Plagiarism Detection in Algorithms - a Case Study Using Algorithmi

António Manso
Techn&Art - Instituto Politécnico de Tomar, Portugal
manso@ipt.pt

Célio Gonçalo Marques
Techn&Art - Instituto Politécnico de Tomar, Portugal
celiomarques@ipt.pt

Paulo Santos
Techn&Art - Instituto Politécnico de Tomar, Portugal
psantos@ipt.pt

Vitor Alencar
Degree Programme in Computer Engineering
Instituto Politécnico de Tomar, Portugal
student 21737@ipt.pt

***Abstract*—Learning to program is crucial in computer science degree programmes. For students to gain this skill, they need to practise a lot, since programming is a difficult and complex process and practice improves it. The courses "Algorithmics" and "Introduction to programming" have therefore become almost insurmountable barriers with high failure rates. To overcome these barriers, researchers have sought to find new ways of teaching. For their part, students have looked for ways to succeed with less effort by repeatedly resorting to plagiarism. The need for classes to be taught remotely due to the pandemic has further aggravated this problem. To fight it, we created the Algorithmi tool and equipped it with a plagiarism detection module. This tool promotes stand-alone study by allowing self-correction of the exercises whilst validating authorship. This paper presents a case study on the application of the plagiarism module to the exercise resolutions submitted by the students in their personal repository. Through the analysis of the results obtained we can conclude that there was a high rate of plagiarism. The current version of the tool needs to be extended with new features, not only to become more accurate, but above all to prevent plagiarism from happening.**

Keywords: Algorithmi, Learning Systems, Programming Languages, Algorithms, tutor, assessment, plagiarism detection

I. INTRODUCTION

Programming is the art of making computers to solve problems automatically. For this to happen the programmer needs to have knowledge of problem-solving and computational logic and master the programming language. A task which is "*considered difficult, complex and categorised as part of the seven grand challenges in computing education*" [1]. We are therefore facing a challenge that is not met by most students in the introductory programming modules, leading to high failure rates [2]. A situation that affects student willingness to participate and makes class management increasingly difficult. Introductory programming modules already have many students due to the demand for professionals in these areas and school failure further aggravates the problem.

This scenario has led many researchers to look for new methodologies, strategies and tools to overcome the problem [3].

Simultaneously, students also try to find other ways to succeed in the module, including plagiarism.

Several studies indicate that academic misconduct, and in particular plagiarism, is particularly problematic in the courses of programming [4].

Plagiarism is defined as the action of copying someone else's work or ideas without giving them credit for it.

The existence of the Internet with original contents indexed, searchable and cloneable makes plagiarism accessible at the distance of a click. On the other hand, the existence of equipment and instantaneous communication applications such as smartphones facilitates the plagiarism of exercises with the transmission of answers to questions between students in an irregular way. In a regular classroom context, these factors can be eliminated or at least diminished through the presence and control of the lecturers. When their presence is not possible (as was the case during the confinement induced by the COVID-19 outbreak), these factors contribute to the increase of plagiarised works from the original sources and among the students.

Intensive practice is the best way for students to learn how to program correctly because it improves abstract thinking for complex problem solving. The need to perform many exercises in conjunction with overcrowded classes means that validation of the authorship of the exercises cannot be done manually by the lecturers in charge. This makes plagiarism the fastest and easiest way for students to succeed.

The pandemic outbreak caused by COVID-19 and the need for classes to be held remotely has further aggravated this situation, leading to the need for a tool that allows stand-alone coursework and detects plagiarism. It is in this scenario that the Algorithmi's plagiarism detection module appears.

This paper presents a case study of the use of the Algorithmi tool in students' answers to programming exercises in the course "Introduction to Programming" of the school year 2019/2020.

We start by addressing the issue of plagiarism, and then explain how Algorithmi detects plagiarism. The case study and the research conclusions are described below.

II. PLAGIARISM

According to [5], there are four categories of plagiarism: accidental, where there is no knowledge both of the attitudes characterised as plagiarism and the understanding of practices of citation and/or reference practised by the different educational institutions; unintentional, where the great amount of information available can, in a certain way, "*influence thoughts and the same ideas can arise through*

spoken or written expressions"; intentional, where the complete work or part of the work of another person is intentionally copied without mentioning the original author and; autoplagiarism, where the author uses his/her own complete work or part of it already published for other purposes without citing the original.

There are several implications regarding the practice of plagiarism, some of these implications in the academic field, according to [6] are: reputation of faculty members and students, legal issues, financial implications and plagiarised research. In other words, in general, the act of plagiarism is considered illegal and can result in severe punishment, such as the imposition of fines, restriction of professional practice in a certain position and even imprisonment.

