Master Thesis
Computing Science

Radboud University

Using graph-based anomaly detection to uncover scientific fraud

Author:
ing. Wibren Wiersma

First supervisor/assessor:
dr. ir. Hugo Jonker

Second assessor:
dr. ir. Erik Poll

September 9, 2022

Summary

The body of scientific research is growing exponentially. To keep up, detection of scientific fraud must scale up as well. While manual human investigation and evaluation is crucial to determine whether specific cases constitute fraud, automation can help to uncover suspect cases. Tools to detect fraud are often targeted on detecting fraud in publications. But those responsible for the fraudulent publications are the authors. Like them, other actors in the publication process, such as reviewers, editors, and venues, are only occasionally the subject of fraud investigation. The goal of our proposed method is to reduce the workload of scientific fraud investigators by providing software-delivered indicators of suspicious behavior of actors in the publication process.

Our approach is based on the assumption that severe cases of scientific fraud will result in outliers compared to their non-fraudulent peers. We devise a method that searches for outlying actors within scientific biographic information. To enable this, we constructed a graph representation of scientific biographic information, including data of DBLP, OpenCitations, scraped editors from ACM journals, and scraped PC members of front matters of Springer LNCS conferences. In comparison with earlier similar efforts, we significantly improved the integration between these data sources.

Our idea is to find outlying actors by detecting outlying numbers of cycles in our graph. This idea is based on the assumption that when an actor commits fraud, it will eventually benefit the actor itself, so there must exist a cycle from and to the fraudulent actor. Westerbaan [Wes22] noticed that several methods of scientific fraud can be described as cycles in a graph. With the cycle detection, we had to face performance issues, since we used a large graph of around 15 million vertices and nearly 45 million edges. We faced this by proposing a more efficient method of cycle detection that fits the purpose of this thesis.

We detected cycles for more than 3 million persons with 6 or fewer edges. These cycles have been grouped by their type. Cycle types describe the pattern of a cycle they contain the node types and edge types included in it, for example, the self-citation cycle type is: $\texttt{Person} \xrightarrow{\texttt{author\_of}} \texttt{Article} \xrightarrow{\texttt{cites}} \texttt{Article} \xleftarrow{\texttt{author\_of}} \texttt{Person}$. Several known methods of scientific fraud can be encoded as cycle types. But, decoding cycle types to find unknown methods of scientific fraud is harder, because domain knowledge of the context is required. But the most important indicator of scientific fraud are not the cycle types, but outlying numbers of cycles of actors.

We propose a statistical method for determining outlier thresholds for these number of cycles, to eliminate the need for domain expertise and the need to ask potential fraudsters for their own thresholds. We validated our method by replicating the PC member demands citations results of Westerbaan. We then applied this method to detect outlying persons with a high percentage of self-citations.We manually checked the top 10 of scientists with the highest self-citation score.

Our proposed method allows us to quickly investigate other cycle types, as long as the question "what is abnormal behavior in which context?" can be answered for the investigated cycle type. Therefore, our method is more flexible than Westerbaan's attack-detection based method, which required each time a new method of detection and querying of data.

With this thesis, we contribute by providing a scientific biographic information graph, a more efficient method for detecting cycles in this graph, and a statistical method of determining outliers.

The improvements in this thesis also opens the possibility to test machine learning or other statistical methods to find scientific fraud within the constructed graph. It is also possible to expand the graph, by integrating more data sets, as well as adding new types of nodes and relationships. Even with our improvements, scientific bibliographic data integration is a challenge (also for our data sources) and can be improved further to enhance the quality of our results.

Contents

Chapter 1

Introduction

Recently, the correctness of several highly-cited foundational publications of Alzheimer research were called into question [Pil22]. These publications are suspected of containing manipulated and reused images. Many studies and methods were developed on the results of these publications. If the results of these studies were indeed faked, millions of dollars on Alzheimer research in the last 16 years were spent on conclusions drawn from faked data.

This example illustrates that scientific fraud can be undetected for a long time, because detection mechanisms are often lacking. According to one of the interviewed scientific experts: "They [the journals and granting institutions] are not subjecting images to sophisticated analysis, even though those tools are very widely available"[Pil22]. For example, only in March 2021 the editors of Molecular Therapy journal started their exploration whether image manipulation detection software could be used for their journal [FH21] and in July 2022 they reported that now all articles will be scanned for image manipulations [FH22]. Besides the fact that scientific fraud detection tools are often not used, there are also forms of scientific fraud that still lack decent detection tools. Examples are the detection of peer reviewing own work, or demanding citations as an editor by a journal.

These kinds of fraud are occasionally still being detected, but are currently mainly reported by human investigators or whistleblowers. Meanwhile, the body of scientific research is growing exponentially, it doubles in size every 10 to 15 years [JCG22; BM15].[1] Given this growth rate, and the lack of detection, there should be lots of scientific fraud waiting to be revealed. For example, Van Noorden [Van21] expects that there are hundreds of generated papers still undetected.

Note that the papers themselves do not commit fraud. Only the authors that write them do. So, we want to shift focus to identifying potentially misconducting actors, like authors. When an actor's behavior is suspect, fraud investigators should focus on this actor, and their related papers. This approach looks for abnormal behavior, so it is possible that the actor did not commit any form of scientific fraud, since the definition of normal, and thus also abnormal behavior, is context dependent. There could be suitable, non-fraudulent explanations why someone is behaving in a particular way. See, for example, the blog post[2] where Mark Griffiths answers questions of a journalist on the suspicion[3] raised by him publishing one article in two days on average.

[1]DBLP shows a graph with publications per year on their website `https://dblp.org/statistics/publicationsperyear.html`

[2]See `https://drmarkgriffiths.wordpress.com/2020/10/22/gambling-with-somebodys-reputation-part-4-the-story-behind-the-story/`

[3]See `https://deevybee.blogspot.com/2020/07/percent-by-most-prolific-author-score.html`

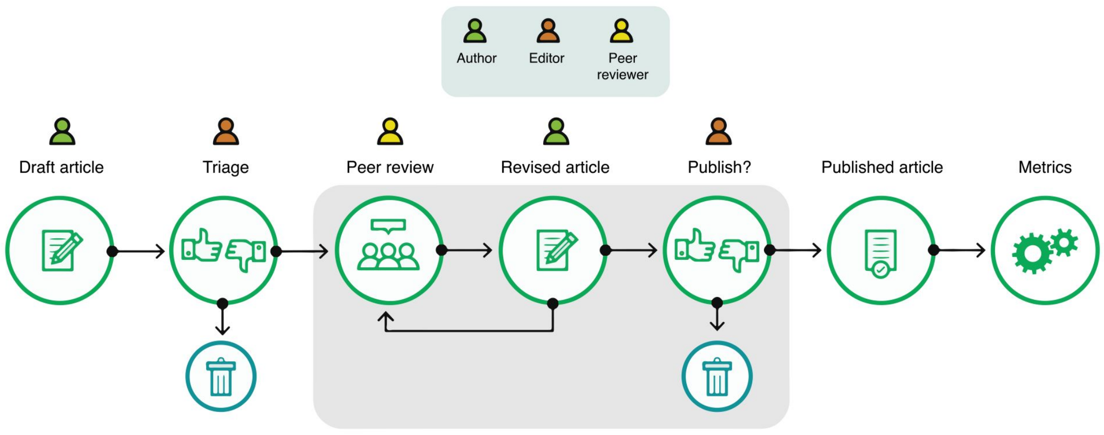


Figure 1.1: Simplified model of academic journal publishing adapted[4] from [SO18]

The number of actors is far less than the number of papers, but is still large. So we propose a method that can be automated and thereby reduce the amount of work for a fraud investigator significantly. Such software tool can help to let humans focus on interesting investigation pointers from the tool and types of fraud detection that are currently not automated.

Our tool searches for abnormal behavior that could be indicative of scientific fraud. With this, we are unlikely to find actors that misconducted once or twice, since they will not differ that much from normal behavior. Instead, we focus on actors that benefit much from their abnormal behavior.

Scientific misconduct is a broad and diverse concept. As fraud can be committed at each step of the scientific publishing process. For illustration purposes, Figure 1.1 shows a simplified model of the scientific publishing process in a journal, including some actors. During the process, many actors are involved, including authors, reviewers, editors, program committee members, publishers, and automated data integration systems. The latter refers to services like Google Scholar or DBLP that automatically parse publications to fill their data repositories and calculate publication metrics. Each actor can influence one, or sometimes several, steps of the scientific publishing process. Below, we give a few examples of known fraudulent or dishonest methods, grouped by some steps of the scientific publishing process.

1. **Writing** The authors can commit fraud by adjusting the publication itself, for example by, plagiarism, citing of own papers, fabrication of data, manipulation of data, manipulating or reusing images, fabricating bogus papers with SCIgen[5] [Van21], buying already manually fabricated bogus papers by so-called "paper mill" companies [EV21], and buying co-author places in articles [Cha22].
2. **Reviewing** An author can, for instance, peer-review their own papers (e.g. Moon [Ora12]), skip reviews by publishing in a predatory journal [BL20][6], or setup fake peer-review rings [BL20]. Also, a journal can "help line up friendly peer reviews for an extra fee" [BL20].
3. **Publishing** An editor or PC member can require co-authorship or citations of their papers before publishing an article in their journal or conference [WF12; Wes22].

[4]Modifications: removed title, moved legend to center, and added metrics step. Image is licensed under CC BY 4.0.

[5]`https://github.com/strib/scigen`

[6]See also `https://beallslist.net/` for a list of potential predatory journals and publishers.

4. **Automated data integration** An author can willingly mislead automated data integration, for example by putting self-citations in white text on a white background, or as text behind an image [Ant20]. Such texts are machine-readable, but not human-readable.

This last type of fraud is unlikely to be detected by human investigators. But when fraud is being investigated by software tools that use data sources of machine reading, like we do, the hidden text is automatically included in the anomaly finding. This can complicate human fraud investigation of the reports of the tool since they are not in accordance with human observation. We remark that there are two decisions to be made: should machines not read text that is not human-readable? and should misleading automated data integration be considered scientific fraud? These questions are to be discussed by publishers and the scientific community. In this thesis, we use the output of automated data integration providers, so if they are misled, so will we.

For the writing phase some automated tools are already known, for example plagiarism scanners[7], SCIgen detection scanners[8], paper mill scanners[9], and image manipulation detectors.[10] But, currently, we are not aware of tools that are in use for detecting scientific fraud that does not falsify the research itself, like boosting citations. To improve the situation, Tielenburg [Tie17] tried to identify scientific outliers and then comparing them with peers, but encountered problems with the latter. Westerbaan [Wes22] swapped this order, by first creating peer groups and then identifying outliers. With this approach, he was able to identify several persons of interest. Both based their detection on known fraud techniques, which requires revealed fraud techniques from manual investigation or whistleblowers.

Westerbaan [Wes22] observed that some of his investigated fraud techniques could be described as cycles in an academic publication graph. So it could be that by detecting cycles in an academic publication graph, scientific fraud could be revealed, also for currently unknown fraud techniques. This is based on the **assumption** that when an actor commits fraud, it will eventually benefit that same actor. This benefit should be measurable because the aim of scientific fraud is often to boost an actor's academic metrics. So in a graph from the academic world, we should be able to find cycles starting and ending by the actor that committed fraud. The pattern of the cycle will describe the kind of fraud. Note that a cycle in itself is not a proof for fraud, since even legal actions, like publishing a paper, will benefit the person publishing it and will lead to a cycle back to that person. Only abnormal amounts (the outliers) can be an indicator for fraud.

Therefore, the main research question of this thesis: **When is the number of cycles of an actor abnormal within an academic publication graph?** As we discussed above, this abnormal number of cycles of an actor can be indicative of scientific fraud. Note that we do not want to assume this approach must work, so never is also a valid answer.

To answer the main research question, the following sub questions need to be answered:

RQ1 **How to integrate publicly available data into an academic publication graph?** While there are various sources of academic publication data, relations between elements in the source are not always explicit. For example, in DBLP, the author field of a publication is a text entry, not a link to an author. Moreover, different sources may use different representations of data. E.g., "Wibren Wiersma", "W. Wiersma", "Wiersma, Wibren", and "W.J. Wiersma" all refer to the same author. Integrating data from multiple sources thus compounds the problem. (Chapter 5 and Chapter 6)

[7]There are many plagiarism scanners, for example Copyleaks.

[8]For example SciDetect and the method by Labbé and Labbé [LL13].

[9]According to Else and Van Noorden [EV21] the paper mill scanners are still have scaling issues.

[10]For example Droplets [Wes22].

RQ2 **How to efficiently detect undirected cycles in a directed academic publication graph?**
The graph resulting from the first research question is directed (e.g., direction of citations). However, fraud does not always adhere to directions. For example, self-citation can be expressed as an author wrote an article that cites an article written by the same author, in other words: WROTE, CITING, WRITTEN BY, or: `author→article→article←author`. The second challenge is efficiency. Straightforward cycle detection is too slow for a realistic setting with millions of authors. (Chapter 7)

RQ3 **How to identify outliers given cycles in an academic publication graph?**
Cycles themselves do not constitute indicators of scientific fraud, as we will discuss in Chapter 8. Our assumption is that an abnormal number of cycles does indicate fraud. This raises the question: when is the number of cycles abnormal? We address this in Chapter 9.

This study contributes to fraud detection in academic publishing with the following:

- An academic publication graph.[11] This includes the data acquisition, parsing, and integration that is required therefore. These contain improvements for the data acquisition, parsing, and integration method of Westerbaan, which are used for constructing the graph.
- An efficient method for detection of undirected cycles in a directed academic publication graph. We discuss performance differences between cycle detection methods and propose a more efficient one.
- An improved method to determine thresholds for outliers by using a statistical approach.

This thesis is structured in the following way. We first provide some background information in Chapter 2. After that, other related work is discussed in Chapter 3. We then describe our methodology in Chapter 4. In Chapter 5, the additional data acquisition and refinement will be discussed. In Chapter 6, we describe how the data sources are integrated into one graph. Then we describe how we efficiently detected cycles in Chapter 7. We discuss the found cycle types in Chapter 8. Then we describe the method and results of detecting outliers for two cycle types in Chapter 9. And finally, we list our discussion, future work, and our conclusions in Chapter 10. In the conclusions, we will directly answer all the research questions of our thesis.

[11]The dump of the Neo4j graph can be requested here `https://doi.org/10.17026/dans-x4m-eda2`

Chapter 2

Background

This chapter contains background information to understand the thesis more in depth. In the first section, Section 2.1, we discuss the incentives for scientific misconduct, which are often directly associated with scientific metrics, but the situation is more complex. In Section 2.2 we provide a small introduction to understand the process of academic publishing, with is relevant for possible methods of scientific fraud. Known methods of scientific fraud that can be expressed as cycles in a graph are discussed in Section 2.3. In Section 2.4, we show how to perform weighted exponential regression, which is used for determining outliers in Chapter 9.

2.1 Incentives for scientific misconduct

It is often suggested that scientific metrics (like h-index, citation counts) allure scientist to misconduct. This suggestion is based on Goodhart's Law: "When a measure becomes a target, it ceases to be a good measure" [Str97]. This claim is applicable because the scientific metrics were invented to measure scientific performance, and since it now becomes the goal, scientists will optimize their way of working to effectively increase these metrics to their full potential. In this way, a scientist could feel the "pressure to publish" to uphold the metrics, which could cause that older measures like quality of work become more disregarded, or even that the scientist will be tempted to commit fraud [Gri20].

There is evidence that scientific metrics have an effect on scientist behavior, for example the number of self-citations increased after scientists are promoted based on their number of citations [See+19]. However, there is no evidence that this change led to more scientific fraud [Lin20; Fan20]. Instead, scientific misconduct could be caused by scientific funding differences between countries, as concluded by Fanelli [Fan20]. Therefore, Fanelli suggests that not "pressure to publish" is a motivator for fraud, but corruption or greed of individual scientists [Fan20]. He suggests that "pressure to publish" led to an increasing number of co-authors within publications, because in the calculation of the metrics the number of co-authors is not included. Authors that are doing a small amount of work in many articles and with many others, have the simplest non-fraudulent way to increase the metrics.

Even the gender of scientist, research funding by industry showed no correlations with risk of scientific misconduct [Fan+22]. Only more productive, more frequently cited, earlier-career researchers working in lower-ranking institutions in countries where high-impact publications are rewarded with cash are more likely to misconduct than their peers in other countries [Fan+22].

Concluding, scientific metrics are no incentives to misconduct, as long as they are not used directly to reward a scientist in cash. Although the metrics are effecting the behavior of scientists even towards gray zones like self-citations. The method we propose in this thesis will besides pointers of misconduct more likely find the effects of behavior, most likely in the gray zones. This underlines the argument that a human investigator is required to determine if someone is exploiting the gray zones in a way that it can be considered fraud.

This conclusion holds only for scientific metrics that target researchers performance. For example, the Journal Impact Factor (JIF) can influence the popularity of a journal. But the journal itself cannot influence its JIF, because the JIF is based on the amount of citations that articles receive in a journal. For a journal it is often hard to guess whether an article will gain a certain number of citations. Journals are therefore tempted to increase their article citations by other means. To improve this situation, a group of scientist, editors, publishers, and scientific funders joined the San Francisco Declaration on Research Assessment (DORA) [12; Van14]. In this declaration, they share their concerns and recommend to not evaluate scientific performance with the JIF.

2.2 Academic publication process

Westerbaan [Wes22] already discussed the academic publication process in full detail in Section 2.1 and 2.2 of his study. Björk and Hedlund [BH04] formulated a formalized model of the scientific publication process. So, for a more detailed description look there, but for the background of this thesis, we keep it simplified. The simplified publication process within a journal is previously shown in Figure 1.1 (repeated in Figure 2.1).

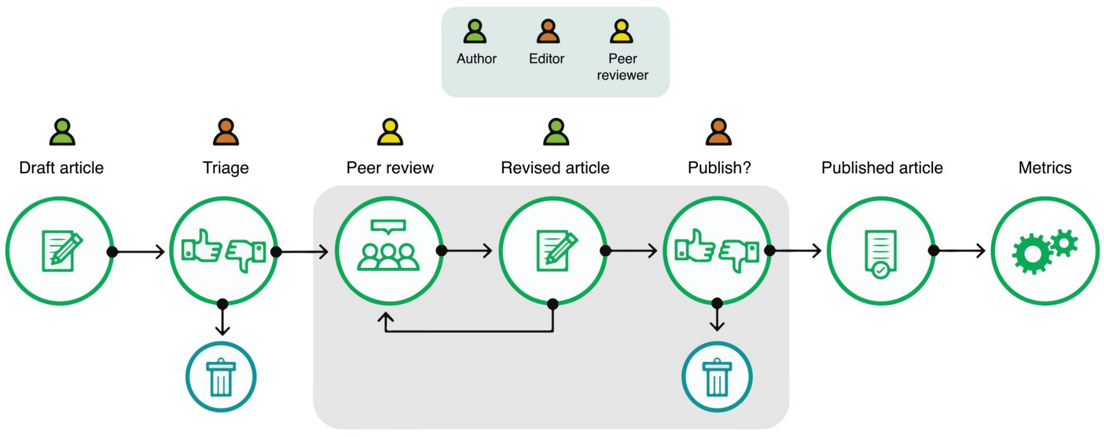


Figure 2.1: Simplified model of academic journal publishing adapted[1] from [SO18]

The process First, the author or authors write an article, then they submit it to a journal. The journal editor or editors do a quick preselect whether the article is of decent quality and should be submitted to peer review. Then someone, with knowledge of the field that the article targets, is asked to peer review the article. Sometimes the authors are asked to suggest a peer reviewer that is able to review their work. The peer reviewer comes with some suggestions for improvement. The authors then have the possibility to improve their work, based on the suggestions of the peer reviewer. Depending on the journal, this process of reviewing and improving can be repeated a

[1]Modifications: removed title, moved legend to centre, and added metrics step. Image is licensed under CC BY 4.0.

number of times. The editors then decide based on the peer review report and the improvements of the authors whether the article should be published or not. After the article is published, crawler bots like those of Google Scholar, the Web of Science and Scopus are reading the published article and processing its metadata like citations, authors and date.

Differences between journals and conferences For the purpose of this paper, the editor can also be replaced by a program committee (PC) member of a conference instead of a journal. Between the publication process of journals and conferences there are differences, for example the editors of a journal are often longer times involved, while program committees may differ per conference. Conferences typically have a deadline for paper submission and the final version, because their occurrence date is fixed. Conferences also have limited space. Journals will often publish the article they approved in the next available issue.

NISP experiment The NISP experiment[2] showed that whether a paper gets approved depends heavily on the composition of the responsible committee. Therefore, subtle differences in opinions of PC members or editors will influence which papers will be published or not. This suggests that the publication process is susceptible to fraud.

2.3 Cyclic expressible scientific fraud

In this section, we discuss several examples of scientific fraud that can be described by cycles in an academic publishing graph. Westerbaan [Wes22] already discovered that some methods of scientific fraud can be expressed as cycles in a graph, these are also included here. Note that in order to see the cycles, we must omit directions, and view the graphs as undirected, this will be explained in more detail in Section 4.2.1. These can be encoded with labels, so they can be detected in our graph, these labels are presented in Section 8.2.

2.3.1 Demanding citations or co-authorship

As mentioned by Westerbaan, editors have power to accept or reject a paper, and the NIPS experiments showed that this process is not deterministic. An editor or PC member can exploit this power by demanding citations (blue) or co-authorship (red) before accepting.

Both methods of fraud can be presented in a graph, as shown in Figure 2.2. Therefore, this kind of fraud is detectable with cycles in a graph. However, not each detected cycle of these types is an indicator for fraud. For example, an editor is often asked because of his/her experience in the field of the venue. Since the publication at that venue possibly covers his/her field of expertise, it is not strange that an author cites the editor's publication by own will. Another example is that the editor is publishing at the venue by him/herself, and therefore the committee decides to avoid conflicts of interest by removing that person from the decision-making of that publication. Personally, we still should be suspicious about the latter, because it is hard to see for an outsider if these conflict of interest provisions were regarded or not. Despite these cautions, it is a valuable pointer for fraud investigation when someone has a lot of cycles in one of the two described cycle types.

In Section 9.1 we replicate a finding of Westerbaan [Wes22] by using the PC member requires citation cycle.

[2]`http://blog.mrtz.org/2014/12/15/the-nips-experiment.html`

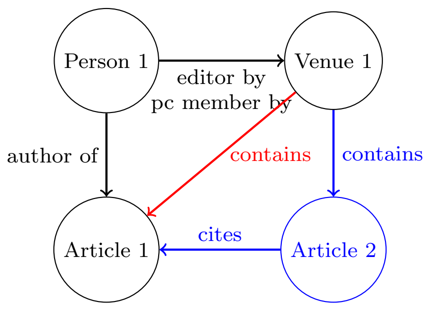


Figure 2.2: Graph representation of require citations or co-authorship, adapted from [Wes22]

2.3.2 Self-citation

The study of Seeber et al. [See+19] showed that self-citations increased after citation numbers became important for promotion. Self-citation is by definition not a fraudulent method, but it depends on the context. It can be an indicator of scientific fraud when self-citation is very frequently used or when citing, own, out of context papers. Westerbaan [Wes22] already showed that this method can be expressed in a graph, like we do in Figure 2.3. In Section 9.2 we discuss our findings of this cycle type and detect persons who abnormally frequently cite themselves.

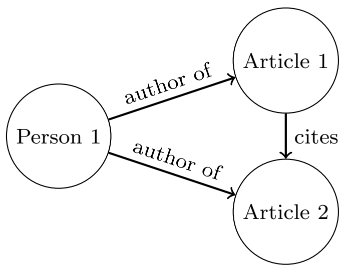


Figure 2.3: Graph representation of self-citation, adapted from [Wes22]

2.3.3 Reviewing own work

Moon submitted bogus e-mail addresses (sometimes related to real people) that belonged to him (red) or to an associative (blue) as potential reviewers for his own work [Ora12]. He also suggested friends and colleagues to review his work (blue). He was detected because of the short time between submission and accordance.

