Enumeration of Subgraph of Interest based on pruning

Maxence Morin[1,2][0009−0006−3789−9904], Baptiste Hemery[2][0000−0003−3923−1829], Fabrice Jeanne[2], and Estelle Pawlowski[1][0009−0008−9872−3667]

[1] Normandie Univ, UNICAEN, ENSICAEN, CNRS, GREYC, 14000 Caen, France
maxence.morin@unicaen.fr https://morin.earth
estelle.pawlowski@ensicaen.fr
[2] Orange Research, 14000 Caen, France
{baptiste.hemery,fabrice.jeanne}@orange.com

Abstract. Graphs are mathematical structures used to model relationships and interactions in diverse fields, including social networks, biology, and industrial applications. In sectors like banking and telecommunications, the exponential growth of data has highlighted the need for efficient graph analysis methods. This paper focuses on the enumeration of SubGraphs of Interest (SGI) within large multigraphs to enhance fraud detection in an industrial context. We present a novel approach that leverages pruning functions to remove less relevant nodes and edges, thereby constructing informative connected components. Using synthetic datasets, we evaluated the performance of the proposed approach and assessed the impact of node and edge classifications on SGI enumeration. We also explore the use of machine learning classifiers, including XGBoost and Random Forest Classifier, to enhance classification performances.

Keywords: Graph pruning, Social Network Analysis, SubGraph of Interest, Machine Learning on Graphs

1 Introduction

Graphs are fundamental mathematical structures that model relationships between objects, and their use extends to many fields, such as social networks, biology, computer science, and data analysis. For example, in social networks, graphs allow for the representation of connections between users, thereby facilitating the analysis of social interactions and group dynamics. In biology, they can model interactions between proteins [6] or relationships between species, providing insight into the structure and functioning of biological systems.

Data-driven decision-making has led companies, especially in banking and telecommunications, to utilize the large volumes of data produced by their services. For instance, the financial sector has seen a significant transformation with the introduction of mobile banking applications, that allow users to manage their finances conveniently from their smartphones [2, 3]. These applications not only

facilitate transactions but also generate extensive data on user behavior, spending patterns, and financial habits [1]. Telecommunication companies have also recognized the potential of this data, utilizing insights from call records and mobile usage to enhance frauds detection [4, 8, 14].

This paper focuses on fraud detection in large and complex transactional graphs. During a fraud detection campaign, our company discovered a small set of frauds that follow a specific bypass fraud pattern [10]. Considering a fraud type, these occurrences are similar but not strictly identical. In the state-of-the-art, there are two approaches to solve fraud detection in complex graphs, namely community detection [4] and subgraph enumeration [62]. Some works also explore the approximate enumeration of subgraphs [13, 16].

We decide to use neither community detection nor subgraphs enumeration due to the complexity of these approaches and the vast size of the graph.

Instead, to cope with the unavoidable variability of identified frauds examples, we propose in this paper an approach using embeddings, graph elements classification and pruning function. By identifying and retaining only the most informative nodes and edges, we aim to create informative connected components that could be considered as fraud candidates.

The remainder of this paper is organized as follows. Section 2 establishes the notations, foundational concepts and presents the datasets we use in this paper. Section 3 outlines our contributions, detailing the various methodologies employed along with their parameters. Finally, Section 4 concludes the paper and presents some further works.

2 Problem Statement

2.1 Notations

Let $G = (V_G, E_G)$ be a multigraph composed of a set of nodes V_G and a set of edges E_G. Each vertex $v \in V_G$ represents a user. Each edge $\{u, v\} \in E_G$ denotes a relationship such as a financial transaction, a communication, or a friendship. Let $\mathcal{S} = (V_\mathcal{S}, E_\mathcal{S})$ be a subgraph, which implies $\mathcal{S} \subseteq G$, $V_\mathcal{S} \subseteq V_G$ and $E_\mathcal{S} \subseteq E_G$.

2.2 Problem Formulation

In this article, we build on the methodology outlined in [12] and the concept of *SubGraph of Interest* (SGI). SGI, denoted $\mathcal{S}$, are adapted to our industrial context where we deal with fraud patterns represented by subgraphs to characterize groups of users and their associated transactions. Moreover, SGI are particular type of subgraphs in that they share structural and semantic similarities. A single user may belong to multiple SGI concurrently. This is a factor that significantly complicates the task of analyzing behaviors across the graph.

