UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Mert Bektas

An Approach for Generating Realistic Synthetic Transaction Data

Master's Thesis (30 ECTS)

Supervisor(s): Alexander Jöhnemark, MSc
Jolanta Goldšteine, Dr. Math
Amnir Hadachi, PhD

Tartu 2024

An Approach for Generating Realistic Synthetic Transaction Data

Abstract:

Privacy factors are crucial in today's constantly evolving financial technology and banking world. Banks are turning to creative and secure solutions to overcome these issues and perform better. Federated Learning (FL) is a novel approach that enables model training between separate organizations. The author collaborated with "Swedbank" and worked on a project to prepare the application of FL with another bank and an intermediary company to enhance our money laundering detection system while preserving data privacy. Both banks used an updated version of the open-source multi-agent-based simulator "AMLSim" to generate synthetic data.

This thesis aims to generate synthetic transaction data close to real-life transactions, making collaborating in anti-financial crime between banks possible. Based on our real-life transaction data, we created features similar to the data generated by AMLSim. Both real and synthetic datasets turned into a graph. The graph evaluation metrics used are In-degree/Out-degree Ratio, PageRank, and Label Propagation. The Snowball sampling algorithm is used to sample real-life transaction data to make it comparable with smaller generated synthetic data. The sampling algorithm is evaluated by generating three different subsamples from the same graph, and their structure is evaluated by the aforementioned evaluation metrics in addition to Graph Density and Graph Components to check if all subsamples are relevant to each other. Finally, generated synthetic graphs are evaluated by the aforementioned evaluation methods to check if their structures are close to real graphs. The results are used to hyperparameter tune AMLSim to generate a more realistic dataset.

Keywords:
synthetic data generation, transaction data, graph subsampling, graph comparison

CERCS: P170 Computer science, numerical analysis, systems, control

Meetod realistlike sünteesitud rahatehingute genereerimiseks

Lühikokkuvõte:

Tänapäeva pidevalt arenevas finantstehnoloogia ja pangandusmaailmas on andmekaitse olulisel kohal. Pangad otsivad innovatiivseid ja turvalisi lahendusi andmekaitsega seotud probleemide ületamiseks ning efektiivsema ärimudeli loomiseks. Federated Learning (FL) on uudne lähenemisviis, mis võimaldab mudeli treenimist eraldiseisvate organisatsioonide vahel. Autor töötas koostöös Swedbankiga projektis, et valmistada ette FL rakendamine koos teise pangaga ja vahendusettevõttega meie rahapesu avastamise süsteemi täiustamiseks, säilitades samal ajal andmete turvalisus. Mõlemad pangad kasutasid avatud lähtekoodiga mitmeagendilise simulatsiooni AMLSim ajakohastatud versiooni, et luua sünteetilisi andmeid ja jagada parameetriväärtuseid.

See magistritöö eesmärgiks on genereerida sünteetilisi andmeid, mis on lähedased Swedbanki reaalsete tehingute andmetele ning leida sobivad parameetriväärtused, mida jagada vahendusettevõttega, et tulevikus FL edukalt taasrakendada. Taoline sünteetiliste andmete genereerimine, mis oleks sarnased reaalsetele tehingutele, võimalikuks pankadevahelise koostöö finantskuritegevuse vastu. Reaalsete tehingute andmed filtreeritakse, et neil oleksid AMLSimi genereeritud andmetega sarnased omadused. Nii reaalsed kui ka sünteetilised andmekogumid muudetakse graafikuks. Graafiku hindamiseks kasutatud mõõdikud olid In-degree/Out-degree Ratio, PageRank ja Label Propagation. Valimi moodustamiseks kasutati Snowball sampling algorithm'i, et muuta need võrreldavaks väiksema sünteetilise andmekogumiga. Selle algoritmi hindamiseks luuakse samast graafikust kolm erinevat alamgraafikut ja nende struktuuri hinnatakse eelmainitud hindamismõõdikute abil ning Graph Componenti ja Graph Density'ga, et veenduda alamgraafikute sarnasuses teiste alamgraafikutega. Viimasena sünteetilised graafikud hinnatakse eelmainitud meetodite abil, et kontrollida, kas nende struktuurid on lähedased reaalsetele graafikutele. Tulemusi kasutatakse AMLSimi hüperparameetrite häälestamiseks, et genereerida realistlik andmekogum.

Võtmesõnad:
sünteetiliste andmete genereerimine, tehingute andmed, alamgraafikute testimine, graafikute võrdlus

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Acknowledgements

I want to show my genuine gratitude to my thesis supervisor, Dr. Amnir Hadachi, for his constant support, for giving the right directions to make things more organized, and for his understanding nature. He was a great companion in this journey. I also want to express my gratefulness to Dr. Jolanta Goldšteine for her positive attitude and support throughout the project. I enjoyed our collaboration on the project very much. I am deeply grateful to my "Industrial Master's Programme in IT" program supervisor and teammate in Swedbank, Alexander Jöhnemark, for continuously checking if I need support and his extremely quick support and input whenever I need it. He always surprises me with how immediately available he becomes whenever I need support. I am also thankful to my friend Li Merila for dedicating her time to assist me in translating the English abstract of this thesis into Estonian. I want to express my most profound thanks to my family for their encouragement, for curiously asking about the status of my thesis regularly, and for motivating me to be on my thesis schedule while doing that. Lastly, I owe a debt of gratitude to Pino D'angio for his amazing music and for making my thesis writing progress easier and more fun.

Contents

1 Introduction

1.1 General Context

Banks increasingly turn to innovative approaches to enhance their services while addressing privacy concerns in the rapidly evolving landscape of finance and financial technology. One such ground-breaking paradigm is Federated Learning (FL). This innovative approach enables model training across multiple decentralized entities without sharing data because only model parameters are shared and aggregated centrally. Therefore, it is helpful for scenarios where data privacy and security are top priorities.

This thesis concerns a project funded by "Vinnova", the Swedish government agency for innovation, under project number 2022-03063. A Swedish intermediary company supported the development and coordination of the project. This project aims to explore the potential of Federated Learning in enhancing collaboration between banks for detecting money laundering while preserving data privacy. "Swedbank" is one of the project participants with whom the author of this thesis worked and contributed to the Federated Learning project.

Expected results of this project include:

1. Enhanced identification of money laundering activities spanning multiple banks.
2. Development of innovative techniques in federated learning customized for anti-money laundering purposes.
3. Strengthened cooperation among banks and external entities, addressing common challenges and fostering innovation.

Since neither bank could share its data directly, synthetic data similar to real data had to be generated. To achieve this, both banks used an updated version of an open-source multi-agent-based simulator called "AMLSim," which the project intermediary provided at the beginning of the project.

For the first part of the project, the author and his team had to generate synthetic data and turn it into a graph, then turn one of Swedbank's datasets into a graph and subsample it if needed. Next, use proper graph comparing metrics to compare the real and synthetic graph's structures, tune hyperparameters of AMLSim to get the optimal synthetic dataset

that is the closest to the real graph. Thus, this thesis primarily focuses on the project's first phase, which involves generating a realistic synthetic graph.

1.2 Roadmap Overview of the Project and Contributions

This subsection provides an overview of the project in which the author has been involved and the author's contributions to it.

The project started with our team getting AMLSim's updated version from the intermediary company to generate synthetic datasets. Then, we implemented graph creation functionality for our real transaction data. Since our real transaction data was big, we had to find and implement a method to subsample our real graph into a smaller graph. It was found that the implemented method is not optimized for big graphs, so our next task was to optimize it to scale well with our big graphs. To ensure that our subsampling method is functioning well, we had to put it into an experiment and create several subsamples to compare them. We also filtered the synthetic dataset to make its structure match the real dataset. After this step, we created a graph from the synthetic data, just like our real dataset. We investigated possible graph comparison methods to evaluate how realistic our synthetic graphs are. We conducted experiments on hyperparameters of AMLSim and generated several datasets to find the optimal hyperparameter combination.

The author's contributions to the project were researching a graph subsampling algorithm, implementing it, and optimizing it for big graphs. Then, experiments are conducted on created subsamples to see if every created subsample is similar to each other to ensure the algorithm is consistent in creating subsamples. To ensure that the comparisons were accurate, the author created a pipeline that filters synthetic datasets after their generation to make them resemble the real graph in terms of structure.

In summary, the roadmap for the project was to:

1. Create a synthetic data by using AMLSim
2. Create a graph from our real data
3. Find a method to subsample our real graph
4. Optimize the method to make it scale well with our big graphs
5. Create subsamples from our real graph

6. Filter the synthetic data to make it suitable to compare with our real graph
7. Create a graph from the filtered synthetic data
8. Find methods to compare real and synthetic graph
9. Compare real and synthetic graph
10. Hyperparameter tuning to create the optimal synthetic graph that is the closest to our real graph

My contributions to the project are:

1. Find a method to subsample our real graph
2. Optimize the method to make it scale well with our big graphs
3. Create subsamples from our real graph
4. Filter the synthetic data to make it suitable to compare with our real graph
5. Create a graph from the filtered synthetic data
6. Hyperparameter tuning to create the optimal synthetic graph that is the closest to our real graph

The details of the work are described in Figure 1. With a reflection on lesson learned from methodological and technical point of view.

2 Literature Review

This section provides relevant theories, concepts, and previous research related to the thesis topic. After providing information from other sources, we discuss how the shared work relates to our topic and how the information can benefit us.

Detecting financial fraud is a challenge for every financial institution. There is a strong need to detect financial fraud to preserve customers' trust and keep assets safe. However, there are obstacles to achieving this. One of these factors is that financial crime happens rarely, and many transaction datasets are not balanced enough due to datasets having fewer examples of fraud. This problem can affect the dependability of fraud detection models. In addition, financial institutions can not share their data with any third party to enable cooperation, creating stronger centralized detection models [5]. This research aligns with what we want to achieve in our project—enabling collaboration with other financial institutes to enhance our financial fraud detection systems. The paper's authors made several modifications to enable FL for fraud detection. They did not use a typical machine learning model update in a centralized training environment, such as Stochastic Gradient Descent (SGD), to achieve this. Instead, they used an averaging algorithm developed by McMahan et al. [18], a modified version of the standard SGD. In this new algorithm, each client computes its update. After these calculations and updates, the server aggregates these updates to form the global model update. This process is repeated until convergence. Even though they did not compare the performance of centralized and FL learning models, they reported that the FL model had a definite accuracy score of 93%. Moreover, they drew attention to the fact that they achieved this by keeping the privacy of their data [5]. In contrast to this paper, we will not use our actual data while training.

We will create and use a realistic synthetic dataset to ensure that our training is fully secure regarding data privacy. Synthetic data generation is important in the financial sector due to privacy concerns and regulatory restrictions, which restrict institutions from sharing their data. A published paper addresses this growing need by exploring three main areas: generating realistic synthetic datasets, measuring their similarity to real data, and ensuring privacy constraints during this data generation process [4]. For generating synthetic tabular data, several methods have been proposed [26]. However, these existing techniques come with their problems. Firstly, most differential privacy frameworks take

each row of a table as a bit string, which is equal in length to the domain size, which makes these techniques impractical. The reason is that the number of columns increases because of the exponential growth. Secondly, sparse high-dimensional datasets often end up having added noise that overpowers the real data. Therefore, the resulting synthetic dataset is a bad approximation of the real dataset [28]. Agent-based modeling (ABM) is also used to generate synthetic datasets. An ABM work has been done by simulating the behavior of multiple agents (representing banks) over multiple periods within a real-time gross settlement (RTGS) payment system to model a bank's payment processing system [10]. ABM has also been used to generate synthetic data for a retail shoe store in the past [17]. When ABM is calibrated manually, the generated synthetic dataset respects privacy constraints. However, it should be kept in mind that automatizing the calibration process of ABM may pose a risk of data leakage [21]. In this thesis, we will use an agent-based simulator, and this research taught us that calibration should be done manually to keep our data safe from leakage.