As shown in Figure 2.4 it is possible to view this fraud method in a graph. Although, there is one problem here: the `reviewer-of` relationship is disguised. Partly because bogus identities were used, therefore having a bogus `Person 1` node, but peer reviews are also often performed double blinded. Therefore, an outsider must deduct a link between reviewer and article from the link between a reviewer and journal issue. This example highlights one of the challenges of our approach in this thesis, sometimes a relationship has been hidden and must be deducted by other methods.

Westerbaan proposed to uncover this method of misconduct by looking at the publication lag (the time between the submission and the acceptation of a paper) [Wes22]. Transforming his method to a cycle detection method in a graph is hard, since the cycle detection must also consider calculations in time based on properties, nodes, or relationships (depending on the chosen data model). We have chosen to not include this attack in our thesis. If publishers, journals, and conferences shared this relationship, we could have detected this fraud method also. With clever thinking and modelling, this challenge can probably be overcome in a future work.

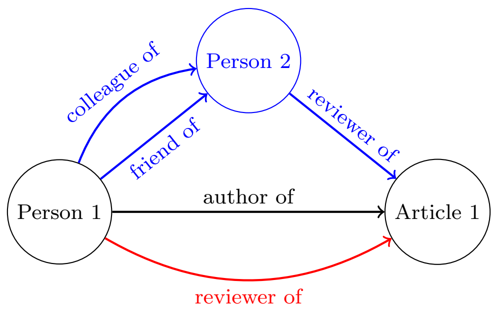


Figure 2.4: Graph representation of reviewing own papers

2.3.4 Venue self-citation and citation stacking

Citations of publications in journals determine the journal's impact factor (JIF, see also Section 2.1). Scientist are rewarded (in fame, also sometimes in cash) when publishing in high impact journals. Some journals were caught to artificially boost the journal impact factor by demanding or inserting citations within publications before publishing it [Van14; WF12]. Other journals were caught to have agreements to extensively citing each other publications, also called 'citation stacking' [Van14].

As shown in Figure 2.5 for self-citation, and Figure 2.6 for citation stacking, both methods of fraud are describable by graph and are cyclic. These cycles are also not direct indicators for fraud. Neophytou [Neo15] noted two possible reasons why journal self-citation is sometimes explainable: firstly, when the journal covers a niche or contains archived publications, and secondly, when other journals in the same discipline are not included in the data sources, thereby missing many citations from those journals to this one, having that a large portion of references to the journal will be from the journal itself.

On citation stacking, Neophytou [Neo15] mentions four reasons to be cautious. Firstly, the journals cover a niche and authors will therefore cite only a few other journals in that niche. Secondly, she has seen that authors that commit extensive self-citation could cause suspicion on venue citation stacking. Thirdly, authors can form citation cartels. And, lastly, small journals can have highly fluctuating metrics, one or two articles could already influence a high percentage of it.

When one of these cycles extensively appear with particular venues, this should lure the attention of a fraud investigator. The fraud investigator should investigate whether these are false positives by the cases mentioned by Neophytou.

These cycles were not investigated in our study because some exceed the length constraint on our paths (see Chapter 7), the reason for this is shown in Section 8.2.

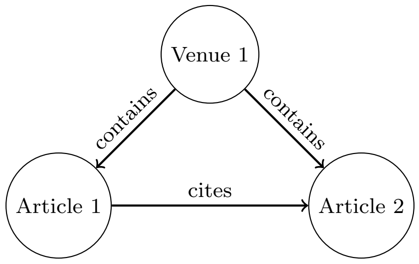


Figure 2.5: Graph representation of venue self-citation, adapted from [Wes22]

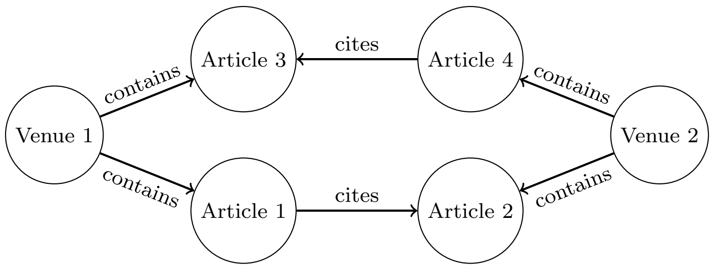


Figure 2.6: Graph representation of venue citation stacking

2.4 Weighted exponential regression

The statistical approach for detecting outliers that we chose is called weighted exponential regression. This is almost the same as exponential regression, but the weight biases the regression to the points with more weight. In our case, we want to apply weighted exponential regression on data where the y-axis describes how often the value on the x-axis occurs in our data, see for example Figure 2.7. We want to prevent a bias to outlying points, by providing a weight with the number of occurrences of each value we have. Normal values occur more often, therefore the exponential regression has to weight these more.

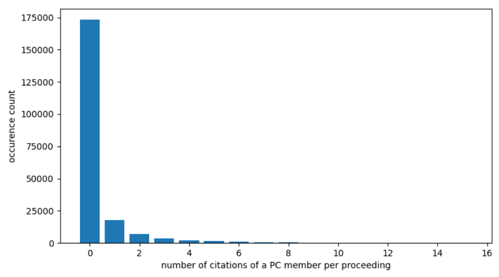


Figure 2.7: Example of exponential distributed data; taken from Figure 9.1a

Weighted exponential regression has been performed on our data in the following steps:

1. We want the result of the exponential regression to be the following formula: $y = a * b^x$.
2. We first transform our values of the y-axis to linear by applying the natural logarithm $y' = \ln(y)$ $(= \log_e(y))$.
3. If the data was exponential distributed, it is now linear. Therefore, we can now apply weighted linear regression. Linear regression results in $y' = m * x + c$.
4. We ask NumPy in Python to do this for us with: `(m, c) = numpy.polyfit(x, numpy.log(y), 1, w=y)`. The weight parameter (w) are the values of the y-axis, which are the number of occurrences of x. We provide y' by taking the log of y as explained in step 2.
5. To transform the result of the linear regression to the exponential regression, we have to undo the ln by raising it to the exponential of e. So $a = e^c$ and $b = e^m$.

Chapter 3

Related work

In this chapter, we discuss related work of others and the connections with our study. First, we discuss some related work on the importance of detecting scientific fraud in Section 3.1. In Section 3.2 several works on scientific fraud detection are discussed. We list some academic publication data sources in Section 3.3, with studies explaining their problems and decisions made on data acquisition and integration. The graph we constructed can be called a knowledge graph, since it describes knowledge of the real world. We address this in Section 3.4. Lastly, we mention some studies on graph based anomaly detection in Section 3.5.

3.1 The importance of scientific fraud detection

This study of Wilhite and Fong [WF12] highlights the importance of the detection of scientific fraud and shows the current scale of misconduct. They analyzed survey responses of 6672 scientists on coercive citations. In these responses, 175 journals were reported of requesting citations, the worst journal was named 49 times. One in 5 responders said to have been asked to include citations. More than half (57%) of the responders told that they would add the required citations if they were asked, while 93% of the responders believed that others would agree to add citations. Wilhite and Fong mention that authors, besides victims, could also be co-conspirators. After their article has been published in such a journal, they also benefit by receiving coercive citations.

According to Biagioli and Lippman [BL20], post-production fraud is targeted at institutional valuation, so the fraud methods change when metrics techniques and markets are changing. Since they are currently changing, it is naive to build a checklist with known fraud methods. They note that there is currently no concept that captures all the various manifestations of post-production misconduct. With our method, we allow switching between different types of scientific fraud if they can be encoded as cycles.

3.2 Detection of scientific fraud

There are several studies that investigate scientific fraud, wherefrom their method or model could be used. In this section, we first discuss the chain of research where this thesis is based on. Then mention our connections with other studies.

Jonker and Mauw [JM17] constructed a basic academic publication model to research the impact of decisions in the research process on scientific metrics. Their publication model is shown in Figure 3.1. Tielenburg [Tie17] used this model for finding outlier with manually crafted heuristics, and then validate them by comparing them with peers, but had no success in the latter. According to Westerbaan [Wes22], Tielenburg was not successful because several scientific fraud forms include editors and reviewers. To include editors and reviewers, Westerbaan expanded the model. The additions of Westerbaan are presented in blue in Figure 3.1. This model is a simplified version of the graph we constructed, not surprisingly because we search for the same kind of fraud. Westerbaan also swapped the order, by first creating peer groups and then detecting outliers. He succeeded in finding interesting outliers.

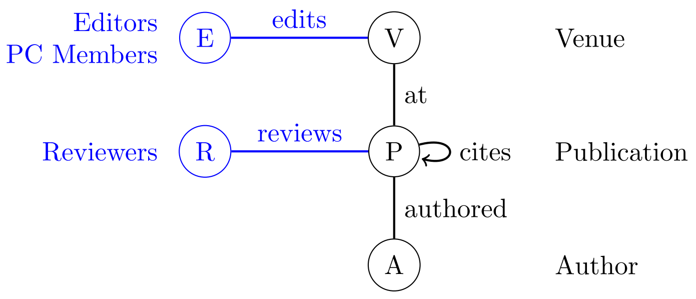


Figure 3.1: Publication view, adapted from [JM17], and in blue from [Wes22]

Kojaku, Livan, and Masuda [KLM21] proposed an algorithm, named CIDRE, to find citations networks of different kinds. They showed various examples of known citation networks and detected new ones. But their algorithm follows an attack-detection based approach like in the study of Westerbaan, their method cannot be converted into cycle based approach.

Frandsen [Fra22] researched authors publishing in predatory journals. The majority, 55%, only published once in a predatory journal, but some are published more often. In these journals, anyone can publish work without a proper peer-review or critical selection procedure. So, it is suspicious if a researcher wants to evade peer-review or other critic. In our study, we wanted to focus on finding outliers in normal journals, since questions can already be raised if someone published in a predatory journal. However, as future work, it is possible to add these predatory journals and encode their detections as cycle types.

Björk and Hedlund [BH04] constructed a formalized model of scientific research. This work does not investigate scientific fraud, but by providing a model, identify points where scientific fraud can be influenced.

3.3 Data sources of academic publications

There are also some studies describing the problems and models of different data sources:

- DBLP [Ley09] describes their imperfections, like problems with name matching and the modelling problems, and whether an article or paper can be considered an `inproceeding` or `incollection`. On the latter, see also Section 6.1 and Section 8.1.
- Aminer [Tan+08] provided an improved method for name to person matching.
- OpenCitations [HPS19] introduced the COCI (Crossref OpenCitation Index) dataset, containing DOI-to-DOI citations. They also retrieve their data from closed sources.

Several issues regarding data sources are discussed in Chapter 5, there we refer also to DBLP and Aminer. We used DBLP and OpenCitations as data sources for our graph.

Van Noorden [Van14] discussed some findings by Clarivate (formerly Thomas Reuters), the company behind the Web of Science. It is possible to download their reports or build queries for the Web of Science to extract the results. However, the Web of Science cannot be integrated with other data sources because access through their API is limited.

3.4 Knowledge graph

The graph constructed in this thesis is a knowledge graph. Hogan et al. [Hog+21] define a knowledge graph as: "a graph of data intended to accumulate and convey knowledge of the real world, whose nodes represent entities of interest and whose edges represent potentially different relations between these entities." This definition fits very well the graph we constructed for this thesis, since our graph describes the relations between articles, persons, and venues which are real world entities. The graph of our study is a directed edge labelled, property graph. Their study helped us to compare different forms of knowledge graphs. Choosing between these forms had little influence on the research, but can be influential on the usefulness of the graph in practice.

3.5 Graph-based anomaly detection

In this section, we discuss several studies on graph based anomaly detection related with fraud investigation. Our thesis briefly touches this topic. But a future work on our study could go more into the direction of anomaly detection, machine learning, and statistics.

Pourhabibi et al. published "a systematic literature review of graph-based anomaly detection approaches" for fraud detection [Pou+20]. Some approaches mentioned contain self constructed algorithms that were context dependent (like for insurance, banking, trading, and social networks), unfortunately there are no academic context studies included. Besides the context, we were not able to find a referenced study that did something similar as that we wanted to: detecting anomalies in an academic publication graph.

But, we found one included study that is slightly related: the study of Carvalho et al. [Car+17]. They decided to detect anomalies in health care, by first giving anomaly scores to consumers (the clients) and then deducting anomaly scores of producers (the hospitals). They used manual investigation afterwards to validate their results, and suspicion of fraud. In our study, we decided not to use anomaly scores, because summing citations was sufficient. We also emphasize the need for manual investigation to validate the results of our method, before raising the suspicion of fraud. We also chose to validate our results manually in Section 9.2.

Another interesting study it that of Li, Xiong, and Liu [LXL12] where they describe an algorithm to find black holes: groups of nodes with more inward edges than outwards. It could be suspicious when certain persons in our graph receive many citations without citing others outside their group frequently, and thereby forming a "citation black hole". Such investigation is a different method than we investigated in our thesis, and requires a fully directed graph, but could be interesting to investigate in a future work.

Schulz et al. created a new metric for scientific performance, for substantiation, they investigated citations in respect to author distances: the amount of co-authors in the graph between two authors [Sch+18]. They found that authors closer to others in the graph are more likely to being cited than those further away. This study can therefore easily be transformed to anomaly detection study: which authors were cited that were not likely to be cited? And this idea of author distance can also be used to construct a k-Nearest Neighbor for comparing authors with each other. For cycle types we did not investigate, these analyses might be useful to construct peer groups.

Chapter 4

Methodology

As also written in Chapter 1, the research question is **When is the number of cycles of an actor abnormal within an academic publication graph?**. In other words, we want to find outlying actors by looking for an outlying number of cycles. In this chapter, we address the methods used to answer this question, by addressing them for each sub question separately.

4.1 RQ1

How to integrate publicly available data into an academic publication graph?
The goal of this question is to define a data model that can be transformed to a graph. This will be discussed in Section 4.1.1. To do that, we had to integrate several data sources, the main challenge for integrating the data sources is people and name ambiguity, as discussed in Section 4.1.2. And, we had to select a graph database software, because this dictates to some extent how we must design our data model. This is discussed in Section 4.1.3.

4.1.1 Data model

For the construction of the academic publication graph, we needed a data model. Westerbaan [Wes22] noted that there is currently not one dataset that allows us to detect all the fraud techniques mentioned in Section 2.3. Microsoft Academic Graph had a lot of this information, but is no longer accessible since December 31, 2021.[1] Therefore, Westerbaan constructed new datasets by combining and integrating existing ones. Besides this reason, we also wanted to compare our PC member demands citation outliers with Westerbaan's to validate our method. Therefore, we are obliged to use the same data sources, because others could potentially lead to different results. Westerbaan constructed a new data model for each fraud technique, instead, we first constructed one graph to find and investigate fraud techniques. We have chosen to integrate DBLP, OpenCitations, scraped editors from ACM editorial boards and scraped PC members from Springer LNCS journals. All these sources were also used by Westerbaan. We did not use Aminer since the extraction of data from Aminer is harder than DBLP. DBLP provides one XML file for download that contains almost all of their data.

Westerbaan already encountered data integration and processing problems. He mitigated these by choosing his queries carefully and checking his results. We cannot do that because we want to construct one graph and perform cycle detection in it. Therefore, wrong relationships

[1] `https://www.microsoft.com/en-us/research/project/academic/articles/microsoft-academic-to-expand-horizons-with-community-driven-approach/`

cannot be ignored by not querying it, since the cycle detection will use them anyway. Therefore, the wrong relationship will show up in many cycles. No relationship is better than a wrong relationship. Therefore, to improve the quality of our detected cycles, we had to improve the data quality and integration. These improvements are listed in Chapter 5. The integration of the data sources to our graph is described in Chapter 6. One of the main challenges in integrating the data sources is people and name ambiguity, which we discuss in the next section.

4.1.2 People and name ambiguity

Authors of papers are always mentioned by their names. However, it could happen that multiple persons share the same name, or name abbreviation [Wes22; Ley09]. This hardens matching the correct persons with the correct article (QJ3). This problem is not only damaging the validity of academic biographic data sources, but could also be frustrating for scientists if their publications are incorrectly matched with another person.

Therefore, academic data sets, like DBLP and Aminer, also had to deal with this problem: DBLP decided to still store person information by name, but to separate persons with the same name they place a postfix at the person name with an incrementing number for person names that already exists in their dataset. For a more detailed description on the way DBLP handles name ambiguity, see [Ley09]. Aminer developed probabilistic models with machine learning to solve this issue [Tan+08].

ORCID To provide a permanent solution to this problem, the ORCID was introduced, to give each researcher a unique ID. As noted by Westerbaan [Wes22] there are persons that have multiple ORCIDs in the DBLP data set. This could be due to data integrity issues of DBLP, but more likely people actually have multiple ORCIDs. On the website of ORCID it is explained that when someone registers a new ORCID other possible matching records are shown to prevent creating multiple ORCIDs for the same persons.[2] This indicates that the organization behind ORCID is aware of this issue and tries to prevent it. Although singularity is not preserved, uniqueness is. We did not find shared ORCIDs between persons in DBLP. ORCIDs are not used very much, but first and foremost for authors of publications. So, unfortunately, matching PC members or editors still has to be based on person names.

Method We used naive name matching with DBLP. Therefore, when there are multiple persons with the same name, we match only with the first. It goes beyond the thesis to develop or train a model for matching the correct person, like in [Tan+08]. Therefore, the fraud investigator has to check afterwards if name matching may invalidate the results of our method. This is one of the limitations of our study that we highly recommend improving in future work.

4.1.3 Graph database software

The graph database software that we used is Neo4j, because of the recommendation in a graph databases comparison study by Fernandes and Bernardino [FB18]. Neo4j fully supports the graph query language Cypher, and also supports GraphQL. Therefore, according to Hogan et al. [Hog+21] Neo4j can handle the most used queries types for graphs: graph patterns, complex graph patterns, and navigational graph patterns.

[2]`https://info.orcid.org/researchers/`

4.2 RQ2

How to efficiently detect undirected cycles in a directed academic publication graph? Because the graph constructed in RQ1 contains 5 million vertices and 44 million edges, detecting cycles can take a long time when not focussing on efficiency. In Chapter 7 we discuss and propose an efficient way of detecting cycles.

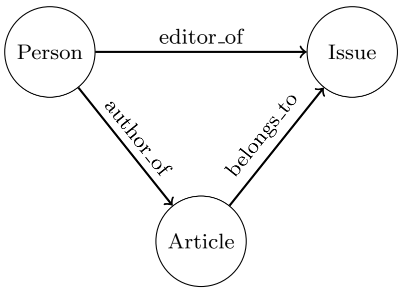
In Section 2.3 we already noticed that for detecting cycles we must omit directions, we explain this in more detail in Section 4.2.1.

4.2.1 Directed vs undirected

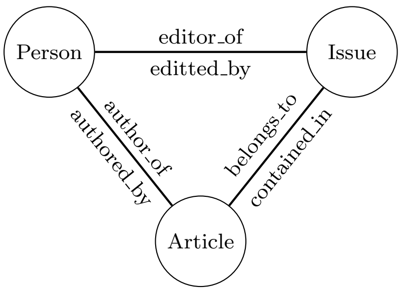
Although the graph constructed for RQ1 is a directed graph, for cycle detection it is better to interpret it as an undirected graph. To explain this, see Figure 4.1a. When we want to detect cycles starting by the person, we will never reach the person again, since all paths end in the `Issue` node. Still, we as human know that all relations in the graph are reversible: `editor_of - has_editor`, `author_of - has_author`, and `belongs_to - contains`. For each relation expressed by an edge in the graph in Chapter 6, there exists an inverse relation. Therefore, to detect cycles, we omit directions on the edges, interpreting the graph as undirected.

At first sight, another solution would be to just add the reverse relations to the graph, but when doing that it becomes hard to prevent that a path does not contain the same relation more than once if the relationship is technically separated in two different edges. For example, we have to prevent false cycle paths like: $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{authored\_by}}\texttt{Person}$. We found it more feasible to interpret the graph as undirected, as to filter or prevent these false cycle paths. Because first detecting and then filtering is not efficient.

Note that although we interpreted the graph as undirected, in the cycles we still need the directions of the relationship. Because the direction of the `cites` relationship matters. For example, if a person cites an editor's article, means something different from if an editor cites a person's article.



(a) Directed



(b) Undirected

Figure 4.1: Graph example

4.3 RQ3

How to identify outliers given cycles in an academic publication graph?
After detecting cycles and group them by types, we can find outliers within a cycle type. The cycle types we found and the possibility of them to described methods of scientific fraud are discussed in Chapter 8. We have chosen two cycle types for outlier detection: the PC member demands citations, and the self-citations cycle type. We chose the first to validate our method by showing we can find the same outliers as Westerbaan [Wes22]. With the second, we illustrated that we can easily switch to other methods of scientific fraud that can be encoded as cycles. We listed the results and discussed our proposed method in Chapter 9.

But first, we explain in Section 4.3.1 other methods of outlier detection and why we propose our method.

4.3.1 Outlier detection

As already discussed in Chapter 3 there are several methods to detect outliers in academic publishing.

1. Manual
   (a) Using heuristics, first selecting possible outliers, secondly perform peer comparison to verify these outliers [Tie17]. This was not a success.
   (b) Using groups, to find and verify outliers in identity groups: program committee members, journal editors, authors based on publication lag. Attack detection based [Wes22].
2. Automatic, using machine learning. See for examples the study of Pourhabibi et al. [Pou+20].
   (a) Supervised,
   (b) Semi-supervised
   (c) Unsupervised

We have chosen for Westerbaan approach, since he proved that his method worked by finding suspicious outliers. Besides, an automated approach requires an integrated, high-quality data model and an automated method for outlier detection, which were both improved in this thesis, but these were not when we started our research.

Instead of the data acquisition and integration process for each attack, we now already have the data available from our cycle detection. Therefore, only gathering extra information is needed to provide context. But this can be done by writing a query for our graph. This approach will be discussed in Chapter 9. There, we also propose a statistical approach for determining when someone is an outlier, this is a small step in the direction of method 2.

Chapter 5

Data acquisition and refinement

In this thesis, the data acquisition is similar to Westerbaan's thesis [Wes22]. However, most of Westerbaan's data has not been reused to include more, recent and refined data. Instead, Westerbaan's data acquisition and parsing tools are reused, but also improved to provide a higher quantity and quality of data. We also developed some new gathering and parsing tools for the same reason.

In Section 5.1 we will discuss why improving the data quality is important before integrating the different sources in one graph. Westerbaan also notices the importance of data quality before integrating, therefore he reserved large parts of his conclusions and future work for discussing data quality issues he encountered. We were able to improve on many of these, as discussed below.

Westerbaan's conclusions and future work about data quality
In his conclusion, Westerbaan listed several problems with datasets and integration in general [Wes22]:

P1 "Source independent key attributes like (DOI and ORCID) are needed to link data across datasets more correctly. However, in practice datasets lack these kinds of keys, or develop their own (like DBLP)."
In this thesis, we improved the extraction of DOI's in DBLP by using a regex (Section 5.2).

P2 "Because of the lack of ORCIDs in the datasets, people have to be matched by other methods. The easy solution, matching by name, is not sufficient." Because there are several people in the body of research that have the same names, most likely Chinese persons, according to [Ley09]. But better methods are too complex for the purpose of our research (see for example [Tan+08]).
In this thesis, we improved the extraction of person names in the PDF data extractor for Springer LNCS front matters (Section 5.3). But still this problem stands, and is discussed in more detail in Section 4.1.2

P3 "Enriching datasets is hard, because the data is diffused and unstructured. Most of the additional data can be found, mostly on the website of the publisher, but is presented in an unstructured form. Which requires custom-made software for each publisher website and form of data presentation." This is most frustrating with (front matter) PDF's, because the way of presenting information changes throughout the years. PDFs are less likely to be updated to recent forms of data presentation, in comparison to webpages.
We improved the parsing of Springer LNCS front matters with 11.30% (Section 5.3). But also rencountered this problem when trying to parse ACM issue front matters (Section 5.5).