During the previously mentioned fraud detection campaign, the company might have identified a homogeneous sample of n groups of fraudsters, designated as $\mathbb{S}_n$. More precisely, the set $\mathbb{S}_n$ is composed of n samples of a unique fraud

pattern. The objective is to enumerate as many SGI as possible within the graph, whose behavioral patterns align closely with those observed in the sample set $\mathbb{S}_n$.

The industrial nature of our context makes the process of SGI enumeration tolerant compared to the isomorphism property of classical subgraph enumeration: we tolerate a few extra or missing nodes or edges, with respect to the examples set $\mathbb{S}_n$. At the end of the process, the obtained SGI will be reviewed by humans. The set of SGI determined to be sufficiently similar to members of $\mathbb{S}_n$ is designated as $\hat{\mathbb{S}}$. Leveraging these identified SGI enables fraud experts to significantly improve the efficiency of fraud detection, thereby reducing detection time.

To formalize this process, we define the ground truth, denoted $\mathbb{S}$, as the complete set of SGI present within the graph. The goal of the company is to recover the best approximation of the ground truth. The set $\mathbb{S}_n$ of samples $\mathcal{S}$ can be viewed as a proper subset of this ground truth: $\mathbb{S}_n \subsetneq \mathbb{S}$. However, obtaining the complete ground truth is inherently impractical. Achieving such comprehensiveness would necessitate querying all users of the service to map their relationships. This is particularly pronounced in the context of fraud detection, where fraudsters are unlikely to provide accurate or honest information.

In this article, we propose two methods that fit within the framework (1) presented in the article [12] to enumerate a set $\hat{\mathbb{S}}$ of SGI as similar as possible to the ground truth $\mathbb{S}$:

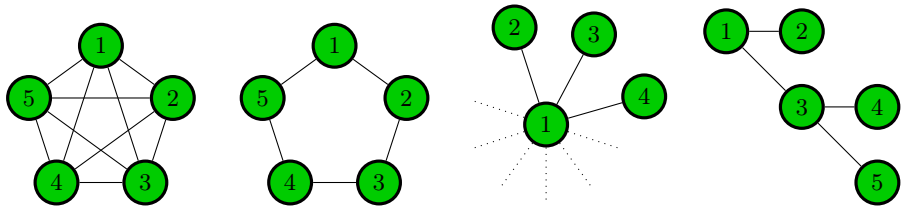
$$(G, \mathbb{S}_n) \rightarrow \boxed{method} \rightarrow \hat{\mathbb{S}} \text{ such that } \hat{\mathbb{S}} \simeq \mathbb{S} \tag{1}$$

2.3 Datasets

In an industrial context in the European Union, the confidential nature of the data and the requirements of the GDPR make access to real datasets particularly complex. Our industrial research problem, presented in Section 2.2, does not correspond to the datasets available in the state-of-the-art [11, 15], due to their lack of diversity and relevance. For example, a dataset dedicated to community detection is often homogeneous in terms of class of communities. Communities represent only groups of friends or only groups of researchers, with no mix of community classes. To our knowledge, there are no datasets that simultaneously integrate tagged groups of friends, family, and fraudsters into a single graph. This limitation makes it difficult to evaluate our algorithms in realistic and varied contexts, which is essential to find a solution that is adapted to the specific needs of the company. In addition, the definition of a community proposed in the state of the art [5] does not perfectly match that of an SGI, which is not necessarily densely connected.

To address these limitations, we developed a synthetic data generator designed to replicate user behavior within a transactional service, such as mobile payment systems. Using this tool, we created datasets composed of subgraphs representing four distinct classes with distinct typical topologies described in

graph theory [7]: Complete graphs (K_n), Circular graphs, Star structures, and Tree structures, as illustrated in Figure 1. These were selected to model the SGI most likely encountered in industrial scenarios of fraud detection.

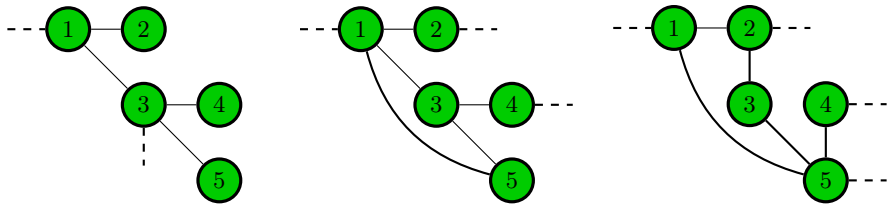


(a) Complete graph (b) Circle (c) Star (d) Tree

Fig. 1: Generic subgraphs

This tool can generate datasets by adjusting two parameters, namely *overlapping* and *noise*. The *overlapping* parameter handles the overlapping property of the subgraphs, i.e. that some nodes can be shared between different subgraphs.

