Improved plagiarism detection with collaboration network visualization based on source-code similarity

Matija Novak
Faculty of Organization and Informatics
University of Zagreb
Varaždin, Croatia
matnovak@foi.hr

Mike S. Joy
Department of Computer Science
University of Warwick
Coventry, United Kingdom
M.S.Joy@warwick.ac.uk

Olfat M. Mirza
College of Computer and Information Systems
Umm Al-Qura University
Makkah, Saudi Arabia
ommirza@uqu.edu.sa

Abstract*—Plagiarism detection is a serious problem in higher education. Teachers use similarity (plagiarism) detection systems, which highlight similarities between student documents, to help them find plagiarism. Most systems are built for text but there are special systems to find similarities between source-code files. In most cases the results are presented in table form showing similarities between pairs of documents in descending order by similarity, and then a teacher is responsible for confirming which similar documents represent cases of plagiarism. While most systems present their results in the form of tables, only few of them present the results as a graph. Some studies indicate that using clustering algorithms to represent such data graphically can improve the speed and accuracy of finding potential instances of plagiarism in large collections of source-code files. The purpose of the study is to answer the following research questions. Can visualization of student solutions (of source-code similarities) in collaboration networks form help identify new cases of plagiarism? What are the steps to do so? The study was designed in a form of two case studies where one was performed on a graduate level university course and one on a course in professional studies. The article presents empirical results describing two cases where a collaboration network (based on source-code similarity) representation has been used. The article argues that the graphical presentation is able to identify new clusters of plagiarised source-code files that would have been missed using existing tabular presentation of data.

***Keywords*—Plagiarism; Visualization; High Education; Source-code; Collaboration Networks**

I. Introduction

Automatic source-code plagiarism detection enables teachers to easily find plagiarised cases in student programming assignments. Various plagiarism detection engines (further referred as tools) exist which can be used for that purpose. There is no perfect tool, so reviews and comparisons of the systems are performed constantly [1, 2], which provide information to make a decision on which tool to use.

Although different, all tools in some way "calculate" similarity between two student solutions and give some kind of a report, which usually indicates potentially plagiarized pairs of solutions. The teacher then needs to check these pairs and decide if some specific cases are really plagiarized or not. The report itself comes in various forms, it can be just a simple listing, table or a graph, or – as Mišić, Siustran, and Protić observe – "Although tools usually present the results as a list of student pairs, it is more natural to interpret those results as a collaboration network." [3]

According to [4]: "A collaborative network (CN) is constituted by a variety of entities (e.g., organizations and people) that are largely autonomous, geographically distributed, and heterogeneous in terms of their: operating environment, culture, social capital, and goals." In this paper the collaboration network is a network of students and the similarities between their solutions for an assignment. The proposed approach involves searching for groups of students collaborating together using clustering algorithms, and visualising the data as a graph. Close collaboration is manifested as a separate group of nodes in the graph (which is refer to as a "cluster") and suggests possible plagiarism between the students in that group. In a network which has no plagiarism only one cluster should be present which includes all students.

In this research a case study is performed to show the usage of collaboration networks when detecting source-code plagiarism in student programming assignments. The rest of the paper is structured as follows. Section 2 describes the related work. Section 3 explains the implementation which forms the basis of the study, and in Section 4 the two cases are discussed. Section 5 describes the steps for performing visualization and gives suggestions on how to start to analyse the graphs. Section 6 suggests ideas for future work and Section 7 concludes.

II. Related work

Plagiarism detection systems had already been built in the 70's and 80's and research papers on the subject focused on the detection of plagiarism rather than the representations of results. Early plagiarism detection systems used only simple text or table reports to present data. Systems like JPlag [5] used improved versions of data presentation with two levels of report. First they gave an overall report and then there was a detailed report. The detailed report usually showed side by side comparison on selected pairs indicating which lines of code were similar.

Later on, systems incorporated graphical displays of results in a form of a simple bar charts, such as those by Gitchell and Tran [6]: "The bars are coloured red, yellow, and green in decreasing similarity to the reference program according to thresholds specified by the user". This way it is much easier to see at first glance to which degree the pairs may be plagiarised. More advanced graphs like those used by [7] show similarities between pairs in the whole corpus of submissions. Using visualization today is normal, for example Makuc [8] uses Force-Directed Graphs and Co-Occurrence matrices to visualize similarities.

