Efficient Graph Partitioning Approaches for Enhanced Accident Prediction

T Divya Teja Reddy[1], S Kartikaaditya[2], S Reshmi Panda[3], and Geetha M[4]

Department of Computer Science and Engineering, Amrita School of Computing, Amrita Vishwa Vidyapeetham, Amritapuri, India.
divyatejareddytadi@gmail.com[1] kartikaditya6@gmail.com[2]
sahukarreshmipanda@gmail.com[3]geetham@am.amrita.edu[4]

Abstract

The increasing amount of data generated by increasing road networks poses a significant challenge in the area of accident prediction. This investigation delves into the use of graph partitioning techniques to address the complexities associated with handling large graph datasets in the context of accident prediction. Graph partitioning techniques are critical in breaking down complex graph structures into manageable components, promoting parallelism, and enabling scalable computations. The computational burden is distributed by strategically dividing the road network into sub-networks, resulting in faster analysis. This study investigates various graph partitioning algorithms and evaluates their effectiveness in maintaining the overall integrity of the road network during the partitioning process. Furthermore, using robust evaluation metrics, this study comprehensively compares various graph partition methods, providing valuable information to choose the most effective strategy for a specific traffic network, thereby advancing robust and optimized solutions for accident prediction.

Keywords: Graph Partitioning · Data Parallelism · Accident Prediction · Road network partitioning.

1 Introduction

The growth of urbanization and the rising complexity of road networks have posed substantial obstacles in the study and administration of traffic data in recent years. The increase in traffic, in addition to the complexity of modern urban road networks, has made it increasingly difficult to foresee and predict possible dangers, particularly the occurrence of Accidents. Analyzing the dynamics of these networks requires methods that are capable of capturing the complex

relationships that exist between road segments and connections.

Managing massive graph datasets efficiently has become a crucial challenge as data volumes continue to increase significantly. Handling such large traffic datasets increases the computational complexity of the model, it grows exponentially as the size of the dataset grows. Storing and processing large graphs may exceed the memory capacity of standard computing devices. This study addresses the challenges posed by urbanization and the increasing complexity of road networks in the context of traffic data analysis.

For evaluating and modeling road networks, graph-based approaches have emerged as a significant tool in recent years. Using graph partitioning techniques, it is possible to handle large graphs while incurring low computational expenses. These techniques facilitate parallel computation and reduce the computational burden by breaking down the network into smaller, more manageable subgraphs. This study contributes to the field by comparing different graph partitioning techniques with a variety of quantitative evaluation metrics, such as node loss, edge loss, and balance, to provide a comprehensive analysis of their performance. In the context of accident prediction, preserving both nodes and edges is critical for accurate modeling. This analysis focuses on evaluating techniques based on their ability to preserve the integrity of nodes and edges in the graph structure. This study provides an in-depth examination of various techniquesto assist in the selection of optimal partitioning algorithms for better Accident predictions.

2 Related work

A recent study emphasized the importance of graph partitioning in optimizing the training of Graph Neural Networks (GNNs). According to the findings in [1], investing time in graph partitioning can significantly reduce GNN training time. This demonstrates how important graph partitioning techniques are in improving the efficiency of GNN training and overall network performance. Prior research revealed that the choice of a partitioning technique heavily impacts theperformance in a real distributed environment[2]. Our research focuses on applying established techniques like METIS, Fennel Partitioning, and Asynchronous Fluid Communities Detection to solve the graph partitioning problem. METIS is a software package for partitioning large irregular graphs[3]. Two versions of METIS (deep graph library implementation and PyTorch library implementation) are taken into account for This study. Fennel is a partitioning strategy whose heuristic combines locality-centric measures with balancing goals [4]. Asynchronous Fluid Communities Detection, on the other hand, is a graph clustering technique for detecting communities in a graph [5].

Taking insights from past research on comparing various graph partition techniques, we present an analytical comparison of these techniques with quantitative evaluation metrics like node loss, edge loss, and, balance.

