TU/e Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Algorithmics Research Cluster

Network Anomaly Detection Enhanced With Link Prediction Methods In the Parallel Computing Model

Master Thesis

Rafał Polak

Supervisors:
Your First Committee Member
Assistant Professor Morteza Monemizadeh
Your Third Committee Member, usually the external member

EMPTY version

Eindhoven, January 2024

Abstract

Modelling real-life phenomena as graphs results in extremely powerful models. Since graphs have been researched in mathematics for many centuries, representing real world as graphs opens a massive toolkit of theoretical ways to analyze those phenomena. One such tool, called anomaly detection, allows the user to algorithmically spot subgraphs of the input network which fall out of line with general aspects of the full graph. In the age of Big Data, the full graph is seldom small enough to be stored and processed locally. Instead, we will focus on graphs represented as edge streams, i.e. streams of data representing the edges, where each item can only be accessed once. Many use cases, such as DDoS detection or financial fraud, require almost immediate action upon the edge stream to minimize the malicious impact of such potential anomaly. With the field of edge stream anomaly detection being widely researched lately, we would like reinforce the state-of-the-art techniques using link prediction. We seek to find out how we can bridge these two topics and mathematically formulate such a connection.

Contents

Contents **v**

List of Figures **vii**

List of Tables **ix**

Listings **xi**

1 Introduction **1**

2 Related work **5**

3 Motivation **9**

4 Methodology and Decomposition **11**

5 Timeline and risks **13**

6 Related work (deepening) **15**

7 Current results **19**

8 Data and code handling **21**

9 Other **23**

9.0.1 Knowledge utilisation/valorization, expected contributions and impact . . . 23

Bibliography **25**

List of Figures

1.1 A biclique of X and Y (with the help of an example by V.L.Nguyen.) 2

1.2 A typical local link prediction method will only require to process information about purple and red nodes in order to compute score(x_7, x_2). 3

1.3 Streaming model visualization. Source: own creation 4

4.1 Top-down predicted approach. The unsure step (parallelization) in red. 12

7.1 Score distribution (density plot) over the [0.2M : 0.9M] row subset of NY Taxi Data (January 2018) [21]. Plotted using Seaborn plotting library for Python [29]. Density's numerical instability is to blame for the seemingly negative results. . . . 20

List of Tables

5.1 The overall plan includes the effort to parallelize the architecture, which might prove outside our reach. However, for now, we would like to remain *dreaming big*, hence it is included 13

6.1 Comparison overview of the most crucial design points for various literature graph Anomaly Detection approaches. For the mathematical notation briefing, see the relevant papers 16

Listings

Chapter 1

Introduction

Graph modelling

Graph theory as a branch of mathematics is assumed to have been born in 1736 in the mind of Leonhard Euler. [20] claims that Euler's paper on the Seven Bridges of Königsberg is regarded as the first paper in the history of graph theory. Since then, its possibilities grew seemingly endlessly. Nowadays, graphs (we will use the name *networks* interchangeably) can express dozens of models, like Bayesian Networks [12] or Markov Networks [12]. Petri Nets [25] span another class of models that their central idea works in a graph fashion. They tweak the underlying assumptions of a graph slightly, allowing edges between more than a pair of nodes. One can observe, however, that such an edge can be translated into a unique type of node, resulting in a bipartite graph, so that the whole model is still describable by a network.

Graph setting in our case

A regular static graph can be described as a pair of two sets:

- V, the set of *vertices* (also known as *nodes*), which usually model entities, like IP addresses, individuals or events
- E, the set of *edges* (also known as *links*), which usually model entities' connections, like correspondence, collaboration or participation. Typically, an edge is represented as a pair of vertices between which, the connection occurs

We will use a slightly altered description to accommodate for the streaming model and augment the edge definition. For us, each edge e will become a quadruplet of values, i.e.:

$$e = (u, v, w, t), \text{ where}$$

- u is a vertex in V
- v is another vertex in V (in '*simple*' graphs, we restrict $u \neq v$)
- w is the *weight* of an edge (for non-weighted graphs, we can set all weights to 1)
- t is a timestamp of the edge creation. Can be continuous or discrete with certain granularity

To understand the rest of the proposal, two specific graphs need to be mentioned.

Biclique (BC) A clique is a fully-connected graph. In the undirected graph example, it means that:

$$\forall_{u,v}[(u, v) \in E]$$

A biclique works in a very similar fashion. Imagine grouping the nodes into two disjoint groups. Then, a biclique is a graph where each vertex from the first group is connected to all vertices of the second group (and vice versa). Mathematically:

$$\forall_u[u \in V_1 : \forall_v[v \in V_2 : (u,v) \in E]]$$

and

$$\forall_u[u \in V_2 : \forall_v[v \in V_1 : (u,v) \in E]]$$

See an example of a BC in Figure 1.1.

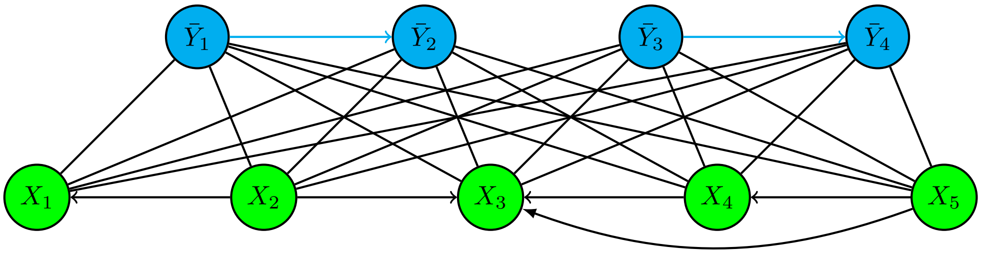

Figure 1.1: A biclique of X and Y (with the help of an example by V.L.Nguyen.)