Researching mobile money transaction frauds are not an easy task. Because finding legitimate mobile money transaction datasets is difficult due to financial institutions' inherent privacy, a paper introduces PaySim, an agent-based mobile money transaction simulator [16]. PaySim has pre-defined transaction types, and the characteristics and behaviors of the customers are based on statistical analysis and distributions gathered from real transaction data. This approach enables PaySim to simulate complex client transactions like in real life. Using PaySim, we can imitate real transaction data and work on it without financial institutions' constraints and legal boundaries. Understanding how PaySim works is crucial for our project since we will use a simulator called AMLSim, built on top of PaySim.

AMLSim is also an agent-based transaction simulator created especially for anti-money laundering (AML) purposes [27]. It is designed to generate synthetic transaction data resembling real-life financial transactions and activities, allowing people to work and research transaction data in a controlled environment. AMLSim operates in two main steps. Firstly, it generates a graph using NetworkX [11] based on pre-defined degree distribution. Second, it generates a time-series of transactions based on observed dynamics from real data using PaySim. Finally, it incorporates domain expertise and defines known suspicious transaction patterns within the generated data. As a result, it generates

a realistic synthetic transaction dataset containing suspicious transactions. This synthetic dataset mimics real-world scenarios and makes the simulator effective in anti-money laundering works.

Graphs are data structures that are essential in representing relationships between entities. Utilizing realistic graphs is important for research and development. However, it is challenging due to obstacles such as sensitive data or data scales, which obstruct the usage of real-world graphs. A synthetic graph can be a solution to such problems. It is possible to hide critical information using a synthetic graph [15]. A way to achieve this is graph anonymization. Graph anonymization aims to keep the identity information hidden in graphs [29]. This can be critical for usage in many areas where graphs contain sensitive information. Differential privacy-based is one solution that hides information by adding noise to statistical queries' answers. Karwa et al. [12] proposed a technique that preserved the privacy in graphs by proving approximated answers to subgraph counting queries. However, this method has the downside of sacrificing some precision in query responses. The same paper [15] talked about graph sampling techniques, which aim to create smaller representations of given graphs. This can be achieved by selectively sampling edges [2] or nodes [13]. This process is similar to reducing the density of the graph. Graph sampling can be useful for big graphs since it creates computationally efficient and more manageable versions of the given graph [3] while still keeping the graph's attributes and characteristics [14]. These approaches can be useful in this thesis since we will be working on big graphs and need to do computations with them.

3 Methodology and Analysis

This chapter overviews the steps towards making synthetic and real graphs ready for hyperparameter tuning and comparison. The following subsections of this chapter will provide an in-depth overview of each aspect. The first subsection explains how we performed graph creation and hyperparameter tuning. The second subsection explains why the project needed a subsampling method, how the research and experiments were done on several subsampling methods, and the chosen method in detail. The third focuses on optimizing and debugging the implemented subsampling method and then subsampling the real graph. The fourth explains the graph creation process from tabular synthetic datasets and the preprocessing steps applied. The fifth subsection presents graph comparison metrics to demonstrate how synthetic graphs are evaluated. The final subsection reports results on which metrics we decided to use for hyperparameter tuning and why.

3.1 Workflow Architecture

Figure 1 explains the workflow loop of generating a more realistic synthetic graph each time. In the beginning, the workflow splits into two separate roads, one for creating a graph from Swedbank's data and one for creating a graph from the generated synthetic data. For synthetic graphs, experiments are done in batches, which means that for each hyperparameter, only the parameters have been changed for a higher or lower value. Then, I compared these synthetic graphs with the real graph to see which range of the chosen hyperparameters performed the best. In order to have them equal in length in the transaction period, we take the last 365 days from both datasets. After this step, we create a graph from the real data, and the real graph is ready for comparison. For the synthetic graph, the source and sink are getting dropped. Then, filtering to have transactions only involves Swedbank customers. After these operations, we also create a graph from the synthetic data. Finally, both of our graphs are ready for comparison.

We will first use the In-degree/Out-degree ratio for the comparison and hyperparameter tuning step. For this comparison, we calculate each node's In-degree/Out-degree ratio in both graphs and create a histogram for the results. Then, we evaluate the results. After the evaluation, if we could not decide which hyperparameter value to pick, we used other

comparison metrics, PageRank and Label Propagation, to finalize the comparison. Next, we calculated and evaluated the scores using the chosen algorithms for real and synthetic graphs. After our evaluations, we checked if the synthetic graph was realistic enough. If not, we changed the AMLSim hyperparameters to create a new batch of experiments to find the most optimal value for the chosen hyperparameter. If it was realistic, we have achieved our goal.

Docker is an open-source platform that makes executing and distributing applications easier. Applications created by Docker are packaged into units called containers, with all the dependencies needed to run the application. [7] As mentioned before, at the project's beginning, our team got an already-built Docker container for AMLSim from the intermediary company. Therefore, we could generate datasets with ease and start this project with synthetic datasets in our hands, with no preparation earlier. Next, other people in our team created a graph from our transaction data using an "Apache Spark" package called "GraphFrames." [9] GraphFrames provides a DataFrame-based API for working with graphs. We decided to work with GraphFrames because our company has big data and already uses Apache Spark, and GraphFrames integrates seamlessly.

3.2 Subsampling

The author researched how to subsample the real graph because it was too big to operate certain comparison methods to evaluate our synthetic graph. A good subsampling algorithm had to generate subgraphs relevant to our real graph and always identical subgraphs, meaning the algorithm should be consistent. Following this research, a decision had to be made between Stratified Sampling and Snowball Sampling, which Snowball Sampling ended up being used in the project.

Stratified sampling is a probability sampling method that is implemented in sample surveys. The target population's elements are divided into distinct groups or strata where within each stratum the elements are similar to each other with respect to select characteristics of importance to the survey [25]. An example of Stratified sampling can be seen from Figure 2. Using this method for graph subsampling would be good because different substructures or communities within the graph are represented in the sample, preserving the overall graph structure. Therefore, it was helpful to have low bias, which is what we intend to have. However, it was computationally heavier than other choices.

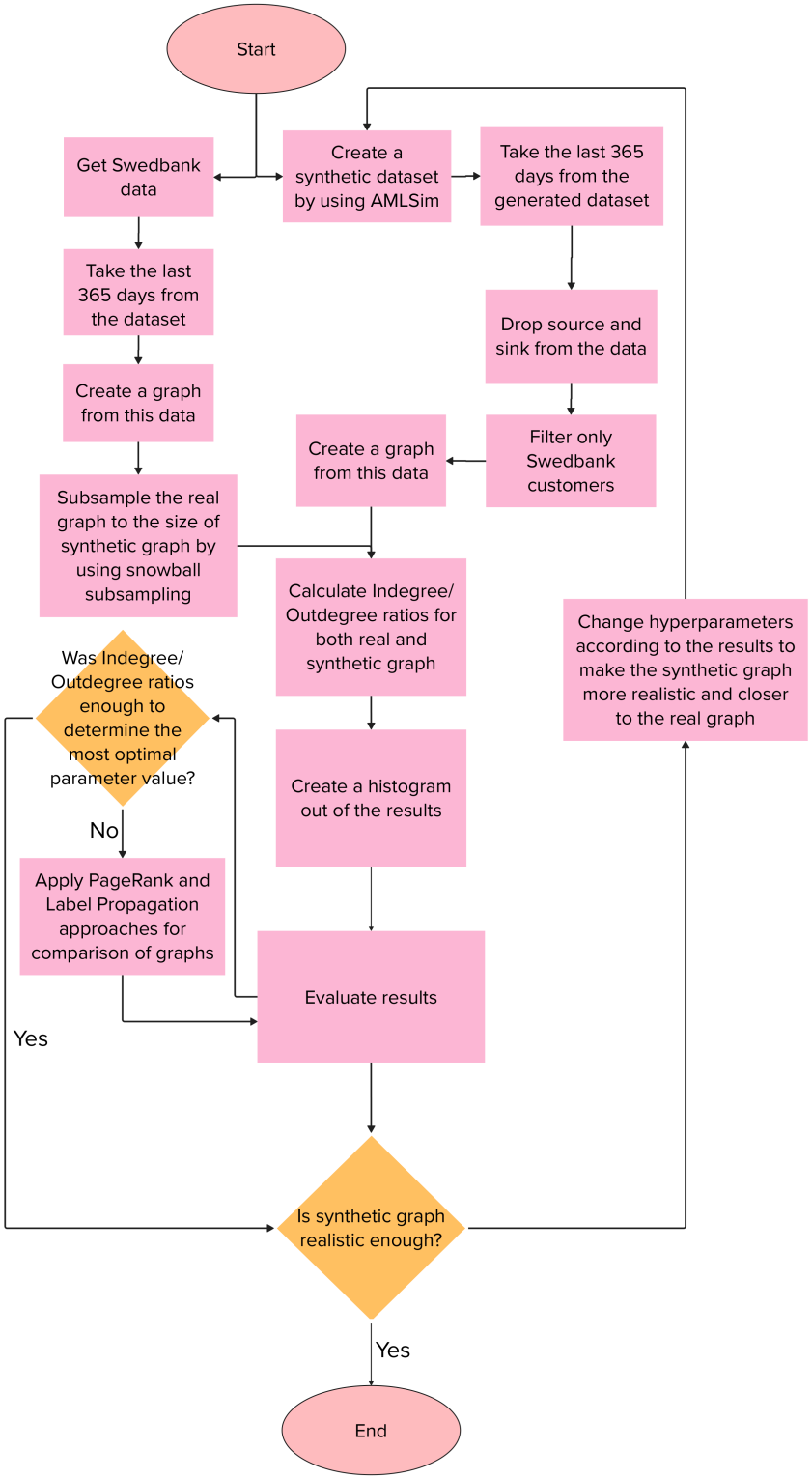

Figure 1. Workflow diagram of realistic synthetic graph generation

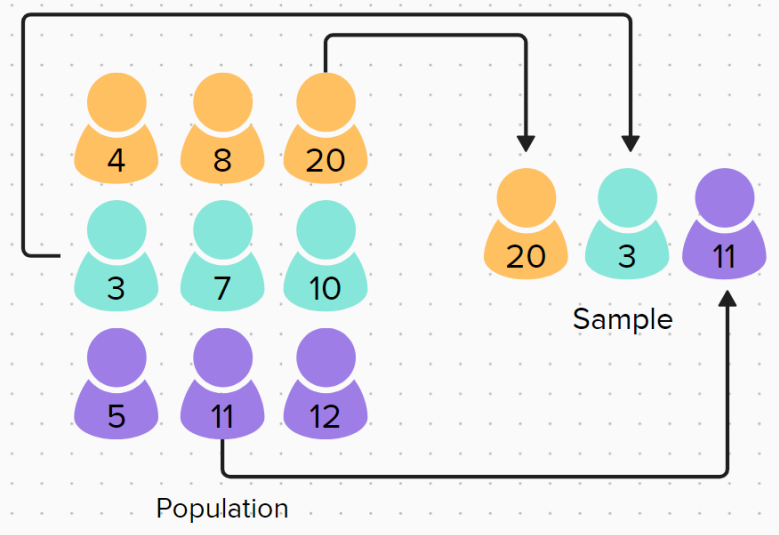

Figure 2. Stratified sampling example

Snowball sampling is one of the most popular methods of sampling in qualitative research, central to which are the characteristics of networking and referral. Figure 3 shows an example of Snowball sampling. The researchers usually start with a small number of initial contacts (seeds) who fit the research criteria and are invited to become participants within the research. The agreeable participants are then asked to recommend other contacts who fit the research criteria and who potentially might also be willing participants, who then, in turn, recommend other potential participants, and so on. Researchers, therefore, use their social networks to establish initial links, with sampling momentum developing from these, capturing an increasing chain of participants. Sampling usually finishes once either a target sample size or saturation point has been reached [20].