Westerbaan also listed some future work on data acquisition in his thesis [Wes22]:

F1 The editorial boards used by Westerbaan were limited to a number of journals to ACM, also only the active board was required, thereby missing past editorial boards on old publications.
We now believe to have included all ACM journals (Section 5.4). We also succeeded in downloading issue front matter's that contain editorial boards and reviewers for each issue. But had some troubles with parsing this data with the front matter parser (Section 5.5)

5.1 Data quality and integration problems

As shown above, most of Westerbaan's problems, conclusions, and recommendations were about the improvement of data quality and integration. Westerbaan did not succeed with integrating the data sources. For our approach, we had to solve or at least improve some of these issues, because we need a graph that consists of integrated data sources. Below, we listed general reasons for the hardness of data integration in our thesis, and refer to specific problems in the other sections to explain the general problem behind the issues in the data sources of this thesis.

The data quality issues in the data sources can be summarized as follows:

Q1 **Completeness** The data sources are incomplete, they do and will not include all scientific biographic information that could be included. There are three reasons why they are incomplete:

Q1.1 **Time** They are time limited: the data is downloaded or scraped at a certain point in time, therefore they do not contain data after that time. We tried to use recent sources, but even the data providers of the data sources need some time to discover and process new data to integrate it into their data set. The data is therefore always behind reality. Some data sources have a history boundary, for example when old publications are not digitalized and therefore not included into the data set.

Q1.2 **Cover** As already mentioned, they do not cover the whole body of scientific biographic information. For example, DBLP, ACM, and LNCS only include computing science biographs. But they even do not cover the whole area of computing science biographic information. This partly caused by the fact that none of the data sources fully manage all data, but draw also from other incomplete data sources. For example, DBLP includes data from ACM and LNCS.

Q1.3 **Unknown vs not exists** Even if potential data is within the cover of the data set, it may happen that the data source does not know everything. For example, DBLP may know a person's name, but not its ORCID. It can be that the person does not have an ORCID, or that DBLP does not know the ORCID of the person. In the first case the data is complete, in the second it is not. In practice, it is often harder to prove that data does not exist than to assume that it is unknown. In the example, DBLP can ask the person if it has an ORCID to know for sure if it does not exist or is unknown, but by asking the person will probably also tell his ORCID if it exists. So unknowing is the problem that it is hard to ask, for example because DBLP has to ask 1,000,000 persons or have to ask dead persons. In addition, proving that something does not exist is even harder because time comes into play, for example a living person may decide to get an ORCID assigned in the future, or an archive investigation may reveal a dead person's ORCID, thereby rendering the earlier data-does-not-exist conclusion invalid. Therefore, when data is not included in the dataset, the assumption that it is not known is better than proving it does not exist, although with this assumption we cannot know whether we have completeness.

Q2 **Integrity** The data in the data sources do not always represent the actual facts. Problems include incorrect matching or classifying data in the data source. For example, DBLP may link articles to the wrong person that has the same name.

Q3 **Processing errors** The parser or data extractor will not correctly extract all data, so parts will be lost and others will be classified wrongly. For example, when the front matter parser did not find PC members in a PDF, it could still be that the PDF does contain PC members. Another example is that the front matter parser classifies the authors of papers as PC members, or the other way around.

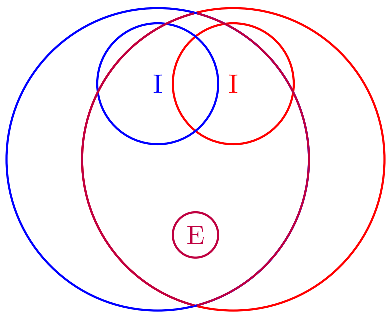


Figure 5.1: Data integration issues illustration

When we integrate data sources with quality issues, the quality problems expand in the integrated data set. To explain this better, look at Figure 5.1, the red and blue circles are two data sets that will be joined. When joining data sets the same quality issues will come back in different forms:

QJ1 **Completeness** Data that is not present in one of the two data sources cannot be linked, therefore the joined data (the purple area in the image) is always smaller than the data sources.

QJ2 **Integrity** Since both data sources have impurities, it may happen that correct data from one set is joined with incorrect data from the other set (I in the image), rendering the join incorrect. In other words, the impurities of one dataset can infect correct parts of the other when joined together.

QJ3 **Processing errors** Another problem, is that the joining method may join two correct data parts that should not have been joined, therefore rendering their join incorrect (E in the image). For example, when joining data of persons by person name, it may match with another person that has the same name, but is not the same person.

Because of the expansion of the data quality problems, it is important to cleanse the data from the data sources and processing software before integrating these into one graph.

5.2 DBLP and DOI

Like persons can be uniquely identified by ORCIDs, articles can be uniquely identified by DOIs. In the DBLP data set, an article can have multiple DOIs and other identifiers, these are included as `EE` elements in the DBLP XML. These DOIs do not all conform to the official format proceeding with `http://doi.org/`, but instead, proceed with, for example the IEEE DOI URL `http://doi.ieeecomputersociety.org/`. To correctly parse all DOIs from those `EE`s, we used the regex proposed by Gilmartin [Gil15] that caught 74.4M of the 74.9M DOIs of CrossRef and follows the modern DOI standard. This regex delivered some clean DOIs to match with the

OpenCitations dataset (Section 6.2).

By using this regex, we found that some articles contain the same DOI multiple times. We removed these DOI duplications within articles. Besides, across the whole dataset 8,441 DOI duplications were found. We found that a large amount was caused by Gilmartin forgetting to include the % char in the regex, though it is officially allowed as an escape character (Q3).[1] We updated the regex to contain the % char as listed in Listing 5.1. After addressing this, 6,216 genuine duplications of 1,072 DOIs were left. A quick investigation showed that some of these duplications were caused because the regex stopped matching when a '<' or '>' was encountered. We choose not to include these in the regex, because in a URLs the '<' and '>' chars need to be officially escaped (as %3C and %3E) (Q2). We chose to remove these duplications from the dataset to prevent incorrect matching (QJ2). The whole validation tree of the parsing of the DOIs using this regex is shown in Figure 5.2.

With this contribution, we improved the DOI based integration between datasets by parsing also non `http://doi.org/` DOIs (QJ1) and removing duplications of DOIs (QJ2). Many DBLP articles lack DOIs: 1,059,750 of 5,839,456 (18%) (Q1). Concluding, we improved the possibility of correct DOI matching, but the lack of DOIs on articles due to missing or non-existing data is great.

```
/(10.\d{4,9}\/[-._;()/:%A-Z0-9]+)/i
```

Listing 5.1: DOI Regex

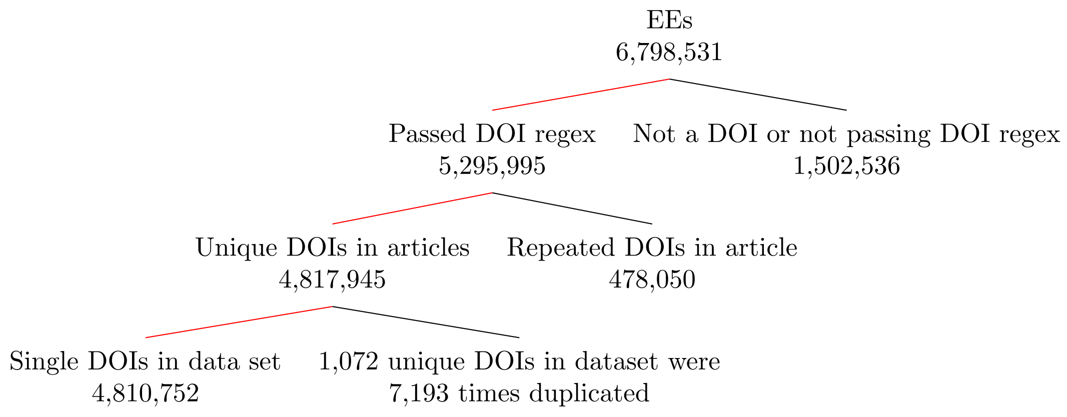


Figure 5.2: DOI Parsing validation tree

5.3 POC front matter parser

Westerbaan downloaded LNCS front matter pages to acquire PC members. To extract the PC members from the front matter pages, he built a proof of concept (POC) front matter parser. He was able to process most of the front matters, but 25.61% of the files could not be processed since there was no organization header found [Wes22]. We were able to reduce this to 10.71% by applying the following changes:

- In addition to search for an `organization` header, we also search for an `organisation` header, since both are valid spelled (Q3).[2]

[1] `https://www.doi.org/doi_handbook/2_Numbering.html#2.2`

[2] `https://writingexplained.org/organisation-vs-organization-difference`

- When no organization header is found, we search for headers containing: `chair`, `committee` or `reviewer`. Because this may indicate that the covering/parent section is an `organization` section in disguise. We test this by checking whether the parent section contains two or more subsections containing `chair`, `committee` or `reviewer` in their headers. By doing so, we now cover front matters that have the title of the conference functioning as the `organization` header, and old front matters that named things differently before the commonly used `organization` section became standard (Q3).

While improving, we noticed that the POC front matter parser was vulnerable to SQL injections, which caused queries to crash when the data contained a ' char. We fixed this by using parameterized SQL statements, besides the improvement of the code security this mainly allowed us to store data in the database that caused it to crash before. With this improvement, we reduced the unknown errors from 113 to 41 (Q3).

Besides, we fixed some parsing issues with spaces in words for big fonts in `PDFLayoutTextStripperFontSize` which now constructs sentences and words based on the current font space size instead of a hard-coded value. This removed spaces in words and double or triple spaces in sentences of large font sizes (like headers). This enabled better detection of section titles (for example when searching for the `organization` section) and member names (Q3).

We also focussed on improving the quality of parsed member names:

- When parentheses are detected at the end of the name, this most frequently indicates an affiliation, for example: `Wibren Wiersma (Radboud University, Nijmegen, NL)`. In such case, the affiliation is stripped from the name and stored as the member's affiliation. Note that parentheses in the middle of names often indicate Western names of Chinese members.
- If the name contains a comma, we assume it is presented as `Wiersma, Wibren`. If it contains more commas we assume that it is represented as `Wiersma, Wibren, Radboud University, Nijmegen, NL`, whereby after the second comma it is all affiliation information, which is stripped from the name and stored as the member's affiliation. When we encountered a `lastname, firstname` format, we transformed it to a `firstname lastname` format to conform with DBLP.

Since affiliation information is now in more cases stripped from the name and `lastname, firstname` is transformed into `firstname lastname` we improved the linking with DBLP names (Q3, QJ1).

All these improvements resulted in a members count increase from 539,176 to 640,583. But note that we cannot give numbers nor differences whether members are actually members or parsed content that is falsely classified as members. In Figure 5.3 an update of the parser validation tree of Westerbaan [Wes22] is shown. The red path is the path that indicates successful parsing, the differences are the differences with the study of Westerbaan [Wes22].

Besides these improvements, which are targeted on the parsing of Springer LNCS front matters, we also tried to use the POC front matter for ACM issue front matter. But failed due to problems which are described in detail in Section 5.5.

5.4 ACM editorial boards

Westerbaan [Wes22] tried to scrape 39 ACM journal editorial boards, wherefrom 25 successful. The 14 that were not successful were generally 'not followed redirects' or incorrect URLs provided to the scraper. We were able to resolve 11 of them to correct URLs, leading to a set of 36 valid ACM journal URLs.

We also noticed that on `https://dl.acm.org/journals`(16-06-2022) there are 66 journals

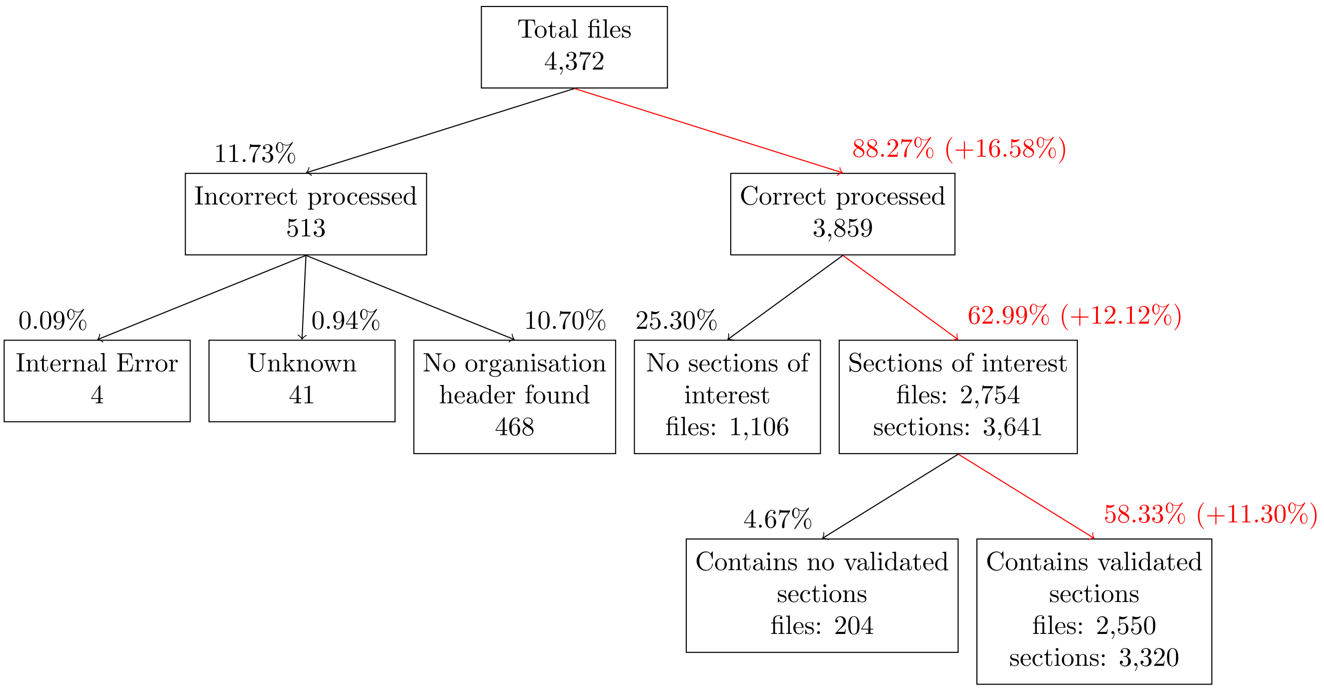


Figure 5.3: Improvement tree of LNCS front matter parsing

listed. After merging the 66 with the 36 of Westerbaan, a set of 67 journals has been constructed. By reusing the scraper and parser of [Wes22] with more correct URLs, more editor boards were scraped. The scraping of 4 journals failed because their websites did not have editorial board pages. Section A.1 contains an overview of all ACM journals used in these study, those used by Westerbaan, and whether scraping was successful or not for each journal.

Parsing of the ACM journals overview page

The journals overview page on `https://dl.acm.org/journals` contains JavaScript to lazy load the journals while scrolling down. Therefore, the page has been downloaded manually after manually scrolling down in a browser until no new journals appeared. From the downloaded page, a parser extracted all journal codes, titles, and URLs (blue parts in Listing 5.2). As shown, we were able to scrape the journal's abbreviations. Since the abbreviations always corresponded with the journals URLs, we extracted the abbreviations from the URLs for the remaining journals. Because the journal abbreviations could be scraped, we did not need an external source to link the journal titles to an abbreviation, unlike Westerbaan [Wes22].

```
<li tabindex="0"
   class="clearfix search__item search__item-browse search__item-journal tweb
  fadeInUp">
      ...
      <div class="browse-item-body">
            ...
            <h4 class="search__item-title">
                  <a href="/journal/tweb" class="browse-link">
```

```
                <span class="browse-code">tweb</span>
                <span class="browse-title">ACM Transactions on the Web</span>
            </a>
        </h4>
        ...
    </div>
</li>
```

Listing 5.2: TWEB Journal item example

(For simplicity, some irrelevant parts in the HTML are replaced with ...)

5.5 ACM issue pages and editors

In the section above, we described how we scraped the editorial boards from ACM. One inconvenience is that we do not know how long these people have been editors of their journals. In other words, in what timeline they had power in the journal. Some ACM journals have past editorial board pages, which could help, but years are not included.

Westerbaan [Wes22] medicated this problem to only investigate the year 2020 and assume that all editorial boards were correct for that year. However, in our method, it is computational performance waste to remove detected cycles that are outside the allowed time bounds after first computing all cycles. To solve this, we propose issue page scraping.

Issue page scraping
We noticed that in some issues, the chief editors are listed as authors in the issue metadata. Therefore, we scraped these to link at least the chief editor's influences to issues. Some issues even have a front matter, containing more metadata information, like the editors and reviewers of that issue (similar to the front matters of conferences in Spring LNCS), others have listed an "Editorial" "article/opinion" in their journals containing more information.

To scrape the issues we started with the most recent issues, since they all have a static URL, following the pattern: `https://dl.acm.org/toc/{journal-code}/current`. From this most recent issue page, we scrape the link to the previous issue. When scraping this previous issue, we search for the link to the preceding issue, repeating this, until there is no previous issue.

Scraping results
From each issue page we tried to scrape the following information:

- Issue metadata
  - Journal name and code
  - Volume and issue number
  - Publication date
  - Currently in progress (yes / no)
  - Publisher (if provided: publisher state and country)
  - Chief editors (if provided)
    - Name
    - Institute (if provided)
    - Profile page (if provided)
  - DOI (if provided)
- Link to previous issue

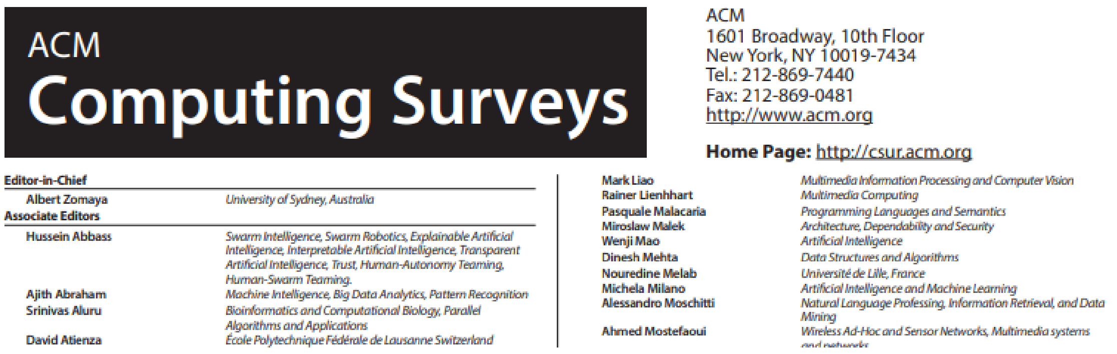
- Front matter download URL (if provided)
- Editorials
  - Title and download URL (if provided)
  - Authors of the editorial inproceeding (which are the chief editors)
    * Name
    * Institute (if provided)
    * Profile page (if provided)

Table 5.1 contains the results of our scraping. We scraped a total of 4,744 issue pages from 61 journals at March 5, 2022. The most success for acquiring editors is to download front matters, which are present on 57% of the issue pages.

What	Acquired	Potential	Rate
Journals	61	66	92%
Contains front matter	2,686	4,744	57%
Contains editorial	863	4,744	18%
Contains editors	1,667	4,744	35%

Table 5.1: Scraping results

We were able to download 2,654 front matters. The next step is to feed these front matters to the POC front matter parser. This sounds easy, but here we encountered some problems. To give an illustration of the ACM front matter, Figure 5.4 shows a modern example. Note that, just like the LNCS front matters, the format differs through the time.



ACM
Computing Surveys

ACM
1601 Broadway, 10th Floor
New York, NY 10019-7434
Tel.: 212-869-7440
Fax: 212-869-0481
http://www.acm.org

Home Page: http://csur.acm.org

Editor-in-Chief	
Albert Zomaya	*University of Sydney, Australia*
Associate Editors	
Hussein Abbass	*Swarm Intelligence, Swarm Robotics, Explainable Artificial Intelligence, Interpretable Artificial Intelligence, Transparent Artificial Intelligence, Trust, Human-Autonomy Teaming, Human-Swarm Teaming.*
Ajith Abraham	*Machine Intelligence, Big Data Analytics, Pattern Recognition*
Srinivas Aluru	*Bioinformatics and Computational Biology, Parallel Algorithms and Applications*
David Atienza	*École Polytechnique Fédérale de Lausanne Switzerland*
Mark Liao	*Multimedia Information Processing and Computer Vision*
Rainer Lienhhart	*Multimedia Computing*
Pasquale Malacaria	*Programming Languages and Semantics*
Miroslaw Malek	*Architecture, Dependability and Security*
Wenji Mao	*Artificial Intelligence*
Dinesh Mehta	*Data Structures and Algorithms*
Nouredine Melab	*Université de Lille, France*
Michela Milano	*Artificial Intelligence and Machine Learning*
Alessandro Moschitti	*Natural Language Professing, Information Retrieval, and Data Mining*
Ahmed Mostefaoui	*Wireless Ad-Hoc and Sensor Networks, Multimedia systems and networks*

Figure 5.4: Clipped illustration of area of interest of front matter of ACM Computing Surveys, Volume 54, Issue 7

POC Front matter parser requirements

The POC front matter parser needs a few instructions in order to work. First it must find its area of interest. Which for ACM front matters always seems to be page 3 of the PDF. If it contains a lot of content, the page size is enlarged or the font size is reduced by ACM, but the content always seems to stay on page 3.

Secondly, the POC front matter parser needs the roles. These are, like in the LNCS front matters, clearly identifiable as headers. And can be recognized by containing words like: `editor` or `reviewer`.

Problems
The challenge of the parsing lays within the fact that the page is split in two columns, containing each two columns (one with name and one with area of expertise or institution). The POC front matter parser of Westerbaan was not build to understand this page splitting, resulting in three main problems/challenges.

One challenge is that the second page column continues on the end of the first, which is clear to the human eye, but not for a machine. The POC front matter parser was build in the way that, for example, the `Editor-in-Chief` role is applied to anyone found below this role title, until a new role title (for example `Associate Editors`) is found. Note the below, so this also includes the second page column in the ACM case as well. To fix this, the POC front matter parser has to first parse the first page column and the second one afterwards, for example by transforming the page such that the second column is placed under the first one.

But the second challenge is that the POC front matter parser first converts the PDF to a text file, trying to fit the text in an ASCII art like raster. Because of the conversion to plain text, the boldness and font sizes of the text is lost. In order to save this information, Westerbaan [Wes22] stored this metadata for each text line. This information is then used to identify headers titles, like the person roles, since they are often of different font sizes and bold. Since the ACM front matters contain two columns, and this information is stored per line, the information in the second column gains the same font size and boldness as the left, for example, the `Editor-in-Chief` font size and boldness is also applied to `Mark Liao`. Resulting in the misinterpretation that the second column also contains a role title (`Mark Liao`).