Erdős–Rényi (ER) model Named after two Hungarian mathematicians, Paul Erdős and Alfréd Rényi, this is a random *model* of a graph. That means, it is essentially a procedure to build graphs according to some principles. Perhaps, the simplest of procedures. Its simplest form works by declaring a set of vertices V and a parameter p - the probability of an edge occurring. Then, for each possible edge, and there are $n = \binom{|V|}{2}$ possibilities, we just "toss a coin of probability of success p" and if the success occurs, add that edge to the graph. The ER model, in expectation, should show no special shape, features or structure.

Link Prediction To put simply, link prediction is a task of predicting the structure of a graph G, given its incomplete version G^o. The incomplete graph G^o could be obtained in numerous ways and one must always consider their dataset as potentially erroneous. Take a graph dataset modelling chemical reactions (where vertices model the substances and edges model their reactivity) as an example. An edge might be missed due to large amount of time needed to observe some reaction, or as a result of an experimenter's error. For a graph modelling people's intimate relations, if the data is based on a survey, the subjects (vertices) may refuse to confess to some relations (edges) which will lead to incomplete data.

So, given a graph

$$G^o(V, E^o) \subseteq G(V, E) \wedge E^o \subseteq E$$

we aim to recover the full $G(V,E)$. This is usually done by assigning each possible edge $(u,v) \notin E^o$ a score, determining how likely this edge is to exist in E. This score can be seen as probability, but frequently will not be numerically bounded to the [0,1] interval. Later on, these scores determine our assumed missing edges. That can be done in various ways, such as:

- by adding top-k highest-scoring edges to $G(V,E)$, for some pre-defined constant k
- by using a threshold classification $(u,v) \notin G^o \wedge \text{score}(u,v) \geq \beta \Rightarrow (u,v) \in G$
- (in the case of a weighted graph) by adding to $G(V,E)$ all scored edges with weights corresponding to their prediction scores.

A more advanced version of link prediction also deals with *spurious* edges, i.e. edges in E^o that do not appear in E and their sampling into G^o was a *false positive*. We will not be dealing with this task in this study and instead just focus in missing edges search.

Link prediction methods can be divided into two main fuzzily-defined categories:

- **Local methods** which evaluate the scores by looking at each possible edge one-by-one, usually computing some statistic about the direct neighbourhood of the vertices inspected. In the example in Figure 1.2, a typical local method would only need to process structural information about nodes in red to quantify the linkage score between the nodes $x7$ and $x2$ given in purple. Any information regarding nodes in blue could be neglected.

- **Global methods** which evaluate the scores based on the entire adjacency matrix of the graph. Due to usual high sparsity of adjacency matrices, these method are almost always significantly more costly. We can use the *Leicht-Holme-Newmann 2 index* as an example [13].

$$R = 2|E^o|\lambda_1 D^{-1}(I - \frac{\alpha}{\lambda_1}A)^{-1}D^{-1}$$

  To use it, one has to compute a diagonal matrix D of all vertex degrees, which is computationally plausible. But then, not only it one to compute λ_1, the largest eigenvalue of the adjacency matrix A, but also invert this middle term, which is incredibly expensive. The resulting matrix R gives the scores for all possible edges. I is an identity matrix and α some constant parameter.

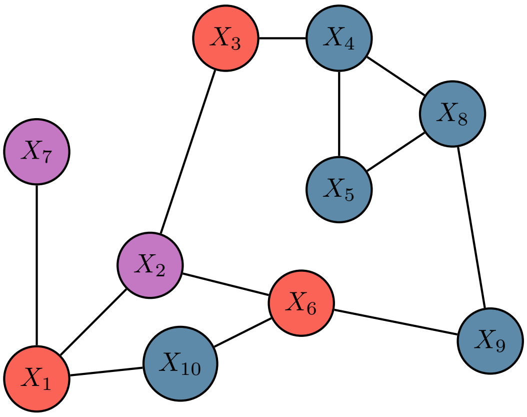

Figure 1.2: A typical local link prediction method will only require to process information about purple and red nodes in order to compute score(x_7, x_2).

Induced graph An induced graph $G_i(V_i, E_i) \subseteq G(V, E)$ is a graph which for a subset of vertices $V_i \subseteq V$ contains all the edges in E that at least one vertex of belongs to V_i.

Anomaly example

Following [15], we can say that "*Anomalies, also referred to outliers, imply something that deviates from what is standard, normal, or expected*". A mathematical definition of an anomaly is strictly case-specific. If the data being modelled is packages exchanged in the IP layer of a computer network, a newly-emerging node that suddenly starts flooding the network with a multitude of edges will typically be an anomaly. If the data being modelled is some social media communication (e.g. tweets at X), then any socially big enough event, like football (soccer) world cup final, will be spotted as an anomaly.

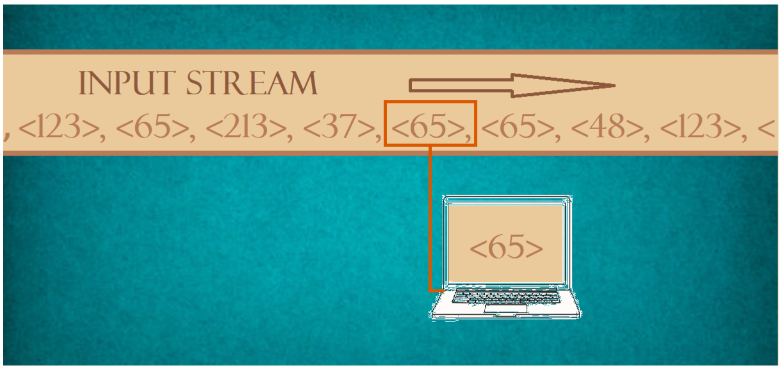

Figure 1.3: Streaming model visualization. Source: own creation

Different approaches to our problem use different precise definition of an anomaly, such that it can be computed. Some studies declare it to be any snapshot of the data that the sum of edge weights within it falls beyond some critical level [27]. These critical levels are either the sum being β times larger than the sum coming on the previous snapshot; or, equivalently, smaller than the $\frac{1}{\beta}$ times the sum within the previous snapshot. Other mathematical formulation suggests that an anomaly exists for a subgraph G if $|dist(G, ER) - dist(G, BC)| > \epsilon$ for some distance function $dist$ [5]. What ER and BC are was mentioned in Section 1.