Snowball sampling was chosen because we could randomly choose a seed node each time we started sampling. This would eliminate bias due to the randomness. Also, it is more cost-effective than stratified sampling. In addition, while researching, we found a Python library for graph subsampling called "Little Ball of Fur." [24] The author got inspired by it and wrote his subsampling code optimized for big graphs with Snowball sampling inside. In this way, implementing a subsampling algorithm from scratch was not necessary and saved much time in the thesis work. The written code's sampling part can be seen from Algorithm 1.

Several experiments have been done for Snowball Sampling to ensure that generated

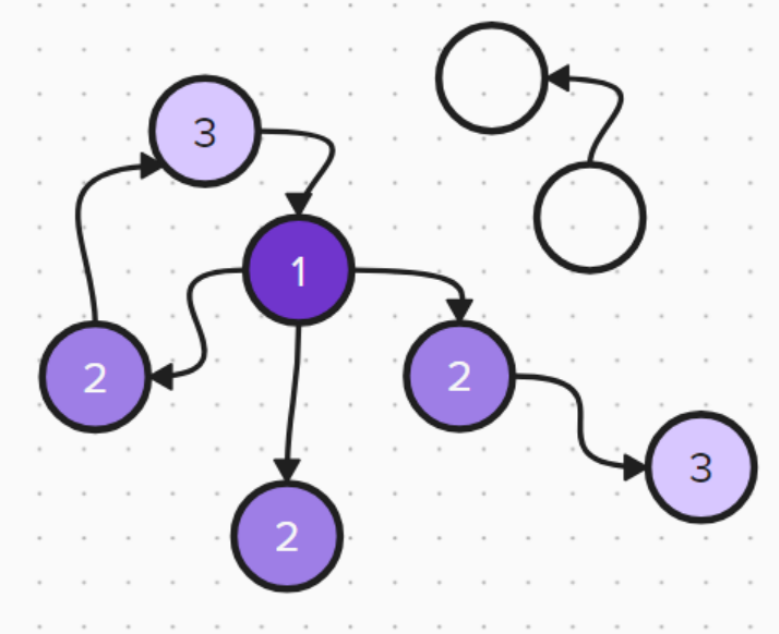

Figure 3. Snowball sampling example

subsamples are not biased and are relevant to each other. Various calculations and algorithms are used to evaluate generated subsamples. These are Graph Components, Graph Density, PageRank, and Label Propagation. Experiments are done by comparing three subsamples of the default synthetic graph with a quarter of its node size.

The results obtained by comparing generated subsamples with the four graph comparison metrics mentioned earlier are displayed in Figures 4, 5, 6, 7. Results show that there are minimal differences between the results of the subsamples. These experiments prove that Snowball sampling is an excellent sampling method for the project; there is no bias in generated subsamples, and they are relevant to each other.

3.3 Snowball Subsampling Implementation

Little Ball of Fur uses another Python library called NetworkX to work with graphs. Moreover, NetworkX graphs differ from GraphFrames graphs, which we used in our project. Therefore, the subsampling function from Little Ball of Fur was incompatible with our graphs. As the solution, the function's source code was found in the library's GitHub repository. Then, the code was customized to make it compatible with Graph-

Algorithm 1: Sampling in Snowball Sampling

Input: Input Graph $graph$,
Desired node size for subsample $desired\_number\_of\_nodes$,
Result: Subsampled Graph $subsample$

```
function Sample(graph, desired_number_of_nodes)
    nodes ← empty set;
    queue ← empty queue;
    nodes ← start_node;
    queue ← start_node;
    while size(nodes) < desired_number_of_nodes do
        temp_node ← queue;
        Find temp_node's neighbors;
        for each neighbor do
            if neighbor is not in nodes set then
                nodes ← neighbor;
                queue ← neighbor;
            if size(nodes) = desired_number_of_nodes then
                break;
        if size(queue) = 0 then
            Select a new node from graph;
            queue ← new node;
    subsample ← Filter graph with nodes in nodes;
    return subsample;
```

Frames instead of NetworkX. This approach was successful, but this solution brought a problem with itself. It was discovered that customized subsampling performs very slowly. This is because NetworkX is optimized for small graphs and does not scale well with big graphs. Therefore, it took a lot of time to finish the execution of our very big graph, which made the project much slower. After an investigation, it was found that the problem was related to the code's random node selection part. Little Ball of Fur uses Python's random module to pick a random node, causing the code to fetch all the nodes in the graph, turn them into a Python list, and then pick a random node. This was making the execution much longer, considering the size of our graph and how often the

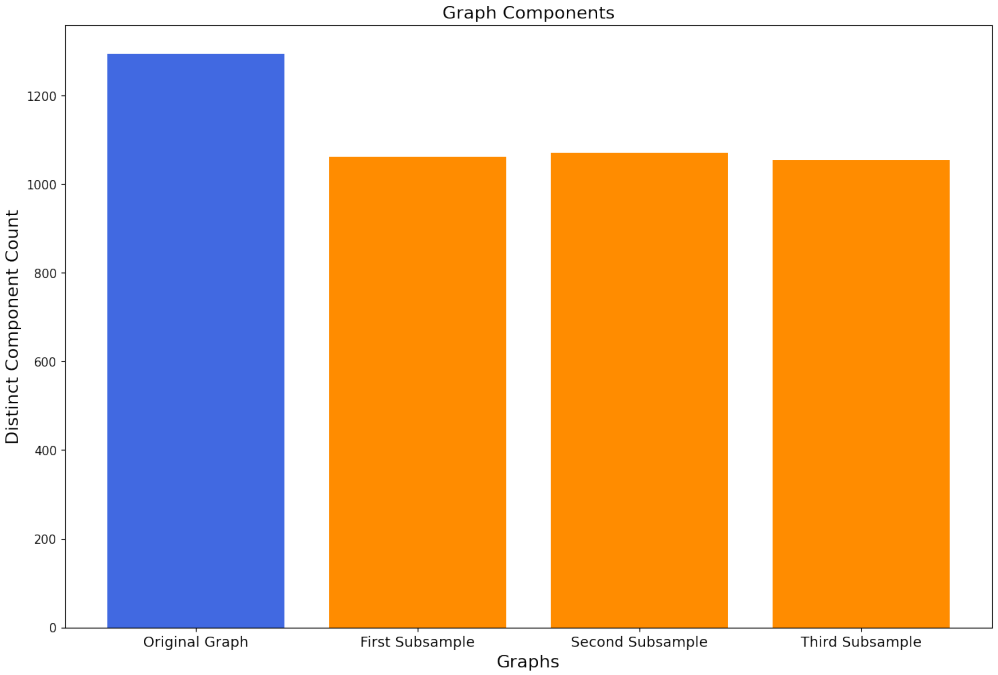

Figure 4. Subsampling experiment using Graph Components

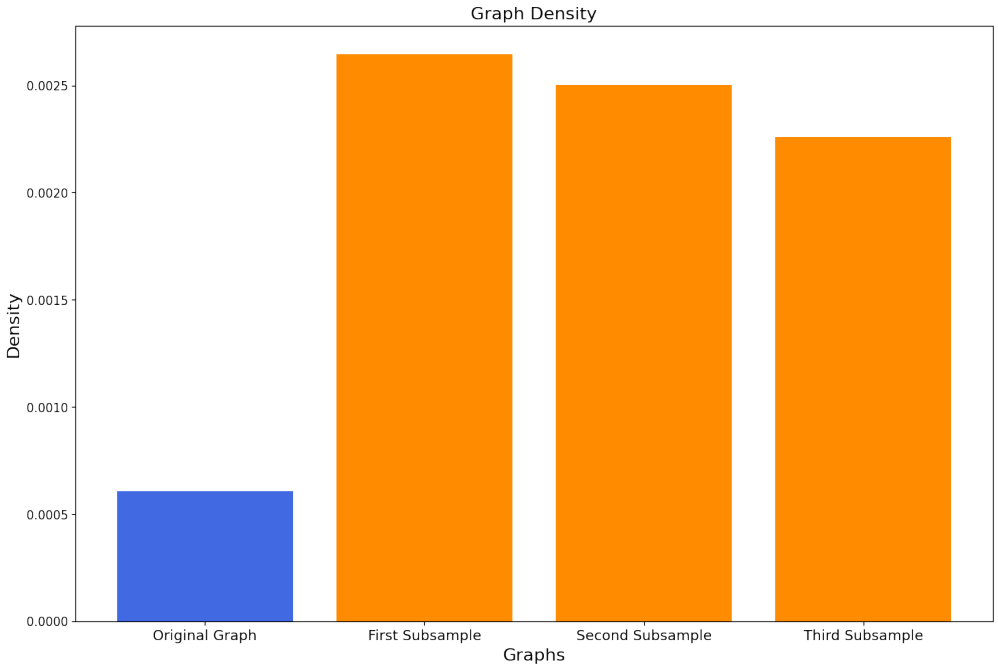

Figure 5. Subsampling experiment using Graph Density

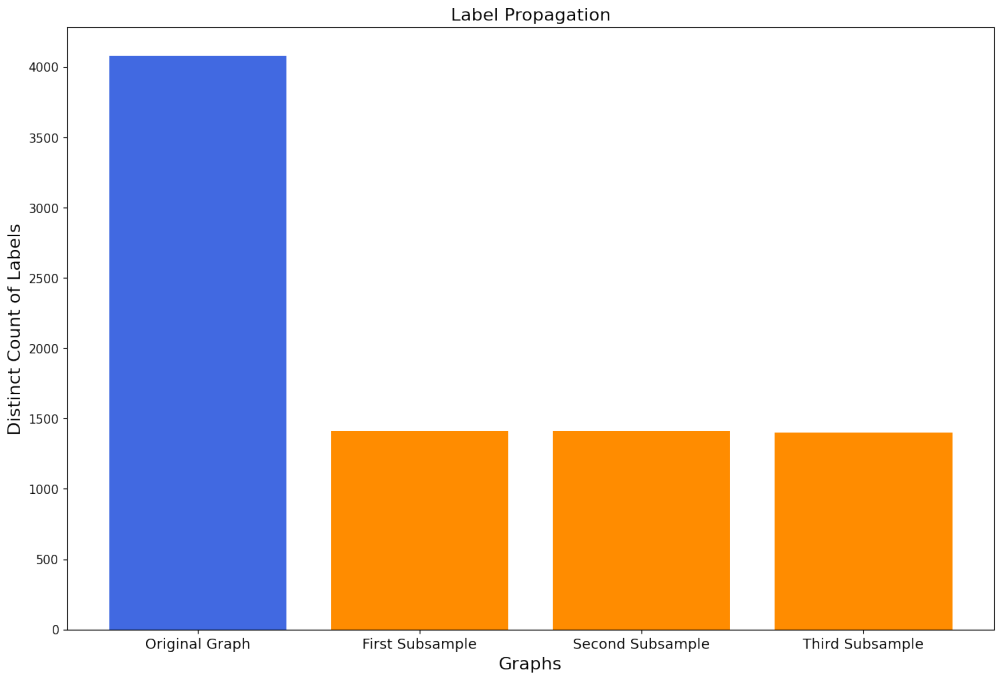

Figure 6. Subsampling experiment using Label Propagation

algorithm had to do this until we got a subsample. The code was operating so slowly, to the extent that it continued running for hours without producing any outcomes. An experiment was done to subsample a graph with 400 nodes, and it took approximately 1.06 days. The solution for this issue was using Dataframe's sample function instead of Python's random module. Currently, the algorithm fetches only ten nodes from the graph at once rather than all the nodes from the input graph. After this step, it picks a random node in these ten nodes. There is a reason why ten nodes are being fetched, rather than such a smaller number since we are only picking a single node in this step. This is because the algorithm has to redo this step if the picked node is already in its subsample. This would cause the algorithm to sample nodes from the graph one more time, therefore making the execution even slower. As a result, subsampling started to operate significantly faster. An experiment was done after these changes, and a graph with 400 nodes was subsampled again. The execution time was 1.53 minutes. As a result of this optimization, we decreased the execution time for the same graph from 1.06 days to 1.53 minutes.