3 Methodology

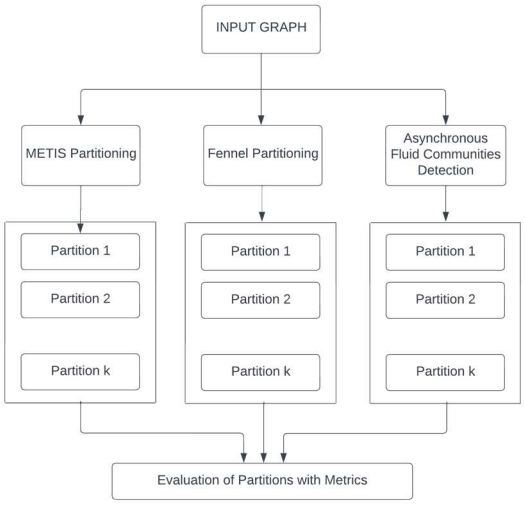

Figure 1: Work Flow of Analysis

Graph partitioning works to divide the set of vertices V into disjoint partitions such that the computational load is balanced. our methodology (fig1) is to apply various graph partitioning and clustering techniques with varying number of partitions to assess the performance of different techniques across the dataset. Distinct Graph partitioning techniques and clustering techniques(METIS, Fennel Partitioning, Asynchronous Fluid Communities Detection) are applied to the road network dataset.

An extensive evaluation is performed on the results obtained of distinct partitioning and clustering techniques to decide which technique is effective in reducing the computational and memory complexities. Later on, the partitioned graph data can be used for Graph Neural Network training allowing for improved accuracy due to reduction of computational complexity.

3.1 Dataset

For this study, the road network dataset of Houston City is considered to evaluate various graph partition techniques. dataset was sourced from [6]. Graph Dataset with a maximum number of nodes and edges is considered to ensure extensive evaluation of the graph partitioning algorithms. The dataset comprises 59,711 nodes and 1,48,937 edges.

3.2 Graph partition techniques

METIS METIS is a multilevel k-way partitioning technique, this algorithm reduces the size ofthe graph by collapsing the vertices and edges in the coarsening stage and performing a k-way partition on the reduced graph then uncoarsen it to construct partitions for the initial graph G. During refinement phase, the focus is on optimizing the region of the graph near partition boundary[3].METIS carves up complex graphs in a multilevel approach. It simplifies the graph first, then partitions it (either splitting in half repeatedly or directly into k groups). Finally, it refines this partition on the original graph, minimizing connections between partitions and balancing their sizes. Deep Graph Library(DGL) library implementation and PyTorch implementation of METIS are used in this study.

Fennel Partitioning Fennel is a unifying framework for graph partitioning that enables a well-principled design of scalable, streaming graph partitioning algorithms that are amenable to distributed implementation[4]. It was developed to handle situations involving parallel processing and distributed computing. It strives to distribute the workload fairly across partitions whilst considering a graph's vertex positions. Fennel tackles the challenge of partitioning huge graphs in a streaming fashion, allocating vertices as they arrive to ensure balanced communication and partition sizes.

Asynchronous Fluid Communities Detection Nodes in a network interact constantly and sequentially in Asynchronous Fluid Community Detection, simulating the movement of a hypothetical" fluid." Instead of assigning nodes to communities in a single step, the algorithm iteratively updates the community assignments of nodes based on local information and the dynamics of the network. This process continues until a stable state is reached, where nodes no longer switch communities, effectively revealing the underlying community structure of the network.

