrt{d(u)}} - \frac{f(v)}{\sqrt{d(v)}} \right)^2. \tag{3}$$

We let

$$\theta = D_v^{-\frac{1}{2}} H W D_e^{-1} H^T D_v^{-\frac{1}{2}}, \tag{4}$$

$$\Delta = \mathcal{I} - \Theta, \tag{5}$$

and $\Omega(f)$ can be written as

$$\Omega(f) = f^T, \tag{6}$$

in the above function, Δ, commonly referred to as the hypergraph Laplacian [6], is positive semi-definite and is often applied in clustering tasks.

B. APT Case Example

We use GhostNet attack as an instance, which is one of the typically notorious APTs [7], stealing diplomatic and political information with influence in multiple countries. It is a targeted cyber attack orchestrated by an organized cybercrime group, where the attacking team sent a message to employees containing a malicious link. When an employee clicked on this malicious link, their computer was remotely controlled for several months. Countless pieces of data were stolen, resulting in immeasurable losses. Details are illustrated in Figure 1.

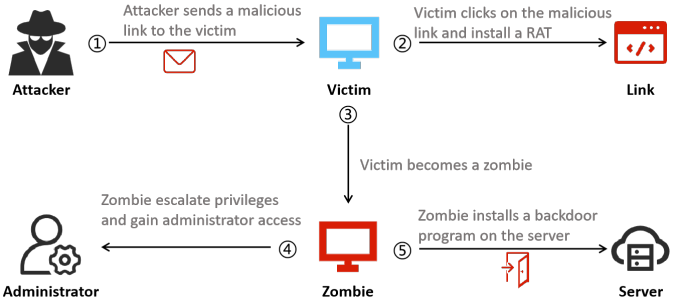

Fig. 1: A Motivating Example.

After obtaining administrator privileges, Attacker navigates the company's network with precision, deliberately avoiding detection by security systems. His primary objective extends beyond initial access, focusing on maintaining a covert presence to maximize the extraction of sensitive information without triggering alerts. Once the company detects the intrusion, Attacker's digital traces vanish, leaving behind a compromised network and disrupted security infrastructure.

III. Methodology: HyperTTC

In this section, we delve into the intricacies of the proposed HyperTTC. It comprises four pivotal steps: 1) Redundant Data Removing; 2) Hypernode Representation; 3) Hyperedge Construction; 4) Behavior Clustering.

A. Redundant Data Removing

HyperTTC harnesses the power of hypergraph construction techniques to preprocess data effectively. Hypergraphs, a generalization of traditional graphs, offer unique advantages in modeling complex relationships among entities. Unlike conventional graphs where edges connect only two nodes, hyperedges in hypergraphs can link any number of nodes, allowing for more flexible and expressive representations of interconnected data.

The first phase in HyperTTC involves two critical steps, aimed at converting raw data into a structured, analyzable format.

Analysis of Redundancy Features. The first step focuses on an in-depth analysis of input records to extract meaningful features. Each record is parsed to identify the entities involved and their interactions. This step lays the foundation for hypergraph construction, where entities are represented as nodes and their interactions form hyperedges. This preliminary hypergraph captures the essential relationships inherent in the dataset, providing a structured representation of the raw data.

Noise Reduction. After constructing the initial hypergraph, the second step addresses data redundancy and clarity enhancement. Redundant edges between entity pairs are removed, simplifying the representation. Furthermore, multiple interactions involving the same entities are consolidated into a single hyperedge, reducing unnecessary complexity. This optimization ensures a cleaner, more concise hypergraph that

retains the essential structural information while improving the overall data quality.

B. Hypernode Learning

We utilized the HyperTTC method to process the dataset through hypergraph construction techniques, leveraging the flexibility of hypergraphs to model complex relationships among entities more effectively than traditional graph structures. The process begins by loading the raw dataset using Python's Pandas library, followed by separating the target variable (label) from the categorical features. Label encoding is then applied to convert categorical features into numerical representations, ensuring compatibility with clustering algorithms.