The *noise* parameter introduces complexity by randomly creating edges between subgraphs and randomly adding or removing nodes or edges within subgraphs. This *noise* allows imperfections within the classes to be more realistic. As the noise increases, we add more edges between SGI and randomly modify the structure of the SGI by adding or removing internal edges or nodes to simulate more human-like behavior. We illustrate the impact of noise level on an example related to a Tree in Figure 2. Here, the Tree varies in size and depth. The dashed lines represent connections to other nodes in the graph.



(a) 5% noise (b) 10% noise (c) 20% noise

Fig. 2: Example of Trees obtained after adding and removing edges

For our study, we tested three noise levels (approximately 5%, 10%, and 20%):

– The first type of dataset, with 5% noise, does not remove any edge. It is therefore composed of unaltered SGI classes. There is an addition of at least 5% of total edges between SGI to obtain large connected components.

- The second type of dataset corresponds to an addition of at least 10% and removal of 10% of the number of edges.
- The last one corresponds to an addition of at least 20% and removal of 20% of the number of edges.

With two *overlapping* values (*true* or *false*) and three *noise* levels, we obtain six combinations. For each combination, we generated ten datasets, finally obtaining sixty datasets. Datasets are all composed of 10 000 nodes. The average number of edges is presented in Table 1. Each dataset contains 250 instances of each class, culminating in a total of 1,000 subgraphs. We are not able to provide any statistical similarity with real-world data as it is concidered confidential by Orange.

Table 1: Average number of edges in datasets

Datasets		Average number of edges
Non-overlapping	5% noise	203 205.5
	10% noise	246 464.0
	20% noise	263 207.2
Overlapping	5% noise	201 371.2
	10% noise	245 758.8
	20% noise	261 457.5

2.4 Experimental Protocol

The evaluation of SGI enumeration involves comparing the framework's (1) output, $\hat{\mathbb{S}}$, with the ground truth, $\mathbb{S}$. In our industrial context, perfect matches are not realistic. Therefore, we introduce the *match* function to introduce a greater degree of tolerance into the matching process. In our context, this *match* function takes as input $\hat{\mathcal{S}} \in \hat{\mathbb{S}}$ and $\mathcal{S} \in \mathbb{S}$. As in [12], the *match* function relies on three boolean functions: φ_{extra}, $\varphi_{missing}$, and φ_{size}:

1. The φ_{extra} boolean function is defined by:

$$\varphi_{extra}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{|V_{\hat{\mathcal{S}}} \setminus V_{\mathcal{S}}|}{|V_{\mathcal{S}}|} < \Gamma_{extra} \tag{2}$$

   Where the threshold Γ_{extra} is set according to the context of the problem.
2. The $\varphi_{missing}$ boolean function is expressed by:

$$\varphi_{missing}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{|V_{\mathcal{S}} \setminus V_{\hat{\mathcal{S}}}|}{|V_{\mathcal{S}}|} < \Gamma_{missing} \tag{3}$$

   Where the threshold $\Gamma_{missing}$ is set according to the context of the problem.

3. The φ_{size} boolean function is defined by:

$$\varphi_{size}(\hat{\mathcal{S}}, \mathcal{S}) = \frac{abs(|V_{\hat{\mathcal{S}}}| - |V_{\mathcal{S}}|)}{|V_{\mathcal{S}}|} < \Gamma_{size} \tag{4}$$

Where the threshold Γ_{size} is set according to the context of the problem.

We now consider the three boolean functions together to decide if two subgraphs are matching:

$$match(\hat{\mathcal{S}}, \mathcal{S}) = \varphi_{\text{extra}}(\hat{\mathcal{S}}, \mathcal{S}) \text{ AND } \varphi_{\text{missing}}(\hat{\mathcal{S}}, \mathcal{S}) \text{ AND } \varphi_{\text{size}}(\hat{\mathcal{S}}, \mathcal{S}) \tag{5}$$

We define a subgraph $\hat{\mathcal{S}}$ as *relevant* if it matches at least one subgraph from $\mathbb{S}$. This function *relevant* relies on the previous *match* function and the Iverson bracket [9], that generalizes the Kronecker delta:

$$[P] = \begin{cases} 1 \text{ if } P \text{ is true;} \\ 0 \quad \text{otherwise} \end{cases} \tag{6}$$

where P is a statement.