As already stated the focus in this research is on using collaboration networks for visualization. Visualization methods operating on all submissions are used for example in the tool Sherlock [7] or by Mišić, Siustran and Protić [3, 9].

Mišić, Siustran and Protić [3] used the results from one course in the academic year 2012/2013, obtained with the MOSS [10] tool, and illustrated collaboration networks using a tool called Gephi [11]. They state that by using graphs and clustering they found three groups of students each containing more than two students whose programs had a high degree of similarity. By manual analysis they confirmed these represented real cases of code sharing and ghost writing.

Research as performed in [12] first created social networks of students' desired partners (team members) for projects, and then performed plagiarism detection on their assignments. Some assignments were done at home, some were done in class. They found out that assignments done at home correlated better with the students' preferred workmates for projects. That tells the teacher that they can tell something about the relationship based on the similarity between pairs. Or, by looking the other way around, it may be an indication that students tend to collude more with their friends than with classmates that they do not know, which gives another good reason for using network analysis to improve plagiarism detection.

Another related area comprises tools which use clustering techniques like Pdetect [13] or the fuzzy-based source-code plagiarism detection system described in [14]. In those systems clustering is used as a way to help improve plagiarism detection. Such tools, instead of just using string comparison and making a decision based on the similarities between two documents, create clusters based on the similarities. It is expected that plagiarised cases should belong to the same cluster. These projects report that clustering seems to have a positive impact on improving plagiarism detection, and that it can be combined with other tools.

In this paper an approach is suggested which uses only free available tools, that can be used by teachers offline, that can run different clustering algorithms and at the same time visualize the data as a graph (a collaboration network). The approach is also important since it offers the possibility to filter out edges to get different clusters, and hence support visualization of the results which potentially allows new potential plagiarism cases to be identified quickly.

III. Case Study Presentation

The case study was performed on a graduate level university course (case 1) and on a course in professional studies (case 2). The student assignments were written in the programming language Java in case 1, and in case 2 the programming language PHP with HTML and CSS was used. Students submitted their solutions to the online e-learning system Moodle and the detection was performed using Sherlock to do the similarity calculation. The detection process and report that were used are the same as used by [15]. The data were stored in a simple MySQL database.

The Sherlock system has been used for few years and has helped to detect plagiarised cases, but for some cases this is still not enough.

In this paper are described the two most interesting cases that were found using visualization in a form of a collaboration network. In each case, after the generation of the report, data were analysed manually for pairs with more than 20% similarity (this was decided by the teachers as a logical cut-off threshold). For example, the results for the first case are presented in Table 1 and show two pairs with high similarity and two with similarity around 20%. The top two pairs were clear cases of plagiarism. The other two, by looking at a side by side comparison, were not so clear. But, since student 17 was in three pairs, student 14 and student 22 in two pairs, further analysis was performed. The analysis showed that different parts of the students' submissions were copied and therefore all pairs were considered plagiarised.

TABLE I. Traditional results – case 1

Student ID 1	***Student ID 2***	***Similarity (0%-100%)***
17	14	45%
14	22	32%
17	22	25%
17	13	24%

Now the idea is to put the gathered data into the Gephi tool to visualize the results in graph form, or more precisely in this case it is a collaboration network (which is henceforth referred to as a *network*) on which some clustering algorithms were performed. This is the same principle as described by Mišić, Siustran and Protić [3, 9]. The terms *graph* and *network* in the rest of the paper are used synonymously.

Once the data were extracted from the database into CSV format, the nodes and edges were imported using the import function in Gephi, which automatically draws a graph (Fig. 1) for the data. After experiment the layout OpenOrb (a way to rearrange the nodes) with Nooverlap (to ensure that no node is presented over another node) was considered suitable in most cases, although sometimes the nodes were rearranged manually based on cluster colours to present a clearer view. Once the graph has been generated the clustering can be performed, and an example of the graph after clustering is shown in Fig. 2. In all figures with graphs the nodes represent students and the edges the similarity between two student solutions. The thicker the line the higher the similarity.