To mitigate class imbalance, we apply the ADASYN oversampling technique, which generates synthetic samples for minority classes to achieve a balanced class distribution. Principal component analysis (PCA) is employed for dimensionality reduction, preserving essential information while decreasing the computational complexity of the clustering process. Feature values are then normalized using Min-Max scaling to improve algorithm convergence and overall performance.

For feature aggregation, we implement Hypernode convolution to consolidate vertex features into the hyperedges they belong to. A transformation matrix T is computed from the vertex features to facilitate feature permutation and weighting. This matrix enables the seamless exchange of inter-vertex and inter-channel information. To compute T, we employ a Multi-Layer Perceptron (MLP), and the transformed features are subsequently compressed using convolution operations, as defined by Equations 7 and 8.

$$T = MLP\left(X_u\right), \tag{7}$$

$$x_e = conv\left(T \cdot MLP\left(X_u\right)\right). \tag{8}$$

In the hypergraph node representation learning phase, the objective function is defined as:

$$\mathcal{L}_{hypernode} = \sum_{i=1}^{N} \sum_{j \in \mathcal{N}(i)} \left\| h_i - h_j \right\|_2^2, \tag{9}$$

where N denotes the total number of nodes, $\mathcal{N}_{(i)}$ represents the neighbors of node i, and h_i is the learned representation of node i. This objective seeks to minimize the Euclidean distance between the embeddings of nodes and their respective neighbors, promoting similarity in representations for nodes with analogous contexts in the hypergraph. Such a formulation effectively captures the complex interrelations among entities, such as processes and files, within the network.

This method demonstrates efficacy in managing complex network traffic data. By retaining critical attributes such as attack types and employing systematic data organization and normalization techniques, it enhances the precision in identifying anomalous behaviors within networks. Consequently, this approach significantly improves the accuracy and effectiveness of network intrusion detection systems.

C. Hyperedge Construction

In conventional graph representations, data points are typically modeled as nodes, with edges representing relationships or interactions. However, in the domain of network intrusion detection, such representations often fail to reflect the complex, multi-party interactions characteristic of cyberattacks. To overcome this limitation, we employ a hypergraph-based approach, where each data point is represented as a node, and hyperedges are constructed based on the attack categories present in the dataset. Unlike traditional edges, hyperedges can connect multiple nodes simultaneously, enabling the representation of intricate relationships among data points.

To construct the hypergraph, we segment the dataset according to attack types, treating each type as a distinct category for forming hyperedges. For instance, attacks such as Denial of Service (DoS), Probe, Remote to Local (R2L), and User to Root (U2R) are categorized separately, with each forming a hyperedge that links the nodes corresponding to instances of that attack type.

This hypergraph structure captures the collective behavior of data points associated with specific attack types, offering a more nuanced representation of network intrusions. Through the use of hyperedges, we construct a hyperedge matrix that effectively encodes the relationships between various attack categories and their corresponding data instances, providing a holistic view of the network intrusion landscape.

The hyperedge convolution process aggregates hyperedge features into the centroid vertex feature. This aggregation is guided by an attention mechanism incorporated within the hyperedge convolution framework. Leveraging a multi-layer perceptron (MLP) [8], weight scores are computed for each hyperedge, enabling the model to prioritize relevant features selectively. As a result, the centroid vertex feature is obtained through a weighted summation of the hyperedge features, encapsulating the selective integration of crucial information. This process can be mathematically expressed as follows:

$$w = softmax\left(x_e W + b\right), \tag{10}$$

$$x_u = \sum_{i=0}^{|Adj(u)|} w^i x_e^i. \tag{11}$$

In the formula above, $|Adj\left(u\right)|$ denotes the size of adjacent hyperedge set, x_e denotes adjacent hyperedge features and x_u denotes hypernode feature. W and b are learnable parameters.

In hyperedge construction and aggregation stage. Our objective function is

$$\mathcal{L}_{edge} = \sum_{e \in E} \left\| x_e - \sum_{u \in e} w_{eu} x_u \right\|_2^2, \tag{12}$$

where E is the set of hyperedges, x_e is the feature vector of hyperedge e , $x_u$is the feature vector of node u in hyperedge e , and w_{eu} are the attention weights. This objective function focuses on the aggregation of node features into hyperedge features. The attention weights w_{eu} are learned to emphasize more relevant nodes, facilitating the effective capture of higher-order relationships.