Now we are able to formalize the *relevant* boolean function as follows:

$$relevant(\hat{\mathcal{S}}, \mathbb{S}) = \big(\sum_{\mathcal{S} \in \mathbb{S}} [match(\hat{\mathcal{S}}, \mathcal{S})] \big) \geq 1 \tag{7}$$

We use the adaptation of *precision* and *recall* metrics as in [12] to evaluate the framework:

$$precision = \frac{\sum_{\hat{\mathcal{S}} \in \hat{\mathbb{S}}} [relevant(\hat{\mathcal{S}}, \mathbb{S})]}{|\hat{\mathbb{S}}|} \tag{8}$$

and:

$$recall = \frac{\sum_{\mathcal{S} \in \mathbb{S}} [relevant(\mathcal{S}, \hat{\mathbb{S}})]}{|\mathbb{S}|} \tag{9}$$

The next section presents the contributions of this paper.

3 Contributions

In this paper, we present the results of experiments using the second approach proposed in [12]. This approach consists in the pruning of nodes and/or edges of a graph, using what is called a ∇ operator. Such operators manage to prune the graph using the results of the node and edge classifications. We compare the four mediation operators ∇ described in the previously mentioned paper and select the best.

We first analyze the performance of each mediation operator by manipulating the classification performance. Then, we show and explain the results of the usage of a cosine similarity as a classifier. Next, we propose to use a machine learning model as a classifier to enhance the performances of node and edge classifications, and consequently the subgraphs enumeration F1 score.

3.1 Performances of Mediation Operators

In this section, we compare the performance of the four mediation operators. We want to know how the node and the edge classifications impact the subgraphs enumeration F1 score. We design an experiment with artificially decreased precision and recall of node and edge classifications to assess the lack of performance in different scenarios.

Methodology To assess the performance of each mediation operator, we manipulate the precision and recall of both node and edge classifications. We randomly select 0.25, 0.50, and 0.75 of the total number of nodes or edges to decrease the recall. To decrease node and edge classification precisions, we add nodes or edges that we know are wrong to achieve a score of 0.25, 0.50 and 0.75. We will compare the results of the mediation operators using the subgraphs enumeration F1 score presented in Section 2.4. An equally impacted by node and edge classifications mediation operator should obtain the results presented in Figure 3.

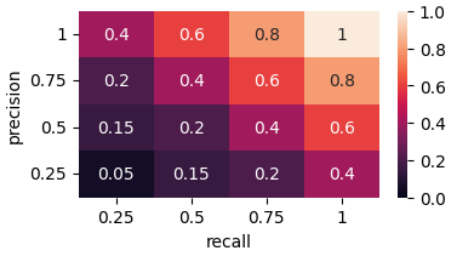
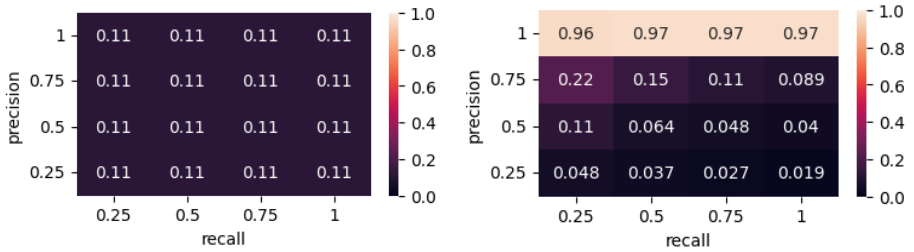


Fig. 3: Expected subgraphs enumeration F1 score for edge and node decreased classifications

However, the reality is more complex. Using the synthetic dataset, we first decrease edge classification performance, then node classification performances, and finally both classification performances.

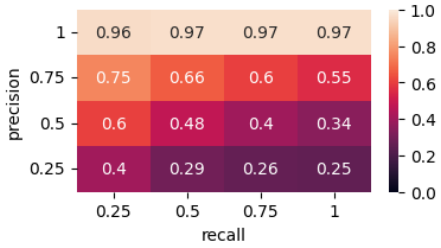
Decreased Edge Classification In this section, we alter the classification of edges to observe its impact on the four mediation operators: ∇_{node}, ∇_{edge}, ∇_{simple} and $\nabla_{majority}$. The average of subgraphs enumeration F1 scores across all datasets are presented in Figure 4.