Moreover, there was a bug in Little Ball of Fur's code. During execution, if no nodes

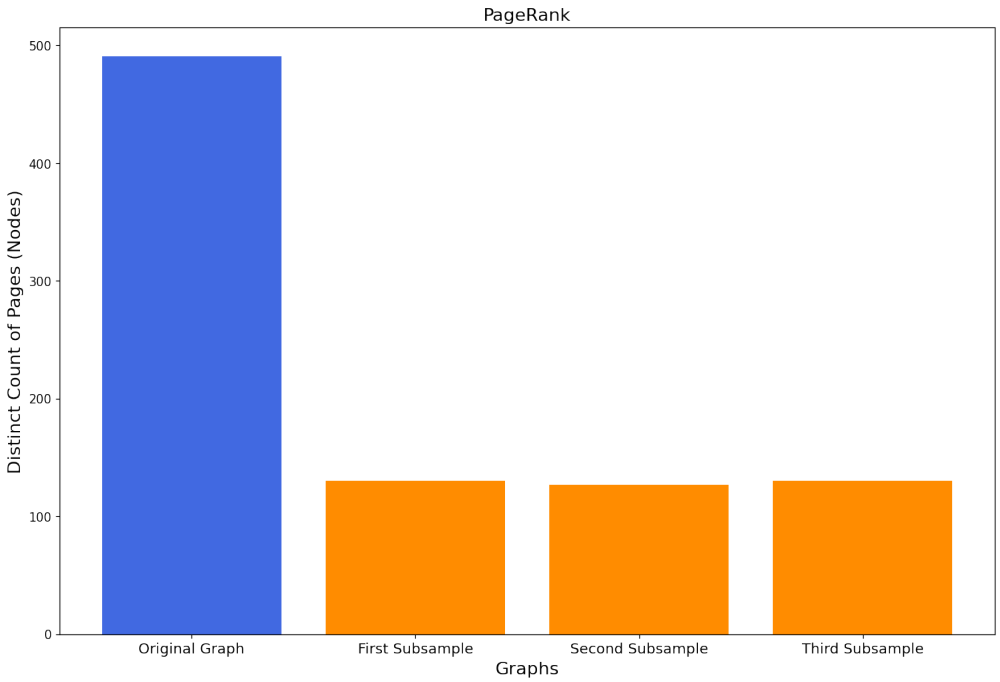

Figure 7. Subsampling experiment using PageRank

are left in the queue, the program goes into an endless loop and does not look for new neighbors to add to the nodes list. This issue is fixed by adding an if statement to check if the current queue is empty. By this change, if the queue is empty, the program picks a new random node from the graph, puts it into the queue, and continues working with no issues.

Thanks to all these works, it became possible to generate subgraphs of any desired size effortlessly and without facing any issues. Based on these results, the work on real graphs was over, and it was ready to be compared with generated synthetic graphs.

3.4 Graph Creation from the Synthetic Data

AMLSim generates transaction data as tabular data. Therefore, it should have been converted into graph format to compare with the real graph. Before creating a graph out of the generated data, a filtering process had to be done to eliminate unwanted and unnecessary features to make it have the same format as the real graph. AMLSim has a mechanism called source and sink. They are pseudo-customers used to adjust

customer account balances in the synthetic dataset. They do this by putting money into the simulation from outside or sending the simulation's money to outside. In summary, the source is the money from outside sources, and the sink is the money that went outside the simulation. This thesis aims to imitate real data structure, and customer account balances are irrelevant to this intent.

For this reason, the account balance information is not included in the real graph. Therefore, all transaction rows that got the money from the source or put its money into the sink have been removed from the generated synthetic data. Only transactions that were linked to Swedbank are kept. Customer ID, transaction origin, and destination column names are changed to make the data suitable for GraphFrames graph creation requirements. Customers and transactions are separated from the generated data. Different DataFrames are created for customers as nodes and transaction parties as edges. These nodes and edges DataFrames are used to create a GraphFrames graph. These steps are combined to create a pipeline for future synthetic graph generation. Finally, our synthetic graphs were ready to be compared with our real graph.

3.5 Graph Comparison Metrics

Research on graph comparison metrics was conducted to evaluate how realistic generated synthetic graphs are. This research also enabled the possibility of hyperparameter tuning of AMLSim to create the optimal synthetic dataset by making changes according to the evaluations. The first attempt towards this was to try the *Graph Edit Distance* (GED) metric. GED is the minimum cost of an edit path between two graphs. An edit path between graphs G and H is a sequence of edit operations that transforms G into H. There are six edit operations, namely, node insertion, deletion, and substitution, as well as edge insertion, deletion, and substitution. Each edit operation comes with an associated non-negative edit cost, and the cost of an edit path is defined as the sum of the costs of its edit operations [6].

PageRank is a method used by Google to evaluate the importance of web pages and their relevance to the search. It works as a voting system, where each link to a page from other pages is considered a vote. Each page has its self-importance score (PageRank). This score is evenly distributed among the page's outgoing edges [23]. While this approach is used to find the most relevant web page, we used it in our graph comparison to find the

distinct PageRank scores in a graph. This way, we can get more insights into the graph's structure since we can calculate how many relevant and irrelevant nodes are in the graph and then compare the score with another graph to see if they have a similar pattern.

Graph Density is a measure of network connectivity that shows how linked the nodes are in a given graph. It is calculated by dividing the number of edges in the graph by the maximum possible number of edges [19]. This simple measure can show us how connected customers are in our real and synthetic graphs, therefore, these graphs' structure.

Graph Components in an undirected graph are defined as connected subgraphs, not part of any larger connected subgraph. Each node in a graph belongs to a component; the whole graph is a structure in which these components come together [8]. Each component is like an island in the whole graph. They come together and create one big graph. We calculated how many components our real and synthetic graphs have to check if they share a similar structure.

Label Propagation is a semi-supervised machine-learning algorithm that assigns labels to unlabeled data points. Firstly, the algorithm starts by labeling a small amount of data points in the whole data. Then, it spreads these labels to unlabeled data points through its execution [30]. In the context of complex networks, where community structures are common, label propagation can be used to identify these communities within the network [22]. We can use Label Propagation to identify communities, customer types, and communication patterns. This can be used as a graph comparison metric to see if the real and synthetic graphs have similar communities.

Except for the In-degree/Out-degree ratio, we implemented all the algorithms and metrics mentioned in this subsection from the GraphFrames library, which is already available for everyone to use [1].

3.6 Graph Comparison Metrics' Results

We found out NetworkX had a function to calculate GED. We converted our GraphFrames graphs to NetworkX graphs to make the function work and start doing tests. We had to use subsamples of the original graph to achieve this because NetworkX had out-of-memory problems with our big graphs. After testing the function, it was discovered that

the function never stopped executing. While trying to find a solution, it is discovered from its documentation that the function has a timeout parameter. However, a new problem arose because there was no clarity on how long to set the timeout. If we set it too short, we could not get the correct results or even get results because the calculation had yet to be done. We tested several timeout values on the same graph to determine the optimal timeout parameter. The results of the tested timeout values can be seen in Table 1.

Number of Nodes	**Timeout Parameter (s)**	**Graph Edit Distance**
100	50	323
100	200	323
100	500	323
500	50	No result
500	200	No result
500	500	4534

Table 1. Graph Edit Distance of graphs with several timeout parameters

However, this test was not helpful because the bigger your graph is, the longer you should set the timeout. Moreover, the only way to find the optimal timeout is to brute force the parameter. Another problem was the need to use bigger graphs to compare and get more accurate results about the similarities in our graphs. However, this was causing out-of-memory problems with the similarity function. To solve this problem, we tried to customize the code, just like we did with the Snowball sampling. However, this idea was quickly dropped since the source code was big and much more complex than the Snowball sampling code.

Graph Components is another algorithm we gave up using in our hyperparameter tuning. This is because we discovered that every time we execute the algorithm on our subsamples, the algorithm finds only one component. This meant that our subsamples had no islands; they were big, fully connected graphs, unlike the real graph. After seeing this, we tried to run the algorithm on our real graph without subsampling it. Then, we saw that the algorithm works slowly on the real graph. For these reasons, we decided that this would not be a proper algorithm for hyperparameter tuning. Therefore, we decided to use this algorithm only for subsampling bias experiments.

We also gave up on Graph Density. After calculating the score for a sampled real graph

and several generated synthetic datasets, we realized their scores were not correlated. This may be because the real graph is sampled by using Snowball sampling, which works by iterating through nodes' neighbors. Furthermore, since the real graph or synthetic data does not have such a structure, they are naturally less connected than the sampled real graph. Therefore, it was not accurate to compare synthetic graphs with a subsampled graph using this algorithm. We had to drop this approach since we could not execute this algorithm on the real graph without subsampling as well.

In-degree/Out-degree ratio is our best metric. Because it is fast to execute, and the results are visible and detailed. We can see which node degree we should improve in order to make the synthetic dataset more realistic. Which enabled us to have more ideas on how to improve the generated datasets.

PageRank and Label Propagation was also successful. They were much faster compared to Graph Components and Graph Density, which enabled us to do these comparisons. And their results were on align with each other and In-degree/Out-degree ratio. Which showed that they were reliable.

Ultimately, we decided to use the nodes' In-degree/Out-degree ratios of graphs as our first choice to compare since In-degree/Out-degree ratios are a great way to see if the generated dataset captures the structure of our real graph and it was our the most reliable and accurate comparison method. If experiments have close results, we will check Label Propagation and PageRank to make the final decision.

4 Hyperparameter Tuning

Fine-tuning AMLSim's hyperparameters was crucial to generating synthetic data that was close to our real data. AMLSim has many hyperparameters that change the behavior of simulated customers and the structure of the generated dataset. We tried to capture our real graph's structure in the synthetic graph rather than focusing on details, such as transaction amounts. Nevertheless, these details affect the general structure. Therefore, paying attention and changing these attributes in our hyperparameters was important. In addition, we had to be careful not to generate a dataset identical to our real dataset due to privacy concerns. Luckily, AMLSim is a simulator that cannot generate precise datasets with no bias. This issue will be addressed in the following subsections, and an example of bias will be provided in the forthcoming sections.

Experiments were done by changing hyperparameters in AMLSim's parameter files. Which are one JSON file, which defines the behaviors of accounts, and one CSV file, which defines the structure of the transaction network. Various hyperparameters are changed during the experiment process.

4.1 Limitations

Hyperparameter tuning required an initial environment setup because of our way of working in the bank and a couple of problems we faced. The first problem was that we could only compare synthetic and real graphs on a cloud platform rather than a local environment. Moreover, synthetic graph generation was the opposite. AMLSim was only available to use in our local environments. Therefore, we had to generate synthetic datasets locally and then commit the generated datasets to our repository. Finally, we had to pull the changes from our repository to our cloud platform to compare the graphs. Here comes the second issue. Our repository had limited space, and we generated datasets containing more than one year of transaction data. This limited space was not big enough to keep the generated datasets. We overcame this problem by creating a pipeline that properly filters synthetic datasets and cuts down the generated datasets to 365 days, which was small enough to keep in the repository. This solution brought a new problem as well. The pipeline was using packages that were unavailable in our local coding environment.

Furthermore, we could not completely switch to a Docker container since AMLSim only works in my local environment. Moreover, our privacy policies made installing packages in our local coding environment impossible. To overcome this issue, we created a Docker container that mounts our local project folder. By doing this, we could work on the same folder but with different environments for different purposes and make our setup ready to compare real and synthetic graphs.

4.2 Experiments with degree.csv

The degree.csv file in AMLSim defines the structure of the transaction network It consists of 3 columns. Explanations of each column can be found in table 2.

Column Name	Explanation
Count (integer)	The number of nodes, as known as accounts.
In-degree (integer)	The in-degree of the nodes.
Out-degree (integer)	The out-degree of the nodes.

Table 2. degree.csv parameter structure

The graph needs to be complete, i.e., the sum of the in-degree and out-degree of all nodes needs to be equal. Furthermore, the total count needs to be equal to the number of accounts in the accounts.csv file.