This hypergraph-based representation offers several advantages over traditional graph-based approaches. It enables the integration of higher-order interactions, facilitating a deeper understanding of the underlying patterns and dependencies present in the network traffic data. Additionally, it provides a more expressive structure for feature extraction and analysis, leading to improved detection and classification performance.

D. Behavior Clustering

We conduct hypergraph-based model training based on the representations of hyperedges and hypernodes obtained in sections III-B and III-C. Our hypergraph model is based on Graph Convolutional Networks (GCNs) [9]. The forward model for a two-layer GCN can be simplified as follow:

$$Z = f_{GCN}(X, A) = softmax\left(\bar{A}ReLU\left(\bar{A}X\Theta_1\right)\Theta_2\right). \tag{13}$$

where $\Theta_1 \in R^{p\times h}$ is an input-to-hidden weight matrix for a hidden layer with h hidden units and $\Theta_2 \in R^{h\times r}$ is a hidden-to-output weight matrix. The softmax activation function is defined as $softmax\left(x_i\right) = \frac{exp(x_i)}{\sum_j exp\left(x_j\right)}$ and applied row-wise. For a supervised classification task with k categories, we minimize the following cross entropy over the labelled dataset V_L:

$$\mathcal{L} = -\sum_{i\in V_L}\sum_{j=1}^{q} Y_{ij} ln Z_{ij}. \tag{14}$$

We employ the cosine function to compare the similarity of each entity in the hypergraph, thus facilitating clustering. Firstly, each entity in the dataset is represented as a vector, reflecting its contained features and attributes. Then, the cosine function is utilized to compute the similarity between these vectors. Cosine similarity measures the cosine value of the angle between two vectors, typically used to gauge directional similarity between them, ranging from -1 to 1, where 1 denotes perfect similarity and -1 denotes complete dissimilarity.

HyperTTC calculates the semantic relationships using cosine similarity as follow:

$$S\left(F_m, F_n\right) = \frac{\sum_{e_i\in F_m}\sum_{e_j\in F_n} e_i \cdot e_j}{\sqrt{\sum_{e_i\in F_n}\left(e_i\right)^2} \times \sqrt{\sum_{e_j\in F_n}\left(e_j\right)^2}}. \tag{15}$$

In this context, F_m and F_n enote the vector representations of two behavior instances, encapsulating the aggregated features derived from their respective constituent events. The cosine similarity score, represented as $S(F_m, F_n)$, measures the alignment between these vectors, with higher scores indicating a stronger semantic similarity between the behaviors.

IV. Experiment

This section starts by describing the experimental setup, as detailed in Section IV-A. Section IV-B presents an overview of the datasets utilized in this study, including a real-world network traffic dataset and widely recognized benchmark datasets. In Section IV-C, we evaluate the efficiency of HyperTTC and perform a comparative analysis, examining its F1-score and Jaccard index relative to other methods.

A. Experiment Setup

We developed HyperTTC using Python 3.11.1. The experiments were carried out on a system running Ubuntu 20.04.6 LTS, equipped with an AMD EPYC 7F72 CPU @ 3.20GHz, NVIDIA A5000 and A6000 GPUs, and 64 GB of RAM.

B. Datasets

UNSW Dataset. In this study, the UNSW dataset [10] serves as the primary resource for our experiments. Developed by the cybersecurity research team at the University of New South Wales, Australia, this dataset is a widely acknowledged benchmark in network intrusion detection research. It includes both normal and malicious traffic, offering diverse network flow data. The dataset features fundamental network characteristics, traffic statistics, protocol-specific attributes, and content-based metrics derived from network traffic, providing a comprehensive foundation for analyzing and distinguishing various network intrusion behaviors.