The ∇_{node} mediation operator exhibits suboptimal performance. Its subgraphs enumeration F1 score remains consistently low, as this operator does not incorporate edge classification into its computations. In contrast, the ∇_{edge} operator focuses exclusively on edge classification, making it highly sensitive to any drop in classification performance. Despite this sensitivity, it achieves strong results when edge classification precision is exceptionally high. However,

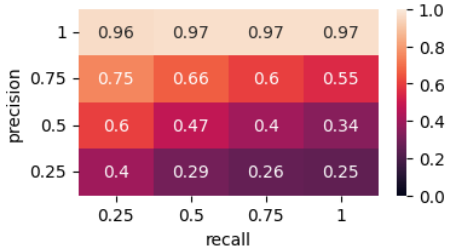


(a) ∇_{node} operator

(b) ∇_{edge} operator



(c) ∇_{simple} operator



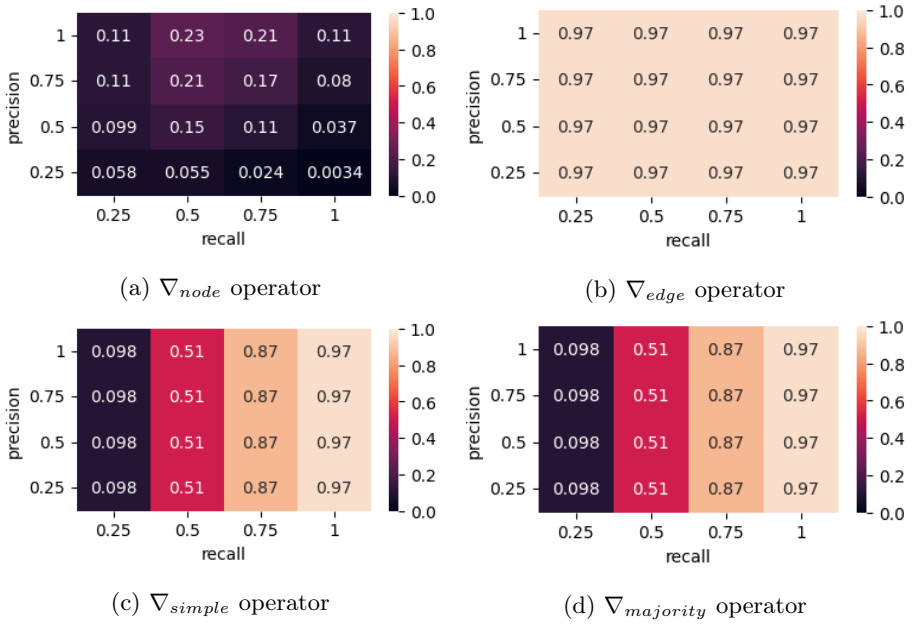
(d) $\nabla_{majority}$ operator

Fig. 4: Average of Subgraphs enumeration F1 scores obtained by decreasing edge classification performance for the four mediation operators

when precision declines, its performance becomes contingent on edge classification recall, improving under conditions of low recall. The ∇_{simple} mediation operator demonstrates robust performance when edge classification precision is high, maintaining satisfactory results even as precision diminishes. In this case, recall plays a less critical role, provided precision remains elevated. Finally, the $\nabla_{majority}$ mediation operator mirrors the performance trends of the ∇_{simple} operator, displaying a similar response to variations in edge classification performance.

The results observed for the ∇_{edge}, ∇_{simple}, and $\nabla_{majority}$ operators reveal an intriguing trend: lower recall often leads to improved outcomes. When recall decreases, fewer edges are retained, and more are removed. This reduction facilitates the separation of subgraphs that share common nodes, which is particularly relevant in scenarios where the SGI overlap. This process effectively acts as a disambiguation mechanism, enabling a clearer identification of the subgraphs that would otherwise remain merged due to their shared edges. While recall decreases in this approach, it enhances the ability to accurately isolate and identify overlapping subgraph structures.

Decreased Node Classification We then alter the classification of nodes to observe and present its impact on the four mediation operators in Figure 5.