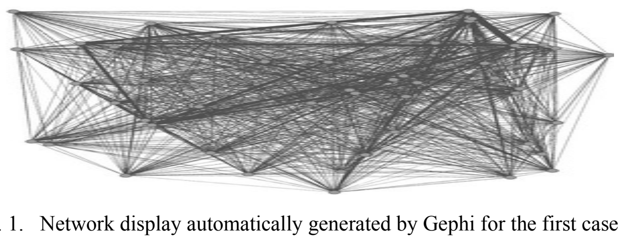
Fig. 1. Network display automatically generated by Gephi for the first case

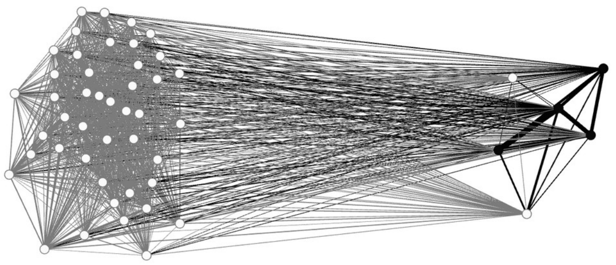

Fig. 2. Network display in OpenOrb with Nooverlap layout and after performing the Chinese Whispers clustering algorithm for the first case

IV. Discussion

In this section the results for each case are discussed. Since each case is different, each will be discussed in a separate subsection. A summary of the steps to get to visualization in Gephi, and some suggestions on how to start looking at graphs, are described in Section 5.

A. *Case 1 - Dataset from graduate level university course*

The first case described in the previous chapter was the first that was analysed using Gephi and is from the academic year 2015/2016. This case was chosen as the first since it was known that there must be at least one cluster as indicated by Table 1. From the collaboration network presented in Fig. 1 nothing can be really concluded, but just by choosing a different layout already it can be seen that some students are separated. But one needs to be careful that this may not mean anything. To further analyse the network, clustering algorithms and partitioning algorithm Community Detection (CD) [16] were performed. Fig. 3 shows the graph based on the Chinese Whispers clustering algorithm [17]. Tested algorithms available in Gephi (v.0.8.2 beta) and their results with default parameters on this first dataset were:

- Markov Cluster Algorithm MCL (experimental) – resulted in 1 cluster with 48 nodes;
- Chinese Whispers (Fig. 3) – resulted in 2 clusters with 5 and 46 nodes;
- KM Clustering – resulted in 1 cluster with all nodes;
- Markov Clustering – although multiple configurations were tried, the algorithm did not manage to deliver results even after 10 minutes of waiting, so it was stopped manually;
- Girvan Newman Clustering – in the setup of this algorithm one needs to select how many clusters should be generated. In this case there was the possibility to choose between 2, 3 or 51 clusters; while 51 is the number of nodes, it does not make sense to select that, so 2 and 3 clusters were tried out, but the resulting clusters were not useful at all;
- Community detection algorithm (CD algorithm) – 4 clusters with 4, 7, 10 and 30 nodes.

The Chinese Whispers algorithm and Community Detection algorithm gave the most interesting results, they formed clusters with the expected students (in Gephi different node colours represent the different clusters). After experimenting with parameters only the MCL algorithm was also able to partially form these expected clusters, all others failed to do that. So only the Chinese Whispers (plugin version 0.8, with 10 iterations, propagation type 'top' and minimum edge weight set to zero is used) and CD algorithms (with resolution set to 1 and use weights checked) were chosen to be used, and when referred to a clustering algorithm in the text, Chinese Whispers is meant if not stated otherwise.

Fig. 3 presents only the interesting part of the network displayed in Fig. 2 using the Chinese Whispers clustering algorithm. To get a better insight, all edges with similarity less than 10% were hidden using filters and the similarity numbers (ranging from 0% to 100%) are displayed.

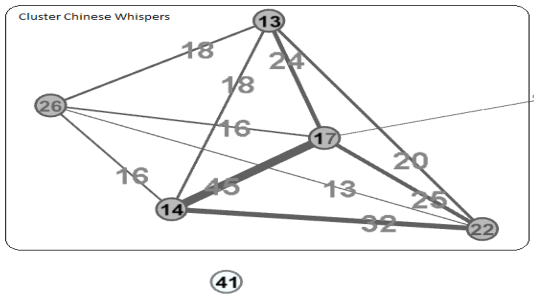

Fig. 3. Graph part with removed edges with less than 10% similarity and with similarity displayed for the first case

From Fig. 3 it can be clearly seen that student 41 does not really belong to this group as correctly indicated by the Chinese Whispers clustering algorithm. Student 47 has a connection to student 17, but by a side by side source-code comparison it was clear that this student also does not belong to this group. The other students 26, 17, 13, 14 and 22 were correctly identified as a group and side by side comparison source-code confirmed it. For case 1 the Chinese Whispers algorithm gave excellent results and the Community Detection algorithm can be considered as very good. As already stated all other algorithms failed to give any satisfactory results.