Streaming model

For most algorithmic applications, one can just assume the input it stored somewhere that is accessible to the processor and operations on it can go on uninterrupted. To give a simple example, in the almighty sorting problem, most basic approaches just consider an array of numbers that simply rests in memory and can be read and written to with no visible overhead. A slightly more complicated model involves the so-called *I/Os*. It assumes the input data being stored in an external storage, e.g. a hard disk, while any operations on the data can only take place in so-called *main memory*. To perform those operations, data has to be fetched from the hard disk into the main memory in an *I/O operation* which is considered slow, what introduces a substantial overhead to the algorithm. The trick is to construct algorithms in a way that minimizes the number of I/Os needed (bearing in mind the structured way data is stored in the hard disk). However, this model also assumes the input data is available to the processor - however costly - at any time.

The streaming model is different. It can be conceptualized as if the data was flowing through a network, or a tube, item by item. The processor can monitor this link, or tube, but is incapable of storing the entire dataset. Hence, in the plain (sometimes called *vanilla*) version of this model, each data item (*token*) can only be seen - processed - once. This is a very harsh requirement and streaming algorithms typically cannot answer most queries on the data precisely, but only approximate them. They do, however, find use in Big Data and applications where the query's answer needs to be found immediately in real-time. Figure 1.3 should help conceptualize this model of computation.

Chapter 2

Related work

In this section, we will introduce the basic ideas present in related literature to get a basic view of what the solution to our problem can look like.

What we expect are not anomalies [15] Our work picks the research up from where it was left in *The Friendships That We Expect Are Not Anomalies*. Its authors were seemingly the first to bridge the gap between seeking an anomaly (something that severely violates its expectations) and actually using the expectation in the process. The expectation is quantified as the value of a link prediction method that decreases the anomaly score if the prediction mechanism claims there is a high chance of the given edge occurring. This allowed them to leverage the static anomaly detection methods.

The specific details on how this link prediction was incorporated into the anomaly scoring remain a crucial point of improvement over this study. In our opinion, while this idea was genius, its detailing was faulty and its results unconvincing. Their incorporation of link prediction into the methods seemingly paid no mind to *robustness* - the aspect of versatility or generalizability of a solution. The overall approach was claimed to improve the state-of-the-art in some cases, while having a slight negative impact in others. Moreover, despite presenting 5 datasets in the report, the experimental results shown only used 2 out of these 5 datasets. We restrain from wondering what was the reason for not presenting the findings over all datasets.

We aim to rectify those problems while reusing the general idea of this very study.

MIDAS [27] *The Friendships That We Expect Are Not Anomalies*[15] uses what it considers the state-of-the-art technique to the anomaly detection problem, called *MIDAS*. *MIDAS* is based on a 'time-tick' principle, where the continuous time intervals between edge arrivals are discretized into 'ticks'. The assumption following is that the mean level (number or summed weight) of edges during each time tick will stay relatively the same. If, in any time tick, the level of edges between any subgraph greatly exceeds our expectations, such edge (and all the following edges in that subgraph until the end of the time tick) will be declared anomalies. However, to account for 'natural' fluctuations in distribution of edges in specific subgraph between time ticks, the level of edges of a 'permanent' graph is also used in the prediction. The 'permanent' graph stores an encoding of the history that does not get reset after any time tick.

SpotLight [5] But what exactly is the encoding? As mentioned in the abstract, the graph object formed by the input edge stream will typically be way too large to store and effectively process. The simplest version of solution to this problem was proposed in a technique called *SpotLight*. *SpotLight* partitions the vertex set randomly into a predefined number of potential subgraphs. Then, the edges' level is being tracked only in these few separate subgraphs, without any holistic view. An anomaly is declared if it is being found in any of those partitions. Such an approach is called a *dense subgraph search* and the presence of dense subgraphs is a typical clue or even a definition of an anomaly to be checked for.

Count-Min Sketch [6] One might notice how to above partitioning method may be cumbersome to use when the vertex set is not known beforehand. A solution to that, assuming each vertex having some intrinsic integer identifier (ID) to it, could be to define a mapping from this ID to the partition number, e.g. have 4 partitions, each for IDs that give a specific solution to ID *mod* 7. This is the basic idea of *Count-Min Sketches* - a very powerful synopsis (data summary object) proposed in 2003. A *Count-Min Sketches* (or CMS) is an extension of a hash table. We assume the reader is familiar with the concept of a hash function. Such a hash table simply stores a set of pairs of data of the form $(ID, value)$ where $value$ is the value that needs to be hashed and ID is a key that determines the output of the hash function. For a hash function h, the hash table T performs

$$T[h(ID)] \leftarrow T[h(ID)] + value$$

For single values, we can simply produce the pairs by setting the pair to $(value, value)$, or count the number of occurrences of ID by using pairs of type $(ID, 1)$

Now, to extract the number of occurrences of ID, simply check $T[h(ID)]$. However, it can happen that for $ID_1 \neq ID_2$, we still have $h(ID_1) = h(ID_2)$. Therefore, what we find in $T[h(ID_1)]$ will actually be the summed count of both ID_1 and ID_2. This is what we call a *collision*. And this is where we get to the difference between a CMS and a simple hash table. Namely, a CMS consists of multiple has tables with different hash functions for each of them. The same $(ID, value)$ pair gets hashed to all the tables with the hopes that if $h_1(ID_1) = h_1(ID_2)$, then $h_2(ID_1) \neq h_2(ID_2)$. Since we can never underestimate the count for $T[ID_1]$, but only overestimate it due to these collisions, the CMS estimation of such a count is given by the minimum of all tables' values corresponding to the given key - hence the name Count-**Min**.