(a) ∇_{node} operator

(b) ∇_{edge} operator

(c) ∇_{simple} operator

(d) $\nabla_{majority}$ operator

Fig. 5: Average of Subgraphs enumeration F1 scores obtained by decreasing node classification performance for the four mediation operators

The performance of the ∇_{node} mediation operator diminishes when the performance of node classification decreases. In contrast, the ∇_{edge} mediation operator delivers exceptional and stable results, as it exclusively relies on edge classification. This stability is reflected in its optimal performance, achieving $precision = 1$ and $recall = 1$, as demonstrated in Figure 4b. The ∇_{simple} mediation operator also performs well, particularly due to its strong recall. However, it exhibits insensitivity to a reduction in node classification precision. Finally, the $\nabla_{majority}$ operator mirrors the performance of the ∇_{simple} operator, offering comparable results.

Discussion The comparison of mediation operators revealed that the ∇_{simple} mediation operator consistently outperformed the others, achieving a good compromise between precision and recall. This operator demonstrated its reliability across varying levels of node and edge classification performance, making it a suitable choice for practical applications. While its performance matches that of the $\nabla_{majority}$ operator, it demonstrates significantly higher efficiency, making it the recommended choice.

It is important to note, however, that these results are based on manipulated classifications of nodes and edges. Identifying a classification method that optimally balances precision and recall remains a critical challenge, which is addressed in the following section.

3.2 Cosine Similarity-Based Classifier

In this section we decide to use a restrained set of features and the cosine similarity as a classifier for nodes and edges. The idea is to analyze the available features to determine which will work the best in all the datasets, regardless of the class we are looking for.

Methodology The first step is to analyze which features will be used to embed nodes and edges. We have ten available properties for nodes (*degree strength*, *hits hub*, *hits authority*, *page rank*, *degree*, *triangles*, *local clustering coefficient* (abbreviated lcc), *core number*, *degree centrality*) and six properties for edges (*betweenness centrality*, *strength*, *strength by type*, *weight*, *multiplicity*, *cut edge*). We want to create vectors that combine between two and all the features and keep the most performing combination to avoid unnecessary computation. We use a classical split of 80% for training and 20% for testing. We configure the cosine similarity thresholds for nodes and edges to respectively $\Gamma_{node} = 0.5$ and $\Gamma_{edge} = 0.5$. The top five combinations of node features are presented in Table 2, and the top five combinations of edge features are presented in Table 3.

Table 2: The five best node features combination, according to the subgraphs enumeration F1 score

Combination	F1 score
hits authority, hits hub, triangles	0.053
lcc, degree centrality, degree strength, hits hub, pagerank	0.053
lcc, degree strength, pagerank	0.052
lcc, degree centrality, degree strength, hits hub	0.051
lcc, degree strength, hits authority, pagerank	0.050

Table 3: The five best edge features combination, according to the subgraphs enumeration F1 score

Combination	F1 score
betweenness centrality, strength	0.046
betweenness centrality, cut edge, strength, strength type, weight	0.046
betweenness centrality, cut edge, multiplicity, strength, strength type	0.046
betweenness centrality, multiplicity, strength, strength type, weight	0.046
betweenness centrality, cut edge, multiplicity, strength type, weight	0.046

The combination of node features *hits authority*, *hits hub*, and *triangles* demonstrates slightly superior performance compared to others. For edges, the

best-performing combination is *betweenness centrality* and *strength*. The objective here is not to achieve optimal results, but rather to identify the most effective combination among all tested options. These outcomes are subject to change as threshold optimization progresses. We recommend employing these features for embedding nodes or edges in graphs similar to our industrial context or similar applications.

To ensure sufficient similarity between elements, we examine the impact of cosine similarity thresholds. Adjustments are made concurrently for both node and edge classification thresholds. The results indicate that the best outcomes are achieved with $\Gamma_{node} = 0.8$ and $\Gamma_{edge} = 0.5$.

Results Using the recommended settings, we applied the method to 20% of the datasets, utilizing *hits authority*, *hits hub*, and *triangles* for node classification with a cosine similarity threshold of 0.8, and *betweenness centrality* and *strength* for edge classification with a threshold of 0.5, alongside the simple mediation operator. For the method to be applicable in an industrial setting, we want its performance to exceed at least the results obtained randomly.

However, even with optimized thresholds, the resulting F1 scores were not enough to be used in our industrial context, and as such, we choose not to present the complete results here. Instead, we focus on discussing an alternative solution in the next section.

3.3 Machine Learning as a Classifier

In this section, we decide to use all the available features combined with a machine learning model to classify nodes and edges. We divide the datasets in a variable percentage for training and use the rest for testing.

Methodology The objective of this study is to develop a model capable of classifying edges and nodes across the six dataset types detailed in Section 2.3.