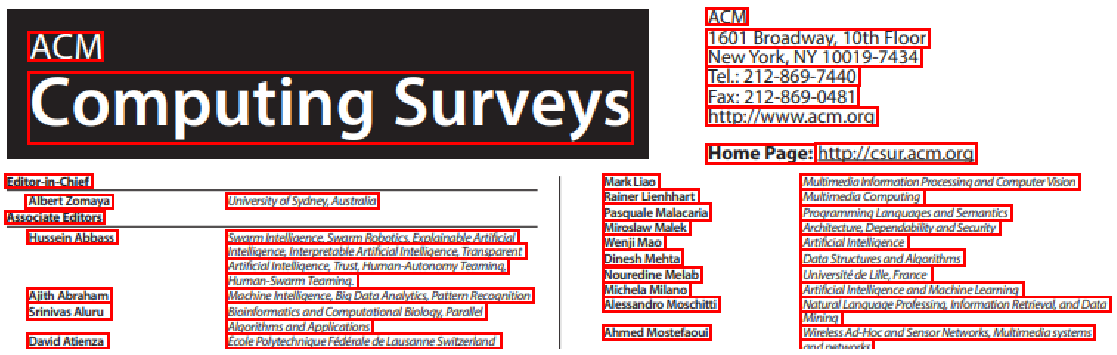
Lastly, because of the conversion to the grid raster, and the small font size, the text that needs to be written to the raster is larger than it, causing it to overflow out its position. This is especially problematic, if the overflow overlaps with text at another position, like the second page column, or exceeds the raster length. See also Listing 5.3.

```
                                                                        ACM
  ACM                                                                   1601 Broadway, 10th Floor
                                                                        New York, NY 10019-7434
  Computing Surveys                                                     Tel.: 212-869-7440
                                                                        Fax: 212-869-0481
                                                                        http://www.acm.org
                                                                        Home Page: http://csur.acm.org
Editor-in-Chief                                               Mark Liao Multimedia Information Processing and Computer
  Albert Zomaya       University of Sydney, Australia         Rainer Lienhhart Multimedia Computing
Associate Editors                                             Pasquale Malacaria Programming Languages and Semantics
                                                              Miroslaw Malek Architecture, Dependability and SecurityH
                      Human-Swarm Teaming.                    Nouredine Melab      Universite de Lille, France
  Ajith Abraham Machine Intelligence, Big Data Analytics, Pattern Recognition MichelaMilano Artifcial Intelligence an
  Srinivas Aluru Bioinformatics and Computational Biology, Parallel AlessandroMoschitti Natural Language Professing,
                      Algorithms and Applications                                  Mining
                                                              Ahmed Mostefaoui Wireless Ad-Hoc and Sensor Networks, Mu
```

Listing 5.3: Contains Figure 5.4 converted to plaintext with the POC front matter parser

Proposal
In order to solve these issues, we must choose a different approach by not first converting the PDF to a text file. Instead, we propose to identify text boxes in the PDF and detecting columns based on the starting positions and sizes of the text boxes in the PDF, see also Figure 5.5. This different approach requires a new POC front matter parser, which is unable to reuse a lot of code from the parser of Westerbaan. Because of our limited time, and expecting the development of the new POC front matter parsers will take some time, we suggest this as future work.

Conclusion
Concluding, we did not manage to include issue editors in our graph. But when the POC front matter parser can be improved in future work to also process ACM front matters, we are basically there. Also, we did scrape the issue dates, therefore we were able to link the editorial boards to recent issues only based on their dates, as will be discussed in Section 6.4.



ACM
Computing Surveys

ACM
1601 Broadway, 10th Floor
New York, NY 10019-7434
Tel.: 212-869-7440
Fax: 212-869-0481
http://www.acm.org

Home Page: http://csur.acm.org

Editor-in-Chief

Albert Zomaya	*University of Sydney, Australia*

Associate Editors

Hussein Abbass	*Swarm Intelligence, Swarm Robotics, Explainable Artificial Intelligence, Interpretable Artificial Intelligence, Transparent Artificial Intelligence, Trust, Human-Autonomy Teaming, Human-Swarm Teaming.*
Ajith Abraham	*Machine Intelligence, Big Data Analytics, Pattern Recognition*
Srinivas Aluru	*Bioinformatics and Computational Biology, Parallel Algorithms and Applications*
David Atienza	*Ecole Polytechnique Fédérale de Lausanne Switzerland*
Mark Liao	*Multimedia Information Processing and Computer Vision*
Rainer Lienhhart	*Multimedia Computing*
Pasquale Malacaria	*Programming Languages and Semantics*
Miroslaw Malek	*Architecture, Dependability and Security*
Wenji Mao	*Artificial Intelligence*
Dinesh Mehta	*Data Structures and Algorithms*
Nouredine Melab	*Université de Lille, France*
Michela Milano	*Artificial Intelligence and Machine Learning*
Alessandro Moschitti	*Natural Language Professing, Information Retrieval, and Data Mining*
Ahmed Mostefaoui	*Wireless Ad-Hoc and Sensor Networks, Multimedia systems and networks*

Figure 5.5: Contains Figure 5.4 with text boxes in red

Chapter 6

Graph construction

From different data sources, including the data acquired as described in Chapter 5, a graph has been constructed. The graph was needed to enable the detection of cycles as described in Chapter 7. As already explained in Chapter 4, we have chosen the following data sources: DBLP, OpenCitations, Springer LNCS front matters, and ACM editorial boards and issue pages. We have chosen this data sources to enable comparison with Westerbaan's thesis, by choosing the data sources that Westerbaan had also chosen in his thesis, but improved these as mentioned in Chapter 5.

The integration of all these data sources delivered the graph model in Figure 6.1. The colors in the figure represent the different data sources: black is DBLP, blue is OpenCitations, violet is ACM editorial boards and issue pages, and teal is Springer LNCS front matters.

DOIs and ORCIDs should typically be modelled as properties of nodes. An article can contain multiple DOIs and a person multiple ORCIDs, therefore an array property is required, but Neo4j does not allow indexes on array properties. To allow indexes on these properties to speed up searching performance, these IDs had to be modelled as separate nodes. They are therefore marked in gray in Figure 6.1.

In this chapter, we discuss the different data sources in each section separately (Section 6.1-6.3). Each section also contains a paragraph regarding the accuracy of the data in the data source and possible problems with the integration with other data sources, referring to the data quality and integration issues listed in Section 5.1.

6.1 DBLP

DBLP is a computing science bibliography which provides an XML file that contains all its data for download.[1] Westerbaan used this source and parsed the XML file to a Postgres database. We downloaded a more recent version of the XML file and we modified the interpreter of Westerbaan to export the data into CSV files instead of the Postgres database, to allow the data to be imported into Neo4j.

Westerbaan chose to import the whole DBLP XML file as general as possible. But we process and parse the data so it fits into a graph model, so we could and will not include everything. The DBLP XML contains the following elements: `article` (article in journal), `inproceeding`

[1]`https://dblp.org/xml/release/`

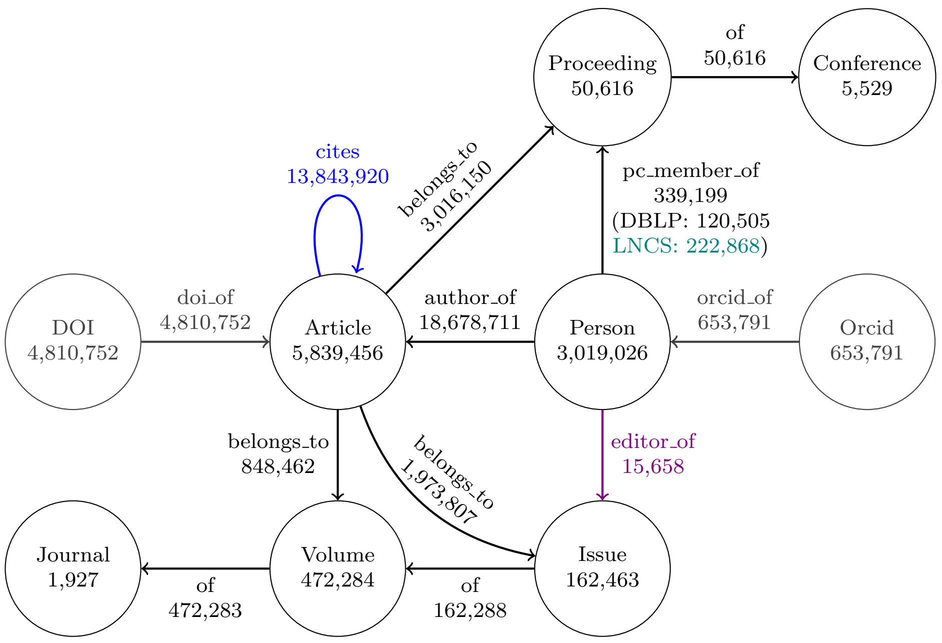


Figure 6.1: Graph model with counts

(article in conference proceeding), `incollection` (article in a collection or book), `proceeding` (proceeding of a conference), `book` (a book written by authors or a collection of book chapters each written by different authors), `phdthesis`, `mastersthesis`, and `www` (webpage, frequently used to link persons to webpages) [Ley09]. We only processed `article`, `inproceeding`, and `proceeding` XML elements, since these can be expected to be properly peer reviewed and cover the methods of fraud described in Section 2.3. Besides, we expect that the inclusion of `phdthesis` and `mastersthesis` does not influence someone's metrics that much, since persons typically don't write multiple of these in their career.

For simplicity, `articles`: articles in journals and `inproceedings`: articles in conference proceedings, are both called articles from now on.

Besides the XML elements, there is information that can be deducted from the XML attributes, or included as XML elements. From these elements and attributes, we extracted extra nodes and relationships, like persons (that can be authors or editors), conferences, issues, volumes, and journals. We learned due to overlap between the LNCS PC members and the editors of DBLP that the editors of proceedings in DBLP are PC members.

After processing the XML file and writing the results to CSV files, the CSV files have been imported to Neo4j.

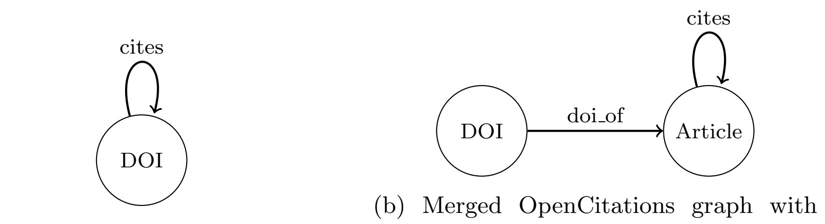
Accuracy

DBLP has some challenges with person names, as described in Section 4.1.2 and DOIs, as described in Section 5.2. All the other accuracies we encountered are listed under the accuracy of integrated data sources, which are all the other data sources.

After our first cycle occurrence analyzation, we noticed some invalid cycles, described in Section 8.1. To correct these, we removed 94 double `author_of` relationships (Q2). We also reinterpreted all 120504 `editor_of` relationships between `Person` and `Proceeding` as `pc_member_of` relationships (Q3, QJ3). And we disallowed double `belongs_to` relationships of `Article` to both `Proceeding` and `Issue` (375 times) or `Volume` (2114 times), in which case we hold the relationship to `Proceeding` and removed the others (Q3).

6.2 OpenCitations

Like Westerbaan, we used the OpenCitations data source to add cite relationships to the graph. Figure 6.2a contains the graph presentation of the OpenCitations data. This dataset has been merged with that of DBLP (Section 6.1) by using the `doi_of` relation leading to the sub graph in Figure 6.2b. The data provided by OpenCitations is already provided as CSV[2] therefore it has been imported directly into Neo4j.



(a) OpenCitations graph

(b) Merged OpenCitations graph with DBLP

Figure 6.2: OpenCitation graphs

Accuracy
Reasons for incorrect data in the join with DBLP:

- The OpenCitations dataset contains a larger set of articles that are out of scope for DBLP (like non ICT articles) (QJ1).
- The article in DBLP did not have a DOI (QJ1, Q3) (see also Section 5.2).
- The article in DBLP did have a different DOI than the DOI used by OpenCitations (QJ1).
- The DOI in the DBLP or OpenCitations dataset is not correct (Q2, QJ2).

6.3 Springer LNCS PC members

We used the LNCS front matters scraped by Westerbaan [Wes22]. We reparsed these front matters after the improvements we made to the POC front matter parser discussed in Section 5.3. The results were written to a SQL database, like in the study of Westerbaan [Wes22].

To acquire the LNCS PC members, we selected all members the role matching the SQL search `'%Program%'`. The roles with the corresponding member counts are listed in Table 6.1. We assume all members in these roles are program committee members (PC members) of their conferences. The graph representation of this relation is shown in Figure 6.3. We saved the results of the SQL query as CSV to import it into Neo4j. From the 294,307 members we were able to link 224,750 (76%) to persons in the DBLP dataset, by naive linking to the DBLP name (for a detailed discussion on persons name matching see also Section 4.1.2).

[2]`https://opencitations.net/download`

Role	**Member count**
Program Committee	131,488
Technical Program Committee	112,416
Program Committee Members	8,687
Programme Committee	3,729
Program Committee and Reviewers	3,602
Senior Program Committee	2,270
Program Chairs	2,238
Programme Committee and Reviewers	2,180
Other matched roles are below 2,000 members	
Total	294,307

Table 6.1: PC Member roles with member counts above 2,000[3]

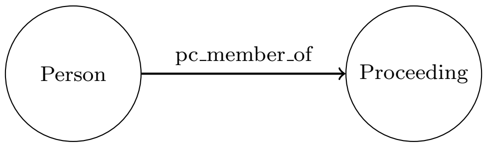


Figure 6.3: LCNS PC members graph

Accuracy
Reasons for invalid data in the join with DBLP:

- The proceeding or the person name or abbreviation is not found in the DBLP dataset. 83,254 of 294,307 person-proceeding combinations (28.29%) (QJ1)
- There may exist multiple persons with the same name in the DBLP dataset, it may happen we made the wrong person PC member. This could be the case for 2,645 of the 53,722 PC Members (4.92%) (QJ3) (see also Section 4.1.2).
- The PC member name is an abbreviation and matched to another person in the DBLP dataset (QJ3) (see also Section 4.1.2).
- The person in the dataset is incorrectly classified as a PC member. As can be seen in the full table of Table 6.1, it is not likely that all roles are PC members. But these numbers are very low (Q2, QJ2).

6.4 ACM editors

Like Westerbaan, we also scraped the ACM editorial boards pages to acquire the editors of journals. In Section 5.4 we discussed our improvements on this. We also discussed in Section 5.5 our idea to scrape ACM issue pages and download front matter pages, to acquire historical editorial boards and thereby directly linking editors and reviewers to issue level instead of journal level. But due to limitations of the POC front matter parser and time constraints to build a new POC front matter parser, we did not succeed in this. However, the issue pages that were scraped contained their publication dates. These publication dates can be used to link editorial boards to recent issues pages instead of the journal. This limits the editorial power only to recent issues, instead of the whole journal always. Westerbaan [Wes22] investigated the year 2020 in

[3]Full table is published on `https://github.com/XibrenX/MasterThesis/blob/main/data/program_search.csv`

his study. So we decided to starting linking issues after January 1, 2020 to also cover the range of Westerbaan.

Figure 6.4 shows the graph after merging, the scraped ACM editorial boards, with the scraped issue pages and DBLP. The gray, dashed relationship shows the graph model when not limiting on issue dates. Not all journals have issues, some only have volumes, in that case we link to volumes instead. Unfortunately, none of these journals matched all merging steps (must have editorial boards, volume pages, and exists in DBLP), so this relationship is not shown in our final graph model.

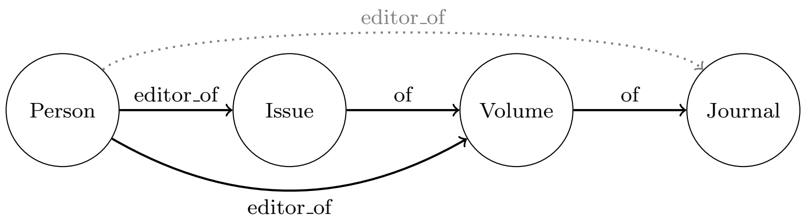


Figure 6.4: ACM editors graph

Accuracy
Reasons for incorrect or missing data:

- Did not have issue page or editorial board page: 10 of 67 journals (QJ1). For more detail, see Section A.1.
  - Did not have an editorial board page: 5 of 67 journals (Q1).
  - Did not have issue page: 6 of 67 journals (Q1). Missed therefore 198 editors.
- Journal abbreviation not found in DBLP: 8 of the 57 journals with issue pages and editorial boards (QJ1).
- ACM journal abbreviations does not match with DBLP journal abbreviations, a quick scan revealed that this is not very likely, but should be regarded in future work (QJ3).
- Issue or volume not found in DBLP because DBLP did not (yet) include it: 102 of 508 volumes and issues (20%) (Q1.1, Q1, QJ1).
- The person name or abbreviation is not found in the DBLP dataset: 7,038 of 30,179 (23%) (QJ1).
- There may exist multiple persons with the same name in the DBLP dataset, it may happen we made the wrong person editor. This could be the case for 236 of the 2361 editors (10%) (QJ3) (see also Section 4.1.2).
- The editor name is an abbreviation and matched to another person in the DBLP dataset (QJ3) (see also Section 4.1.2).
- The ACM editorial board webpage is not displaying the current editorial board (Q1.1, Q2, QJ2).
- Recent changes in the editorial board are enlarged by us over a time period of 3 years, which may wrongly enlarged the power of new editors, and neglected that of recently resigned ones (Q1.1, Q2, QJ2).
- The editor is linked to an issue where he/she was not editor, because he/she joined the editorial board recently (Q2)

Note that we are paying a high price to link editorial boards to recent issues, but we have to do it to prevent that editors were linked to the whole journal. Meaning, they are linked to all issues and volumes, even very old ones. A better solution would be to redesign the POC front matter parser to also parse ACM front matters, and thereby acquiring more accurate data for old issues (see also Section 5.5).

Chapter 7

Efficiently detecting cycles

The idea behind cycles is that when someone or some group commits fraud, they do so to benefit from it. Therefore, the fraudulent action will, in the chain of events, eventually end up with the person or group that committed the fraud. So when correctly modelled, methods of fraud should be detectable by cycles in a graph is our assumption. For trying this idea, cycle detection was needed. We start in Section 7.1 with discussing the terminology used in this chapter. In Section 7.2 we discuss several approaches of retrieving or detecting cycles, and chose the one that performed the best. The next step after detecting the cycles is to group them by their type, this is harder than it seems and will be discussed in Section 7.4. After grouping the cycles, we discuss the found cycle types in Chapter 8. Two detected cycle types were taken for detecting outliers in Chapter 9.

Regarding performance, we limited the number of edges in a cycle to 6. More than 6 relations/edges resulted in too long runtimes. Most of the fraud methods mentioned in Section 2.3 could be expressed with less than 6 edges, see also Section 8.2 for the encoding of these fraud methods as cycle types.

7.1 Cycle terminology

In this thesis, we use the following definition of a cycle.

Definition 7.1.1 (Cycle)**.** A cycle is a path in a graph that starts and ends with the same node, not containing an edge or node multiple times in between.

Start node (S) As start/end node we choose the person or group (conference, journal, institute) we want to investigate. From this node, we try to return to this node by following a path.

Path In a graph, a path can be formed as a set of edges with their nodes. A path is not allowed to contain an edge or node multiple times.

Path length The length of a path is described in the number of edges it contains. When the path is a cycle, the path length is also the **cycle length**. In our graph, we don't have edges from and to the same node, so the cycle length is always larger than 1.

Type Cycles can be grouped by their type or pattern. This type is derived from the node types, relationship types and relationship directions in the path. For example, there are many cycles in our graph containing different nodes, like different articles and persons, that all have the self-citation cycle type: $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$

Label The label of a cycle is an abbreviation of its type. Each node in the path is labelled by a capital letter, the first letter of its node type. For example, `Article` becomes A. One exception: `Proceeding` becomes `R` since the `P` has already been given to `Person`. Each relationship is labelled by a lower letter, the first letter of the relationship type. For example: `editor_of` becomes `e`. Here also one exception: `of` becomes `f` since `o` has already been given to `orcid_of`. Relationship directions can be derived from the neighbour node labels, because most relationship types are always between different node types. The exception is the `cites(c)` relationship, since is goes from `Article(A)` to `Article(A)`. For the `cites(c)` relationship type, an arrow is needed to indicate the direction: $\overleftarrow{\texttt{c}}$ $\overrightarrow{\texttt{c}}$. For example, a cycle with type: $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xleftarrow{\texttt{pc\_member\_of}}\texttt{Person}$ has label $\texttt{PaA}\overrightarrow{\texttt{c}}\texttt{AbRpP}$. The full list of node and relationship abbreviations can be found in Section A.2.

Mirror Each cycle path always has a mirrored path that describes the same cycle in the opposite direction, because it is possible to walk a cycle clockwise or anti-clockwise. For example, the following path labels describe the same cycle but in opposite direction:

- $\texttt{PaA}\overrightarrow{\texttt{c}}\texttt{AbRpP}$
- $\texttt{PpRbA}\overleftarrow{\texttt{c}}\texttt{AaP}$

This is even the case with palindrome cycle labels like `PaAaPaAaP` (co-author cycle type), but the mirror is not visible from its label but on its node level. Since the same edge is not allowed to be visited twice, we know that both nodes with label `A` should be different, so its mirror has the same label, but swapped both `A` labelled nodes and their connected relationships. For example, the mirror of $P_1 a_2 A_3 a_4 P_5 a_6 A_7 a_8 P_1$ is $P_1 a_8 A_7 a_6 P_5 a_4 A_3 a_2 P_1$.

7.2 Methods

There are two main ways to retrieve or detect cycles with Neo4j.

1. Using Neo4j to construct and return the cycles.
2. Retrieving a subgraph with relevant nodes/paths from Neo4j and detect the cycles by ourself.

For the first way, we investigated the following approaches:

1a. Using the help library apoc for Neo4j `apoc.nodes.cycles`.
Cypher: `MATCH (s) WITH collect(s) as n CALL apoc.nodes.cycles(n, maxDepth: 6) YIELD path RETURN path`
Unfortunately, this method contained no option to ignore directions in our graph.

1b. Using `(s)-[*]-(s)` a Cypher language query that describes cycles.
Cypher: `MATCH p=(s)-[*2..6]-(s) RETURN p`

For the second way, we investigated the following approaches to retrieve a sub graph to detect the cycles in. Note that to construct cycles up to 6 edges, we only have to retrieve a subgraph with max depth 3 from the start node, since to reach a leaf node is 3 steps, and back is also 3, summing up to 6. (see for example Figure 7.1).

2a. Retrieving all paths.
Cypher: `MATCH p=(s)-[*0..3]-() RETURN p`

2b. Retrieving the end of all paths, Neo4j returns the paths in order of increasing length.
Cypher: `MATCH (s)-[*0..2]-()-[r]-(n) RETURN r,n`

2c. Using `apoc.path.subgraphAll`.
Cypher: `MATCH (s) WITH s CALL apoc.path.subgraphAll(s, {maxLevel:3}) YIELD nodes, relationships RETURN nodes, relationships`

2d. Using the Neo4j library Graph Data Science Library with BFS algorithm. This requires creating an in memory Graph Data Science project with nodes and relationships we want to include in our investigation. We created a project with all nodes and relationships except the property nodes with relationships: `DOI` and `Orcid` since they are not bound to multiple nodes and therefore will not be part of cycles. In this project, we had to explicitly list all relationships to be interpreted as undirected.
Cypher: `MATCH (s) WITH s CALL gds.bfs.stream('project', {sourceNode:ID(s), maxDepth:3}) YIELD nodeIds MATCH (n) WHERE ID(n) in nodeIds WITH collect(n) AS n, nodeIds MATCH (n1)-[r]->(n2) WHERE ID(n1) in nodeIds AND ID(n2) in nodeIds RETURN n, collect(r) AS r`

The performance and cycle count per method is listed in Table 7.1. To compare performance, we detected cycles for person X. Person X was encountered during our analysis and had many cycles, we choose person X since more cycles means more computing time, therefore the time differences between the methods are less influenced by other running computer processes. We ran the queries 10 times to compensate for Neo4j query cash.[1] The calculation times are hard to compare, approximately 90% of the calculation time is the parsing of the query data stream to programming language objects, and filtering and storing these objects. Besides, the calculation times we experienced were only a few seconds, while the retrieval time often exceeds the calculation time. The calculation time for method 2 is the time it cost to run the method described in Section 7.3.