Configuring degrees of accounts were important for us, since we observed a bump in our synthetic graph's histograms, which we did not have in our real graph. And we eliminated this bump by changing degrees in the file according to our real data. As a result, we had a more realistic synthetic graph.

From Figure 8, the modified synthetic graphs from Experiments 3, 4, and 5 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because other experiments do not have a smooth ratio decrease while moving toward the right in the plot.

Since we could not decide which one to choose, we looked at the results of other comparison algorithms.

According to the test results in Table 3, in PageRank comparison, experiment 3 is closest to the sampled real graph. In Label Propagation comparison, experiments 3 and 5 have

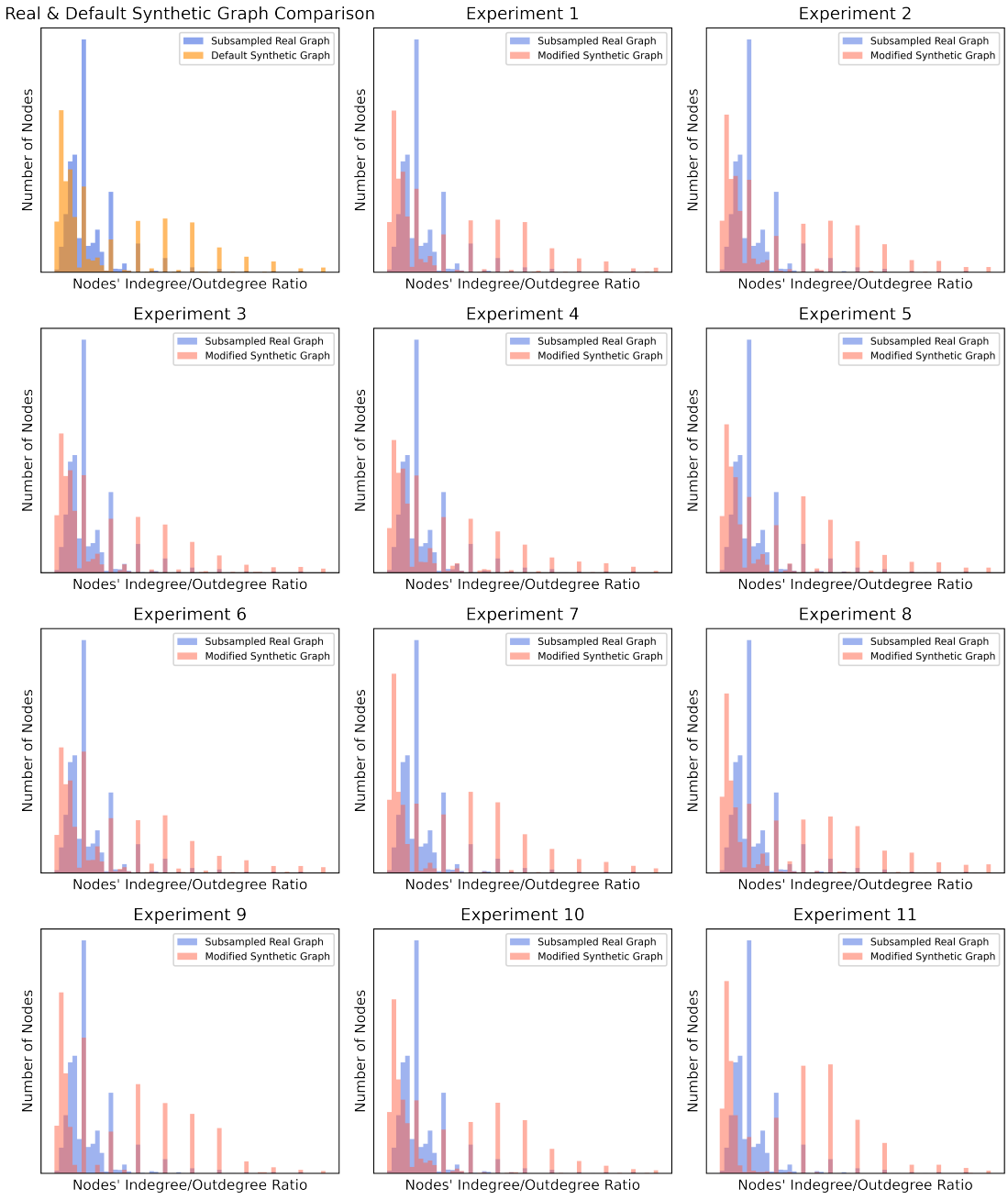

Figure 8. In-degree/Out-degree experiments for degree.csv

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 3	509 Distinct Pages (Nodes)
	Experiment 4	487 Distinct Pages (Nodes)
	Experiment 5	489 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 3	4089 Distinct Labels
	Experiment 4	4094 Distinct Labels
	Experiment 5	4089 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 3. degree.csv experiments with several algorithms

the same result and are closest to the sampled real graph. Since experiment 3 had good results in both comparisons, we chose experiment 3's parameters as our final decision in degree.csv experiments.

In comparison to the default synthetic dataset, experiment 3 had lower In-degree/Out-degree ratios and more nodes with less In-degree/Out-degree ratios in degree.csv.

4.3 Experiments with General Parameters

The conf.json file in AMLSim contains general parameters of the behavior of generated accounts. Experiments are conducted by picking a hyperparameter and creating new synthetic datasets by choosing a lower or higher value and then comparing it with a real graph. First experiments have the lowest value in their hyperparameter experiment group, and final experiments always have the highest value in their experiment cases. For example, experiment 1 in parameter Alpha is the lowest value in all experiments belonging to parameter Alpha. Experiment 7, the last experiment in parameter Alpha, has the highest value in all parameter Alpha experiments. In all experiments, hyperparameter values increase along with their experiment numbers. In each parameter experiment, every experiment number also belongs to its parameter's number in the belonging parameter type. For example, parameter Alpha's experiment 1's parameter value is mentioned as α_1. The goal of these experiments is to find the best general parameter combinations that lead to generating the most optimal realistic synthetic dataset. Due to working on this project with another bank, we are prohibited from sharing our chosen hyperparameters and their values openly and with details. Therefore, instead of directly

disclosing parameter names, we will use mathematical symbols when explaining them.

Parameter Alpha (α)

From Figure 9, the modified synthetic graphs from experiments 1, 4, and 7 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because their degree distribution is closer to the left, like how it is in the real graph. Their bump also moved more to the left. After looking at the histograms closely, we saw that experiment 1 is less realistic than others because experiment 4 and 7's fourth bar is closer to the real graph. Therefore, we have two options left: experiments 4 and 7.

Since they had similar results, we could not decide which one to choose, so we looked at the results of other comparison algorithms.

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 4	477 Distinct Pages (Nodes)
	Experiment 7	506 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 4	4100 Distinct Labels
	Experiment 7	4076 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 4. Parameter α experiments with several algorithms

According to the test results in Table 4, in PageRank comparison, experiment 7 is closest to the sampled real graph. In Label Propagation comparison, experiment 7 is the closest to the sampled real graph. Since experiment 7 had good results in both comparisons, we chose experiment 7: α_7 parameter value as our final decision in parameter alpha experiments.

Compared to the default synthetic dataset, experiment 7 had a higher value for parameter α.

Parameter Beta (β)

From Figure 10, the modified synthetic graph from experiment 1 resembles the sub-sampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose experiment 1 because it has a smooth ratio decrease towards the right, without ups

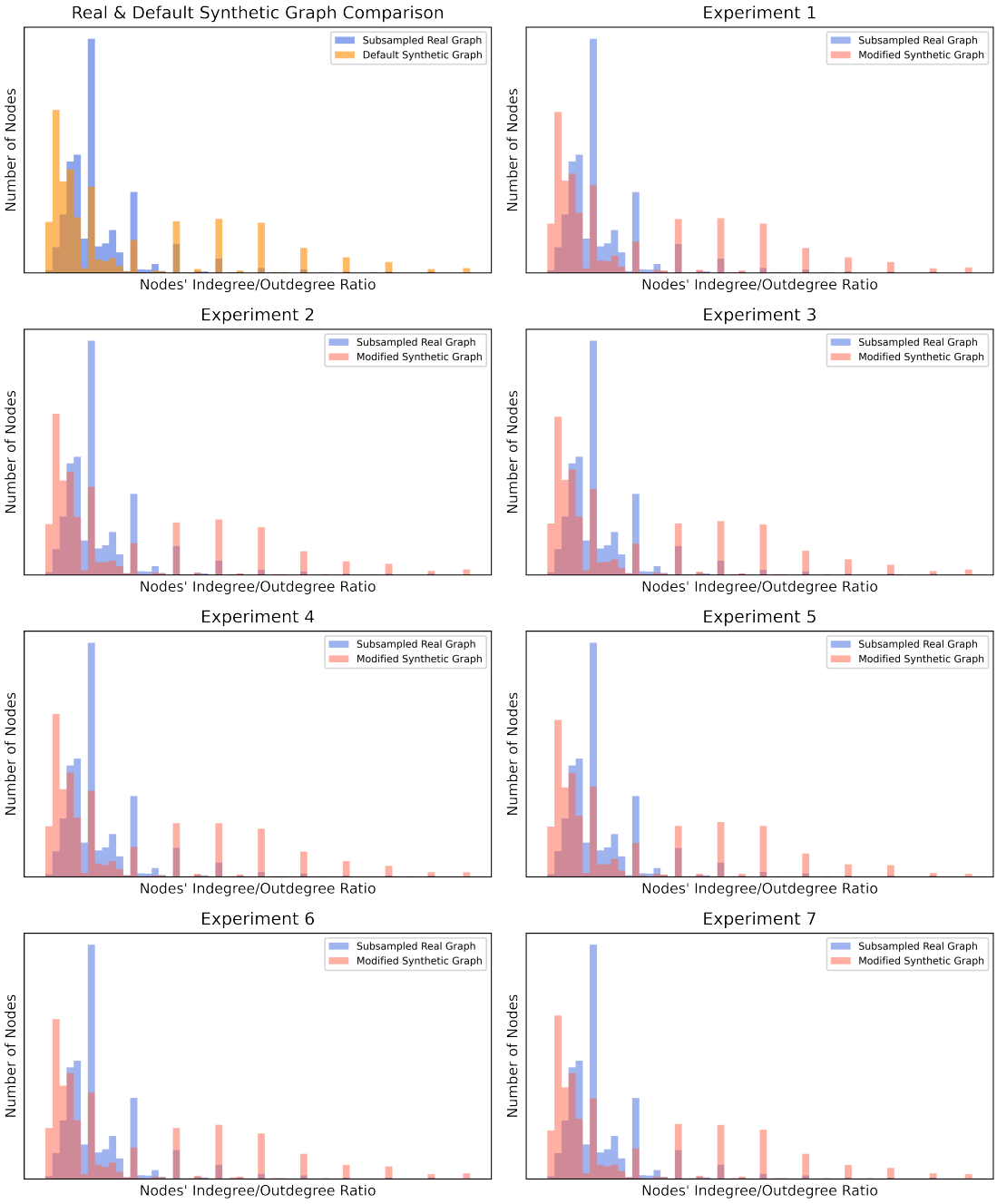

Figure 9. In-degree/Out-degree experiments with parameter α

and downs in bars. Its bump is also more to the left, unlike other experiments. Since experiment 1: β_1 parameter value is a clear winner, we didn't have to look at the results

of other comparison algorithms.