3.3 Evaluation Metrics

Node Loss Node loss refers to loss in the nodes after the partitioning of the initial graph. Alower node value is desirable as it gives a more effective partition.

$$\text{Node Loss} = \frac{\text{Total Nodes in Original Graph} - \text{Total Nodes in Subgraphs}}{\text{Total Nodes in Original Graph}} \tag{1}$$

Edge Loss Edge loss refers to the reduction of edges after the partitioning of the initial graph.A lower edge loss value is desirable as it results in a more effective partition.

$$\text{Edge Loss} = \frac{\text{Total Edges in Original Graph} - \text{Total Edges in Subgraphs}}{\text{Total Edges in Original Graph}} \tag{2}$$

Balance Balance is used to measure the ratio between the maximum number of edges in one partition and the average number of edges in all subsets[7]. signifies the distribution of nodes or edges among the various partitions of the graph network. For optimal performance, partitions should be well-balanced. An ideal value for balance is often considered as 1, indicating perfect and effective partitions.

4 Experimental Analysis

This section presents our findings from a thorough analysis of graph partitioning methods on three datasets mentioned in the section 3.1. our analysis includes visual representations of the partitions along with quantitative evaluation metrics. our objective is to evaluate various graph partitioning and clustering techniques with varying number of partitions to assess the performance in preserving the integrity of the road network and effectiveness in achieving balanced partitions.

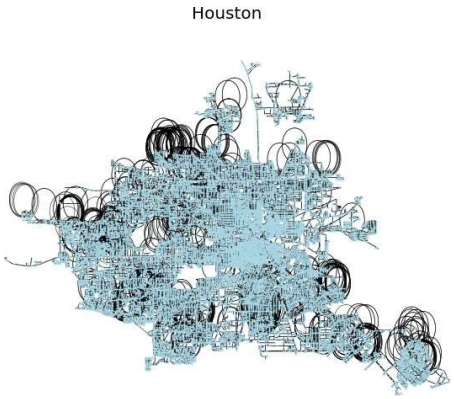

Figure 2: Unpartitioned Road Network of Houston City

Figure 2 depicts an unpartitioned view of Houston city's road network (light blue color represents nodes and black color represents edges), with nodes and edges overlapping, reflecting

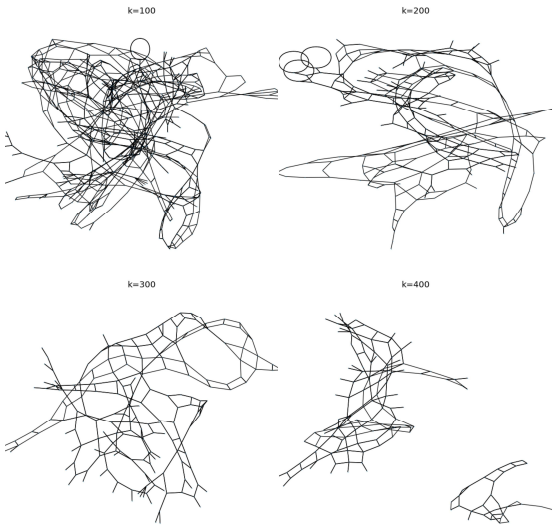

Figure 3: METIS (Pytorch) Subgraphs with varying number of partitions(k) 100 - 400

the complex connections within. Training the GNN model on this dataset poses significant challenges due to inherent Complexity.

Figure 3 reveals that at k=100, the subgraph appears denser, suggesting well-preserved connectivity with minimal edge loss. Despite a reduction in density as k increases, the subgraphs, including the k=400 partition, still exhibit a substantial number of edges, indicating effective preservation of connectivity even at higher partition sizes.