Method	**Retrieval time**	**Calculation time**	**Cycle count**
1b	2:55:23.504	approx 1 sec	31,020,124 (including mirrors and duplications)
2a	11.281	approx 3 sec	2,208,358 (78,321 recursive)
2b	2.650	approx 3 sec	2,208,358 (78,321 recursive)
2c	12:58.443	approx 3 sec	2,208,358 (78,321 recursive)
2d	11.037	approx 3 sec	2,208,358 (78,321 recursive)

Table 7.1: Performance of queries per method for person X

From the results in Table 7.1 we have drawn our conclusions. Approach 1b works, but has very long runtimes in comparison to all others, also some post-processing is needed to filter out cycle mirrors and other increasing cycle number factors. Surprisingly, we conclude that detecting cycles by ourselves is performing far better than letting Neo4j do it, partly because we can do some optimizations in the method itself, limiting the amount of wasted cycle detections. Method

[1] `https://stackoverflow.com/questions/31579763/how-to-compare-performance-on-neo4j-queries-without-cache`

2b performed the best, so we chose for 2b.

During the performance analyses, we observed that Neo4j was using one core at 100%. When looking at the table and this observation, we think that the most performance issues come from streaming the data from the database to the query requestor, since method 2a and 2b should not differ too much in calculation but do differ in keeping track of walked paths and amount of streamed data.

7.3 Detecting the cycles

We chose for detecting the cycles by ourselves. This approach is further explained in the following sections, each describing a step.

Section 7.3.1 The cycles must be detected in a graph, therefore first a subgraph must be constructed.

Section 7.3.2 Next, we must choose a cycle detection algorithm.

Section 7.3.3 For efficiency reasons, we introduce recursive paths to group equally labelled paths from and to the same nodes.

Section 7.3.4 This recursive paths allow simple construction of recursive cycles, from this recursive cycles normal cycles can be reconstructed if needed, in practice we do not need this, because of the grouping of cycles explained in Section 7.4.

7.3.1 Extracting the subgraph

Method 2b provides us with the paths and nodes from the start node nicely in layers. So first all paths with length 1, then 2, then 3. In order to construct the graph, we add the retrieved edges and nodes, layer by layer to construct a subgraph. See also Figure 7.1.

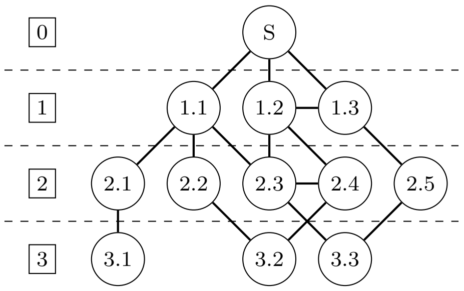


Figure 7.1: Subgraph layer example

Remove leaf nodes The next step is to remove leaf nodes with only one relation. These leaf nodes will never be present in a cycle since they only have one relationship. For example, node 3.1 in the subgraph of Figure 7.1. This operation is recursive, since 2.1 becomes a leaf node after node 3.1 has been deleted, so node 2.1 has to be checked again. We do this by iterating over the nodes, checking if it is a leaf node, and when it is deleting it. When deleting a node, we also check (sometimes again) the node on the other end of the deleted relation.

7.3.2 Cycle detection algoritms

For the cycle detection, we tried two algorithms: Depth First Search (DFS) and Breath First Search (BFS).

DFS First we tried Depth First Search (DFS). We started in the start node, and when we found a path to the start node again, we have a cycle, when we find a path to an already visited node we have found an inner cycle. For example, in Figure 7.1, S-1.2-2.3-2.4-1.2 contains the inner cycle 1.2-2.3-2.4. This algorithm was dismissed for the reason that we did not have a way to constrain path length. For example, in the subgraph a cycle S-1.1-2.2-3.2-2.4-1.2-2.3-3.3-2.5-1.3-S is possible, which is way too long and complicated for our analysis, and in large subgraphs this kind of paths result in stack overflow exceptions. We could decide to stop searcher after for example length 5, but this raised the problem that we visited 1.2 and stopped there while not fully exploring its edges. This leaves some nodes visited but not fully explored. This breaks the algorithm.

BFS Second thought is a layered approach where only depth labelled: 0-1-0, 0-1-1-0, 0-1-2-1-0, 0-1-2-2-1-0, 0-1-2-3-2-1-0, or 0-1-2-3-3-2-1-0 cycles are allowed. For this, the Breath First Search (BFS) algorithm fits the best since it uses a layered way of exploring. We already knew the layers from our subgraph construction, so we do not need the queue of the algorithm, instead we order all nodes by layer in one queue. When visiting a node, we explore all paths to lower layered nodes and store the shortest paths to the start node. A node can have multiple shortest paths, for example, in Figure 7.1, node 2.3 has 2.3-1.1-S and 2.3-1.2-S. When a node has multiple shortest paths, it is part of a cycle. When combining all possible shortest path in a node we construct the cycles for example 2.3-1.1-S and 2.3-1.2-S become S-1.1-2.3-1.2-S. After exploring all paths to lower layered nodes, we look for paths to same level nodes which have already been visited (when they have not been visited no shortest paths are known, but they will look at this node when they are visited), if such node exists we combine the shortest paths of two nodes to get new cycles. For example, we already knew that node 2.3 has paths 2.3-1.1-S and 2.3-1.2-S, node 2.4 has the shortest path 2.4-1.1-S. So combining the paths gives the cycles S-1.1-2.3-2.4-1.1-S and S-1.2-2.3-2.4-1.1-S. In this way, we find all cycles with max depth 3 and max length 6 in our subgraph.

7.3.3 Recursive paths

The shortest paths in the BFS algorithm can often be simplified. For example, node with type C in Figure 7.2 contains 3 shortest paths to S: $\texttt{Sa}_1\texttt{A}_1\texttt{c}_1\texttt{C}$, $\texttt{Sa}_2\texttt{A}_2\texttt{c}_2\texttt{C}$, and $\texttt{Sb}_1\texttt{Bc}_3\texttt{C}$. So despite the fact that there are two different nodes, their paths have the same labels: `SaAcC`. To combine paths in a recursive path, there are two rules:

1. The paths must be of the same type.
2. The end node (E) of the paths must be the same (C in the example)

After applying, we can say that C has two shortest recursive paths to S: $\texttt{Sa}_{\{1,2\}}\texttt{A}_{\{1,2\}}\texttt{c}_{\{1,2\}}\texttt{C}$ and $\texttt{Sb}_1\texttt{Bc}_3\texttt{C}$.

Definition 7.3.1 (Recursive path)**.** A recursive path (R) is a set of recursive path segments where $\forall x, y \in R[\texttt{type}(x) = \texttt{type}(y) \wedge x_E = y_E]$.

Definition 7.3.2 (Recursive path segment)**.** A recursive path segment (T) is defined as the triple: $T = \langle r, E, R \vee S \rangle$ where r is the relation between the end node E with the recursive path of the previous node R or the start node S.

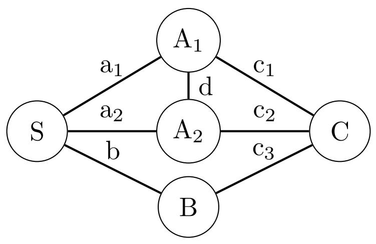


Figure 7.2: Example graph to illustrate recursive paths

This solution simplifies less relevant parts of cycles. A great example are the co-author paths ($\texttt{P}_1\texttt{aAaP}_2$), since the definition of a co-author is that a person writes articles with another person (the co-author), however a frequent publisher can write many articles with the same co-author. The fact that someone is a co-author can be considered more important than all separate article paths between the two persons, sometimes the number of articles are important, but not each single article. Using this solution, all paths to the co-author can be simplified as one recursive path and enables analysis as the total number of articles between co-authors ($\{\{\texttt{P}_1\texttt{aA}\}\texttt{aP}_2\}$).

Counting sub paths To retrieve the sub path count of a recursive path, all sub paths have to be counted. We use a recursive algorithm to sum up all paths from the start node to the end of the recursive path.

$$\texttt{count}(S) = 1$$
$$\texttt{count}(R) = \sum_{t \in R} \texttt{count}(t_{R \vee S})$$

7.3.4 Recursive cycles

Recursive paths can form recursive cycles. This can happen in 3 ways, described as 3 kinds of recursive cycles.

1. A recursive path can itself be a recursive cycle. This is the case when a recursive path contains 2 or more segments. For example, the `SaAcC` recursive path in Figure 7.2 a cycle exists from and to S: $\texttt{Sa}_1\texttt{A}_1\texttt{c}_1\texttt{Cc}_2\texttt{A}_2\texttt{a}_2\texttt{S}$. This recursive cycle kind has always a palindrome label, since the label of the recursive path (`SaAcC`) is attached in mirrored order (`SaAcCcAaS`).
2. Secondly, if there are 2 or more recursive paths to a node, all these recursive paths can be combined to form cycles. For example, the `SaAcC` and `PbBcC` in Figure 7.2 can be combined as `SaAcCcBbS`. Containing 2 cycles: $\texttt{Sa}_1\texttt{A}_1\texttt{c}_1\texttt{Cc}_3\texttt{BbS}$ and $\texttt{Sa}_2\texttt{A}_2\texttt{c}_2\texttt{Cc}_3\texttt{BbS}$.
3. Thirdly, if there exists a relation between two nodes of the same depth, for example `d` in Figure 7.2. All recursive paths of both nodes can be combined to form recursive cycles. In the example $\texttt{Sa}_1\texttt{A}_1$ and $\texttt{Sa}_2\texttt{A}_2$ will be combined using `d` as $\texttt{Sa}_1\texttt{A}_1\texttt{dA}_2\texttt{a}_2\texttt{S}$.

The number of normal cycles in a recursive cycle depends on its type:

1. Number of cycles in recursive cycle kind 1 R:
   $\texttt{index}(x, X)$ is the position of x in set X.

$$\sum_{x,y \in R} \texttt{index}(x, R) < \texttt{index}(y, R) \wedge x_r \neq y_r \rightarrow \texttt{count}(x) * \texttt{count}(y)$$

2. Number of cycles in recursive cycle kind 2 $\langle R1, R2\rangle$:

$$\sum_{x\in R1}\sum_{y\in R2} x_r \neq y_r \rightarrow \texttt{count}(x) * \texttt{count}(y)$$

3. Number of cycles in recursive cycle kind 3 $\langle R1, r, R2\rangle$:

$$\sum_{x\in R1}\sum_{y\in R2} \texttt{count}(x) * \texttt{count}(y)$$

The $x_r \neq y_r$ part is needed to prevent that the same path is used multiple times in one circle, this could happen when two recursive paths that have different labels but with the same edge (r) are tried to combine. For example, `SaAbBcC` may try to combine with `SAAeBcC`, this is denied for cycles that use the same c relationship.

7.4 Grouping cycles and removing duplicates

As mentioned in Section 7.1 each cycle can be described by two possible paths, clockwise or its mirror, anti-clockwise. So to eliminate possible cycle duplicates in a set, we must check if one of these two possible paths is already present in the set. This problem becomes even harder when grouping cycles by label, since the grouping should group based on both possible path labels. For example, when grouping self-citation cycles, the grouping algorithm needs to check for two labels $\texttt{PaA}\overrightarrow{\texttt{c}}\texttt{AaP}$ and its mirror $\texttt{PaA}\overleftarrow{\texttt{c}}\texttt{AaP}$.

Instead of checking for both directions of the path, we propose to prefer one of the two possible paths and transform the other always to the preferred one. Like, clockwise is preferred for clocks to "increase performance" and avoid confusion when reading time. We propose the following preference for cycles in our graph:

For non palindrome labelled cycles We prefer the direction with the earliest alphabetic letter on the left. For example, we prefer `PaAbRpP` over `PpRbAaP` since the letter "a" occurs earlier in the alphabet than "p". For palindrome labelled cycles, this does not work, since its mirror has the same label.

Palindrome labelled cycles that contain `c` as explained in Section 7.1 the `cites` relationship needs its direction to distinguish between two different cycle types. So when we have a palindrome labelled cycle with a `c`, we prefer left-to-right over right-to-left relationships. For example, $\texttt{PaA}\overleftarrow{\texttt{c}}\texttt{AaP}$ is preferred over $\texttt{PaA}\overrightarrow{\texttt{c}}\texttt{AaP}$. There are palindrome labelled cycles that contain `c` which cannot be ordered in this way, like $\texttt{PaA}\overrightarrow{\texttt{c}}\texttt{A}\overleftarrow{\texttt{c}}\texttt{AaP}$, they are also allowed in the method below.

Palindrome labelled cycles The cycles that are labelled in this way cannot be preferred by one of their two possible path labels since their mirrors have exactly the same labels. This is because they are duplicates by themselves. For example, take the cycle $\texttt{P}_1\texttt{a}_2\texttt{A}_3\texttt{a}_4\texttt{P}_5\texttt{a}_6\texttt{A}_7\texttt{a}_8\texttt{P}_1$ and its mirror $\texttt{P}_1\texttt{a}_8\texttt{A}_7\texttt{a}_6\texttt{P}_5\texttt{a}_4\texttt{A}_3\texttt{a}_2\texttt{P}_1$ are both cyclic representations of the path $\texttt{P}_1\{\texttt{a}_2\texttt{A}_3\texttt{a}_4,\ \texttt{a}_8\texttt{A}_7\texttt{a}_6\}\texttt{P}_5$. Note that this is actually a recursive path, as discussed in Section 7.3.3, and a recursive cycle of kind 1 in Section 7.3.4. So when grouping these cycle types, we double fold them as in the recursive path. By doing so, the ordering becomes clear: the start node is preferred left, and inside the recursive path cycles can be reconstructed by combining each half on the condition that they have the same start and end nodes. We can remove duplicates since each half is only allowed to occur once.

Chapter 8

Cycles types

By using the method described in Chapter 7 to detect cycles, we let our cycle detection algorithm detect all cycles starting from all 3,019,026 `Person` nodes in our graph, with max 6 edges. It took us around 4 days, with an average detection time of 10 persons each second, on a large desktop computer.

We ran this analysis two times. The first time we encountered some invalid cycles, due to problems in our graph, data quality, and integration. These are described in Section 8.1. After resolving these problems, we ran the analysis again, reducing the amount of found cycle types from 366 to 233.

The results of the second run can be found in Section A.3, the table there contains cycle types and the number of occurrences. When looking at the cycle types, we tried to classify those types as indicative for scientific fraud. Some scientific fraud can be described in a graph, see also Section 2.3, and can be translated to cycle types in our graph, which can also be found in our results. These cycle types are listed in Section 8.2. However, transforming cycle types back to possible methods of scientific fraud is hard and requires domain knowledge, which is explained in Section 8.3.

8.1 Invalid cycles types

In this section, we describe the invalid cycles we encountered the first time, and how we prevented them from occurring in the next run.

- One of those is the $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ cycle. In other words, a person that is a co-author of his/her own paper. A quick investigation showed us that this happened most often with Chinese named persons, probably because one person wrote a paper with another person with the same name and DBLP linked it wrongly two times to the same person (Q2).
- Another example is that a person can have multiple roles in the same proceeding, providing the cycle $\texttt{Person}\xrightarrow{\texttt{pc\_member\_of}}\texttt{Proceeding}\xleftarrow{\texttt{pc\_member\_of}}\texttt{Person}$. For example, Antonio Cerone has the roles “InSuEdu 2012 Program Co-chairs”, “MoKMaSD 2012 Program Committee”, “MoKMaSD 2012 Program Co-chairs”, and “InSuEdu 2012 Program Committee” in the “sefm-2012s” conference code, leading to 4 `pc_member_of` relations to the same proceeding.
- The same cycle also occurred when a PC member was registered both in DBLP and LNCS. This is also the case in the example with ’Antonio Cerone’, resulting in a total of 5 `pc_member_of` relations to the same proceeding.

- We detected parts in cycles that contained $\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\{\texttt{Issue,Volume}\}$. In other words, an article that is both a journal article and a conference article. In the study regarding DBLP [Ley09] they notice this because Springer LNCS articles are published in a proceeding, but are also part of a journal, where the journal represents the conference. The proceedings themselves are also published as volumes in the Springer LNCS journal.

This examples results in cycle types that should actually not exist and increases the numbers of other cycles types that should have been lower. We solved this by removing duplicated `author_of` relationships, and merging multiple `pc_member_of` relationship to one, while collecting the roles in an array property of the relationship.

Since DBLP names the relationship between $\texttt{Person}\rightarrow\texttt{Proceeding}$ an editor, we first added it as an editor relationship in our graph. This resulted in complex cases since the editor relationship could not be merged with the `pc_member_of` relations, leading to many forms of the same cycle type (`PeRpP` = `PpReP` = `PpRpP`). Therefore, we decided to rename `editor_of` when pointing to a `Proceeding` to `pc_member_of` to enable merging with LNCS `pc_member_of` relations. Now, only the cycle type `PpRpP` should occur.

To prevent the Springer LNCS journal-proceeding cycles, and to avoid confusion of articles belonging to both a `Proceeding` and `Issue` or `Volume` we have chosen to ignore relations to `Issue` or `Volume` when there is a reference to a `Proceeding` in DBLP. We already ignored the volume references of `Proceedings`.

8.2 Known suspicious cycles types

From literature and our academic experience, some methods to fraud technics are already known and can be expressed by cycles in a graph, as explained in Section 2.3. These fraud methods can be detected as cycles in our graph since they can be translated to cycle types, as we show below. Some of these cycle types exceed the max length of 6 edges, therefore they cannot be found in our list of found cycle types (in Section A.3).

Demanding citations as editor or PC member: (Section 2.3.1)

$\texttt{Person}\xrightarrow{\texttt{editor\_of}}\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}_x\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PeIbA}_x\vec{\texttt{c}}\texttt{AaP}$)

where not $\texttt{Article}_x\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{A}_x\texttt{aP}$)

$\texttt{Person}\xrightarrow{\texttt{pc\_member\_of}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PpRbA}_x\vec{\texttt{c}}\texttt{AaP}$)

where not $\texttt{Article}_x\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{A}_x\texttt{aP}$)

Demanding co-authorship as editor or PC member: (Section 2.3.1)

$\texttt{Person}\xrightarrow{\texttt{editor\_of}}\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PeIbAaP}$)

$\texttt{Person}\xrightarrow{\texttt{pc\_member\_of}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PpRbAaP}$)

Self-citation as author: (Section 2.3.2)

$\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PaA}\vec{\texttt{c}}\texttt{AaP}$)

Citation stacking as author:

$\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}_x\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PaA}\vec{\texttt{c}}\texttt{A}_x\vec{\texttt{c}}\texttt{AaP}$)

where not $\texttt{Article}_x\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{A}_x\texttt{aP}$)

$\texttt{Person}_1\xrightarrow{\texttt{author\_of}}\texttt{Article}_1\xrightarrow{\texttt{cites}}\texttt{Article}_2\xleftarrow{\texttt{author\_of}}\texttt{Person}_2\xrightarrow{\texttt{author\_of}}\texttt{Article}_3\xrightarrow{\texttt{cites}}\texttt{Article}_4\xleftarrow{\texttt{author\_of}}$

$\texttt{Person}_1$ ($\texttt{P}_1\texttt{aA}_1\vec{\texttt{c}}\texttt{A}_2\texttt{aP}_2\texttt{aA}_3\vec{\texttt{c}}\texttt{A}_4\texttt{aP}_1$)

where not $\texttt{Article}_{\{1,4\}}\xleftarrow{\texttt{author\_of}}\texttt{Person}_2$ ($\texttt{A}_{\{1,4\}}\texttt{aP}_2$) and not $\texttt{Article}_{\{2,3\}}\xleftarrow{\texttt{author\_of}}\texttt{Person}_1$ ($\texttt{A}_{\{2,3\}}\texttt{aP}_1$)

Venue self-citation as journal or conference: (Section 2.3.4)

$\texttt{Journal}\xleftarrow{\texttt{of}}\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{JfVfIbA}\vec{\texttt{c}}\texttt{AbIfVfJ}$) (length is 7, so exceeds max length of 6)

$\texttt{Conference}\xleftarrow{\texttt{of}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xrightarrow{\texttt{of}}\texttt{Conference}$ ($\texttt{CfPbA}\vec{\texttt{c}}\texttt{AbPfC}$)

Or sub cycles like:

$\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}$ ($\texttt{VfIbA}\vec{\texttt{c}}\texttt{AbIfV}$)

$\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}$ ($\texttt{IbA}\vec{\texttt{c}}\texttt{AbI}$)

$\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}$ ($\texttt{RbA}\vec{\texttt{c}}\texttt{AbR}$)

Venue citation stacking as journal or conference: (Section 2.3.4)

$\texttt{Journal}\xleftarrow{\texttt{of}}\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xrightarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ $\xleftarrow{\texttt{of}}\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{JfVfIbA}\vec{\texttt{c}}\texttt{AbIfVfJfVfIbA}$
(length is 14, so exceeds max length of 6)

$\texttt{Conference}\xleftarrow{\texttt{of}}\texttt{Proceeding}\xrightarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xrightarrow{\texttt{of}}\texttt{Conference}$ $\xleftarrow{\texttt{of}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xrightarrow{\texttt{of}}\texttt{Conference}$ ($\texttt{CfPbA}\vec{\texttt{c}}\texttt{AbPfCfPbA}\vec{\texttt{c}}\texttt{AbPfC}$)
(length is 10, so exceeds max length of 6)

Or sub cycles like:

$\texttt{Journal}\xleftarrow{\texttt{of}}\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xrightarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}_x\xleftarrow{\texttt{of}}\texttt{Issue}$ $\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{JfVfIbA}\vec{\texttt{c}}\texttt{AbIfV}_x\texttt{fIbA}\vec{\texttt{c}}\texttt{AbIfVfJ}$)

where not $\texttt{Volume}_x\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{V}_x\texttt{fJ}$) (length is 12, so exceeds max length of 6)

$\texttt{Journal}\xleftarrow{\texttt{of}}\texttt{Volume}\xleftarrow{\texttt{of}}\texttt{Issue}\xrightarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}_x\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}$ $\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Issue}\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{JfVfIbA}\vec{\texttt{c}}\texttt{AbIbA}\vec{\texttt{c}}\texttt{AbIfVfJ}$)

where not $\texttt{Issue}_x\xrightarrow{\texttt{of}}\texttt{Volume}\xrightarrow{\texttt{of}}\texttt{Journal}$ ($\texttt{I}_x\texttt{fVfJ}$) (length is 10, so exceeds max length of 6)

$\texttt{Conference}\xleftarrow{\texttt{of}}\texttt{Proceeding}\xrightarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}$ $\xrightarrow{\texttt{cites}}\texttt{Article}\xrightarrow{\texttt{belongs\_to}}\texttt{Proceeding}\xrightarrow{\texttt{of}}\texttt{Conference}$ ($\texttt{CfPbA}\vec{\texttt{c}}\texttt{AbPfCfPbA}\vec{\texttt{c}}\texttt{AbPfC}$)

where not $\texttt{Proceeding}_x\xrightarrow{\texttt{of}}\texttt{Conference}$ ($\texttt{P}_x\texttt{fC}$) (length is 8, so exceeds max length of 6)

8.3 Gray-zone cycle types

A cycle type describes a behavior of a person, which, depending on the context and on the occurrence amount within the context, can be indicators for scientific fraud. This requires to determine the context and to define normal behavior for each cycle type differently, which we will demonstrate in Chapter 9.

But, some cycles are more likely an indicator of scientific fraud than others, regardless of the number of occurrences. Take, for example, $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}\xrightarrow{\texttt{pc\_member\_of}}$ $\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PaAaPpRbAaP}$), the person published an article in a proceeding where a coauthor is a PC member. This cycle type may imply conflicts of interest. But, maybe this conference has very good rules about such cases, and the PC member is excluded from the publication decision of this article. Also, this cycle type is not very strange, since the PC member and the person will probably work in the same field, and since they were coauthors, the conference will probably cover this field.