Another thing that needs to be noticed is that this analysis showed one more student solution which was not identified in a traditional report. It is also visible from Fig. 3 that every student in this group has some similarity around 20% with every other student in the group. The side by side analysis of the source-code also showed that this involved different parts of code when looking at one student and comparing their code to others'. Also it was interesting to see (in Fig. 3) that almost all students had similarity greater than 10%, which tells the teacher that all cases could have been identified if the similarity cut off was 10% instead of 20%. But then again they would also get many other pairs which would lead to more false positives. So when lowering the cut-off is not possible the network analysis can help.

B. *Case 2 – Dataset from course in professional studies*

In the second case the interesting thing was that there were many pairs with high similarity. The dataset was from the academic year 2012/2013 and in this academic year no plagiarism detection tool was used, only manual inspection by the teacher was performed. It was interesting to see how high the similarity rates actually were. The initial graph was again not useful as in case 1. After performing the Chinese Whispers clustering algorithm and the CD algorithm nothing interesting was found. There was one big cluster and several with only one node.

The Chinese Whispers clustering algorithm had the possibility to perform clustering only on the visible part of the graph, but this was not possible with the CD algorithm, so only the Chinese Whispers clustering algorithm was used further. This feature was the key to case 2. So instead of performing the clustering and then filtering based on similarity as in the first case, the filtering was performed first. Multiple versions were tried out. The first version removed all similarities below 30% and performed clustering on the new graph, and this filter is presented in Fig. 4.

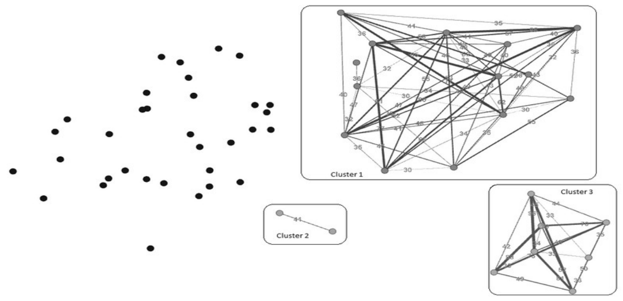

Fig. 4. Network display after Chinese Whispers clustering with 30%-100% similarity filter for case 2

In Fig. 4 one pair (cluster 2) was extracted as a separate cluster, and side by side comparison confirms it was a plagiarized pair but also there are two large clusters (cluster 1 and 3 which analysed further. Since the two bigger clusters contained many nodes the first instinctive idea was to remove everything with a high percentage similarity, so the percentage was incrementally increased to 65%. This situation with all edges filtered below 65% and the clusters looked fine at first, but by analysing the clusters it was found out that all the pairs had already been found by reading the existing report produced by the plagiarism detection system. So, the problem was that the clusters contained a lot of high similarity (more than 65%) which are clear plagiarized cases and were already found in the traditional way. So the new idea was to remove the high similarities instead of the lower ones.

Fig. 5 presents the network leaving only the edges with 30%–47% similarity. Although most of the found pairs remained, some new nodes were added into the clusters. By analysing the source-code side by side for the newly added nodes (in comparison to filtering out similarities above 65%) it was found out that the larger cluster (cluster 1) contained six new matches. In the smaller cluster (cluster 3) it was found out that it contained one new match. Thus 7 more students did plagiarize and were not found by only using the traditional report. Although they were in the report there were too many pairs with higher similarity so they were overlooked.

So this shows that visualization can help but it has to be mentioned that it is not always easy and it takes time to "play" with the filtering. But it can be "rewarding" as it was in this case.