To achieve this, we evaluated four classifiers: XGBoost (XGB), Multi-Layer Perceptron (MLP), Random Forest Classifier (RFC), and Decision Tree Classifier (DTC). Each model was trained using the standard 80/20 split, meaning eight out of ten datasets were employed for training for each dataset parameter combination. This approach resulted in a total of 48 datasets for training and 12 datasets reserved for testing.

Subsequently, we examined the impact of reducing the number of the training datasets to simulate industrial scenarios where only a limited portion of the ground truth is available. To this end, we tested various training split percentages, specifically 10%, 20%, and 50%.

Results We tested those models on the remaining 20% of the data. We present the average (μ) and the standard deviation (σ) of subgraphs enumeration F1 scores obtained during the testing phase in Table 4.

Table 4: Average and standard deviation of subgraphs enumeration F1 scores obtained during the training phase, viewed by class and for each model

Class	Kn		Circle		Star		Tree		All classes	
	μ	σ	μ	σ	μ	σ	μ	σ	μ	σ
DTC	0.637	0.320	0.823	0.092	0.639	0.106	0.667	0.245	0.691	0.220
MLP	0.517	0.378	0.717	0.130	0.013	0.025	0.535	0.359	0.445	0.371
RFC	0.620	0.356	0.857	0.127	0.607	0.302	0.674	0.342	0.690	0.304
XGB	0.636	0.370	0.855	0.120	0.618	0.295	0.723	0.301	0.708	0.293
All models	0.602	0.349	0.813	0.128	0.469	0.340	0.650	0.313	0.634	0.319

Table 4 presents the resulting F1 scores on average for all datasets and for each model viewed by class. Here, the XGBoost model achieves the best performance, with an average F1 score of 0.708. The Decision Tree Classifier and Random Forest Classifier show similar performances, taking the second position. In contrast, the Multilayer Perceptron model yields the poorest results, performing particularly poorly when tasked with enumerating star subgraphs. During the training phase, the edges from the Kn class and the nodes from the Circle class appeared to be the easiest to identify. However, in the testing phase, the best results for subgraphs enumeration were achieved for the Circle class and not for the Kn class. On the one hand, this can be interpreted as the results of a greater impact of the node classification on the ∇_{simple} operator. On the other hand, this can be interpreted as a lack of expressiveness in edge features.

The results of the different splits in training and testing partitions are presented in Table 5.

Table 5: Subgraph enumeration F1 score obtained using different splits

Split	10 90		20 80		50 50		80 20	
Model	μ	σ	μ	σ	μ	σ	μ	σ
DTC	0.599	0.263	0.601	0.270	0.643	0.258	0.691	0.220
MLP	0.546	0.301	0.409	0.359	0.538	0.317	0.445	0.371
RFC	0.619	0.320	0.622	0.318	0.644	0.323	0.690	0.304
XGB	**0.631**	0.319	**0.635**	0.318	**0.664**	0.317	**0.708**	0.293
All models	0.599	0.301	0.567	0.328	0.622	0.307	0.634	0.319

The results shown in Table 5 indicate a decline in overall average performance as the amount of training data decreases. These outcomes are consistent and were anticipated.

However, what was unexpected is that the results remain sufficiently strong to consider applying this method in our industrial context. Notably, even with

only 10% of the data used for training, the XGBoost model achieves acceptable performance.

Discussion The evaluation of machine learning classifiers showed that XGBoost achieved the highest F1 scores for subgraphs enumeration, even when the training data was significantly reduced. This robustness to limited training data is of particular importance in industrial settings, where obtaining large labeled datasets is often impractical due to constraints such as data confidentiality and the General Data Protection Regulation (GDPR).

Interestingly, while the Decision Tree Classifier and Random Forest Classifier performed comparably to XGBoost in some scenarios, their results were slightly less consistent when training data was reduced. The Multilayer Perceptron (MLP), on the other hand, consistently underperformed, particularly in enumerating star subgraphs, suggesting that it is less suited for this specific task.

4 Conclusion and Further Work

This study introduced a pruning-based framework for the enumeration of SubGraphs of Interest (SGI) in large-scale multigraphs, with a particular emphasis on industrial applications. The results underscore the efficacy of the proposed pruning-based methodology, especially in scenarios such as fraud detection, where available training data are scarce.

By employing synthetic datasets designed to emulate user behaviors in transactional services, we assessed the influence of node and edge classification on subgraphs enumeration using one of the four mediation operators, providing insights into their respective contributions to the framework's effectiveness.