Other cycles, like the coauthor cycle, ($\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$, `PaAaPaAaP`) are not likely indicators of scientific fraud. Although, very frequently writing with the same coauthor may be suspicious. Also, the coauthors-are-also-coauthors cycle, ($\texttt{Person}\rightarrow\texttt{Article}\leftarrow\texttt{Person}\rightarrow\texttt{Article}\leftarrow\texttt{Person}\rightarrow\texttt{Article}\leftarrow\texttt{Person}$, `PaAaPaAaPaAaP`) is even lesser an indicator of scientific fraud, only in very extreme cases it may be.

Therefore, reasoning about the cycle types requires domain expertise and may involve ethical discussion of what is normal and abnormal. Domain expertise is also required to translate the cycles back to possible methods of scientific fraud. This should be done with caution, since legal and fraud actions may result in the same cycle type.

These examples also show that flagging behavior as suspicious depends on the context, which is one of the foundations of Westerbaan's thesis [Wes22]. This has to be considered before deducting a fraud method from a cycle type.

Chapter 9

Outlier identification

In the last chapter (Chapter 8) we discussed the cycle types we detected. In this chapter, we take two cycle types for the detection of outliers. We start with the PC member requires citations in Section 9.1, to show we can replicate the findings of Westerbaan. After that, we also search for outliers in the self-citations cycle type in Section 9.2 with the same method. We improved the outlier identification of Westerbaan by choosing for a statistical approach, while still showing similar results. This statistical approach removes the limitations of determining thresholds per peer group or context.

Before investigating a fraud method or cycle type, two question needs to be answered: "Where to can an actor be compared, or who are similar actors?" (context), and "When is someone an outlier in its context?" (outlier). We first discuss both questions in general.

Context
Tielenburg [Tie17] tried to first identify outliers and then compare them with peer groups, he did not succeed in the latter, Westerbaan [Wes22] turned this order around and succeeded. Therefore, before detecting outliers, we must first select actors that are similar to the investigated actor. An easy solution are so-called peer-groups: groups of actors that can be considered similar. It is also possible to select peers dynamically (like persons that to some degrees write the same amount of articles and publish in the same journals) or with algorithms like K-nearest neighbor. This will to some extent determine the "normal" in the outlier detection, so we must be careful to not compare fraudsters with other fraudsters and thereby deciding that they are not, although their behavior is probably similar.

Choosing a good peer group also improves post investigation by fraud investigators, since if the peer group is correctly chosen, it may also represent a culture group in the real world. For example: "In this journal, we do it always in this way, so this behavior is not strange". This is especially the case for niche area's in science. We know for example about the e-voting. In this niche, about four professors and some of their co-authors are publishing frequently, and sometimes outsiders also write several articles. It may be possible that citation stacking between these four professors is relatively high, but once it is clear that the peer group is considered: e-voting a fraud investigator may quickly decide that the report is a false positive. It will be harder to decide when these peer groups are dynamic or determined by algorithms.

Outlier
The definition of a (statistical) outlier is according to NIST: "An outlier is an observation that lies an abnormal distance from other values in a random sample from a population." [NIS] So to

find outliers we need methods that measure distance from normal observation/behavior, when this distance becomes abnormal, one can be considered an outlier. Therefore, to detect outliers in a dataset, we have to answer two questions:

1. What is normal?
2. What is an abnormal distance from the normal?

Some outlier methods may classify normal incorrectly when used in a wrong dataset. For example: a very naive way is to just take the 0.01% values that distance the most from the median and classify those as an outlier. In that case, we consider the median as normal, but if all values just perfectly fit a parabolic curve, should not the curve itself be the normal? And secondly, are these values, when the dataset just contains three numbers repeated many times, at an abnormal distance? We do not want a method that assumes that there must be outliers, because is sometimes there are none.

From this example, note that outlier classification depends on the data distribution classification, which may differ per cycle type. When investigating outliers in cycles, we have to manually select an outlier classification method per cycle type.

9.1 PC member demanding citations

In this section, we investigate the PC member requires citation cycle type. This fraud method was introduced in Section 2.3.1. The cycle type:
$\texttt{Person}\xrightarrow{\texttt{pc\_member\_of}}\texttt{Proceeding}\xleftarrow{\texttt{belongs\_to}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Person}$ ($\texttt{PpRbA}\vec{\texttt{c}}\texttt{AaP}$)

The goal of this investigation is to replicate the finding of Westerbaan [Wes22]. He found an interesting outlier in the Spire conference. Therefore, we follow his approach.

We did not use the results of our cycle detection, but the Cypher query listed in Listing 9.1. Because, we had to exclude article citations that were made by the PC members themselves. Of course, a PC member will not require himself to include reference to his own work, this is called self-citation and not included in this investigation. We also included the conference of the proceeding for the context.

```
MATCH (c:Conference)<-[:of]-(r:Proceeding)<-[:pc_member_of]-(p:Person)
OPTIONAL MATCH P=(p)-[:author_of]->()<-[:cites]-(a)-[:belongs_to]->(r)
WHERE NOT exists((p)-[:author_of]->(a))
RETURN c.Code, r.dblpKey, p.dblpName, count(P)
```

Listing 9.1: Cypher query for PC member requires citation

Context Westerbaan compared PC-members within conferences. Table 9.1 shows that comparing cycles across conferences is hard, since each conference has another PC member citation distribution. For example, Crypto differs much from Aweb, in the maximum number of citations and the number of 0 citations. Asia crypt also differs from Crypto, but there are more like each other, suggesting that in the same area of expertise conferences could be more comparable to each other. Spire is also shown to enable comparison with Westerbaan's study.

Normal Because of the inability to compare cycles across conferences, we have to find a way to spot outliers for each conference. Westerbaan proposed to ask each conference what a good threshold value should be [Wes22]. But since we have 228 conferences with PC members in our dataset, it would be a pain to ask each of them for a threshold value, so we propose a more statistical approach for outlier detection. Which will be explained below.

Citations	**Crypto**	**Asiacrypt**	**Apweb**	**Spire**
0	39	20	458	110
1	22	21	64	23
2	13	8	26	15
3	12	10	12	10
4	15	9	7	7
5	7	11	5	7
6-9	37	30	6	16
10-19	41	34	0	24
20-34	33	17	1	3
35-49	9	6		4
50-74	9	6		2
74-124	6			0
124-199	5			0
$\geq$200				1
Total PC Members	249	204	581	223
Total Years	16	18	14	16

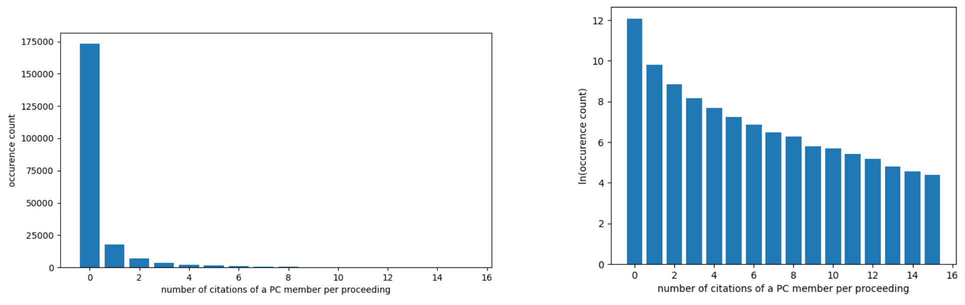
Table 9.1: Number of PC member with citations per range of citations in four conferences

Selection of a statistical outlier classification method
First, we tried to use Tukey's fences (also known as outliers in box plots) to find outliers. This failed since the data is not a normal distributed, but exponential, see Figure 9.1. Because most PC members do not have any citations, very often the 75 percentile was 0. So when using Tukey's fences, all non-zero citations were classified as outliers. But a few PC member citations do not have to be strange, since they are hopefully experts in the field of the conference. And as can be seen in Table 9.1, this is especially common in the Crypto conference.

So strait forward outlier detection with box plots did not work. But we did not find a suitable outlier detection method for this dataset type in the literature. The reasons for this are:

- Our dataset is grouped at integers therefore we have many 0's and some 1's, instead of exponential decrease of data points between 0 and 1. Mathematically speaking, we need a method for $\mathbb{N}$ not $\mathbb{R}$.
- Our dataset is 1D, not 2D, so the data itself is not exponential (samples), only the occurrence or distribution is exponential. In Figure 9.1 the occurrence is the vertical-axis, but in fact it describes the amount of points on the horizontal-axis. Mathematically speaking, we need a method for exponential distribution, not exponential samples. The key difference between these two methods, is that we need the amount of points to be included, since they give weight to the method. Not each point in Figure 9.1 has the same weight, the amount of occurrence is its weight. This weight has to be included in the calculation to prevent a bias to high values, which are actually the outliers we want to detect.
- We want to exclude naive approaches like the 0.01% most citations are outliers, as explained in the beginning of this chapter.

One method that comes close is that of Raybon [Ray19], when applying his method we calculate the probability of having 0, 1, 2, ... citations. However, we still have to decide from where on we respect the probability as too low, and consider it an outlier. He defines the threshold on 1%. So, this method did not fulfill requirement 3.



(a) Normal (b) Ln

Figure 9.1: Citations of PC members are distributed exponentially (from 0 up to 15)

Our proposal: weighted exponential regression
But Raybon's approach inspired us to propose the use of weighted exponential regression to draw a trend line. As seen from Figure 9.1 the citation data is an exponential distribution, therefore we are able to apply exponential regression to find the trend in decline. When a PC member exceeds the expected trend, these PC members get an outlier flag. We can sum up the outlier flags, and the PC members with the most flags should be high on the list for fraud investigation. There is still one problem: normal exponential regression treats all points as equal in searching a fitting formula: leading to a bias to the less occurred citation numbers in our data, we correct this by providing a weight: the number of occurrences, since the 0 citations occurs far more often than the 12 citations and should therefore not be treated equally, but in proportion to its number occurrences. For a mathematical and technological implementation of weighted exponential regression, see Section 2.4.

Outlier in conference
The flagging of an outlier is based on the difference of the outlier with the trend. This difference (d) can be measured by the distance between the bar of a PC member and the trend line. We have to come up with a threshold value, after what distance one becomes an outlier. Note that while this still requires a threshold value, it is now a threshold value for all conferences. And by using the trend line, conferences that perfectly fit the trend line will not have outliers, regardless of the threshold value. The threshold value should be within 1 (above 1 only marks outliers that by accident had more times the same amount of citations, which is too easy to escapes as fraudster) and 0 (when the trend line is below 1, all is considered an outlier). In this study we use 0.5 but the value can be tweaked to get more or lesser outliers. This tweaking includes or excludes an amount of small outliers, big outliers (with $d \geq 0.8$) will only begin to be excluded by high threshold values.

Usually, the value of d will be between 0 and, 1. But, d could become bigger than 1 if a PC member has for multiple proceedings the same number of received citations. For example, a PC member could have been cited 33 times in the year 2001 and 33 times in the year 2020. If the trend line is 0.2 on 33 then the d in this case is 1.8.

9.1.1 Spire conference to compare with Westerbaan

To illustrate the method we give an example with the Spire conference, the same as Westerbaan used for his example to validate our method. In Figure 9.2 the blue bars are the number of citations, and the orange line is the trend line from the weighted exponential regression. As seen in Figure 9.2 the trend line fits very well, but we can see that after 11 citations of a PC member some citations numbers are not in the trend. Table 9.2 contains the outliers found in Spire using this method, the names of the PC members are anonymized. Person F is an outlier of all the other outliers in this conference, this Person F is the same person that Westerbaan considered an outlier in his study.

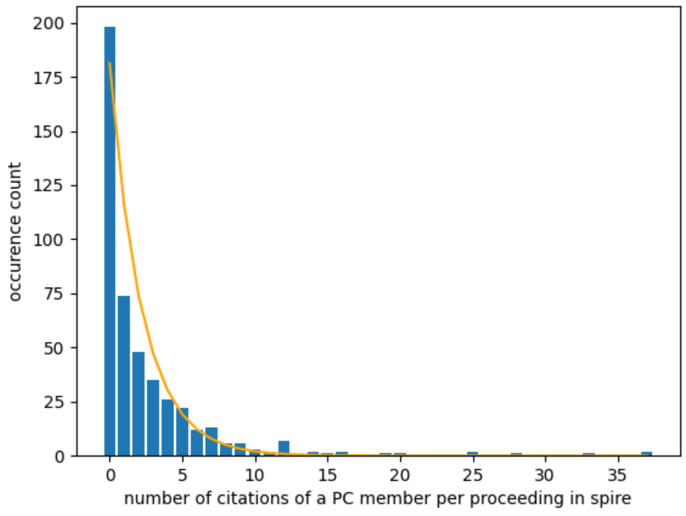


Figure 9.2: Citations of PC members in Spire

9.1.2 Across all conferences

We applied our method of detecting outliers on all conferences. We then summed the amount of outlying citations per PC member across all conferences. The top 10 of PC members with the highest sum of outlying citations is listed in Table 9.3. Person F from Spire holds position 6 in this list. The additional outlying citations this person received are coming from the Cpm conference.

9.2 Self-citations

In this section, we detect outliers in the self-citation cycle type. This fraud method was introduced in Section 2.3.2. We used the results of the cycle detection to investigate outliers in this cycle type. The cycle type:
$\texttt{PaA}\vec{\texttt{c}}\texttt{AaP}$: $\texttt{Person}\xrightarrow{\texttt{author\_of}}\texttt{Article}\xrightarrow{\texttt{cites}}\texttt{Article}\xleftarrow{\texttt{author\_of}}\texttt{Article}$.

Context For the context, we chose to use the amount of self-citation cycles as a percentage of the total number of citations. To acquire the total number of citations per person, we used the Cypher query listed in Listing 9.2. To eliminate persons that did only write a few papers, and are therefore less interesting for investigating large scale scientific fraud, we chose to only

PC Member	**Citations**	**Distance d**
A	14	0.66301
B	15	0.78493
C	16	0.86274
D	16	0.86274
E	28	0.99937
F	14	0.66301
F	19	0.96432
F	20	0.97722
F	25	1.99758
F	33	0.99993
F	37	1.99999
Total F	148	

Table 9.2: Outliers in Spire

Top	**Sum of outlying citations**
1	423
2	307
3	293
4	276
5	246
6/F	225
7	224
8	210
9	202
10	201

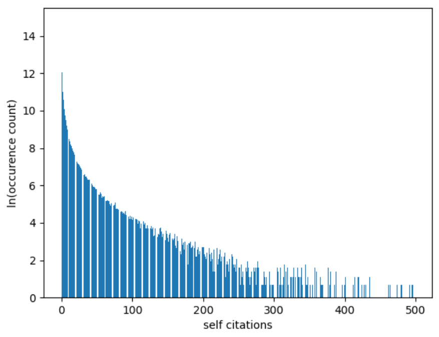
Table 9.3: Top 10 of PC members with the highest sum of outlying citations across all conferences

include persons that received 50 or more citations. When self-citations are used to improve a person's h-index, we start detecting around h-index 7 (7 articles with each at least 7 citations results in at least 49 citations). There are 156,552 persons of a total of 3,019,026 that received 50 or more citations. The distribution of the self-citation occurrences before this limitation is shown in Figure 9.3a.

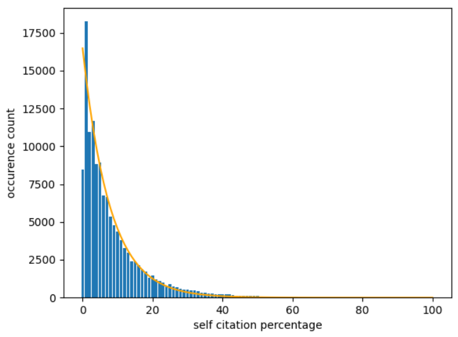
```
MATCH P=(p:Person)-[:author_of]->(:Article)<-[:cites]-(:Article)
RETURN p.dblpName, count(P) AS citations
```

Listing 9.2: Cypher query for acquiring total number of citations per person

Normal A scientist with more than 50 citations should not have a high percentage of self-citations. But, what is high? The self-citations percentages (p) are skewed distributed, see Figure 9.3b. But this skewed distribution is very close to an exponential distribution, so we can also apply weighted exponential regression here. After performing weighted exponential regression, we noticed that $d \geq 0.5$ for $p \geq 81\%$, so self-citation percentages equal or above 81% are classified as outliers by our method.



(a) Log (ln) of the occurrence count of self-citation amounts from 0 to 500



(b) Occurrences of self-citations percentages for scientists with more than 50 citations

Figure 9.3: Self-citation distributions

Results There are 64 scientists that have a self-citation percentage above 81%. The top 10 of scientists with the highest self-citation percentages is displayed in Table 9.4. We manually validated the top 10. Some of the scientists are coauthors from each other, these are the groups in the table.

Top	Coauthor group	Self-citations	Total citations	Percentage
1	A	70	70	100%
2	A	70	71	99%
3		66	67	99%
4	B	58	59	98%
5	B	55	56	98%
6		148	152	97%
7	C	148	154	97%
8	C	148	152	97%
9	C	148	155	95%
10		188	197	95%

Table 9.4: Top 10 of scientists with the highest self-citation percentages

Validation For each scientist or coauthor group, we investigated the number of coauthors for the self-cited articles, the number of self-cited articles, and the publication year range of these articles. We also validated for a random sample of 2 articles whether the same citations are also displayed by the publisher to ensure our data is valid. And we tried to find Google Scholar or other scientific profiles to check if the number of citations we found match with the total number of citations in these other online profiles.

- Group A has 15 self-citing articles, and worked also with 8 more coauthors who only contributed once or a few times with them. The articles are written in across multiple years from 2013 to 2018 and published in multiple conferences and journals. Their citations seem not to come from invalid data, since most references are also shown on the cite of

the publisher. On their online profiles they much more publications outside the area of computing science, although the also work togheter in these areas.
- Person 3 has 18 self-citing articles with 5 coauthors working on multiple of these. The article data from the years 1984 to 1989. The citations across them are few, some citation also loop back. Regarding the publications years, this could be caused by invalid data. The online profile of this person shows simular data.
- Group B are the only coauthors of 13 articles, published in the last 4 years, all citing past articles. The citions match with the cite of the publisher. According to their Aminer profiles, they have written more articles and received more citations outside the scope of our graph.
- Person 6 has 30 articles with 34 coauthors that contributed only once or twice. Most articles cite others in this collection of articles. They are all published in the last 4 years. According to the website of the publisher, the citation seem not come from invalid data. On the online profile of this person, much more articles and citations are listed, this person also publishes outside computing science.
- Group C are coauthors from each other and with another holding place 32. All their self-citations flow from 17 articles that cite each other. All these artciles are published in 2008 in the same issue. The citations seem not to come from invalid data, since most references are also shown on the cite of the publisher. Their profile pages seem to match our data, except person 9. Some of them also wrote a few more articles together in 2010 and 2013. Person 9 has written more articles, very old ones in 1985 until 2013.
- Person 10 has 51 articles with self-citations, mostly written alone, but several with 19 coauthors each contributed only once, only two coauthors contributed to three articles. The article's dates are in long range: from 1986 until 2018. According to the website of the publisher, the citation seem not come from invalid data. According to this person online profile the person published much more articles and even collected many more citations, because this person is publishes outside computing science.

Conclusion Person 3 could be caused by invalid data (Q2). We missed data from outside computing science to classify group A, and B; and person 6, and 10 (Q1). Group C should be further investigated by a fraud investigator.

Chapter 10

Conclusions

In this chapter we list our discussion (Section 10.1), future work (Section 10.2), and conclusions (Section 10.3). In the conclusions, we will answer the research questions mentioned in Chapter 1.

10.1 Discussion

In this thesis, we described a method to search some forms of scientific fraud. Our method does not cover all forms of scientific fraud, for example it does not find plagiarism or image manipulation, other tools are a better fit for these forms. Scientific fraud is like an arms race, people will find new ways to commit fraud, and the fraud investigators will have to find new ways to detect these. Our method is just a proof of concept ready to expanded and improved, but has already some interesting results. Our method is limited by methods of fraud that can be encoded as cycle types.

There are some cycle types and form of scientific fraud, that can be described as cycles in a graph and can be detected in that way, but representing them as cycles is not the best way. For example, the co-author cycle: $\texttt{Person}_1 \rightarrow \texttt{Article} \leftarrow \texttt{Person}_2 \rightarrow \texttt{Article} \leftarrow \texttt{Person}_1$, for every person that wrote at least two different articles with the same person. In analyzing these cycles types, it is better to analyze it as a path: $\texttt{Person}_1 \rightarrow \texttt{Article} \leftarrow \texttt{Person}_2$. This eliminates the exponential amount of cycles (c) with the amount (a) of articles written with the coauthor: $c = 1+2+3+...+a-1$. To avoid this unnecessary amount of cycles, we have introduced recursive paths and cycles (Section 7.3.3-7.3.4). Firstly, it reduced the amount of paths to the amount of articles, so the exponential factor is lost, and it allows more easily to analyze its context which is key for outlier detection Chapter 9. Note that when viewing the normal amount of cycles, this exponential factor is present and results in a biased amount of cycles. So for analyzing these types of cycles, transforming back to paths or using the recursive cycles is highly recommended.

Ethics

In our study, we place high importance on the work of the human fraud investigator. The methods proposed in our study are not fit to flag behavior as fraud, only to highlight some suspicious behavior for further investigation by human fraud investigators. Reasons like false data or context based explanations of the anomaly should be regarded, which cannot be included in full proportion into our method. The Committee on Publication Ethics (COPE) publishes guides on what to do when scientific fraud is suspected. We highly recommend being cautious when accusing persons or venues of scientific fraud.

10.2 Future work

The most future work we see is in the area of data sciences and machine learning, we therefore recommend that persons performing future work are capable in at least one of these areas.

10.2.1 Improving data acquisition and refinement

In this study, a beginning was made with constructing a scientific biographic information graph. This graph is still far from complete and contains invalid information. Efforts to enlarge the graph and improve the quality of it, will enable the development of a more wholistic fraud investigation system, that can hint fraud investigators where to look first.

Person and name matching One of the biggest data integration issue is that of matching persons based on name. This issue is discussed in more detail in Section 4.1.2, and by Westerbaan; Ley; Tang et al. [Wes22; Ley09; Tan+08].

POC front matter parser In Section 5.5 we explained our struggle with parsing ACM issue front matters, this failed because the original front matter parser did not work well with font sizes and multiple page columns. We suggest building a new POC front matter parser, not based on a text based grid, but on positions of text boxes. But there could be also other solutions, for example `https://spacy.io` looks also useful. This will enable the inclusion of ACM editors and reviewers from the front matters in the data model.

Fresh data Besides these improvements, it is also possible to host the data set or Neo4j server publicly. This allows to frequently re-scraping and updating the graph data sources, so it always contains the most recent data, this also limits the time part of the data completeness issue (Q1.1 in Section 5.1).

Altmetrics It is also possible to integrate data from outside the academic world to improve outlier detection. This includes so called “altmetrics”, for example, comments on a website or social media about publications or authors. Lin describes the problems and solutions this addition offers [Lin20].

10.2.2 Advanced methodology

Besides future work on data integration and refinement, we also have some suggestions for the methodology.

Anomaly detection - machine learning The data has been integrated and parsed in a graph. This method can still be improved as noted above. But this new way of data representation opens the possibility for anomaly detection in combination with machine learning. This also requires a statistical method of outlier detection, which we suggested in Chapter 9. The study of Pourhabibi et al. list’s many methods for graph based anomaly detection, some of these can be used as inspiration for a future study [Pou+20].

For example, in the study of Carvalho et al. [Car+17], they transmuted the anomaly scores from an easy identifiable group to a difficult one. We can do something similar by transmuting the anomaly scores of PC members in Section 9.1 to their conferences.