The Count-Min Sketch will be useful for storing subgraphs of the data stream using constant memory size. Just as in *SpotLight* we were interested in keeping separate structures for a number of random subgraphs of the graph input, we can now use a CMS to do so. For each incoming edge, we can simply use its vertices as the ID and its weight (if the graph is unweighted, set each weight to 1) as $value$. But since each edge happens between two vertices, won't we need two keys?

Improved CMS A very recent scientific paper [28] has proposed an improved data summary object which can resolve that issue. It simply adds a dimension to the Count-Min Sketch by hashing the ingoing vertex and the outgoing vertex separately, while maintaining the multiple hashes in the third dimension. Under the assumption of a perfect hashing (which can be closely enough obtained in practice), such a sketch (or equivalently, *synopsis*) should hold more-or-less the same value in all cells and submatrices across itself. If this is not the case and there are some significantly denser regions in this 3-dimensional matrix, then such a dense region may point to a dense subgraph of the original dataset, hence, revealing an anomaly. The mathematical details regarding all the aforementioned techniques are omitted here for clarity.

Neural Network techniques Oh course, with the power that neural networks have recently brought us, it bears no surprise that they have also been employed to solve the problem in question, i.e. graph anomaly detection (in edge streams). A recent survey [11] lists those approaches - coming from 2019 up to 2022 - and divides them into four self-explainable categories:

1. **AND:** Anomalous node detection
2. **AED:** Anomalous edge detection
3. **ASD:** Anomalous subgraph detection
4. **GLAD:** Graph-level anomaly detection

Due to working with edge streams, our approach falls primarily into the AED category. For this category, the survey [11] lists 3 works of research. 2 of them are tailored to work in relatively specific dataset types: detecting fraud in an e-commerce platform. All of them are unsupervised learning methods, despite those supervised ones prevailing in the AND problem. In one of them, called *eFraudCom* [30], representative normal data is being sampled and the differences between those samples and the graph in question is being investigated.

Chapter 3

Motivation

The anomaly detection problem itself is as popular among the (algorithmic) scientific community as it gets. There are not many applications that would **not** benefit from being able to procedurally spot irregularities, outliers or questionable subsets in the data they are working on; and trying to list possible use cases would be Herculean task on its own. Taking a step forward, due to immense power that graph modelling offers (explained in Section 1), it is fairly straightforward that a mix of these two fields must be, and is, relatively well-researched and have countless applications. The next step in narrowing the scope down is looking purely at graphs coming from the streaming model. Within that scope, enumerating concrete use cases finally becomes feasible. And the use case that we would like to address the most, partially due to personal reasons, is the *DDoS* prevention.

Distributed Denial of Service (DDoS) The DDoS - an improvement over Denial of Service (DoS) - is an incredibly common and plain type of a network attack in computer science. It capitalizes on the fact that a server in a computer network, once queried, will always try to answer that query by design. There is no way to tell apart which queries are benevolent and legitimately seek answers, and which queries are malicious and only aimed at exhausting the server's resources. A server, typically having access to high computational power and bandwidth, will be able to handle thousands of queries per second, but if the requests come densely enough, it will not be able to handle them all, struggling to answer ALL queries - legitimate and malicious ones. This is how a DoS work. But since typically a PC can't 'flood' the server alone, such attack originates from a large set of machines that flood the server by combined effort, hence the 'Distributed' in the name.

A recent survey [4] reveals shocking numbers for the popularity of the DDoS attack in early 2020's. According to this survey, in the first half of 2022 alone, 'experts' identified 6,019,888 different attacks [4][19]. While large web services are hard to attack due to their employed countermeasures, it is still possible to do so. The New Zealand stock exchange has experienced a DDoS attack for two days [4][2]. In 2022, Microsoft Corporation reported that it was the target of a DDoS attack that, in one minute, reached the level of 3.47 terabits per second (*sic*!) [4] (the internal reference is no longer available). Large web-based services may have proprietary ways to deal DDoS attacks, but many smaller providers are not protected due to lack of awareness or costs associated with implementing a DDoS protection system.

Since a graph anomaly detector in a graph streaming scenario can effectively protect against DDoS attacks, we have chosen this to be the main motivation behind our work. Note that the biggest issue with anomaly-based mechanisms is the high rate of false positives, as an anomaly does not always represent an ongoing attack [4][7].

Chapter 4

Methodology and Decomposition

The following flowchart, at Figure 4.1, summarizes briefly the main decomposition of process that currently is expected to be achieved.