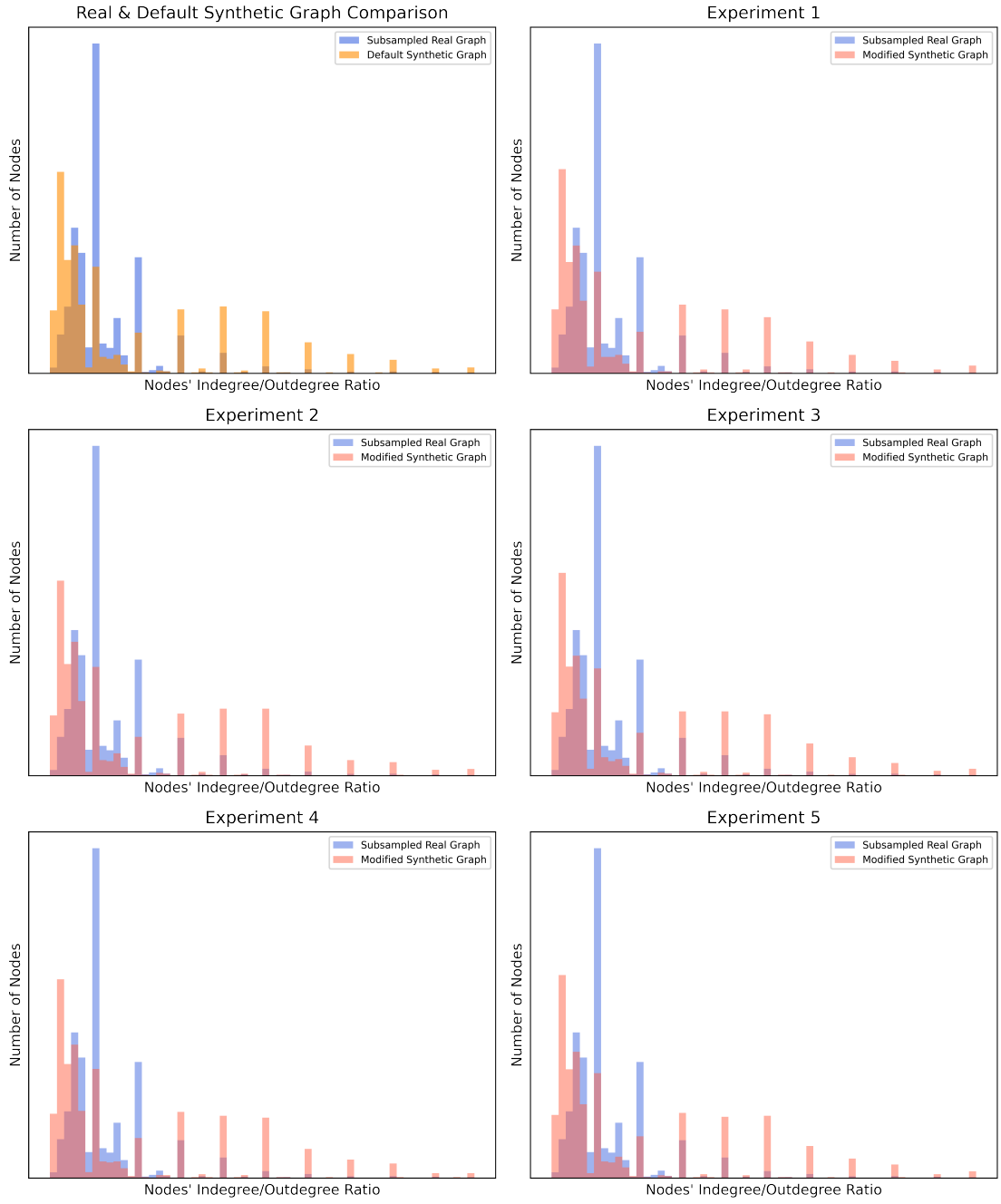

Figure 10. In-degree/Out-degree experiments with parameter β

Compared to the default synthetic dataset, experiment 1 had a lower value for parameter

β.

Parameter Delta (δ)

From Figure 11, the modified synthetic graphs from experiments 1 and 5 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because they have a smooth ratio decrease towards the right, without ups and downs in bars, like how it is in the real graph.

Since they had similar results, we could not decide which one to choose, so we looked at the results of other comparison algorithms.

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 1	491 Distinct Pages (Nodes)
	Experiment 5	489 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 4	4083 Distinct Labels
	Experiment 7	4089 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 5. Parameter δ experiments with several algorithms

According to the test results in Table 5, in PageRank comparison, experiment 1 is the closest to the sampled real graph. In Label Propagation comparison, experiment 1 is also the closest to the sampled real graph. Since experiment 1 had good results in both comparisons, we chose experiment 1: δ_1 parameter value as our final decision in parameter δ experiments.

Compared to the default synthetic dataset, experiment 1 had a lower value for parameter δ.

Parameter Epsilon (ϵ)

From Figure 12, the modified synthetic graphs from experiments 1, 2 and 5 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because they have a smooth ratio decrease towards the right, without ups and downs in bars, like how it is in the real graph.

Since they had similar results, we could not decide which one to choose, so we looked at the results of other comparison algorithms.

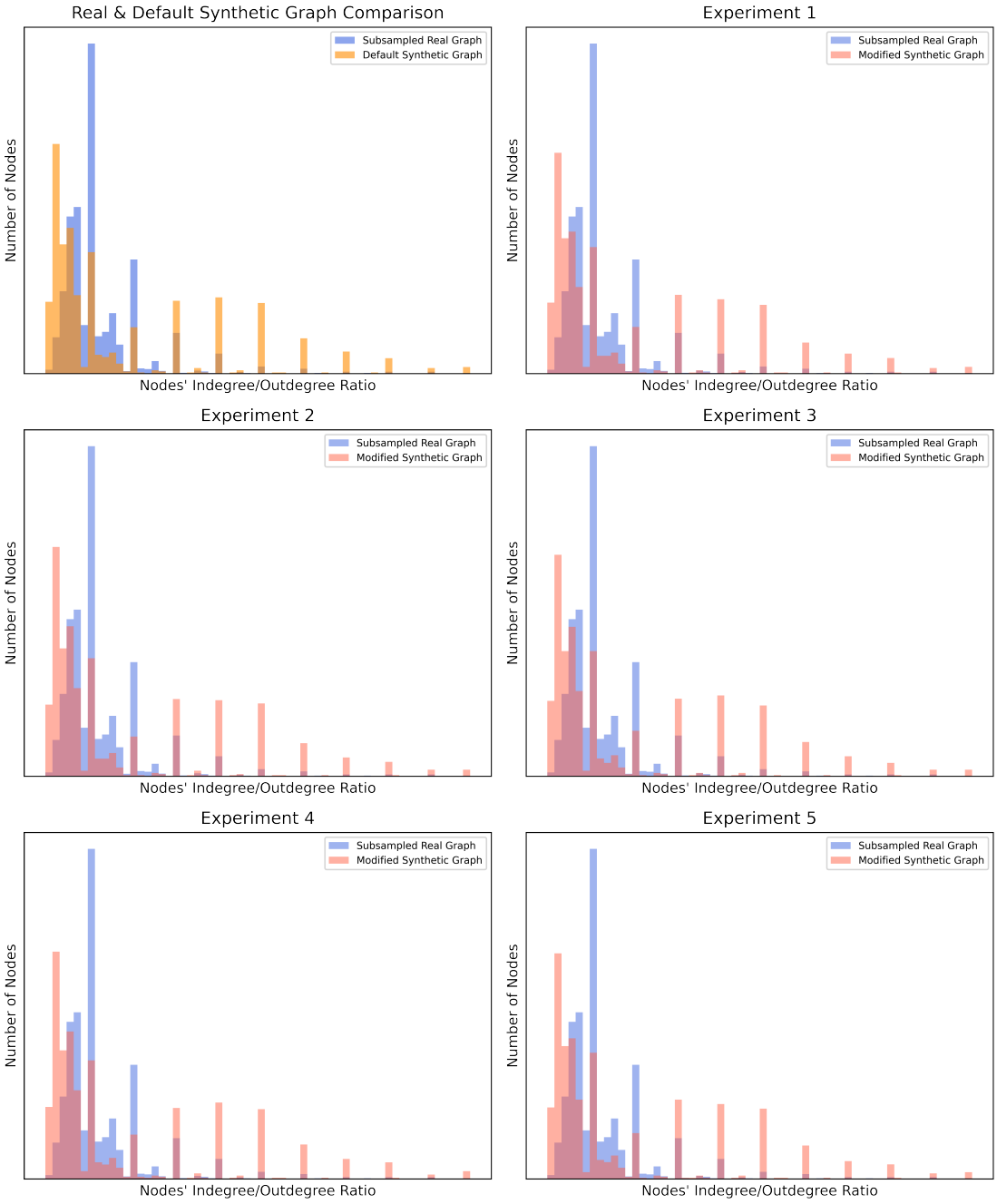

Figure 11. In-degree/Out-degree experiments with parameter δ

According to the test results in Table 4, in PageRank comparison, experiment 2 is the closest to the sampled real graph. In Label Propagation comparison, experiment 2 is

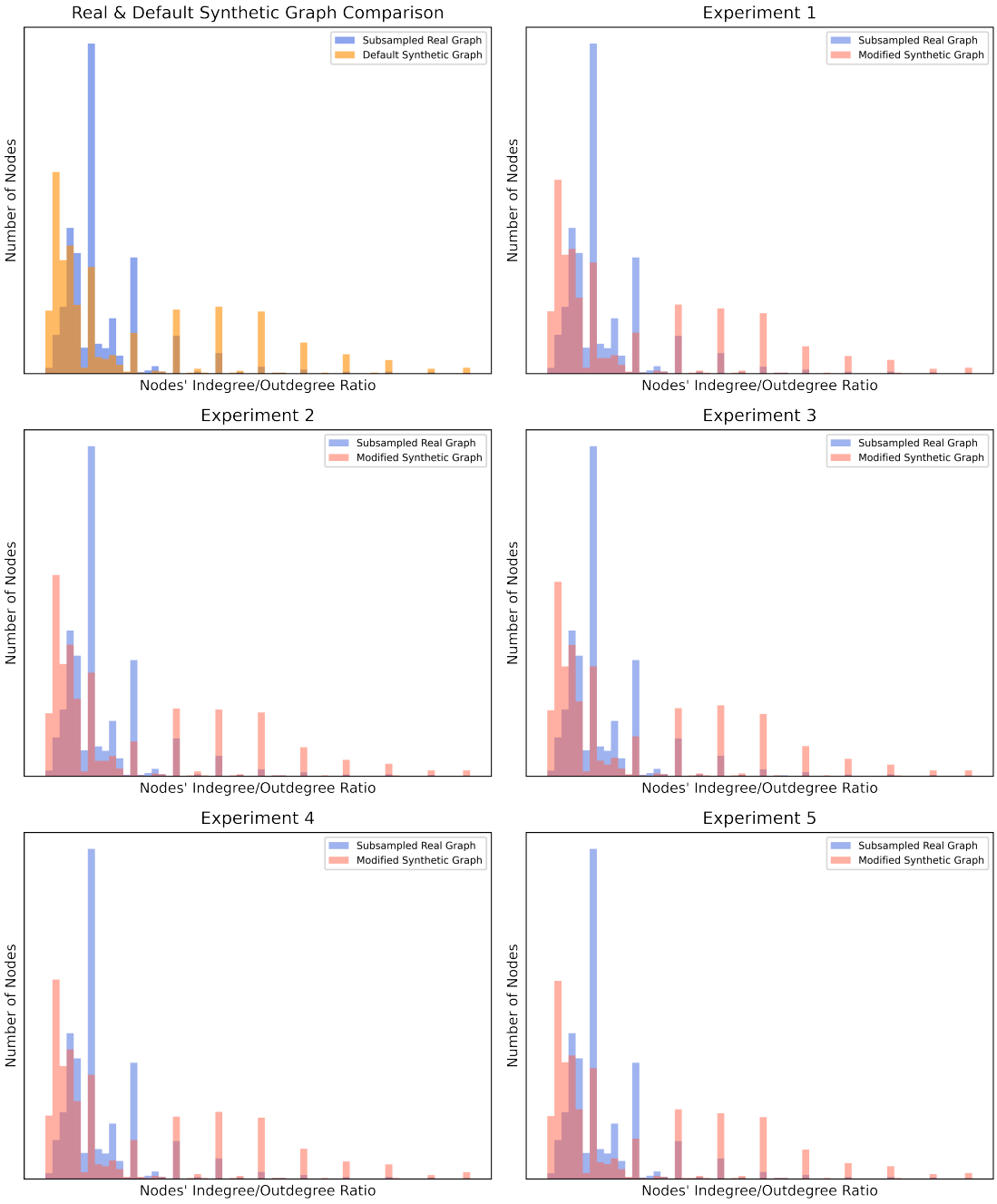

Figure 12. In-degree/Out-degree experiments with parameter ϵ

also the closest to the sampled real graph. Since experiment 2 had good results in both comparisons, we chose experiment 2: ϵ_2 parameter value as our final decision in

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 1	491 Distinct Pages (Nodes)
	Experiment 2	493 Distinct Pages (Nodes)
	Experiment 5	488 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 1	4086 Distinct Labels
	Experiment 2	4079 Distinct Labels
	Experiment 5	4090 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 6. Parameter ϵ experiments with several algorithms

parameter ϵ experiments.