Ideas from related work Also some ideas of related work could be integrated in this study. One is the detection of black holes in a graph [LXL12], this could detect certain communities, of persons, articles or venues that for example receive many citations, without citing very much outside their community. We do not know if this could lead to interesting cases but probably highly cited scientist should be found with this method, and maybe this will give new insights for finding outliers.

Differences in time Yet another possible method is to bring the time dimension to the table. Sometimes, a person may start to commit fraud at a certain point in time, which can be seen by a sudden increase for example in publications or citations, when such events occur it is useful to inform a fraud investigator about this, so he/she can monitor the person and investigate the cause of the increase.

Calculating the influence on metrics A additional step could be to calculate the influence that outliers have on their own metrics. Since we have the data available to calculate scientist's h-index, we could also calculate the influence on the h-index when removing the anomaly citations of PC members. This is especially interesting when scientist commit targeted fraud to improve their metrics.

10.3 Conclusion

Our main research question is **When is the number of cycles of an actor abnormal within an academic publication graph?** To answer this question, we first answer our sub research questions.

RQ1 How to integrate publicly available data into an academic publication graph?
As also noted by Westerbaan [Wes22] data availability, quality, and integration is a problem. Therefore, we underline the recommendations of Westerbaan for better data quality provided by publishers. But in this thesis, we also improved the integration and quality at certain points. For example, by acquiring additional data with higher quality by scraping and parsing data from the ACM website. We now have a full list of all ACM journals, and the scraped issue pages allowed us to more link the editorial boards to specific issues instead of the journal in general. We also improved Westerbaan's proof-of-concept LNCS front matter parser, leading to more and better results. We improved the extraction of article DOIs in DBLP to improve the linking with OpenCitations. We also improved the name detection in the POC front matter parser to increase the likeliness of finding persons in DBLP. We succeeded in combining the data of DBLP, OpenCitations, ACM website and Springer LNCS front matters in one academic publication graph, with room for expansion.

RQ2 How to efficiently detect undirected cycles in a directed academic publication graph?
We tackled the challenge of detecting undirected cycles in a directed graph to chose for an undirected cycle detection algorithm that nevertheless still report the directions of the edges in the cycles. We did this by proposing our own cycle detection method, based on BFS (Breadth first search) algorithm, that has better performance in comparison to straightforward solutions of Neo4j. This performance increase was needed to allow cycle detection of cycles not larger than 6 edges in a graph containing out of 15 million vertices, and 44 million edges. Still, detection of all cycles for all 3 million persons ran for 4 days.

RQ3 How to identify outliers given cycles in an academic publication graph?
The results showed that we also detected cycle types that are related to known methods of fraud. Also, some unknown methods of fraud can be detected when cycle types can be decoded with domain knowledge and in the right context. But more rewarding is it to investigate outliers within cycles types.

With our method, we can find outlying actors by encoding methods of scientific fraud as cycle types and detecting outlying numbers of cycles. We showed we can reproduce Westerbaan's evaluation of the SPIRE conference.

Furthermore, we improved detection of outliers within attacks by providing a statistical method to determine thresholds. Thus, the threshold no longer has to be based on domain knowledge, thereby eliminating the risk of basing thresholds on suggestions of fraudsters.

We reapplied the method to detect persons with high self-citation numbers in comparison to their overall citations. With this, we demonstrated that we can easily apply our method to other cycle types that indicate scientific fraud.

The main research question
To answer the main research question: it depends on the context and peers of an outlying person. With our statistical method proposed in Chapter 9, we can calculate when the number of cycles of an actor becomes outlying. We did these on cycle types that are indicative for scientific fraud. With our method, we can also investigate other methods of scientific fraud if they can be encoded as cycle types in our academic publication graph.

Bibliography

[12] *San Francisco Declaration on Research Assessment.* 2012. URL: https://sfdora.org/read/ (visited on 08/29/2022).

[Ant20] Ike Antkare. “Ike Antkare, His Publications, and Those of His Disciples”. In: *Gaming the Metrics : Misconduct and Manipulation in Academic Research.* Ed. by Mario Biagioli and Alexandra Lippman. Infrastructures. Cambridge, Massachusetts: The MIT Press, 2020. Chap. 14, pp. 177–200. ISBN: 978-0-262-53793-3. (Visited on 03/03/2022).

[BH04] Bo-Christer Björk and Turid Hedlund. “A Formalised Model of the Scientific Publication Process”. In: *Online Information Review* 28.1 (Jan. 1, 2004), pp. 8–21. ISSN: 1468-4527. DOI: 10.1108/14684520410522411.

[BL20] Mario Biagioli and Alexandra Lippman. *Gaming the Metrics : Misconduct and Manipulation in Academic Research.* Infrastructures. Cambridge, Massachusetts: The MIT Press, 2020. ISBN: 978-0-262-53793-3.

[BM15] Lutz Bornmann and Rüdiger Mutz. “Growth Rates of Modern Science: A Bibliometric Analysis Based on the Number of Publications and Cited References”. In: *Journal of the Association for Information Science and Technology* 66.11 (2015), pp. 2215–2222. ISSN: 2330-1643. DOI: 10.1002/asi.23329.

[Car+17] Luiz F. M. Carvalho, Carlos H. C. Teixeira, Wagner Meira, Martin Ester, Osvaldo Carvalho, and Maria Helena Brandao. “Provider-Consumer Anomaly Detection for Healthcare Systems”. In: *2017 IEEE International Conference on Healthcare Informatics (ICHI).* 2017 IEEE International Conference on Healthcare Informatics (ICHI). Aug. 2017, pp. 229–238. DOI: 10.1109/ICHI.2017.75.

[Cha22] Dalmeet Singh Chawla. *How a Site Peddles Author Slots in Reputable Publishers’ Journals.* 6590. Apr. 6, 2022. DOI: 10.1126/science.abq4276.

[EV21] Holly Else and Richard Van Noorden. “The Fight against Fake-Paper Factories That Churn out Sham Science”. In: *Nature* 591.7851 (7851 Mar. 23, 2021), pp. 516–519. DOI: 10.1038/d41586-021-00733-5.

[Fan+22] Daniele Fanelli, Matteo Schleicher, Ferric C. Fang, Arturo Casadevall, and Elisabeth M. Bik. “Do Individual and Institutional Predictors of Misconduct Vary by Country? Results of a Matched-Control Analysis of Problematic Image Duplications”. In: *PLOS ONE* 17.3 (Mar. 2, 2022), e0255334. ISSN: 1932-6203. DOI: 10.1371/journal.pone.0255334.

[Fan20] Daniele Fanelli. “Pressures to Publish: What Effects Do We See?” In: *Gaming the Metrics.* Ed. by Mario Biagioli and Alexandra Lippman. Cambridge, Massachusetts: The MIT Press, 2020. Chap. 8, pp. 111–122. ISBN: 978-0-262-53793-3.

[FB18] Diogo Fernandes and Jorge Bernardino. “Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB:” in: *Proceedings of the 7th International Conference on Data Science, Technology and Applications.* 7th International Conference on Data Science, Technology and Applications. Porto, Portugal: SCITEPRESS - Science and Technology Publications, 2018, pp. 373–380. ISBN: 978-989-758-318-6. DOI: `10.5220/0006910203730380`.

[FH21] Robert M. Frederickson and Roland W. Herzog. “Keeping Them Honest: Fighting Fraud in Academic Publishing”. In: *Molecular Therapy* 29.3 (Mar. 3, 2021), pp. 889–890. ISSN: 1525-0016. DOI: `10.1016/j.ymthe.2021.02.011`.

[FH22] Robert M. Frederickson and Roland W. Herzog. “Addressing the Big Business of Fake Science”. In: *Molecular Therapy* 30.7 (July 6, 2022), p. 2390. ISSN: 1525-0016. DOI: `10.1016/j.ymthe.2022.06.001`.

[Fra22] Tove Faber Frandsen. “Authors Publishing Repeatedly in Predatory Journals: An Analysis of Scopus Articles”. In: *Learned Publishing* (Aug. 4, 2022). ISSN: 0953-1513, 1741-4857. DOI: `10.1002/leap.1489`.

[Gil15] Adrew Gilmartin. *DOIs and Matching Regular Expressions.* Crossref. Aug. 11, 2015. URL: `https://www.crossref.org/blog/dois-and-matching-regular-expressions/` (visited on 04/06/2022).

[Gri20] James Griesemer. “Taking Goodhart’s Law Meta”. In: *Gaming the Metrics : Misconduct and Manipulation in Academic Research.* Ed. by Mario Biagioli and Alexandra Lippman. Cambridge, Massachusetts: The MIT Press, 2020. Chap. 5, pp. 77–87. ISBN: 978-0-262-53793-3.

[Hog+21] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia d’Amato, Gerard de Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, Axel-Cyrille Ngonga Ngomo, Axel Polleres, Sabbir M. Rashid, Anisa Rula, Lukas Schmelzeisen, Juan Sequeda, Steffen Staab, and Antoine Zimmermann. “Knowledge Graphs”. In: *Synthesis Lectures on Data, Semantics, and Knowledge* 12.2 (Nov. 8, 2021), pp. 1–257. ISSN: 2691-2023. DOI: `10.2200/S01125ED1V01Y202109DSK022`.

[HPS19] Ivan Heibi, Silvio Peroni, and David Shotton. “Software Review: COCI, the OpenCitations Index of Crossref Open DOI-to-DOI Citations”. In: *Scientometrics* 121.2 (Nov. 2019), pp. 1213–1228. ISSN: 0138-9130, 1588-2861. DOI: `10.1007/s11192-019-03217-6`.

[JCG22] Tiago S. Jesus, Greta Castellini, and Silvia Gianola. “Global Health Workforce Research: Comparative Analyses of the Scientific Publication Trends in PubMed”. In: *The International Journal of Health Planning and Management* 37.3 (2022), pp. 1351–1365. ISSN: 1099-1751. DOI: `10.1002/hpm.3401`.

[JM17] Hugo Jonker and Sjouke Mauw. “A Security Perspective on Publication Metrics”. In: *Security Protocols XXV.* Ed. by Frank Stajano, Jonathan Anderson, Bruce Christianson, and Vashek Matyáš. Vol. 10476. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2017, pp. 186–200. ISBN: 978-3-319-71075-4. DOI: `10.1007/978-3-319-71075-4_21`.

[KLM21] Sadamori Kojaku, Giacomo Livan, and Naoki Masuda. “Detecting Anomalous Citation Groups in Journal Networks”. In: *Scientific Reports* 11.1 (1 July 15, 2021), pp. 1–11. ISSN: 2045-2322. DOI: `10.1038/s41598-021-93572-3`.

[Ley09] Michael Ley. “DBLP: Some Lessons Learned”. In: *Proceedings of the VLDB Endowment* 2.2 (Aug. 1, 2009), pp. 1493–1500. ISSN: 2150-8097. DOI: 10.14778/1687553.1687577.

[Lin20] Jennifer Lin. “Altmetrics Gaming: Beast Within or Without?” In: *Gaming the Metrics : Misconduct and Manipulation in Academic Research*. Ed. by Mario Biagioli and Alexandra Lippman. Cambridge, Massachusetts: The MIT Press, 2020. Chap. 16, pp. 213–227. ISBN: 978-0-262-53793-3.

[LL13] Cyril Labbé and Dominique Labbé. “Duplicate and Fake Publications in the Scientific Literature: How Many SCIgen Papers in Computer Science?” In: *Scientometrics* 94.1 (Jan. 2013), pp. 379–396. ISSN: 0138-9130, 1588-2861. DOI: 10.1007/s11192-012-0781-y.

[LXL12] Zhongmou Li, Hui Xiong, and Yanchi Liu. “Mining Blackhole and Volcano Patterns in Directed Graphs: A General Approach”. In: *Data Mining and Knowledge Discovery* 25.3 (Nov. 1, 2012), pp. 577–602. ISSN: 1573-756X. DOI: 10.1007/s10618-012-0255-0.

[Neo15] Jenny Neophytou. *Anomalous Citation Patterns in the World of Citation Metrics*. Willy. Sept. 9, 2015. URL: https://www.wiley.com/network/archive/anomalous-citation-patterns-in-the-world-of-citation-metrics (visited on 07/11/2022).

[NIS] NIST/SEMATECH. “7.1.6. What Are Outliers in the Data?” In: *E-Handbook of Statistical Methods*. URL: https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm (visited on 06/07/2022).

[Ora12] Ivan Oransky. *South Korean Plant Compound Researcher Faked Email Addresses so He Could Review His Own Studies*. Retraction Watch. Aug. 24, 2012. URL: https://retractionwatch.com/2012/08/24/korean-plant-compound-researcher-faked-email-addresses-so-he-could-review-his-own-studies/ (visited on 07/19/2022).

[Pil22] Charles Piller. “Blots on a Field?” In: *Science* 377.6604 (July 22, 2022), pp. 358–363. ISSN: 0036-8075. DOI: 10.1126/science.add9993.

[Pou+20] Tahereh Pourhabibi, Kok-Leong Ong, Booi H. Kam, and Yee Ling Boo. “Fraud Detection: A Systematic Literature Review of Graph-Based Anomaly Detection Approaches”. In: *Decision Support Systems* 133 (June 1, 2020), p. 113303. ISSN: 0167-9236. DOI: 10.1016/j.dss.2020.113303.

[Ray19] Steven Raybon. *Real-Time Anomaly Detection with Exponentially-Distrubted Data*. Towards Data Science. July 23, 2019. URL: https://towardsdatascience.com/real-time-anomaly-detection-with-exponentially-distrubted-data-205e0df32096 (visited on 06/08/2022).

[Sch+18] Christian Schulz, Brian Uzzi, Dirk Helbing, and Olivia Woolley-Meza. “A Network-Based Citation Indicator of Scientific Performance”. July 12, 2018. arXiv: 1807.04712 [physics]. URL: http://arxiv.org/abs/1807.04712 (visited on 03/15/2022).

[See+19] Marco Seeber, Mattia Cattaneo, Michele Meoli, and Paolo Malighetti. “Self-Citations as Strategic Response to the Use of Metrics for Career Decisions”. In: *Research Policy*. Academic Misconduct, Misrepresentation, and Gaming 48.2 (Mar. 1, 2019), pp. 478–491. ISSN: 0048-7333. DOI: 10.1016/j.respol.2017.12.004.

[SO18] Bodo M. Stern and Erin K. O’Shea. *Scientific Publishing in the Digital Age*. ASAPbio. Jan. 26, 2018. URL: https://asapbio.org/digital-age (visited on 07/15/2022).

[Str97] Marilyn Strathern. "'Improving Ratings': Audit in the British University System". In: *European Review* 5.3 (July 1997), pp. 305–321. ISSN: 1474-0575, 1062-7987. DOI: 10.1002/(SICI)1234-981X(199707)5:3<305::AID-EURO184>3.0.CO;2-4.

[Tan+08] Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. "ArnetMiner: Extraction and Mining of Academic Social Networks". In: *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.* KDD '08. New York, NY, USA: Association for Computing Machinery, Aug. 24, 2008, pp. 990–998. ISBN: 978-1-60558-193-4. DOI: 10.1145/1401890.1402008.

[Tie17] Niels Tielenburg. "Automating Outlier Detection in Academic Publishing". MA thesis. Open University of the Netherlands, June 21, 2017. 62 pp. URL: https://www.open.ou.nl/hjo/supervision/2017-niels.tielenburg/2017-n.tielenburg-msc-thesis.pdf (visited on 03/07/2022).

[Van14] Richard Van Noorden. "Transparency Promised for Vilified Impact Factor". In: *Nature* (July 29, 2014). ISSN: 1476-4687. DOI: 10.1038/nature.2014.15642.

[Van21] Richard Van Noorden. "Hundreds of Gibberish Papers Still Lurk in the Scientific Literature". In: *Nature* 594.7862 (7862 May 27, 2021), pp. 160–161. DOI: 10.1038/d41586-021-01436-7.

[Wes22] Ewoud Westerbaan. "Acquisition and Integration of Public Data to Improve Detection of Scientific Fraud". MA thesis. Open University of the Netherlands, Apr. 1, 2022. URL: https://www.open.ou.nl/hjo/supervision/2022-e.westerbaan-msc-thesis.pdf.

[WF12] Allen W. Wilhite and Eric A. Fong. "Coercive Citation in Academic Publishing". In: *Science* 335.6068 (Feb. 3, 2012), pp. 542–543. DOI: 10.1126/science.1212540.

Appendix A

Additional tables

This appendix contains several tables that provide additional information, like an overview of the ACM journals scraped (Section A.1), the abbreviations used for nodes and relationship types in path labels (Section A.2), and the list of all found person cycles (Section A.3).

A.1 ACM journals

Table A.1 contains a comparison between the ACM journals that Westerbaan [Wes22] scraped and the ones we scraped. For the ones we scraped, we tried scraping the editorial boards (ED) and issue pages (IP). Y means scraped; N means tried, but failed; empty means not tried. All the failed editorial boards scraped journals had no editorial board page. All the failed issue page scraped journals had no issue pages on their website.

Westerbaan also tried to scrape the following three URLs, in which he did not succeed because they are not typically ACM journals URLs. We ignored them.

- `http://www.is.umbc.edu/taccess/index.h`
- `https://ubiquity.acm.org/meet_the_editors`
- `https://campus.acm.org/public/genpubqj/genpubqj_control.cfm?promo=QJPUB&product=11240&form_type=PUB`

During our study, the journals listed on the ACM journals page (`https://dl.acm.org/journals`) changed, finally stabilizing on 66 of 67 journals listed in the table, only `tslp` is currently not listed on the ACM journals page. This journal page was not available at the time of Westerbaan's study.

Code	Title	[Wes22]	ED	IP
cola	Collective Intelligence		Y	N
csur	ACM Computing Surveys	N	Y	Y
dgov	Digital Government: Research and Practice		Y	Y
dlt	Distributed Ledger Technologies: Research and Practice		Y	N
dtrap	Digital Threats: Research and Practice		Y	Y
fac	Formal Aspects of Computing		Y	Y
games	ACM Games: Research and Practice		Y	N

Table continues on next page

Code	Title	[Wes22]	ED	IP
health	ACM Transactions on Computing for Healthcare		Y	Y
imwut	Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies		Y	Y
jacm	Journal of the ACM	Y	Y	Y
jats	ACM Journal on Autonomous Transportation Systems		Y	N
jdiq	Journal of Data and Information Quality	Y	Y	Y
jea	ACM Journal of Experimental Algorithmics	N	Y	Y
jetc	ACM Journal on Emerging Technologies in Computing Systems	N	Y	Y
jocch	Journal on Computing and Cultural Heritage	Y	Y	Y
jrc	ACM Journal on Responsible Computing		N	N
pacmcgit	Proceedings of the ACM on Computer Graphics and Interactive Techniques		Y	Y
pacmhci	Proceedings of the ACM on Human-Computer Interaction		Y	Y
pacmpl	Proceedings of the ACM on Programming Languages		Y	Y
pomacs	Proceedings of the ACM on Measurement and Analysis of Computing Systems		Y	Y
taas	ACM Transactions on Autonomous and Adaptive Systems	Y	Y	Y
taccess	ACM Transactions on Accessible Computing		Y	Y
taco	ACM Transactions on Architecture and Code Optimization	Y	Y	Y
talg	ACM Transactions on Algorithms	Y	Y	Y
tallip	ACM Transactions on Asian and Low-Resource Language Information Processing	N	Y	Y
tap	ACM Transactions on Applied Perception	Y	Y	Y
taslp	IEEE/ACM Transactions on Audio, Speech and Language Processing		N	Y
tcbb	IEEE/ACM Transactions on Computational Biology and Bioinformatics	N	N	Y
tcps	ACM Transactions on Cyber-Physical Systems		Y	Y
tds	ACM/IMS Transactions on Data Science		Y	Y
teac	ACM Transactions on Economics and Computation		Y	Y
tecs	ACM Transactions on Embedded Computing Systems	N	Y	Y
telo	ACM Transactions on Evolutionary Learning and Optimization		Y	Y
thri	ACM Transactions on Human-Robot Interaction		Y	Y
tiis	ACM Transactions on Interactive Intelligent Systems		Y	Y
tiot	ACM Transactions on Internet of Things		Y	Y

Table continues on next page

Code	Title	[Wes22]	ED	IP
tist	ACM Transactions on Intelligent Systems and Technology		Y	Y
tkdd	ACM Transactions on Knowledge Discovery from Data	N	Y	Y
tmis	ACM Transactions on Management Information Systems		Y	Y
toce	ACM Transactions on Computing Education	Y	Y	Y
tochi	ACM Transactions on Computer-Human Interaction	Y	Y	Y
tocl	ACM Transactions on Computational Logic	Y	Y	Y
tocs	ACM Transactions on Computer Systems	Y	Y	Y
toct	ACM Transactions on Computation Theory	Y	Y	Y
todaes	ACM Transactions on Design Automation of Electronic Systems	Y	Y	Y
tods	ACM Transactions on Database Systems	Y	Y	Y
tog	ACM Transactions on Graphics	Y	Y	Y
tois	ACM Transactions on Information Systems	Y	Y	Y
toit	ACM Transactions on Internet Technology	Y	Y	Y
tomacs	ACM Transactions on Modeling and Computer Simulation	Y	Y	Y
tomm	ACM Transactions on Multimedia Computing, Communications, and Applications	Y	Y	Y
tompecs	ACM Transactions on Modeling and Performance Evaluation of Computing Systems		Y	Y
toms	ACM Transactions on Mathematical Software	Y	Y	Y
ton	IEEE/ACM Transactions on Networking	N	N	Y
topc	ACM Transactions on Parallel Computing		Y	Y
toplas	ACM Transactions on Programming Languages and Systems	Y	Y	Y
tops	ACM Transactions on Privacy and Security	N	Y	Y
tors	ACM Transactions on Recommender Systems		Y	N
tos	ACM Transactions on Storage	Y	Y	Y
tosem	ACM Transactions on Software Engineering and Methodology	Y	Y	Y
tosn	ACM Transactions on Sensor Networks	Y	Y	Y
tqc	ACM Transactions on Quantum Computing		Y	Y
trets	ACM Transactions on Reconfigurable Technology and Systems	N	Y	Y
tsas	ACM Transactions on Spatial Algorithms and Systems		Y	Y
tsc	ACM Transactions on Social Computing		Y	Y
tweb	ACM Transactions on the Web	Y	Y	Y
tslp	ACM Transactions on Speech and Language Processing	N	N	Y

Table A.1: ACM Journals data sources

A.2 Node and relationship types abbreviations

Table A.2 and Table A.3 contain the full list of label abbreviations of nodes and relationships. Where the abbreviation does not follow the first letter of the type rule, the abbreviation is written in italics.

Node type	**Abbreviation**
Article	A
Conference	C
DOI	D
Issue	I
Journal	J
Orcid	O
Person	P
Proceeding	*R*
Volume	V

Table A.2: Node label abbriviations

Relationship type	**Abbreviation**
author_of	a
belongs_to	b
cites	c
doi_of	d
editor_of	e
of	*f*
orcid_of	o
pc_member_of	p

Table A.3: Relationship label abbriviations

A.3 Person cycles per type

In Table A.4 all cycles starting from all 3,019,026 persons with max 6 edges are listed. We mention the label, by how many persons this cycle type occurrence, the total amount of recursive cycles, the median (50th percentile), the 75th percentile (p75), and the largest amount of a person (max or 100th percentile). Most of the cycle types will probably have an exponential distribution, so the 50th (median), 75th, and 100th (max) percentile are shown to give a first sight indication of the distribution.