Parallelization in practice Despite the bold name, the parallel computing model is just an idea for the project that is not central to its design. The project is meant to work using regular, central computing and then, if the time constraints permit, be scaled up to a parallel computing model. Such a design would be immensely helpful for the cases where the data flow is decentralized. This can happen either when the data control center is decentralized (e.g. in the case of a large financial institution which cannot afford to bear a single point of failure) or simple the data generation process is decentralized, e.g. in so-called 'smart buildings'. Scaling up to compute the anomaly detection with link prediction will require smart distribution of the data over several machines in a way that each of them is still able to compute some meaningful characteristic of its data subset. This requirement forces us to use the **local methods** of link prediction only, as each peripheral machine would not have the access to the entire adjacency matrix of the dataset. If this was the case, the entire streaming model would not be considered for the task at all. Most likely, each machine will need to receive the subset of the data corresponding to an induced graph on a subset of nodes. This way, each machine would have sufficient information about a subset of nodes to compute the link prediction scores in their neighbourhood. However, scores between vertices of different subsets (belonging to different machines) could not be found, unless communication between machines is allowed. Since this idea is not central to the project, we leave the details to be figured out later.

Notice how due to constant space, the sketches about the entire dataset need not use any parallelization, only the link prediction part.

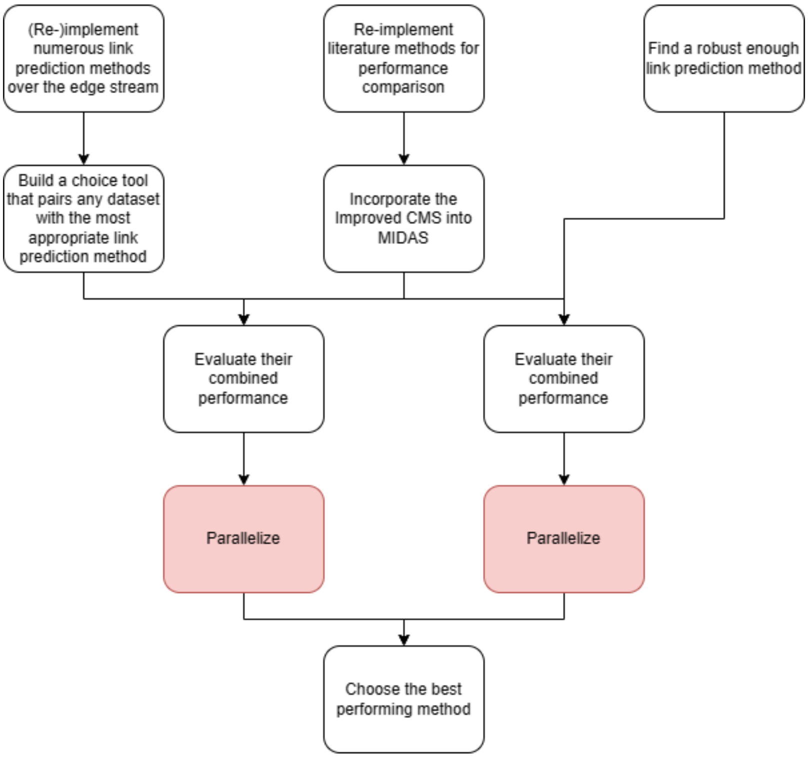
(Re-)implement numerous link prediction methods over the edge stream
Re-implement literature methods for performance comparison
Find a robust enough link prediction method
Build a choice tool that pairs any dataset with the most appropriate link prediction method
Incorporate the Improved CMS into MIDAS
Evaluate their combined performance
Evaluate their combined performance
Parallelize
Parallelize
Choose the best performing method

Figure 4.1: Top-down predicted approach. The unsure step (parallelization) in red.

Chapter 5

Timeline and risks

Timeline See Table 5.1 for an approximate planning schedule for the project.

Week	Description	Expected Result	Deliverable
1–2	Deciding on the topic	Project field	None
3–6	Literature exploration	Method general idea	Final project name
7	Literature analysis	Method more precise idea	Preliminary code
8	Finalizing the draft project proposal	Finished draft	Draft in PDF
9–10	Re-implementing literature methods	Getting at least 3 of them working	Preliminary results and graphs
11–12	Experimenting with literature methods	Runtimes and accuracies at few datasets	Partial final results and graphs
13–21	Own solution implementation	Working (non-parallel) implementation	Code uploaded
22–24	Experiments conduct on the same datasets as above	Runtimes and accuracies comparable with above	Final graphs and tables
25	Expanding the experiments	Runtimes and accuracies on more datasets	Even more infographics, perhaps more advanced visuals
26–29	Parallelization	Fully-operational parallel implementaition	Spark code
30	Parallel experimenting	Parallel runtimes and accuracies	Infographics about the results
31	Thesis main writing (substantial part will be being written in parallel to coding)	Thesis draft obtained	Draft to be reviewed
32	Thesis polishing	Increasing clarity and presence	Complete thesis
33	Defence	Obtaining our MSc	Final thesis

Table 5.1: The overall plan includes the effort to parallelize the architecture, which might prove outside our reach. However, for now, we would like to remain *dreaming big*, hence it is included

”It doesn’t work” Perhaps the largest pitfall any research of this kind may fall into could be plainly explained by ”*it doesn’t work*”. To put it very simply, if one enhances the state-of-the-art method that achieves a 92% accuracy on some test data and their enhancement achieves 90%

accuracy - the goal was not achieved. Unless, of course, the new method is significantly faster in computation or relies on weaker assumptions etc. Since our research is focused on doing just that - reinforcing the current best-performing methods, the risk of "*not working*" is looming there. And it seems to be the largest motor fueling the haste in implementing our solution - the earlier we realize it potentially not working, the more time there is to try to tweak the method or come up with an alternative solution. This is why the basic version of the method has already been coded and could almost be used to process the same datasets as the original methods, in order to compare the performances.