	Number of Partitions(k)			
Metrics	**100**	**200**	**300**	**400**
Node loss	0.0	0.0	0.0	0.0
Edge loss	0.01723	0.02828	0.03534	0.04253
Node Balance	1.02996	1.02828	1.02996	1.02493
Edge Balance	1.32267	1.32528	1.39065	1.38006
avg nodes	597.11	298.555	199.036	149.2775
avg Edges	1463.7	723.62	478.91	356.505

Table 1. Performance Metrics for METIS(Pytorch) Across Different Partition Sizes

From table 1, Node loss is consistently zero across all partition sizes this implies that the METIS algorithm preserves all the nodes while performing partitioning. Edge balance values are higher in larger partition sizes, indicating less balanced partitions when number of partitions is large. However, the values of both node and edge balance values are close to 1 indicating the METIS produced well-balanced partitions. Average nodes and average edges are expected to decrease as the number of partitions increases subgraphs tend to have fewer nodes and edges. Average nodes and edges are decreasing as expected. For the task of accident prediction, the preservation of data integrity emerges as a pivotal factor, minimal edge loss is evident, highlighting the remarkable strength of METIS in edge preservation, As evidenced by the consistentlylow

edge loss across different partition sizes.

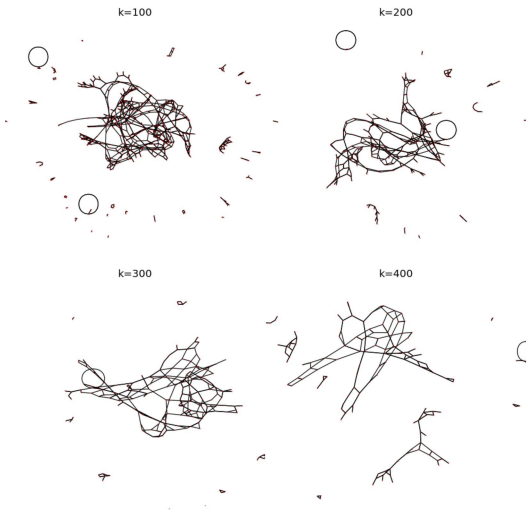

Figure 4: METIS (DGL) Subgraphs with varying number of partitions(k) 100 - 400

In the analysis of Figure 4, it is noted that the subgraphs exhibit a significantly lower density compared to those in Figure 3. One of the reasons for the sparsity in the subgraphs could be the increased number of isolated nodes without connecting edges, especially when the number of partitions is 100. Furthermore, as k increases, in comparison to PyTorch METIS, the graphs exhibit increasing sparsity, reflecting significant node and edge loss.

	Number of Partitions(k)			
Metrics	**100**	**200**	**300**	**400**
Node loss	0.0	0.0	0.0	0.0
Edge loss	0.15720	0.16902	0.18940	0.19740
Node Balance	1.17063	1.18236	1.30629	1.29289
Edge Balance	1.08665	1.11503	1.15052	1.17788
avg nodes	597.11	298.55	199.036	149.27
avg Edges	1255.23	618.815	402.42	298.84

Table 2. Performance Metrics for METIS(DGL) Across Different Partition Sizes

The table 2 summarizes evaluation metrics of METIS(DGL), offering insights into the performance of the technique. As the number of partitions increases edges loss exhibits an upward trend. The algorithm performed effectively in preserving the nodes contributing to the overall integrity of the road network. Our primary objective is to achieve balanced partitions. Later in this section, focused analysis on Node Balance and Edge Balance is carried to out evaluate the effectiveness of this technique.

From the figure 5, reveal that with the increase in partition, the connectivity decreases resulting in lesser edges relative to vertices. While the graphs with k=100 and k=200 showdecent

connectivity between points the graphs with partitions 300 and 400 (k=300 and k=400) show a significant decrease in edges which leads to the isolation of nodes. Moreover, uneven distribution of edges across partitions is observed which implies that some portions of datashow higher connectivity than others which indicates varying degrees of relationships among nodes within the dataset.