Existing Benchmark Datasets. To thoroughly evaluate the performance of HyperTTC, we also leverage three citation datasets: Cora-author and Citeseer-citation [11]. The Cora-author dataset comprises academic articles categorized into diverse disciplines along with their associated authors. Meanwhile, the Citeseer-citation dataset captures citation relationships among documents within the Citeseer repository, offering valuable insights for research in information retrieval and related fields.

C. Efficiency Analysis

In this section, we assess the effectiveness of HyperTTC for multi-granularity APT detection using the UNSW-NB dataset alongside other benchmark datasets, while performing comparative analyses with various clustering methodologies.

The evaluation employs two widely recognized metrics for community detection: the Jaccard score [12] and the F1 score [13]. The data is divided into training, validation, and testing sets in a 60%/20%/20% ratio. Each method undergoes three runs, with the final scores averaged. Notably, HyperTTC achieves an F1-score of 70.449 ± 1.538% on the UNSW dataset, highlighting its strong precision and recall rates. This demonstrates HyperTTC's effectiveness in accurately identifying atomized APTs at multiple levels of granularity.

Our analysis includes a comparative evaluation of several clustering methods, including GCN [14], GAT [15], CLARE [16], Bespoke [13], HCHA [17], and HGNN [18], which represent diverse approaches applied across clustering tasks in various domains. Tables I and II present the F1 and Jaccard scores achieved on the UNSW, Cora-authorship, and Citeseer-citation datasets. To provide a more intuitive comparison, four line charts are used to visualize the results.

Compared to other methods, HyperTTC demonstrates superior performance in multi-granularity APT detection, offering enhanced interpretability and effectiveness on both real-world and benchmark datasets. This establishes HyperTTC as a dependable tool for cybersecurity applications involving APT detection.

TABLE I: Performance (Mean ± std%) of different methods on UNSW. Higher F1 indicates better overall clustering performance and higher Jaccard indicates greater similarity between predicted clusters and true clusters. The best result is shown in bold.

Dataset	**Metric**	GCN	GAT	CLARE
UNSW	F1	48.550 ± 2.652	51.612 ± 3.516	53.360 ± 3.573
	Jaccard	46.841 ± 5.264	43.415 ± 2.487	47.903 ± 4.789
Dataset	**Metric**	HGNN	HCHA	**HyperTTC**
UNSW	F1	62.290 ± 1.873	60.239 ± 2.775	**70.449 ± 1.538**
	Jaccard	53.022 ± 2.483	50.553 ± 1.995	**57.993 ± 0.732**

TABLE II: Performance (Mean ± std%) of Different Methods in Transfer Learning Tasks

Method	**Dataset**	**F1**	**Jaccard**
GCN	Citeseer-citation	49.879 ± 1.705	43.968 ± 1.545
	Cora-authorship	66.936 ± 6.322	39.201 ± 3.398
GAT	Citeseer-citation	33.829 ± 5.016	22.058 ± 3.696
	Cora-authorship	63.555 ± 8.144	47.381 ± 7.941
CLARE	Citeseer-citation	58.813 ± 2.553	44.685 ± 5.403
	Cora-authorship	66.517 ± 5.491	49.203 ± 5.540
HGNN	Citeseer-citation	60.614 ± 0.798	53.040 ± 1.220
	Cora-authorship	71.547 ± 0.875	62.917 ± 1.418
HCHA	Citeseer-citation	61.228 ± 1.763	49.756 ± 1.632
	Cora-authorship	71.525 ± 1.279	51.154 ± 2.913
HyperTTC	Citeseer-citation	**68.441 ± 2.287**	**50.261 ± 2.655**
	Cora-authorship	**75.731 ± 1.712**	**63.678 ± 2.497**

V. Conclusion

The paper introduces a novel tactic-specific traffic clustering method based on hypergraph to detect atomized APTs. Specifically, in the proposed method, each entry in the raw traffic data is firstly considered as one node in the hypergraph structure, with interactions between entries represented by hyperedges. Then a two-layer GCN is applied on hypergraph to model higher-order relationships between entities and hyperedges. Finally, HyperTTC employs a downstream clusterer to learn benign and malicious behaviors. Extensive experiments based on various datasets and settings demonstrate the effectiveness of our hypergraph-empowered clustering techniques in discovering atomized APT patterns within complex network traffic data. As Tables I and II illustrate, HyperTTC gets higher F1-score and Jaccard-score compared with baselines. The results show that our approach outperforms traditional provenance graph-based methods by using hypergraphs to capture complex relationships in cybersecurity data, leading to significantly better clustering accuracy.