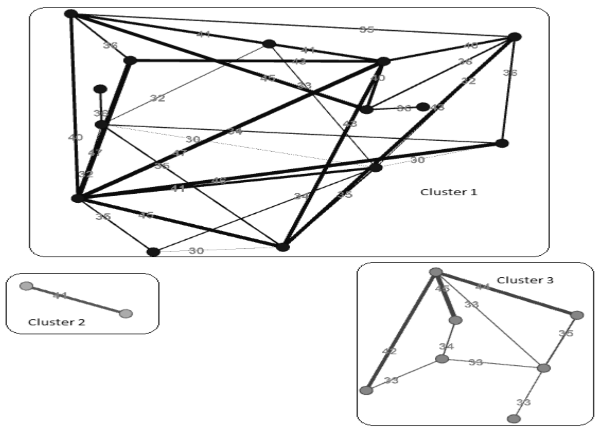

Fig. 5. Network display after Chinese Whispers clustering with 30% - 47% similarity filter for case 2

V. Steps and Suggestions for Using Visualization Techniques

From the two cases presented it is clear that this technique can be useful, but since it has to be performed manually, in this section it is described how to use the technique with Gephi and some suggestions as to what to look for. Note, the following suggestions are made based on previous experiences in using Gephi on the two presented cases (Section IV) but also by using this approach on other cases over the years.

More detailed information on how to use Gephi is available on the official website (https://gephi.org/) and [11]. To enable visualization using Gephi it is suggested using the following steps.

1) Perform detection with a traditional plagiarism detection tool like Sherlock, MOSS, JPlag or some other tool.

2) Extract the information about similarities between student pairs into a database or Excel.

3) Transform the data into Gephi readable format by creating two CSV files, one containing all nodes (representing the individual students) and one containing all edges (similarities between student submissions). The names of the fields are important so the nodes CSV must contain Label and Id fields. Labels will be displayed on the graph and Ids are the matches to the edges file. The edges CSV must contain the following fields: Source, Target, Type, Weight, and Label. Where the Source and Target are the Ids from the nodes CSV, the weight represents the similarity.

4) Import the two files into Gephi, first the nodes and second the edges.

5) Choose and run the desired clustering algorithm. To begin with suggestion is to use the Chinese Whispers clustering algorithm as it was shown that it can give good results in the described cases.

6) Analyse the graphs and use the filters to change the information displayed on the graphs. Note that if filters are applied you can rerun the clustering algorithm only on the visible graph part. The most useful filter is the edge weight filter, which is specified with two numbers, meaning remove everything below the first number and remove everything above the second number. Edge weight in this case represents the similarity between two student solutions.

Once the data are displayed and a clustering algorithm is run, the question becomes what to look for. To simplify the manual analysis for the teachers, presenting two main graphs were found to have the best cost/benefit ratios. Of course, it can be the case that there is no additional plagiarism to be found, or conversely that substantial extra manipulation of the parameters and filters would be needed to find new plagiarism cases.

A. *Analysing the full graph*

The first kind of graph – the whole graph – is the easiest to get, since no filtering is applied. Using filters to filter out desired similarity intervals can be helpful to reduce the noise created by "crossing lines", but since each cluster is colour coded (images in the paper are manually transformed to black and white) in Gephi it is easy to spot the clusters and it is not always necessary to use the filters. For quick removal of noise it is sufficient to filter the "uninteresting"' edges weighted less that (say) 10%.

A second action is to look for matches (suspected plagiarized pairs) already known to exist, to see if the clustering algorithm places the nodes of one pair in the same cluster. If not, an analysis (side by side comparison) of those nodes and the clusters containing the pairs will indicate why there are differences. Also, this is an easy way of testing new clustering algorithms. Sometimes analysing already known groups can suggest some new cases (as for example in case 1). Just check if the suspicious groups of students have some new students that have smaller similarities (maybe around 20%) but may have connections to other students in the same group.

Looking for small clusters with 2 to 5 nodes and establishing why they were extracted as separate cluster is the next stage in analysing a graph, and this is especially useful with clusters of only 2 nodes, which should always be analysed by pairwise comparison. Since the graph only gives numbers, the teacher needs to scrutinise the detailed report of the tool that was used to perform the detection in the first place. If there are many small clusters, it is useful to start looking at those containing the highest similarities.

Clusters with 6 or more nodes may take too much time to analyse pairwise, unless the similarities between most of the nodes in cluster are reasonably high (say, more than 20%, but judgment must be applied). Visualization here is a big help since Gephi uses line thickness to indicate the similarity value, and hence clusters with more thick lines are likely to be of most interest.