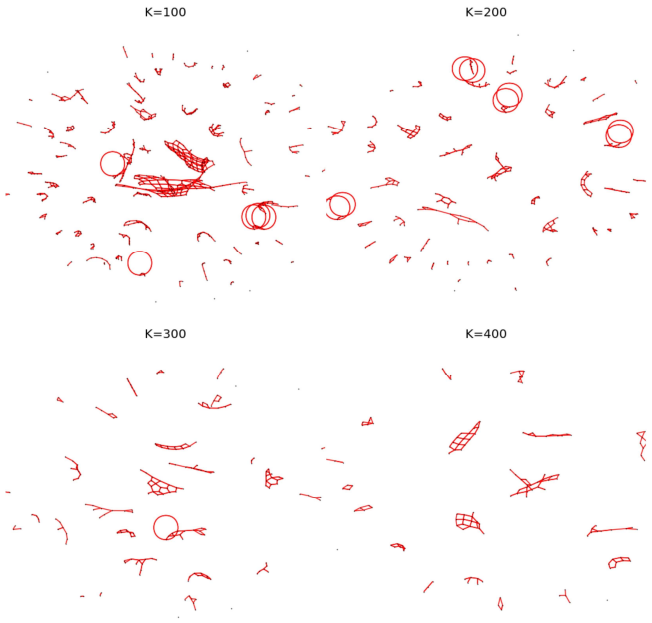

Figure 5: Fennel partitioning Subgraphs with varying number of partitions(k) 100 - 400

	Number of Partitions(k)			
Metrics	**100**	**200**	**300**	**400**
Node loss	0.0	0.0	0.0	0.0
Edge loss	0.28373	0.28479	0.28653	0.28664
Node Balance	1.01653	1.04838	1.10029	1.10532
Edge Balance	1.10518	1.24294	1.31280	1.46077
avg nodes	597.11	298.555	199.0366	149.277
avg Edges	1066.79	532.605	354.203	265.612

Table 3. Performance Metrics for Fennel partitioning Across Different Partition Sizes

From the table 3, node loss is not observed. The edge loss increases with an increase in the number of partitions affecting connectivity between nodes. It is observed that as the number of partitions increase the balance between nodes and edges decreases implying a growing imbalancein the partition of the graph which would affect load balancing negatively.

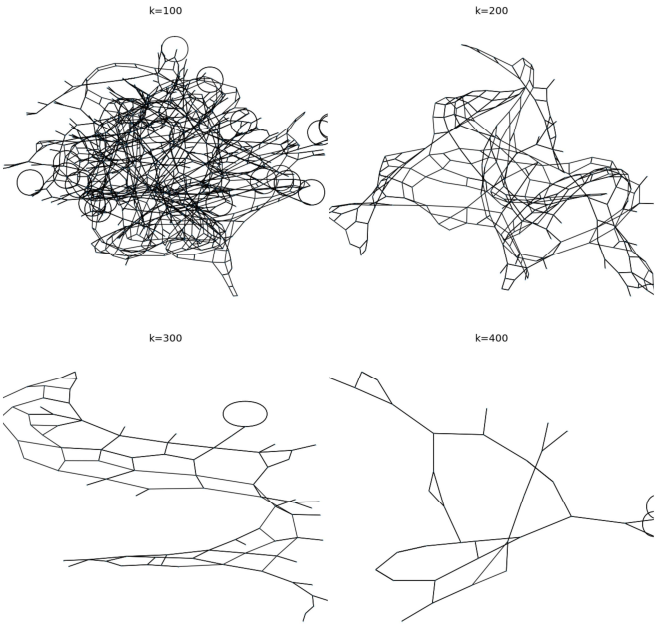

Figure 6: Asynchronous clustering Subgraphs with varying number of partitions(k) 100 - 400

As depicted in Figure 6, initial observations reveal that as the number of partitions increases from 100 to 400, the partitioned graph exhibits a denser structure, indicative of a more detailed breakdown of the network into smaller communities. This density gradually diminishes with the rise in partition sizes, signifying a sparser graph. This shift in density can be interpreted as a potential sign of edge loss.