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PaAaPaAaP	1,312,006	10,490,094	4	8	1,030	343,337,520
PaAaPaAaPaAaP	1,237,134	107,131,832	24	75	44,500	12,081,321,020
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AaP	868,531	456,984,876	88	386	84,364	8,597,894,806
PaAaPaAbRbAaP	834,650	26,613,452	4	14	25,416	294,051,200
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	769,845	19,182,625	7	20	5,932	134,269,713
PaA$\overleftarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	726,988	29,812,211	11	36	7,013	438,404,844
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AaPaAaP	712,850	32,507,817	8	26	24,661	652,871,850
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	707,054	33,432,919	11	37	12,049	282,526,178
PaA$\overrightarrow{c}$A$\overleftarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AaP	663,634	59,106,868	16	61	28,324	651,534,629
PaA$\overleftarrow{c}$AaPaA$\overrightarrow{c}$AaP	656,855	10,093,436	5	14	1,891	142,692,006
PaA$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AaP	634,588	21,066,702	6	21	11,356	186,224,711
PaA$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$AaP	611,921	53,621,360	16	60	27,311	1,624,682,094
PaA$\overleftarrow{c}$AaPaAaP	595,025	5,060,641	3	8	1,322	64,140,125
PaA$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	565,378	20,627,009	7	25	9,767	219,629,230
PaA$\overrightarrow{c}$AaPaAaP	519,240	5,096,820	3	8	1,844	67,135,002
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$AaP	512,925	17,465,000	5	19	16,047	82,658,708
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AaP	505,417	43,797,072	10	44	29,291	423,451,399
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AaPaAaP	501,339	10,490,559	5	14	5,396	173,889,535
PaAbRfCfRbAaP	494,611	1,129,650	1	2	127	31,510,876
PaA$\overrightarrow{c}$AaPaA$\overleftarrow{c}$AaP	494,102	13,792,665	6	19	13,140	179,279,214
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AaP	431,647	4,712,151	4	10	1,119	29,094,420
PaA$\overleftarrow{c}$A$\overleftarrow{c}$AaPaAaP	428,644	5,266,214	4	10	2,150	65,825,761
PaA$\overrightarrow{c}$AaP	419,836	2,102,583	2	4	661	3,283,060
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$A$\overleftarrow{c}$AaP	412,942	27,611,610	8	35	26,773	295,264,382
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AbIbAaP	390,334	6,552,177	4	11	4,468	20,149,885
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$AaP	372,161	14,386,600	6	21	13,525	95,174,377
PaA$\overrightarrow{c}$AaPaA$\overrightarrow{c}$AaP	365,800	4,383,853	3	9	2,053	39,221,232

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AbRbAaP	351,761	13,186,715	6	20	8,533	60,528,191
PaAbRbAaP	339,423	968,316	1	3	181	4,215,448
PaAbIbAaPaAaP	335,052	2,906,327	2	5	7,075	20,739,569
PaA$\overrightarrow{c}$A$\overrightarrow{c}$AaPaAaP	326,169	5,088,825	3	10	4,600	61,715,996
PaAbVfJfVbAaP	293,459	390,966	1	1	30	31,704,486
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AbIbAaP	285,932	3,086,269	3	8	2,018	8,517,169
PaA$\overleftarrow{c}$AbIbA$\overrightarrow{c}$AaP	277,395	902,635	1	3	444	7,878,658
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AaP	271,346	3,295,770	3	8	3,243	13,723,948
PaAaPpRpPaAaP	265,889	4,965,157	5	30	720	1,184,794,304
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AbVbAaP	225,069	4,045,737	4	13	3,660	11,781,184
PaAbRpPpRbAaP	224,618	4,376,853	3	7	3,847	35,999,484
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AbRbAaP	215,665	4,998,204	4	12	6,309	18,262,545
PaAaPaAbVbAaP	213,080	8,049,874	4	17	10,880	59,074,182
PaA$\overleftarrow{c}$AbRbA$\overrightarrow{c}$AaP	207,961	1,014,401	2	4	379	12,367,332
PaA$\overleftarrow{c}$AaPpRbAaP	189,686	1,243,456	2	6	605	5,854,239
PaA$\overrightarrow{c}$AbIbA$\overleftarrow{c}$AaP	183,420	1,055,082	2	4	1,413	9,575,086
PaAaPpRbAaP	177,083	663,272	1	3	354	7,298,425
PaA$\overrightarrow{c}$AbRbA$\overleftarrow{c}$AaP	152,605	1,153,270	2	5	1,414	16,946,364
PaAbIfVfIbAaP	146,315	310,406	1	2	157	1,397,222
PaA$\overrightarrow{c}$A$\overrightarrow{c}$AaP	142,254	723,798	2	4	962	1,824,508
PaA$\overrightarrow{c}$AaPpRbAaP	134,316	933,354	2	5	1,897	4,553,713
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AbVbAaP	128,660	912,866	2	6	1,025	1,900,446
PaA$\overrightarrow{c}$AbRpPaAaP	119,810	632,036	2	4	480	15,907,203
PaA$\overrightarrow{c}$AbIbAaP	117,988	377,967	1	2	629	772,119
PaA$\overleftarrow{c}$AbRpPaAaP	117,691	465,642	2	4	137	8,852,541
PaA$\overrightarrow{c}$AbVbA$\overleftarrow{c}$AaP	112,064	492,732	1	3	1,146	5,723,572
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	98,509	1,159,394	3	8	2,402	5,041,511
PaA$\overleftarrow{c}$AbRbAaP	96,699	490,652	2	4	651	1,241,550
PaA$\overleftarrow{c}$AbVbA$\overrightarrow{c}$AaP	95,927	219,299	1	2	290	1,396,394
PaAbIbAaP	93,211	162,340	1	2	170	474,136
PaA$\overrightarrow{c}$AbRbAaP	92,624	537,095	2	4	763	1,454,735
PaA$\overrightarrow{c}$A$\overrightarrow{c}$AbIbAaP	86,992	870,842	2	6	1,853	2,014,918
PaA$\overleftarrow{c}$AbIbAaP	85,127	258,780	1	3	484	591,737
PaA$\overleftarrow{c}$A$\overleftarrow{c}$AbIbAaP	81,933	587,567	2	5	900	1,328,107

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PaA$\overleftarrow{c}$A$\overleftarrow{c}$AbRbAaP	80,335	1,105,559	3	9	1,904	2,879,118
PaA$\overrightarrow{c}$AbIbA$\overrightarrow{c}$AaP	77,063	295,387	2	4	381	1,394,601
PaA$\overrightarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AaP	76,672	1,029,957	3	10	1,866	4,496,523
PaA$\overrightarrow{c}$A$\overrightarrow{c}$AbRbAaP	74,040	1,467,322	3	10	2,973	4,202,284
PaAbVbAaP	57,718	90,775	1	2	53	508,438
PpRbAaPaAbRpP	57,032	6,119,678	25	89	7,486	45,267,970
PaA$\overrightarrow{c}$AbRbA$\overrightarrow{c}$AaP	54,873	266,551	2	5	195	2,071,448
PaA$\overrightarrow{c}$AbVbAaP	54,698	141,166	1	2	456	301,678
PaA$\overrightarrow{c}$A$\overrightarrow{c}$AbVbAaP	52,228	387,778	2	5	1,815	802,772
PpRbAaPpRpP	51,898	6,113,222	13	63	4,254	159,721,715
PaAbRbAaPpRpP	50,701	20,123,471	35	173	27,842	748,934,174
PaAaPaAaPpRpP	49,136	18,853,380	70	264	59,756	1,381,319,803
PpRpPpRfCfRpP	48,439	1,832,932	18	50	975	36,753,584,791
PpRpPaAaPpRpP	47,830	127,232,103	259	1,321	84,119	229,059,092,932
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AaPpRpP	46,998	27,866,583	73	450	40,438	891,818,818
PpRbA$\overleftarrow{c}$A$\overrightarrow{c}$AbRpP	45,008	10,978,841	40	164	6,985	609,027,768
PaAaPaAbRpP	43,734	883,830	6	18	1,773	5,891,245
PpRpPpRpPpRpP	42,955	8,205,259	98	290	3,541	18,483,115,279,482
PpRbA$\overleftarrow{c}$AaPpRpP	42,072	12,292,291	45	204	9,073	821,906,725
PaAaPpRpPpRpP	41,125	2,175,175	22	71	1,176	27,945,993,665
PaA$\overrightarrow{c}$A$\overleftarrow{c}$AaPpRpP	40,997	4,666,334	22	94	5,926	69,055,851
PpRbA$\overrightarrow{c}$A$\overleftarrow{c}$AbRpP	40,804	11,638,984	12	68	11,585	75,601,342
PaAaPpRpP	40,290	223,310	3	6	265	6,144,750
PaA$\overleftarrow{c}$AaPaAbRpP	40,071	873,919	7	23	1,115	4,379,418
PpRbA$\overrightarrow{c}$AaPpRpP	40,063	10,023,768	22	117	9,061	432,232,351
PaA$\overrightarrow{c}$AbVbA$\overrightarrow{c}$AaP	39,963	125,945	1	3	432	535,840
PaAaPpRfCfRpP	39,797	681,620	7	21	436	85,653,311
PaAbRpPaAbRpP	39,455	695,900	5	14	724	2,300,508
PaA$\overrightarrow{c}$AaPaAbRpP	38,796	1,136,292	8	27	2,191	5,480,420
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AbRpP	38,658	6,354,001	35	140	5,780	48,488,538
PaA$\overleftarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AbRpP	38,527	2,689,671	20	68	2,929	21,819,731
PaAbRfCfRpP	37,789	157,355	2	5	88	1,031,700
PaA$\overrightarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AbRpP	37,766	3,879,772	26	91	4,351	41,436,717
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overleftarrow{c}$AbRpP	37,509	9,626,130	24	123	14,267	59,362,167

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	Persons	Total	Median	P75	Max	Total
$\mathtt{PaA\overleftarrow{c}A\overleftarrow{c}AaPpRpP}$	36,575	2,171,032	14	55	3,056	30,746,499
$\mathtt{PaA\overleftarrow{c}AaPpRpP}$	35,536	797,378	7	22	1,114	8,361,823
PaAbRpPpRpP	35,502	1,457,763	6	25	1,372	29,750,726
$\mathtt{PaAaPaA\overrightarrow{c}AbRpP}$	35,092	1,449,219	11	37	1,987	13,300,323
$\mathtt{PaA\overleftarrow{c}A\overrightarrow{c}AbRpP}$	35,078	2,437,394	13	48	2,820	8,607,166
$\mathtt{PaA\overleftarrow{c}AbVbAaP}$	34,729	77,863	1	2	180	166,852
$\mathtt{PaA\overrightarrow{c}A\overrightarrow{c}AaPpRpP}$	34,146	2,724,440	13	57	6,061	31,310,409
$\mathtt{PaA\overrightarrow{c}AaPpRpP}$	33,961	885,945	7	22	1,679	8,485,080
$\mathtt{PaA\overleftarrow{c}A\overleftarrow{c}AbVbAaP}$	33,904	139,304	2	4	381	253,512
PaAbIbAaPpRpP	33,364	805,989	6	19	3,544	10,924,549
$\mathtt{PaAaPaA\overleftarrow{c}AbRpP}$	30,665	1,157,972	7	26	3,474	8,287,109
PpRfCfRpP	30,547	45,717	1	2	24	4,593,786
$\mathtt{PaA\overrightarrow{c}AbRpPpRpP}$	30,081	526,904	5	17	568	153,390,939
PpRpPpRpP	29,893	10,788,540	49	265	5,069	5,988,883,392
$\mathtt{PaAbRbA\overrightarrow{c}AbRpP}$	29,691	1,227,065	11	36	4,242	2,801,026
$\mathtt{PpRbA\overrightarrow{c}AbRpP}$	28,380	2,726,079	8	42	3,600	4,982,053
$\mathtt{PaA\overleftarrow{c}AbRpPpRpP}$	28,325	326,870	5	13	283	239,915,885
PaAbRpP	27,936	67,156	1	3	87	101,098
$\mathtt{PaA\overrightarrow{c}A\overrightarrow{c}A\overleftarrow{c}AbRpP}$	26,565	2,095,227	9	42	7,411	8,036,581
$\mathtt{PaA\overrightarrow{c}AbRfCfRpP}$	26,221	222,424	4	9	312	2,051,334
$\mathtt{PaA\overleftarrow{c}AbRfCfRpP}$	25,308	196,914	4	9	145	2,429,915
$\mathtt{PaA\overrightarrow{c}A\overleftarrow{c}A\overleftarrow{c}AbRpP}$	24,033	1,114,425	7	30	2,767	4,642,093
$\mathtt{PaA\overrightarrow{c}A\overleftarrow{c}AbRpP}$	23,215	698,514	6	20	2,484	1,781,902
$\mathtt{PpRbA\overrightarrow{c}A\overrightarrow{c}AbRpP}$	22,957	1,362,965	7	25	2,839	5,318,193
$\mathtt{PaAbRbA\overleftarrow{c}AbRpP}$	22,738	853,168	7	26	3,691	1,831,412
PaAbVbAaPpRpP	20,595	4,429,186	8	52	8,988	166,071,666
$\mathtt{PaA\overrightarrow{c}AbIePaAaP}$	19,362	47,913	1	3	68	752,092
PpRpPeIePpRpP	19,313	748,000	29	54	237	1,817,295,782
$\mathtt{PaA\overrightarrow{c}A\overrightarrow{c}AbRpP}$	19,009	372,079	5	15	1,662	813,171
$\mathtt{PaA\overrightarrow{c}A\overrightarrow{c}A\overrightarrow{c}AbRpP}$	18,900	551,064	6	21	2,612	1,970,884
PaAaPeIePaAaP	18,566	262,689	10	18	137	23,425,922
$\mathtt{PaA\overrightarrow{c}AbRpP}$	17,929	128,604	3	7	377	177,595
$\mathtt{PaAbIbA\overrightarrow{c}AbRpP}$	14,544	85,040	2	5	410	152,165
$\mathtt{PaA\overleftarrow{c}A\overleftarrow{c}AbRpP}$	12,449	146,620	3	10	533	311,798

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PaA$\overleftarrow{c}$AbRpP	12,398	63,326	2	5	206	88,580
PaA$\overleftarrow{c}$A$\overleftarrow{c}$A$\overleftarrow{c}$AbRpP	12,376	195,573	4	13	849	583,145
PaAbIbA$\overleftarrow{c}$AbRpP	12,032	86,893	3	7	334	149,271
PaAaPeIePpRpP	9,945	212,219	14	27	197	26,286,992
PaA$\overleftarrow{c}$AaPeIbAaP	9,619	52,120	3	6	99	232,060
PaA$\overrightarrow{c}$AbIePpRpP	9,341	27,451	2	4	63	1,005,184
PaAbIePpRbAaP	6,241	24,227	2	5	77	56,211
PaA$\overrightarrow{c}$AaPeIbAaP	5,695	25,599	2	5	130	121,573
PaAbVbA$\overrightarrow{c}$AbRpP	5,584	18,473	2	3	77	26,573
PaAbIePaAaP	4,349	9,119	1	2	38	68,269
PaAbVbA$\overleftarrow{c}$AbRpP	4,259	19,145	2	4	239	28,953
PaA$\overleftarrow{c}$AbIePaAaP	2,945	3,905	1	1	6	35,240
PaAbIePaAbRpP	2,011	7,373	2	4	85	19,074
PaAbIePeIbAaP	1,865	80,303	25	37	402	408,668
PaAbIePpRpP	1,617	7,197	2	5	54	61,468
PaA$\overleftarrow{c}$AbIePpRpP	1,545	2,033	1	1	6	66,570
PaA$\overleftarrow{c}$AbVfIbAaP	1,443	1,794	1	1	6	4,781
PeIePeIeP	1,443	92,340	40	107	289	12,010,524
PeIfVfIeP	1,443	3,543	2	3	9	56,386
PeIePaAaPeIeP	1,441	544,149	181	511	2,061	172,362,924
PeIbA$\overleftarrow{c}$AaPeIeP	1,425	285,626	78	252	1,517	5,279,814
PeIbA$\overleftarrow{c}$A$\overrightarrow{c}$AbIeP	1,425	545,987	120	445	2,595	4,697,076
PeIbAaPaAbIeP	1,423	90,589	30	79	480	473,408
PeIbAaPeIeP	1,415	34,570	12	31	144	394,042
PeIfVfJfVfIeP	1,411	1,531	1	1	4	109,386
PeIePpRpPeIeP	1,376	109,849	42	104	402	188,559,138
PaAbRbAaPeIeP	1,338	342,132	119	342	2,346	6,420,541
PaA$\overleftarrow{c}$A$\overrightarrow{c}$AaPeIeP	1,331	476,561	195	491	2,448	18,365,592
PaAaPaAaPeIeP	1,325	320,770	133	322	3,279	21,447,282
PaA$\overrightarrow{c}$AbVfIbAaP	1,321	1,720	1	1	9	5,783
PaA$\overleftarrow{c}$A$\overrightarrow{c}$A$\overrightarrow{c}$AbIeP	1,304	412,332	138	437	3,543	2,205,277
PeIbA$\overrightarrow{c}$AbIeP	1,291	41,172	9	26	383	56,299
PaAaPaA$\overrightarrow{c}$AbIeP	1,282	86,107	32	85	842	534,105
PaA$\overleftarrow{c}$A$\overleftarrow{c}$A$\overrightarrow{c}$AbIeP	1,278	147,363	51	146	1,586	673,152

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PaA$\vec{c}$A$\overleftarrow{c}$A$\vec{c}$AbIeP	1,268	303,855	98	308	2,427	1,908,570
PaAaPpRpPeIeP	1,260	64,856	33	67	487	18,835,433
PaA$\vec{c}$A$\overleftarrow{c}$AaPeIeP	1,236	215,585	68	184	1,918	6,726,872
PaA$\overleftarrow{c}$A$\overleftarrow{c}$AaPeIeP	1,223	89,480	29	82	715	2,111,940
PaA$\overleftarrow{c}$A$\vec{c}$AbIeP	1,220	62,432	24	63	746	208,091
PaA$\vec{c}$AaPaAbIeP	1,177	44,658	17	45	505	208,837
PaA$\overleftarrow{c}$AaPaAbIeP	1,172	25,091	12	26	337	120,432
PeIbA$\vec{c}$AaPeIeP	1,172	18,896	6	26	103	287,995
PaAbIbAaPeIeP	1,164	24,521	10	25	189	320,759
PaA$\vec{c}$A$\vec{c}$AaPeIeP	1,149	103,667	31	93	1,426	2,202,105
PaA$\overleftarrow{c}$AaPeIeP	1,139	25,301	10	26	358	524,337
PaAbRbA$\vec{c}$AbIeP	1,131	123,546	29	112	1,054	275,918
PaA$\overleftarrow{c}$A$\vec{c}$A$\overleftarrow{c}$AbIeP	1,130	40,644	11	43	710	148,092
PeIbA$\vec{c}$A$\overleftarrow{c}$AbIeP	1,108	31,153	5	64	152	131,696
PaA$\vec{c}$A$\vec{c}$A$\vec{c}$AbIeP	1,098	83,451	21	80	1,609	316,754
PaAbRpPaAbIeP	1,080	7,373	4	9	91	19,074
PaAbRpPeIeP	1,076	9,616	5	12	77	171,780
PaAaPaAbIeP	1,072	11,166	5	12	141	60,172
PaA$\overleftarrow{c}$AbRpPeIeP	1,070	11,242	7	15	58	452,934
PaA$\vec{c}$AaPeIeP	1,067	27,479	11	30	414	538,946
PaA$\vec{c}$AbRpPeIeP	1,059	17,919	11	23	174	842,472
PaA$\vec{c}$A$\vec{c}$AbIeP	1,008	29,845	9	28	613	69,736
PeIePaAaPpRpP	969	378,520	122	510	4,847	28,384,600
PeIePeIePeIeP	957	30,851	35	38	106	28,811,492
PeIbA$\overleftarrow{c}$AaPpRpP	945	111,139	34	117	1,581	1,093,438
PeIbAaPaAbRpP	942	24,227	12	30	318	56,211
PaAaPeIeP	930	3,248	3	4	26	176,008
PaA$\vec{c}$A$\vec{c}$A$\overleftarrow{c}$AbIeP	917	16,151	6	22	175	49,040
PeIbA$\overleftarrow{c}$A$\vec{c}$AbRpP	905	127,915	52	150	1,487	628,594
PaAbIbA$\vec{c}$AbIeP	901	25,977	5	13	1,130	93,252
PeIePpRpPpRpP	899	95,128	57	151	717	431,244,178
PeIePpRfCfRpP	863	16,833	11	28	190	2,196,395
PeIePaA$\vec{c}$AbRpP	859	78,666	36	108	1,216	1,384,720
PeIePaAbRpP	855	18,876	10	25	222	206,027

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
PeIePaA$\overleftarrow{c}$AbRpP	829	59,329	22	80	945	837,942
PaAaPaA$\overleftarrow{c}$AbIeP	827	5,532	3	8	142	27,509
PeIbA$\vec{c}$A$\vec{c}$AbIeP	767	5,410	2	14	28	8,429
PeIbA$\overleftarrow{c}$A$\overleftarrow{c}$AbRpP	766	36,974	14	51	730	77,751
PaA$\vec{c}$AbIeP	765	4,380	3	6	77	6,373
PaAaPeIePeIeP	738	15,458	17	28	69	2,999,906
PeIbAaPpRpP	727	10,164	5	15	182	67,862
PaAbIfVbAaP	697	818	1	1	5	1,112
PeIePpRpP	674	4,442	3	9	54	181,524
PaA$\vec{c}$AbIePeIeP	663	2,392	2	5	25	91,315
PaA$\vec{c}$A$\overleftarrow{c}$AbIeP	629	4,795	3	7	95	8,609
PaAbVbAaPeIeP	618	21,382	6	35	456	622,783
PeIbA$\vec{c}$AaPpRpP	599	7,563	3	12	152	56,594
PeIbA$\overleftarrow{c}$AbRpP	592	8,359	5	14	183	10,289
PeIePeIePpRpP	563	17,331	22	46	96	7,549,910
PaAbRbA$\overleftarrow{c}$AbIeP	548	6,188	5	16	81	12,535
PeIbA$\vec{c}$A$\overleftarrow{c}$AbRpP	521	5,311	3	10	111	9,864
PaA$\vec{c}$A$\overleftarrow{c}$A$\overleftarrow{c}$AbIeP	404	1,143	2	3	29	1,980
PaAbIeP	333	534	1	2	10	598
PaAbVbA$\vec{c}$AbIeP	304	1,230	2	4	52	1,487
PaA$\vec{c}$AbIfVbAaP	264	339	1	1	7	483
PaAbIbA$\overleftarrow{c}$AbIeP	204	759	2	4	64	1,044
PaAbIePaAbIeP	168	496	2	3	16	834
PaAbIePeIeP	168	478	2	4	9	6,685
PaA$\overleftarrow{c}$AbIfVbAaP	157	181	1	1	5	277
PaA$\overleftarrow{c}$AbIeP	149	284	1	2	18	314
PaAbVbA$\overleftarrow{c}$AbIeP	127	556	2	4	69	738
PaA$\overleftarrow{c}$AbIePeIeP	120	156	1	2	3	4,538
PeIbA$\vec{c}$AbRpP	101	461	3	5	39	503
PaA$\overleftarrow{c}$A$\overleftarrow{c}$AbIeP	72	144	1	2	14	175
PaA$\vec{c}$AbIfVfIeP	70	73	1	1	2	518
PeIbA$\vec{c}$A$\vec{c}$AbRpP	44	101	1	3	12	155
PaAbIfVfIeP	29	29	1	1	1	116
PaA$\overleftarrow{c}$A$\overleftarrow{c}$A$\overleftarrow{c}$AbIeP	21	30	1	1	3	36

Table continues on next page

		Recursive cycles				Cycles
Cycle type label	**Persons**	**Total**	**Median**	**P75**	**Max**	**Total**
`PaA`$\overleftarrow{\texttt{c}}$`AbIfVfIeP`	10	10	1	1	1	53
`PaAbVfJfVfIeP`	2	2	1	1	1	16

Table A.4: Cycle occurrences