Big clusters are usually not of interest since they contain many nodes (students) and probably deal with common uninteresting similarities. Note that it is hard to specify what is meant by a "big" cluster, and this will depend on the size of dataset (class). For example, if there are 300 students in total then a cluster of 10 may be seen as small, but in case of 50 students, a cluster of 10 is considered big. Also, in a large group of students a cluster containing only 2 students is more suspicious than in a smaller group. We suggest that a rule of thumb is that a cluster is small if it contains up to 5 students, or in large classrooms 5% or fewer of the students, and that a cluster is big if it contains more than 20% of the students.

This graph will probably not identify many new cases, since the highest similarities were already found by the tool's pairwise comparison which would have placed those highest similarities at the top of its list.

B. *Analysing the filtered graph*

Running the clustering algorithm on the whole graph, then filtering out the highest and the lowest similarities and rerunning the clustering algorithm (as described above), will in most cases exclude the highest similarities which would already have been identified had a table format been used, and are therefore not new information. Typically these are similarities above 80%, but this figure is heavily dependent on the dataset.

The lowest similarities (as already suggested up to 10% similarity) are probably too low to be considered plagiarism and represent only common similarities. The interesting similarities are those in the middle range where new suggestions of plagiarised cases are most likely to be found. This is the biggest benefit of the proposed collaboration network analysis.

Once the new clusters are available every cluster that has few nodes (say, 2-5) is suspicious. Big clusters usually suggest that similarities are there because of the template. But the teacher knows best, and if there was a template given to students to be used in the assignment then more than one big cluster might be very interesting. It is important to note that some clustering algorithms create one cluster which will contain all nodes that are not related and some will create many clusters contain only one node – both are uninteresting.

One could say that visualization is not necessary for identifying the clusters – and this is undoubtedly true – but visualization gives the teacher a quick insight into the clusters and their sizes, and enables them to speed up the process of analysing the graphs. We already gave one example in the previous subsection regarding the decision as to which clusters to analyse. Another benefit is that one can quickly change the high and low cutoff points, rerun the algorithm, and quickly see changes in the clusters and if some new interesting cluster has emerged.

It is not possible to define more precise instructions (in terms of percentages) since every case and every dataset is different. The teacher must use their own judgement where they think (based on the percentages) the missed cases might hide. Usually after one year of using a plagiarism detection tool, and dealing with plagiarism, the teacher gets a feeling for where to look and the percentages for their own assignments, and the same is true for pairwise comparisons. For the teachers who are not used to plagiarism detection, a suggestion is to first get familiar with one plagiarism detection tool and its report, and only then follow the given instructions for visualization and clustering, and always remember that no tool can find plagiarism – a tool only finds similarities and suggests potential plagiarism.

It helps to know if someone has plagiarised before and which students are collaborating together in other courses or projects, since students often have high similarities with their friends or with the students that they work closest with (as shown by [12]), and such patterns of behaviour can be confirmed by your own data. Many students who were accused of plagiarism were room-mates or had bought the project from the same third party – but data protection rules mean that they can not be asked about their social status.

VI. Future work

Since network representation is not incorporated into all plagiarism detection tools, performing the analysis is potentially time consuming, so in the future it would be beneficial to incorporate the visualization software into plagiarism detection tools. There are some exceptions like Sherlock which has network presentations, but it is not always acceptable, as described for example in [11], and it does not include the multiple clustering algorithms and possibilities which Gephi provides. Of course, Gephi is not a tool specialized for plagiarism detection, so there is the problem of trying out various clustering algorithms and experimenting with their properties and other filters to get to best suitable ones for each particular scenario, as it can be seen from the cases presented. In the future a tool which combines Gephi (or similar software) with traditional plagiarism detection tools should be developed which then will have only the functionalities useful for plagiarism detection.

At the current time, it is good to use standard procedures (like side by side comparison) and incorporating collaboration networks analysis as an additional functionality, since collaboration network analysis created from similarity data can help finding new cases of plagiarism (as it was argued in this paper).

For future work it would also be useful to do a side by side comparison of a whole group based on the clusters. This would ease the process of side by side manual analysis following the visual analysis.

VII. Conclusion

The empirical results confirm that using visualization in the form of a collaboration network combined with clustering can improve plagiarism detection and help find new cases of plagiarism, and this agrees with the results from related work. But also, it has to be stated that collaboration networks will not always give new results, especially if there is none to be found.