Data-specific prediction Link prediction is generally a data-specific method, even though there exist methods that are said to work well without any domain-specific tweaking [14]. Still, using link prediction for a generalized framework raises serious questions about the possible robustness of it, which can greatly vary depending on the specific method used. To mitigate this issue, we strive to either employ the most robust prediction technique possible, or possible automate the process of choice of appropriate technique. The network could be probed for some key features indicating which link prediction method to use and then this specific technique would be used. This would be another possible mitigation of the robustness problem.

Strict assumptions Multiple literature pieces analyzed assume their input coming from a very rigid distribution. See those assumptions in [**?**]. This further hinders the robustness of the method, which would ideally work under weaker assumptions. The good thing is, the main thing that 'breaks' when such assumptions are violated, are the theoretical guarantees that approaches like [27] use. The method itself will most likely continue to work properly in most cases. An assumption-testing step can be merged into the initial step of data processing, to check whether the theoretical guarantees are still valid.

Chapter 6

Related work (deepening)

Table 6.1 summarizes the aforementioned relevant literature in terms of various repeating aspects of the methods. Note that many terms used might be vague without reading the whole paper. The table serves as a medium of comparison so that the best parts of each of the works can be utilized in our technique.

Aspect	**MIDAS**[27]	**The Friendships...**[15]	**SpotLight**[5]	**August**[28]
Anomaly intuition	Microcluster	Microcluster	Swiftly-emerging dense subgraph	Dense subgraph in the sketch
Anomaly definition	Microcluster if: $\frac{c(e,(n+1)T)}{c(e,nT)} > \beta$ or $\frac{c(e,(n+1)T)}{c(e,nT)} < \frac{1}{\beta}$ For an edge: $score(u,v,t) = (a_{uv} - \frac{s_{uv}}{t})^2 \frac{t^2}{s_{uv}(t-1)}$	MIDAS score divided by the link prediction score	*SL-Distance* defined as $E\|\|v(G_1) - v(G_2)\|\|_2^2$. Anomaly if $SLdist(G + ER) - SLdist(G + BC) > \epsilon$ The definition of v function is vague	No explicit condition mentioned, just an algorithm
Dealing with	Edge streams	Edge streams	Graph streams (snapshots)	Edge streams or graph streams
Assumptions on input distribution	Mean edge level is the current time tick = mean edge level in the previous time tick	Same as MIDAS	In each time tick, we shall receive an ER-shaped ("non focus-aware") addition to the graph	Density of the graph remains uniform after hashing

Points made	For MIDAS-F version, evade poisoning by conditional merge. Do not encode an edge into the CMSes if it is declared an anomaly	Fixes the weakness of MIDAS which is its lack of spatial relations	-	Procedurally seeks a dense submatrix around a newcoming edge's hash cell. The hash functions for in- and out- vertices are equal to preserve a sense of connectivity
Sketch used	Two CMSes - one for the current time tick and one ”global”. In the MIDAS-R version, the ”global” sketch is decaying	Same as MIDAS	Total edge weight in any sampled subgraph. Many subgraphs monitored	Two higher-dimensional CMSes (one decaying and one global)
Pros	The scoring function comes from the solution of a χ^2 distribution with 1 degree of freedom	Uses so-called ”online” and ”offline” approach. ”Online” scores edges on the spot. ”Offline” seeks the proper parameters using a training and test set split	Has theoretical bounds on false positive probability	Seemingly pretty robust
Cons	If the underlying data distribution has changed, MIDAS is working against it. The MIDAS-F's conditional merging preserves errors	Only deals with local link prediction methods. Explains not their application on a sketch or a snapshot. Does not take robustnes into account	Since it's working with snapshots, a wrong time tick setting can miss an anomaly	The dense submatrix search method is very complicated and complex

Table 6.1: Comparison overview of the most crucial design points for various literature graph Anomaly Detection approaches. For the mathematical notation briefing, see the relevant papers

Looking back at the identified risk arising from predictions that may be too data-specific (Chapter 5, paragraph *Data-specific predictions*), we would like to elaborate how this might not be an insurmountable problem after all. The problem, it seems to be, is that using a link prediction method that properly exploits a given input dataset's features, is artificially strengthening the relations in this dataset. But our take is that such an approach is not problematic *per se*; it can be commonly seen in various Data Scientific approaches.

For example, the data generation problem works under such condition. Solutions to the data creation problem seek to create as 'fresh' data as possible, from a given set of primitive instances. In 1999, even a genetic algorithm for new test cases generation was proposed [23]. It combines algorithmic *recombination* with algorithmic *mutation*, where the first one essentially creates plain variations of pre-existing instances and the latter solely tweaks individual test cases' numerical values one at a time. Given the genetic nature of this algorithm, it purely works by artificially strengthening the relations of the data and yet, remains a common type of solution in practice.

Such 'tweaking' of original input is a common argument point for various art-related Artifical Intelligence (AI) projects - that they are not capable of bringing any originality to their outcomes, and instead just explore the realm of interpolations between input data. In [16], we read that there is a substantial difference in how an AI model and a human child replicates the patterns found in the world around them, as the AI model exists in "*an artificial world, far removed from the heterogeneity, diversity and noiseless of the real world that a child is exposed to*". It is no surprise that "*computer that learned patterns from a training set can generate new artifacts with the same patterns.*" [16], while breaking out of those '*meta-patterns*' is beyond their reach. Despite that, AI art projects have gotten a huge traction and public attention.

Those two examples further hint that relying on strengthening already-present features of the data, can yield satisfactory results - perhaps in numerous fields of research. Therefore, we are convinced to accept it as plausible in our solution as well.