VI. Acknowledgments

This research work is funded by National Nature Science Foundation of China under Grant No. 62202303, 62471301 and National Key Laboratory of Security Communication Foundation, Grant No. 6142103042310.

References

[1] Y. Hu, J. Wu, G. Li, J. Li, and J. Cheng, "Privacy-preserving few-shot traffic detection against advanced persistent threats via federated meta learning," *IEEE Transactions on Network Science and Engineering*, 2023.

[2] A. Alsaheel, Y. Nan, S. Ma, L. Yu, G. Walkup, Z. B. Celik, X. Zhang, and D. Xu, "Atlas: A sequence-based learning approach for attack investigation," in *USENIX security symposium*, 2021.

[3] X. Han, T. Pasquier, A. Bates, J. Mickens, and M. Seltzer, "Unicorn: Runtime provenance-based detector for advanced persistent threats," *arXiv preprint*, 2020.

[4] T. Van Ede, H. Aghakhani, N. Spahn, R. Bortolameotti, M. Cova, A. Continella, M. van Steen, A. Peter, C. Kruegel, and G. Vigna, "Deepcase: Semi-supervised contextual analysis of security events," in *IEEE Symposium on Security and Privacy*, 2022.

[5] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou, "Hypergraph learning: Methods and practices," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 44, no. 5, pp. 2548–2566, 2020.

[6] I. Duta, G. Cassarà, F. Silvestri, and P. Liò, "Sheaf hypergraph networks," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[7] Y. Li, S. Bai, Y. Zhou, C. Xie, Z. Zhang, and A. Yuille, "Learning transferable adversarial examples via ghost networks," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 458–11 465.

[8] W. Zhang, Z. Yin, Z. Sheng, Y. Li, W. Ouyang, X. Li, Y. Tao, Z. Yang, and B. Cui, "Graph attention multi-layer perceptron," in *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2022, pp. 4560–4570.

[9] D. Jin, Z. Yu, C. Huo, R. Wang, X. Wang, D. He, and J. Han, "Universal graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 10 654–10 664, 2021.

[10] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems," in *Military communications and information systems conference*, 2015.

[11] N. Yadati, M. Nimishakavi, P. Yadav, V. Nitin, A. Louis, and P. Talukdar, "Hypergcn: A new method for training graph convolutional networks on hypergraphs," *Advances in neural information processing systems*, vol. 32, 2019.

[12] T. Chakraborty, A. Dalmia, A. Mukherjee, and N. Ganguly, "Metrics for community analysis: A survey," *ACM Computing Surveys*, vol. 50, no. 4, pp. 1–37, 2017.

[13] A. Bakshi, S. Parthasarathy, and K. Srinivasan, "Semi-supervised community detection using structure and size," in *IEEE International Conference on Data Mining*, 2018.

[14] Y. Yang, Z. Feng, M. Song, and X. Wang, "Factorizable graph convolutional networks," *Advances in Neural Information Processing Systems*, vol. 33, pp. 20 286–20 296, 2020.

[15] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "Kgat: Knowledge graph attention network for recommendation," in *ACM SIGKDD conference on knowledge discovery and data mining*, 2019.

[16] X. Wu, Y. Xiong, Y. Zhang, Y. Jiao, C. Shan, Y. Sun, Y. Zhu, and P. S. Yu, "Clare: A semi-supervised community detection algorithm," in *ACM SIGKDD conference on knowledge discovery and data mining*, 2022.

[17] S. Bai, F. Zhang, and P. H. Torr, "Hypergraph convolution and hypergraph attention," *Pattern Recognition*, vol. 110, p. 107637, 2021.

[18] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao, "Hypergraph neural networks," in *AAAI conference on artificial intelligence*, 2019.