Future research will investigate the robustness of this approach under even more constrained data availability, exploring scenarios with merely 1% to 5% of training data. We are also currently working on publishing synthetic datasets. Furthermore, incorporating advanced graph machine learning models, such as Graph Isomorphism Networks (GIN) or Node2Vec for node embeddings and Edge2Vec for edge embeddings, holds potential for enhancing representation quality. Employing different models for node and edge classifications may also yield notable improvements in predictive performance.

References

1. Antonio, B.O., Juan, L.R., Ana, I.D., Francisco, L.C.: Examining user behavior with machine learning for effective mobile peer-to-peer payment adoption. Financial Innovation **10**(1) (Mar 2024). https://doi.org/10.1186/s40854-024-00625-3
2. Aron, J.: Mobile money and the economy: A review of the evidence. The World Bank Research Observer **33**(2), 135–188 (10 2018). https://doi.org/10.1093/wbro/lky001

3. Coulibaly, S., Fofana, S.I., Dembele, M.I.: The orange money service in the niamakoro and faladié districts of bamako: Economic benefits and constraints. European Journal of Science, Innovation and Technology **3**(3), 358–366 (Jul 2023), https://www.ejsit-journal.com/index.php/ejsit/article/view/223
4. El Ayeb, S., Hemery, B., Jeanne, F., Cherrier, E.P.: Community Detection for Mobile Money Fraud Detection. In: 2020 Seventh International Conference on Social Networks Analysis, Management and Security (SNAMS). Paris, France (Dec 2020). https://doi.org/10.1109/SNAMS52053.2020.9336578, https://normandie-univ.hal.science/hal-03949051
5. Fortunato, S.: Community detection in graphs. Physics Reports **486**(3–5), 75–174 (Feb 2010). https://doi.org/10.1016/j.physrep.2009.11.002
6. Giuliani, A., Krishnan, A., Zbilut, J., Tomita, M.: Proteins as networks: Usefulness of graph theory in protein science. Current Protein & Peptide Science **9**(1), 28–38 (Feb 2008). https://doi.org/10.2174/138920308783565705
7. Grinberg, D.: An introduction to graph theory (2023). https://doi.org/10.48550/arxiv.2308.04512
8. Huang, M., Liu, Y., Ao, X., Li, K., Chi, J., Feng, J., Yang, H., He, Q.: Auc-oriented graph neural network for fraud detection. In: Proceedings of the ACM Web Conference 2022. p. 1311–1321. WWW '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3485447.3512178
9. Iverson, K.E.: A programming language. John Wiley & Sons, Inc., USA (1962)
10. Kouam, A.J., Viana, A.C., Tchana, A.: Simbox bypass frauds in cellular networks: Strategies, evolution, detection, and future directions. IEEE Communications Surveys & Tutorials **23**(4), 2295–2323 (2021). https://doi.org/10.1109/COMST.2021.3100916
11. Leskovec, J., Lang, K.J., Dasgupta, A., Mahoney, M.W.: Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters (2008). https://doi.org/10.48550/arxiv.0810.1355
12. Morin, M., Hemery, B., Jeanne, F., Pawlowski-Cherrier, E.: Methodology for identifying social groups within a transactional graph. In: Aiello, L.M., Chakraborty, T., Gaito, S. (eds.) Social Networks Analysis and Mining. pp. 117–130. Springer Nature Switzerland, Cham (2025). https://doi.org/10.1007/978-3-031-78548-1_11
13. Pienta, R., Tamersoy, A., Tong, H., Chau, D.H.: Mage: Matching approximate patterns in richly-attributed graphs. 2014 IEEE International Conference on Big Data (Big Data) (Oct 2014). https://doi.org/10.1109/BigData.2014.7004278
14. Sarma, D., Alam, W., Saha, I., Alam, M.N., Alam, M.J., Hossain, S.: Bank fraud detection using community detection algorithm. In: 2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA). pp. 642–646 (2020). https://doi.org/10.1109/ICIRCA48905.2020.9182954
15. Ugander, J., Karrer, B., Backstrom, L., Marlow, C.: The anatomy of the facebook social graph (2011). https://doi.org/10.48550/arxiv.1111.4503
16. Zampelli, S., Deville, Y., Dupont, P.: Approximate constrained subgraph matching. In: van Beek, P. (ed.) Principles and Practice of Constraint Programming - CP 2005. pp. 832–836. Springer Berlin Heidelberg, Berlin, Heidelberg (2005)