Chapter 7

Current results

So far, we cannot provide any outstanding results, as the project has just begun (or, considering that it is currently the *preparation phase*, has not yet begun). But to prove that some - strongly preliminary - results were obtained, i.e. the basic code is running error-free, MIDAS [27] over the Improved Count-Min Sketch [28] was implemented and its scores gathered. Specifically, its scores over the *NYC Taxi Data* [21] in January 2018 were gathered, with the time interval set to 1 day. Due to the outstanding size of the dataset (2.9 million rows) and our PC's incapability of single-processing it, a subset of 0.7-million-rows subset, specifically [0.2M : 0.9M], was chosen to be scored. Since this subset involves a large number of rows with t=1 (1^{st} of January) and all its corresponding scores are 0 by design, a large peak around $score = 0$ can be seen.

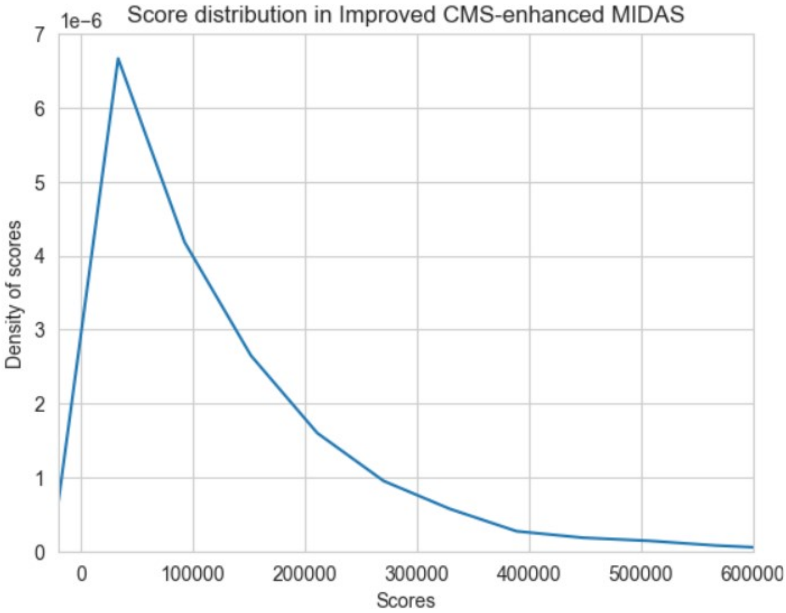

Figure 7.1: Score distribution (density plot) over the [0.2M : 0.9M] row subset of NY Taxi Data (January 2018) [21]. Plotted using Seaborn plotting library for Python [29]. Density's numerical instability is to blame for the seemingly negative results.

Chapter 8

Data and code handling

Datasets Our study - so far - is not aiming for any crucial data creation. Since we are dealing with a problem in a relatively well-researched field, it is of utmost importance to compare our solution on the same dataset that other algorithms were tested on. Only in this way we can truly see if we manage to outperform those. The desired subset of datasets that the work would be going to be judged on is:

- CIC-DDoS2019 [10]
- DARPA [24]
- CIC-IDS2018 [9]
- CTU-13 [26]
- UNSW-NB15 [18]
- NYC-taxi-2018 [21]
- ISCX-IDS2012 [1]

With more datasets potentially to follow. The subset aforementioned is created by a union operator of the sets of datasets used in scientific articles which were most critical in developing this paper. In other words, it encapsulates all datasets across the papers that we have already analyzed and drew ideas and conclusions from. This list is most likely to get filled with additional datasets, depending on the future of the project. For example, if parallelization of the method successfully gets incorporated into the project, then we would need some massive datasets to truly utilize and showcase the power of parallel computing.

There are no unethical implications of using any of the aforementioned datasets, to the best of our knowledge.

No new datasets are planned to be created as a result of this project.

Code The implementation written for the project will be self-made, with the exception of numerous code libraries (packages) that can be considered *standard*. Some notable examples include NumPy [8], Pandas [22] and tqdm [3]. For the rest, the code will be fully developed from scratch. Crucially, no implementations of the methods from literature will be used. This is to mitigate the performance differences arising from implementation quality and - most importantly - programming language used.

The full code for this project will be hosted publicly under MIT licence, imposing little to no restrictions to its users. The users will be allows to use, copy, modify, merge publish, distribute, sublicense and sell copies of the software [17].

For clarity and ease of reproduction of the work, there will be a README.txt file, alongside the others .py and .ipynb files hosted publicly, probably either as a GitHub/GitLab repository, or simply as a .zip file at Google Drive.

Chapter 9

Other

9.0.1 Knowledge utilisation/valorization, expected contributions and impact

To start with, we are fairly confident in doing research as a valuable and ethical deed even with neutral or unknown outcome for its utilization. That being said, the positivity of a more accurate anomaly detection system is purely in the hands of the user. To the best of our knowledge, the cutting-edge recent improvements in graph anomaly detection mechanisms have not yet been implemented into industry, mainly due to one key reason: it's too fresh. One of the most influential works in writing our thesis [28] was released in August 2023. Numerous other influential findings were presented - similarly - only dozens of months ago. It makes it hard to properly assess the subfield's contribution into academia, industry and society. We can theorize, though. The input into academia seems the most straightforward - if the method proves successful, it can turn the eyes of other researchers into its direction, resulting in even more robust and accurate systems. If performance turns out to be improved as a result of our research, then usefulness in industry is guaranteed. Maintaining the current applications with lower entry barrier and resources usage can help allocate the freed resources in businesses elsewhere, for the greater good of us all. For the social part, one outcome is very obvious and it deals with potentially unpleasant false positives. As any person who has been evacuated from a building due to a false fire alarm will tell you - such false positives get annoying, but better than remaining in a burning building. Lowering the number of false positives for our anomaly detection system (while maintaining the same share of true positives) would be a change uniformly perceived as good. For all the other concerns - especially regarding the employment of the system in some financial setting - it is up to the user to use the system ethically and to global benefit.