This approach is particularly effective for cases where similarity between two pairs is average (to low) and there is substantial total similarity distributed amongst a group of students. In other words, collaboration network analysis can help identify groups of people who "cooperate" together. What can also be seen from the cases presented is that there are no specific parameters that give the best results. Everything depends on the data available, but by trying different parameters and configurations some useful results can be obtained. Also, with practice the teacher will find out which parameters work better for their datasets and it gets easier and faster to obtain useful results. One algorithm has been found to be very good and useful in all three cases, namely the Chinese Whispers clustering algorithm, and if one is a newcomer to visual analysis this algorithm is good to start with, even with the default parameters

As with other plagiarism detection methods a created network cannot be used directly to accuse somebody of plagiarism, but rather to indicate that plagiarism might be taking place and that there is evidence of similarities. The teacher still needs to check the indicated pairs manually. For future work it would be useful to do a further side by side comparison of a whole group based on the clusters, to confirm the results reported here.

References

[1] Verco KL, Wise MJ (1996) Plagiarism à la mode: A comparison of automated systems for detecting suspected plagiarism. Comput J 39:749–750

[2] Novak M, Joy M, Kermek D (2019) Source-code similarity detection and detection tools used in academia: a systematic review. ACM Trans Comput Educ 19:1–37 . https://doi.org/10.1145/3313290

[3] Mišić M, Siustran Z, Protić J (2016) A comparison of software tools for plagiarism detection in programming assignments. Int J Eng Educ 32:738–748

[4] Camarinha-Matos LM, Afsarmanesh H (2005) Collaborative networks: a new scientific discipline. J Intell Manuf 16:439–452

[5] Prechelt L, Malpohl G, Philippsen M (2002) Finding plagiarisms among a set of programs with JPlag. J Univers Comput Sci 8:1016–1038 . https://doi.org/10.3217/jucs-008-11-1016

[6] Gitchell D, Tran N (1999) Sim: A utility for detecting similarity in computer programs. In: The proceedings of the thirtieth SIGCSE technical symposium on Computer science education. ACM Press, New York, New York, USA, pp 266–270

[7] Joy M, Luck M (1999) Plagiarism in programming assignments. IEEE Trans Educ 42:129–133 . https://doi.org/10.1109/13.762946

[8] Makuc Ž (2013) Methods to Assist Plagiarism Detection. University of Ljubljana

[9] Mišić MJ, Protić J, Tomašević M V. (2018) Improving source code plagiarism detection: Lessons learned. In: 25th Telecommunication Forum. IEEE, Belgrade, Serbia, pp 1–8

[10] Schleimer S, Wilkerson DS, Aiken A (2003) Winnowing: local algorithms for document fingerprinting. In: Proceedings of the 2003 ACM SIGMOD international conference on on Management of data. ACM Press, New York, New York, USA, pp 76–85

[11] Bastian M, Heymann S, Jacomy M, others (2009) Gephi: an open source software for exploring and manipulating networks. ICWSM 8:361–362

[12] Luquini E, Omar N (2011) Programming plagiarism as a social phenomenon. In: IEEE Global Engineering Education Conference. IEEE, São Paulo, Brazil, pp 895–902

[13] Moussiades L, Vakali A (2005) PDetect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets. Comput J 48:651–661 . https://doi.org/10.1093/comjnl/bxh119

[14] Acampora G, Cosma G (2015) A Fuzzy-based approach to programming language independent source-code plagiarism detection. In: IEEE International Conference on Fuzzy Systems. IEEE, Istanbul, Turkey, pp 1–8

[15] Kermek D, Novak M (2016) Process Model Improvement for Source Code Plagiarism Detection in Student Programming Assignments. Informatics Educ 15:103–126 . https://doi.org/10.15388/infedu.2016.06

[16] Blondel VD, Guillaume J-L, Lambiotte R, Lefebvre E (2008) Fast unfolding of communities in large networks. J Stat Mech Theory Exp 2008:P10008 . https://doi.org/10.1088/1742-5468/2008/10/P10008

[17] Biemann C (2006) Chinese whispers: an efficient graph clustering algorithm and its application to natural language processing problems. In: Proceedings of TextGraphs: the First Workshop on Graph Based Methods for Natural Language Processing on the First Workshop on Graph Based Methods for Natural Language Processing. Association for Computational Linguistics, Morristown, NJ, USA, pp 73–80