	Number of Partitions(k)			
Metrics	**100**	**200**	**300**	**400**
Node loss	0.0	0.0	0.0	0.0
Edge loss	0.035518	0.04693	0.06267	0.06430
Node Balance	2.75158	2.67287	3.84853	3.53033
Edge Balance	2.56462	2.55871	3.49571	3.74285
avg nodes	597.11	298.555	199.0366	149.277
avg Edges	1436.47	709.73	468.86	348.397

Table 4. Performance Metrics for Asynchronous Fluid Community Across Different Partition Sizes

Especially at K=100, there is a potential core-periphery arrangement indicating there is a central group of nodes that are tightly connected which are then surrounded by less connected nodes, Implying an organized structure where few nodes have relatively stronger connections.

After a thorough analysis of the table 4, it is understood that Asynchronous fluid community detection displays efficient conservation of nodes and very low edge loss varying from 0.032 to 0.064 signaling good connectivity. In terms of node and edge balance with the value running from

2.5 to 3.8 which indicates an imbalanced distribution of nodes and edges across partitions, This negatively impacts load balancing.

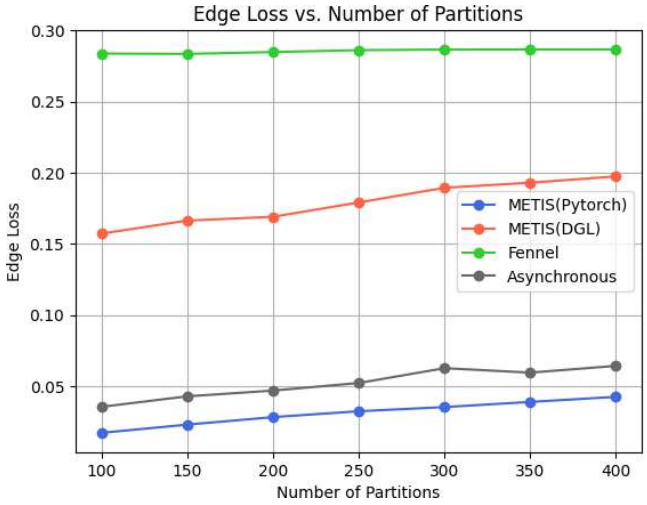

Figure 7: Plot of Edge Loss vs Number of Partitions across Partitioning Techniques

Our main aim revolves around partitioning the graphs while preserving nodes and edges. From figure 7, notably, edge loss in the fennel partitioning technique is relatively higher than in other techniques, making it unsuitable for instances where Preserving edges is critical. In the context of an Accident Prediction task where relations between nodes are crucial, minimal edge loss is desirable. METIS(DGL) also has a moderately high edge loss value this is evident inFigure 4 where the count of isolated nodes is high. Asynchronous and METIS(PyTorch) versionsexhibited low edge loss values. As the value of k increases there is a detectable upward trend inedge loss. It is suggested to stop increasing the k value at an optimal point to avoid the excessiveloss of edges. Node Loss is consistently zero across all the techniques, indicating the efficiency oftechniques in the preservation of nodes contributing to the overall integrity of the road network. From Fig 8, Node Balance serves as a pivotal metric in evaluating the quality of subgraphs.

Our objective is to achieve balanced partitions, ensuring an equal distribution of nodes across the partitioned graphs. The Asynchronous Fluid clustering technique demonstrate a high node balance value almost equal to 3.5 at k=350, signifying unequal distribution of nodes among partitions. Fennel Partitioning and METIS(PyTorch) exhibit node balance equal to 1, signifying well-balanced partitions. Although fennel partitioning succeeded in producing balanced partitioning it is not effective in preserving edges.

Upon analyzing Fig 9, Asynchronous fluid community detection technique demonstrates high edge balance values, suggesting imbalanced partitions with balance values exhibiting a fluctuating trend. METIS(DGL) outperformed other techniques with edge balance values close to 1across the partition sizes. Fennel Partitioning initiates with a near-optimal value but experiences an increase with higher partition sizes. METIS(PyTorch) exhibits comparable edge balance values.