Compared to the default synthetic dataset, experiment 2 had a higher value for parameter ϵ.

Parameter Lambda (λ)

From Figure 13, the modified synthetic graphs from experiments 4 and 5 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because they have a smooth ratio decrease towards the right, without ups and downs in bars, like how it is in the real graph.

Since they had similar results, we could not decide which one to choose, so we looked at the results of other comparison algorithms.

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 4	480 Distinct Pages (Nodes)
	Experiment 5	464 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 4	4037 Distinct Labels
	Experiment 5	4055 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 7. Parameter λ experiments with several algorithms

According to the test results in Table 7, in PageRank comparison, experiment 4 is the closest to the sampled real graph. In Label Propagation comparison, experiment 4 is also the closest to the sampled real graph. Since experiment 4 had good results in

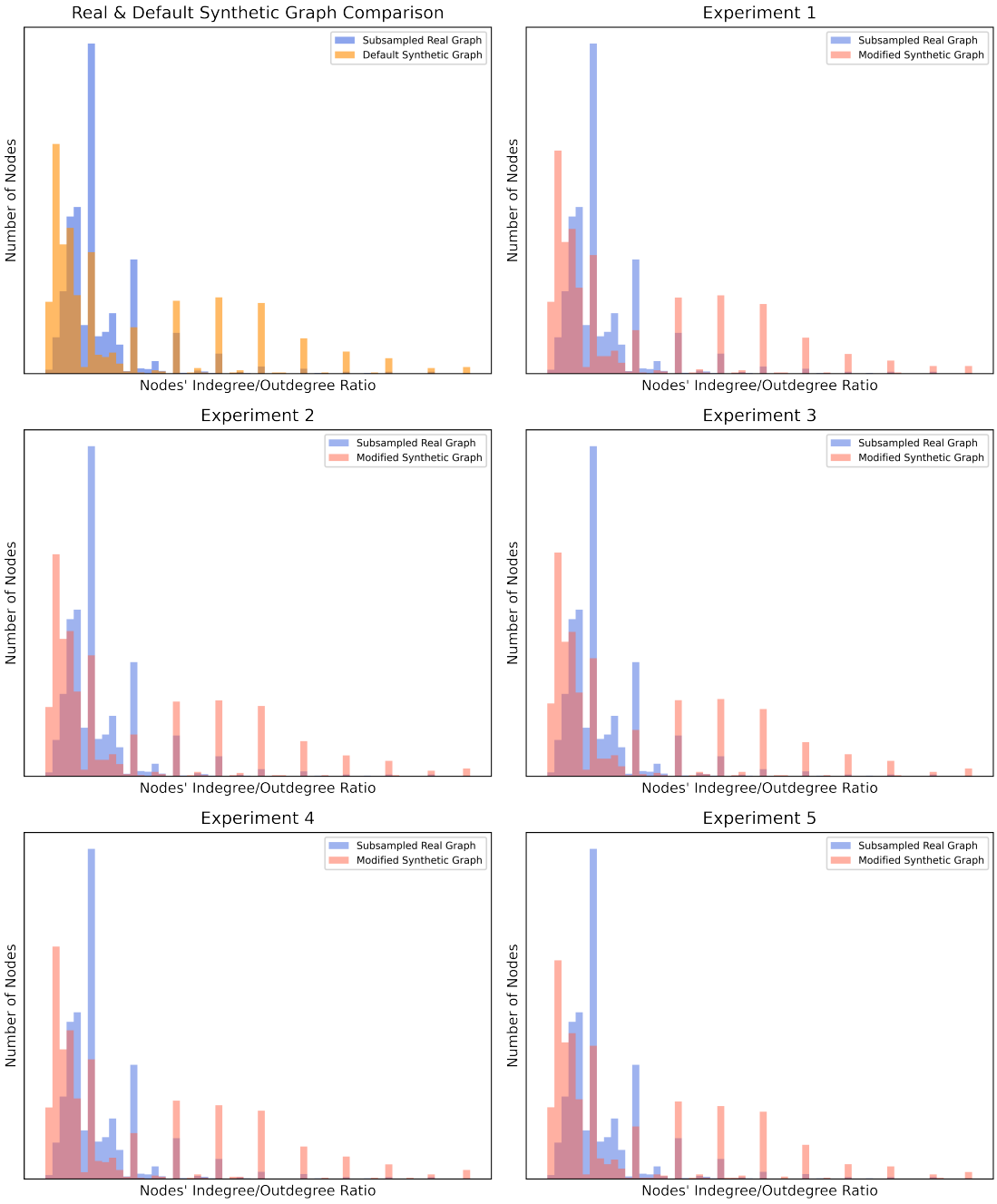

Figure 13. In-degree/Out-degree experiments with parameter λ

both comparisons, we chose experiment 4: λ_4 parameter value as our final decision in parameter λ experiments.

Compared to the default synthetic dataset, experiment 4 had a higher value for parameter λ.

Parameter Omega (ω)

From Figure 14, the modified synthetic graphs from experiments 1, 2 and 4 are the ones that resemble the subsampled real graph's pattern the most in the In-degree/Out-degree ratios histogram. We chose the mentioned experiments because they have a smooth ratio decrease towards the right, without ups and downs in bars, like how it is in the real graph. After looking at the histograms closely, we saw that experiment 2 is less realistic than others because experiment 1 and 4's fourth bar is closer to the real graph. Therefore, we have two options left: experiments 1 and 4.

Since they had similar results, we could not decide which one to choose, so we looked at the results of other comparison algorithms.

Comparison Method	Graphs to Compare	Scores
PageRank	Experiment 1	479 Distinct Pages (Nodes)
	Experiment 4	484 Distinct Pages (Nodes)
	Sampled real graph	3009 Distinct Pages (Nodes)
Label Propagation	Experiment 1	4080 Distinct Labels
	Experiment 4	4066 Distinct Labels
	Sampled real graph	1175 Distinct Labels

Table 8. Parameter ω experiments with several algorithms

According to the test results in Table 8, in PageRank comparison, experiment 4 is the closest to the sampled real graph. In Label Propagation comparison, experiment 4 is also the closest to the sampled real graph. Since experiment 4 had good results in both comparisons, we chose experiment 4: ω_4 parameter value as our final decision in parameter ω experiments.

Compared to the default synthetic dataset, experiment 4 had a higher value for parameter ω.

4.4 Final Results

In order to have the final synthetic dataset, we combined parameters of the best results from our degree.csv and general parameter experiments. In Figure 15, the first plot

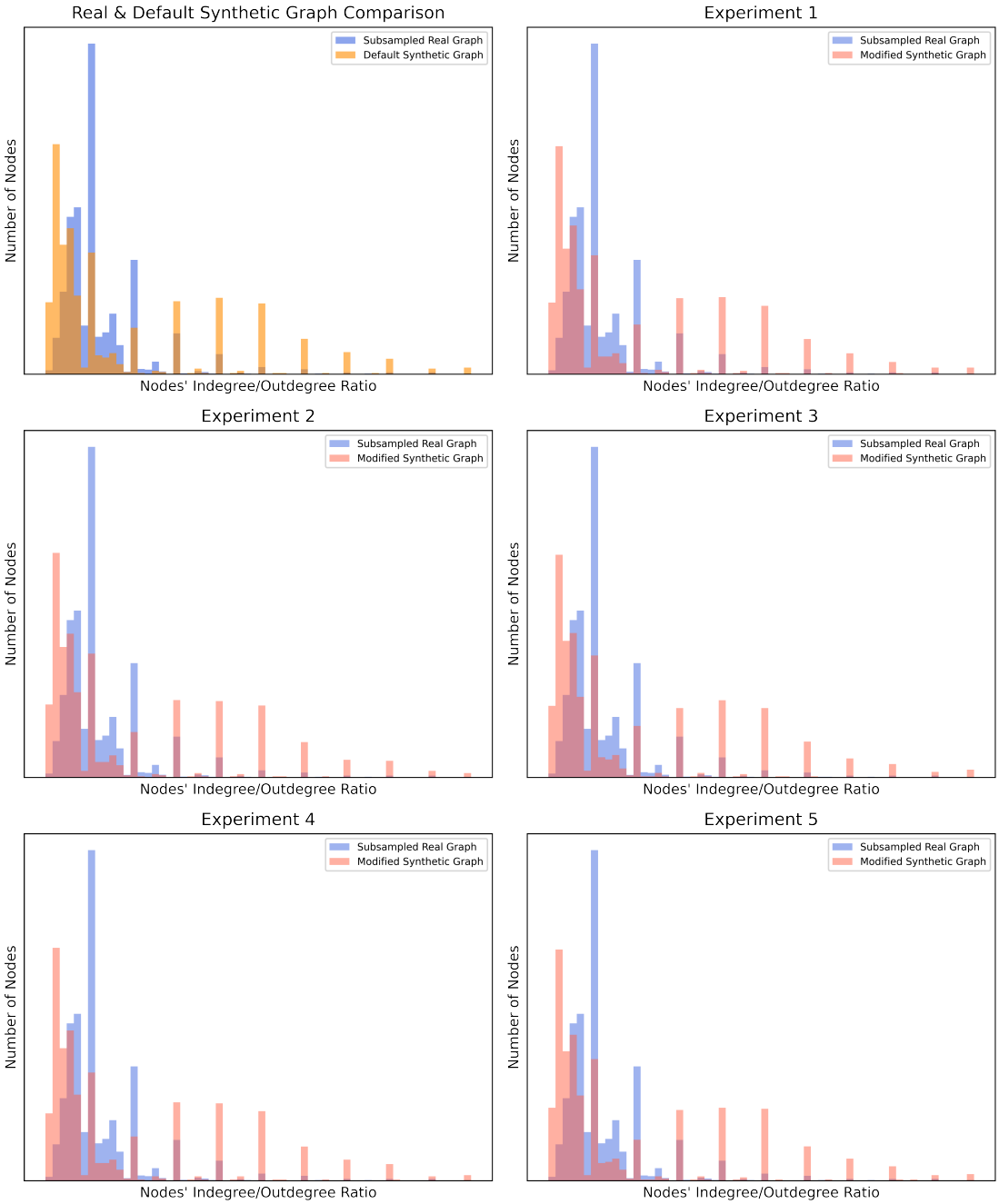

Figure 14. In-degree/Out-degree experiments with parameter ω

compares the real and default synthetic graphs. On the synthetic graph's histogram, it is visible that there is a bump in the middle, which does not exist in the real graph.

We had several experiments on hyperparameters, and our main goal was to eliminate this bump. The second plot on the top right shows the comparison between the real graph and the synthetic graph, which is modified only on its general parameters. The difference between the default synthetic graph and the synthetic graph with changed general parameters is not much. The reason is that the general parameters do not affect the graph too much without degree.csv changes. The third plot on the bottom left shows the comparison between the real graph and the synthetic graph that has a modified degree.csv but has default general parameters. It is visible that degree.csv is the most effective method to make AMLSim graphs more realistic. The bump is almost gone, the degree ratios decrease smoothly to the right, and it resembles the ratio pattern of the real graph. However, the last plot on the bottom right shows that combining degree.csv and general parameters gives the best results to generate the most realistic graph. The final synthetic graph looks similar to the synthetic graph with modified degree.csv and default general parameters, but the ratios of the degrees are lower and, therefore, closer to the real graph.