Bibliography

[1] M. Tavallaee A. Shiravi, H. Shiravi and A. A. Ghorbani. Toward developing a systematic approach to generate benchmark datasets for intrusion detection. *computers security*, 2012. 21

[2] https://www.bbc.com/news/53918580. 9

[3] Casper da Costa-Luis, Stephen Karl Larroque, Kyle Altendorf, Hadrien Mary, richardsheridan, Mikhail Korobov, Noam Yorav-Raphael, Ivan Ivanov, Marcel Bargull, Nishant Rodrigues, Guangshuo Chen, Antony Lee, Charles Newey, CrazyPython, JC, Martin Zugnoni, Matthew D. Pagel, mjstevens777, Mikhail Dektyarev, Alex Rothberg, Alexander Plavin, Fabian Dill, FichteFoll, Gregor Sturm, HeoHeo, Hugo van Kemenade, Jack McCracken, MapleCCC, Max Nordlund, and Mike Boyle. tqdm: A fast, Extensible Progress Bar for Python and CLI, 2023. 21

[4] Anderson Bergamini de Neira, Burak Kantarci, and Michele Nogueira. Distributed denial of service attack prediction: Challenges, open issues and opportunities. *Computer Networks*, 222:109553, 2023. 9

[5] Sudipto Guha Dhivya Eswaran, Christos Faloutsos and Nina Mishra. SpotLight: Detecting Anomalies in Streaming Graphs. *KDD '18*, 2018. 4, 5, 15

[6] S. Muthukrishnan Graham Cormode. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms 55*, 2005. 6

[7] Brij B Gupta and Amrita Dahiya. *Distributed Denial of Service (DDoS) Attacks: Classification, Attacks, Challenges and Countermeasures*. CRC press, 2021. 9

[8] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. 21

[9] Arash Habibi Lashkari Iman Sharafaldin and Ali A Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. *ICISSp 1 (2018), 108–116.*, 2018. 21

[10] Saqib Hakak Iman Sharafaldin, Arash Habibi Lashkari and Ali A Ghorbani. Developing realistic distributed denial of service (ddos) attack dataset and taxonomy. *In 2019 International Carnahan Conference on Security Technology (ICCST). IEEE, 1–8.*, 2019. 21

[11] Hwan Kim, Byung Suk Lee, Won-Yong Shin, and Sungsu Lim. Graph anomaly detection with graph neural networks: Current status and challenges. *IEEE Access*, 10:111820–111829, 2022. 6, 7

[12] Thomas Krak. Lecture notes in generative ai models, 2022. 1

[13] E. A. Leicht, Petter Holme, and M. E. J. Newman. Vertex similarity in networks. *Phys. Rev. E*, 73:026120, Feb 2006. 3

[14] Ryan N. Lichtenwalter, Jake T. Lussier, and Nitesh V. Chawla. New perspectives and methods in link prediction. In *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '10, page 243–252, New York, NY, USA, 2010. Association for Computing Machinery. 14

[15] Yao Luo and Morteza Monemizadeh. The Friendships That We Expect Are Not Anomalies. Master's thesis, Eindhoven University of Technology, 2018. 3, 5, 15

[16] Lev Manovich. Defining ai arts: Three proposals. *AI and dialog of cultures" exhibition catalog. Saint-Petersburg: Hermitage Museum*, 2019. 17

[17] https://pitt.libguides.com/openlicensing/MIT. 21

[18] Nour Moustafa and Jill Slay. Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). *In 2015 military communications and information systems conference (MilCIS). IEEE, 16.*, 2015. 21

[19] www.netscout.com/threatreport/global-highlights. 9

[20] Robin J. Wilson Norman Biggs, E. Keith Lloyd. *Graph Theory, 1736-1936.* Clarendon Press, 1986. 1

[21] http://www.nyc.gov/html/tlc/html/about/trip_record_data.shtml. vii, 19, 20, 21

[22] The pandas development team. pandas-dev/pandas: Pandas, February 2020. 21

[23] Roy P. Pargas, Mary Jean Harrold, and Robert R. Peck. Test-data generation using genetic algorithms. *Software Testing, Verification and Reliability*, 9(4):263–282, 1999. 17

[24] David J Fried Isaac Graf Kris R Kendall Seth E Webster Richard Lippmann, Robert K Cunningham and Marc A Zissman. Results of the darpa 1998 offline intrusion detection evaluation. *In Recent advances in intrusion detection, Vol. 99. 829–835*, 1999. 21

[25] Grzegorz Rozenberg and Joost Engelfriet. *Lecture Notes in Computer Science book series (LNCS,volume 1491).* Springer, Berlin, 2005. 1

[26] Jan Stiborek Sebastian Garcia, Martin Grill and Alejandro Zunino. An empirical comparison of botnet detection methods. *computers security 45 (2014), 100–123*, 2014. 21

[27] Bryan Hooi Minji Yoon Kijung Shin Siddharth Bhatia, Rui Liu and Christos Faloutsos. Real-Time Anomaly Detection in Edge Streams. *ACM Trans. Knowl. Discov. Data. 16*, 2022. 4, 5, 14, 15, 19

[28] Kenji Kawaguchi Neil Shah Philip S. Yu Siddharth Bhatia, Mohit Wadhwa and Bryan Hooi. Sketch-Based Anomaly Detection in Streaming Graphs. *KDD '23*, 2023. 6, 15, 19, 23

[29] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021. vii, 20

[30] Ge Zhang, Zhao Li, Jiaming Huang, Jia Wu, Chuan Zhou, Jian Yang, and Jianliang Gao. efraudcom: An e-commerce fraud detection system via competitive graph neural networks. *ACM Trans. Inf. Syst.*, 40(3), mar 2022. 7