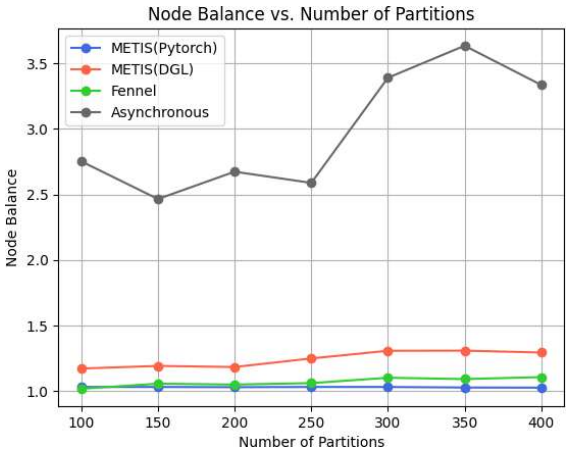

Figure 8: Plot of Node Balance Vs Number of Partitions across Partitioning Techniques

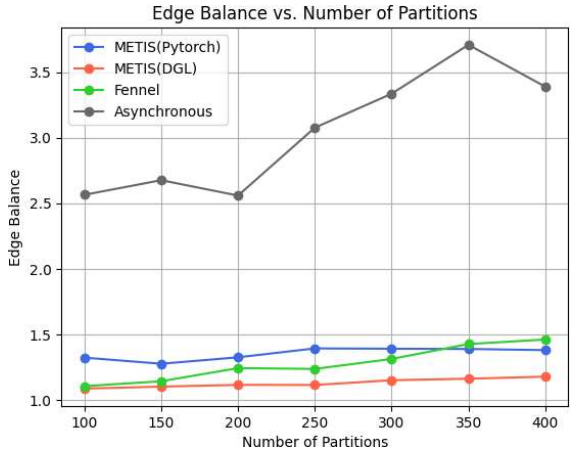

Figure 9: Plot of Edge Balance vs Number of Partitions across Partitioning Techniques

5 Conclusion

In Conclusion, analysis focused on analyzing different graph partitioning techniques including METIS (PyTorch), Fennel partitioning, METIS(DGL), and Asynchronous Fluid Community detection are applied to the Houston city road network dataset, has provided insights for optimizing accident prediction models. This analysis provided a comprehensive understanding of the performance of the various techniques by utilizing visualizations of subgraphs and evaluation metrics.

Upon analyzing the partitioning techniques, our results reveal that METIS(DGL) excels in maintaining low balance values suggesting the technique's ability to effectively preserve edges, and distribute edges equally across partitions. However, METIS(DGL) exhibited high Edge Loss comparatively. Asynchronous Fluid community detection demonstrated lower edge loss

values but resulted in imbalanced partitions. The observed imbalanced partitions could be its clusteringnature. Asynchronous fluid community detection focuses on grouping similar nodes, this leads to uneven partitions. On the other hand, Fennel partitioning provides well-balanced partitions but it does come with a trade-off, it exhibits higher edge loss. METIS(Pytorch) outperformed other techniques by preserving edges while achieving balanced partitions, making it suitable for accident prediction tasks.

The choice of Partitioning Technique should align with specific objectives, such as Balanced partitioning, node preserving, and edge-preserving, which are critical for achieving accurate results. This analysis summarizes the insights that contribute to a more informed decision-making process in choosing a partition technique.