As a result, we got a synthetic graph that resembles the real graph's pattern and degree ratios. We could not get a synthetic graph identical to the real one. This is because AMLSim is not able to generate precise graphs. These results show the most realistic synthetic graph we could get. This is good for us since generating an identical synthetic graph would be dangerous. Because it could give important information about our data, creating a risk of reverse engineering.

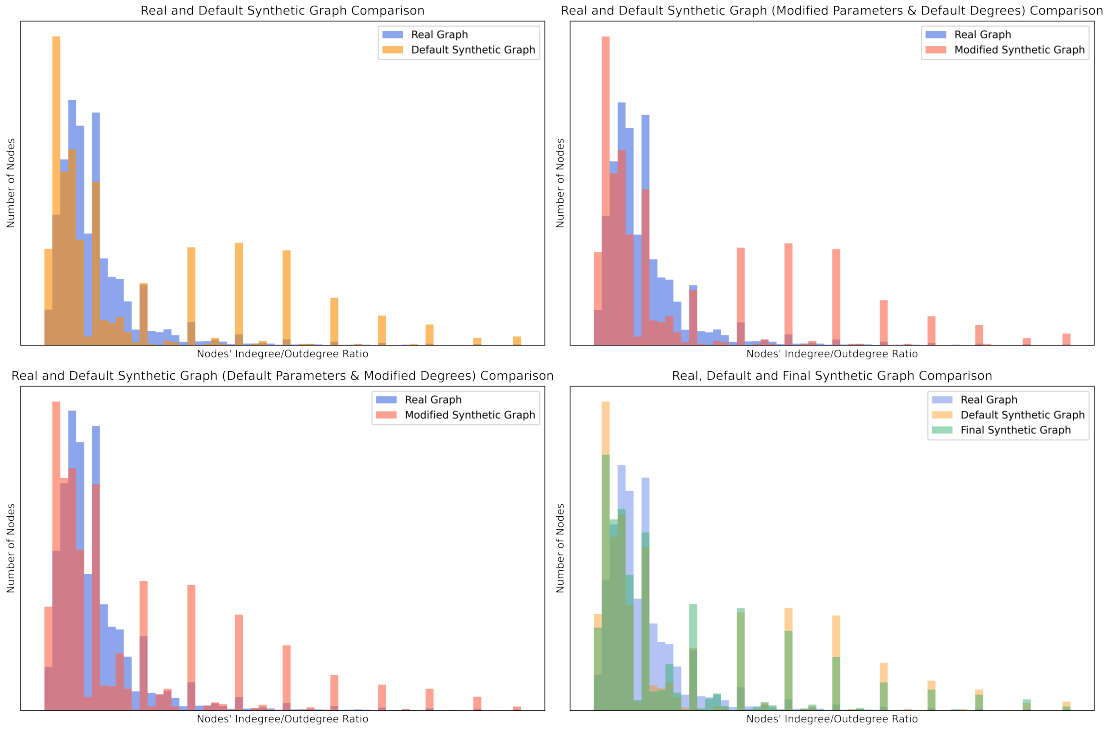

Figure 15. Results of hyperparameter tuning

5 Discussion

This chapter will cover the conclusions of this thesis and explain the limitations that affected the result of the work.

5.1 Context and Outcome

This thesis investigated methods to produce a synthetic graph with a similar structure to real transaction data for FL needs. We conducted graph creation for synthetic and real datasets, paying attention to the fact that they have similar features. Then, we implemented Snowball sampling to sample our real graph, make computationally expensive comparison algorithms work faster with our big graphs, and make the real graph's node count equal to make their comparison meaningful. Next, we used In-degree/Out-degree Ratio, PageRank, and Label Propagation methods to compare graphs and performed experiments on AMLSim's hyperparameters by comparing generated synthetic graphs and a real graph generated from Swedbank's transaction data. Then, we made changes to hyperparameters accordingly in order to create more realistic synthetic data.

The results showed that the degree.csv file plays a crucial role in generating realistic datasets in AMLSim compared to its general parameters. Combining our best results from general parameters and degree.csv experiments, we generated the most optimal synthetic dataset that resembles the real graph's structure. When we first started doing experiments, there was a bump in histograms of synthetic graphs that did not occur in real graphs. This bump was the most unrealistic part of the generated dataset, which we achieved by eliminating and moving to the left and tightening it to give it a similar distribution to our real graph.

5.2 Limitations

Generating synthetic data identical to our real transaction data was impossible with AMLSim because AMLSim does not have the capability to generate precise datasets. The result of the work is based on the most realistic dataset we have achieved by using AMLSim and experimenting with its hyperparameters.

6 Conclusion and Future Work

6.1 Conclusion

In this thesis, the author collaborated with Swedbank and contributed to a project to generate a realistic synthetic dataset to enable Federated Learning between banks. At first, the author found a method to subsample a graph created from Swedbank's transaction data. Next, the method was optimized to make it scale well with Swedbank's big graphs, and subsamples were created. In addition, several methods are added to compare graphs. To ensure that the subsampling method is working well, the author conducted experiments to ensure that subsampling creates subgraphs similar to each other using implemented comparison methods. Then, synthetic data was filtered in order to make it suitable for comparison with the real graph. After creating a graph with the filtered synthetic dataset, the author worked on hyperparameter tuning to find the optimal AMLSim hyperparameters to create a realistic synthetic graph.

The results are that the subsampling method works as it should. Generated subsamples are close to each other in structure, and the method is persistent in creating similar subsamples. Hyperparameter tuning was successful. We experimented on different hyperparameters by changing their values and comparing them with a subsampled real graph. After finding the most optimal hyperparameter values and combining them, we created a synthetic graph that represents the pattern of the real graph well. As a result, we achieved our goal in this thesis work and generated a realistic synthetic graph. Our final synthetic graph resembles our real graph, and the FL project is still progressing. However, these results are not the final work in the project to generate a realistic dataset, and the realistic synthetic graph generation work is still ongoing.

6.2 Future Work

In future work, different paths can be taken. If the work is done on an individual, AMLSim can be customized and improved to generate more realistic datasets. It is possible to modify the simulation's agent's behavior and the general structure of how the simulation works. If the work is done with another company, like how we worked, research on comparison methods can be considered to improve comparison and get more

insights about the graphs. Furthermore, the hyperparameter tuning can be automatized to make the process faster. It also may make the tuning more successful, as precise parameter values can be found after experiments to create more realistic datasets. However, one should pay attention to data leakage while automatizing tuning.

References

[1] Graphframes graph algorithms. https://graphframes.github.io/graphframes/docs/_site/user-guide.html#graph-algorithms.

[2] Nesreen K Ahmed, Nick Duffield, Jennifer Neville, and Ramana Kompella. Graph sample and hold: A framework for big-graph analytics. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1446–1455, 2014.

[3] Nesreen K Ahmed, Jennifer Neville, and Ramana Kompella. Network sampling: From static to streaming graphs. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 8(2):1–56, 2013.

[4] Samuel A Assefa, Danial Dervovic, Mahmoud Mahfouz, Robert E Tillman, Prashant Reddy, and Manuela Veloso. Generating synthetic data in finance: opportunities, challenges and pitfalls. In *Proceedings of the First ACM International Conference on AI in Finance*, pages 1–8, 2020.

[5] Tomisin Awosika, Raj Mani Shukla, and Bernardi Pranggono. Transparency and privacy: the role of explainable ai and federated learning in financial fraud detection. *IEEE Access*, 2024.

[6] David B Blumenthal. New techniques for graph edit distance computation. *arXiv preprint arXiv:1908.00265*, 2019.

[7] Carl Boettiger. An introduction to docker for reproducible research. *ACM SIGOPS Operating Systems Review*, 49(1):71–79, 2015.

[8] John Clark and Derek Allan Holton. *A first look at graph theory*. World Scientific, 1991.

[9] Ankur Dave, Alekh Jindal, Li Erran Li, Reynold Xin, Joseph Gonzalez, and Matei Zaharia. Graphframes: an integrated api for mixing graph and relational queries. In *Proceedings of the fourth international workshop on graph data management experiences and systems*, pages 1–8, 2016.

[10] Marco Galbiati and Kimmo Soramäki. An agent-based model of payment systems. *Journal of Economic Dynamics and Control*, 35(6):859–875, 2011.

[11] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[12] Vishesh Karwa, Sofya Raskhodnikova, Adam Smith, and Grigory Yaroslavtsev. Private analysis of graph structure. *Proceedings of the VLDB Endowment*, 4(11):1146–1157, 2011.

[13] Jure Leskovec and Christos Faloutsos. Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636, 2006.

[14] Jure Leskovec and Christos Faloutsos. Scalable modeling of real graphs using kronecker multiplication. In *Proceedings of the 24th international conference on Machine learning*, pages 497–504, 2007.

[15] Seung-Hwan Lim, Sangkeun Lee, Sarah S Powers, Mallikarjun Shankar, and Neena Imam. Survey of approaches to generate realistic synthetic graphs. 2016.

[16] Edgar Lopez-Rojas, Ahmad Elmir, and Stefan Axelsson. Paysim: A financial mobile money simulator for fraud detection. In *28th European Modeling and Simulation Symposium, EMSS, Larnaca*, pages 249–255. Dime University of Genoa, 2016.

[17] Edgar Alonso Lopez-Rojas and Stefan Axelsson. Using the retsim fraud simulation tool to set thresholds for triage of retail fraud. In *Nordic Conference on Secure IT Systems*, pages 156–171. Springer, 2015.

[18] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pages 1273–1282. PMLR, 2017.

[19] Salvatore Menniti, Emanuele Castagna, and Tommaso Mazza. Estimating the global density of graphs by a sparseness index. *Applied Mathematics and Computation*, 224:346–357, 2013.

[20] Charlie Parker, Sam Scott, and Alistair Geddes. Snowball sampling. *SAGE research methods foundations*, 2019.

[21] Donovan Platt. A comparison of economic agent-based model calibration methods. *Journal of Economic Dynamics and Control*, 113:103859, 2020.

[22] Usha Nandini Raghavan, Réka Albert, and Soundar Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106, 2007.

[23] Bernhard Rieder. What is in pagerank? a historical and conceptual investigation of a recursive status index. *Computational Culture*, (2), 2012.

[24] Benedek Rozemberczki, Oliver Kiss, and Rik Sarkar. Little ball of fur: a python library for graph sampling. In *Proceedings of the 29th ACM international conference on information & knowledge management*, pages 3133–3140, 2020.

[25] Ravindra Singh, Naurang Singh Mangat, Ravindra Singh, and Naurang Singh Mangat. Stratified sampling. *Elements of survey sampling*, pages 102–144, 1996.

[26] HMHS Surendra and HS Mohan. A review of synthetic data generation methods for privacy preserving data publishing. *International Journal of Scientific & Technology Research*, 6(3):96–99, 2017.

[27] Mark Weber, Jie Chen, Toyotaro Suzumura, Aldo Pareja, Tengfei Ma, Hiroki Kanezashi, Tim Kaler, Charles E Leiserson, and Tao B Schardl. Scalable graph learning for anti-money laundering: A first look. *arXiv preprint arXiv:1812.00076*, pages 1–7, 2018.

[28] Jun Zhang. *Algorithms for synthetic data release under differential privacy*. PhD thesis, 2016.

[29] Bin Zhou, Jian Pei, and WoShun Luk. A brief survey on anonymization techniques for privacy preserving publishing of social network data. *ACM Sigkdd Explorations Newsletter*, 10(2):12–22, 2008.

[30] Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. *ProQuest number: information to all users*, 2002.

Appendix

I. Glossary

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Mert Bektas**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   An Approach for Generating Realistic Synthetic Transaction Data,
   *(*title of thesis)

   supervised by Alexander Jöhnemark, Jolanta Goldšteine and Amnir Hadachi.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Mert Bektas
15/05/2024