References

1. Merkel, Nikolai & Stoll, Daniel & Mayer, Ruben & Jacobsen, Hans-arno. (2023). An Experimental Comparison of Partitioning Strategies for Distributed Graph Neural Network Training.
2. Antelmi, Alessia & Cordasco, Gennaro & Spagnuolo, Carmine & Vicidomini, Luca. (2015). On Evaluating Graph Partitioning Algorithms for Distributed Agent Based Models on Networks. 9523. 367-378. 10.1007/978-3-319-27308-2 30.
3. Karypis, George & Kumar, Vipin. (1997). METIS—A Software Package for Partitioning Unstructured Graphs, Partitioning Meshes and Computing Fill-Reducing Ordering of Sparse Matrices.
4. Abbas, Zainab & Kalavri, Vasiliki & Carbone, Paris & Vlassov, Vladimir. (2018). Streaming graph partitioning: an experimental study. Proceedings of the VLDB Endowment. 11. 1590-1603. 10.14778/3236187.3236208.
5. Parés, Ferran & Garcia-Gasulla, Dario & Vilalta, Armand & Moreno, Jonatan & Ayguadé, Eduard & Labarta, Jesús & Cortés, Ulises & Suzumura, Toyotaro. (2018). Fluid Communities: A Competitive, Scalable and Diverse Community Detection Algorithm. 229-240. 10.1007/978-3-319-72150-7 19.
6. Huang, Baixiang & Hooi, Bryan & Shu, Kai. (2023). TAP: A Comprehensive Data Repository for Traffic Accident Prediction in Road Networks. 1-4. 10.1145/3589132.3625655.
7. Sani, Mustapha & Ahmad, Abdulmalik & Mohammed, Ayaz & Ahmad, Abdulkadir & Haruna, Yusuf& Ayaz, Khalid & Mohammed, Abdulkadir & Ahmad, Yusuf & Overview,. (2022). An Overview of Metrics for Evaluating the Quality of Graph Partitioners. Mathematics and Computer Science. 7.1-8. 10.11648/j.mcs.20220701.11.
8. A. B. Nair, G. Surendran, K. P. Prathyun, V. Sivaprasad and C. P. Prathibhamol, ”Comparative study of Centrality based Adversarial Attacks on Graph Convolutional Network model for Node classification,” 2022 7th International Conference on Communication and Electronics Systems (IC- CES), Coimbatore, India, 2022, pp. 731-736, doi: 10.1109/ICCES54183.2022.9835948.
9. M. Lolla, A. Barik and B. K R, ”GNN Based Trainable Dependency Parser,” 2023 International Conference on Recent Advances in Information Technology for Sustainable Development (ICRAIS),Manipal, India, 2023, pp. 194-200, doi: 10.1109/ICRAIS59684.2023.10367102.
10. S. M. Devi Thumati, K. Dhanya, H. Sathish, K. C. Sekhar Madan and S. Rani, ”A Comparative Study on the Working of GNN and CNN on Panoramic X-Rays in Prediction of Dental Diseases,”2023 8th International Conference on Communication and Electronics Systems (ICCES), Coimbat- ore, India, 2023, pp. 755-762, doi: 10.1109/IC-CES57224.2023.10192836.
11. Gutta Akshitha sai, Komma Naga Sai Likhitha, Maddi Pavan Kalyan, Perisetla Anjani Devi, and Prathibhamol C. 2022. Combinational Features with centrality measurements on GCN+LR classification of Adversarial Attacks in homogenous Graphs. In

Proceedings of the 2022 Fourteenth International Conference on Contemporary Computing (IC3- 2022). Association for Computing Machinery, New York, NY, USA, 573–581. https://doi.org/10.1145/3549206.3549305
12. Anjan Chowdhury, Sriram Srinivasan, Animesh Mukherjee, Sanjukta Bhowmick, and Kuntal Ghosh. 2023. Improving Node Classification Accuracy of GNN through Input and Output Intervention. ACM Trans. Knowl. Discov. Data 18, 1, Article 17 (January 2024), 31 pages. https://doi.org/10.1145/3610535
13. Vijai, Anupa and Padmavathi, S and Venkataraman, D, A Review of Object Detection in Aerial Images by Deep Learning Approaches. Available at SSRN: https://ssrn.com/abstract=4705432 or http://dx.doi.org/10.2139/ssrn.4705432