The project Impact of Policies for Plagiarism in Higher Education Across Europe (IPPHEAE), which took place from October 2010 to September 2013, makes a comparison of policies for academic integrity in Higher Education in the European Union [7]. In this study, a "maturity level" was calculated for each country using an Academic Integrity Maturity Model (AIMM). Portugal was ranked 12.79/36, 18th out of the 27 countries involved in the study. In addition, it was found that: 1) Many of the faculty members and students who responded to the survey did not understand very well what is considered plagiarism; 2) few Portuguese Universities use software to verify the originality of the work submitted by students; and 3) 77% of the faculty members and 79% of the students interviewed showed interest in having more training and information to avoid plagiarism and academic fraud.

The scenario where students and faculty have no knowledge about conduct that may constitute plagiarism or fraud is worrying, because in addition to the legal implications, plagiarism can also affect student learning, considering that the simple fact of "copying and pasting" inhibits critical thinking and the development of logical reasoning. According to [8], in students' perception, the practice of plagiarism occurs because: they think that their practice of plagiarism will not be detected; it is very easy to copy and paste information from the Internet; ignorance about how to quote and provide references, the objective is to complete the coursework rather than focusing on learning; not being able to express other people's ideas and other coursework in their own words; not being able to deal with the workload; not being aware of the penalties related to plagiarism; not having control on the part of the lecturer regarding the detection of plagiarism; among other causes.

In addition, [9] identified possible negative emotions that can predict plagiarism conduct such as stress, depression or anxiety.

The most common methods of plagiarism according to [5], are: copy and paste a certain content word by word, plagiarise an idea, reaffirm the same content using synonyms or changing the order of sentences, artistic plagiarism, not citing or referencing the original work, not using quotation marks (“ ”) when the content used is exactly the same as the original content, use of incorrect or non-existent sources and references, translation of content without reference to the original work and, finally, plagiarism of source code, the main focus of this work.

A. Plagiarism Detection in Source Code

As already mentioned, plagiarism does not occur only in textual documents, or image files, but also in source codes. According to [10], beginner programmers often send plagiarised codes in practical tasks during the introductory programming modules.

The possible reasons for the practice of plagiarism as described by [11] and [12] are: 1) difficulty level of the exercise higher than the student's programming level; 2) interest of the student in minimising a large amount of the task by plagiarising the task of the colleagues; 3) insufficient time for the completion of the exercises; 4) fear of failure, 5) lack of interest on the part of the student regarding practical tasks; 6) lack of merit/recognition and 7) inadequate resources such as hardware, software, library, teaching staff, among others.

There are several approaches in the literature regarding the detection of plagiarism, varying in implementation style and source code. Above all, it is important to emphasise initially the difference between detecting plagiarism in text files and in files derived from codes and algorithms.

According to [13] the verification in text files consists essentially in looking for direct similarities, verifying the similarity "line by line" in relation to some other text.

Text file detectors are not effective for verifying similarities between source files because this approach cannot verify changes made to the copied code such as changing variables and possible changes to the structure of the document [13]. When checking for plagiarism in files defined as "source code", which are described by languages and code structures, they are notoriously different and need to undergo a modelling process in order to be able to compare them with other source files.

This modelling process usually follows some pre-defined steps that are established based on each objective and need. These factors may vary depending on the wide variety of plagiarism detectors available on the market. The most common practices for plagiarising source code files imply changing the code. Such practices range from changing the name of variables to inserting redundant code snippets. Figure 1 illustrates some of the most common techniques in source code plagiarism. In literature, these practices are characterised by "blinding methods", where the plagiarist aims to hide code snippets that are originally from another person.

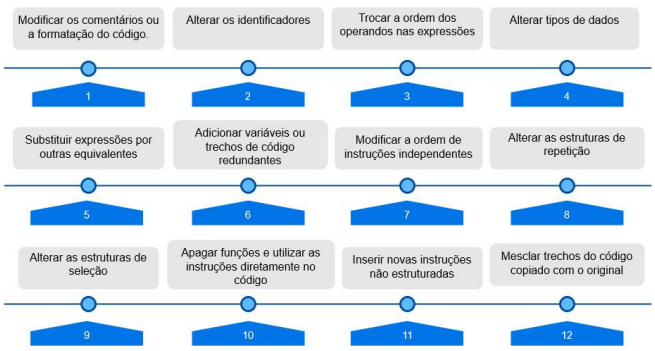


Figure 1 - Source code obfuscation techniques

Determining the level of similarity between two source codes is a difficult task and several automatic tools have been

developed to help lecturers in the arduous task of detecting cases of plagiarism. Examples of these software tools are MOSS (Measure of Software Similarity) [14] and Jplag [15]. These tools use advanced techniques to compare small fragments of source code at the syntactic level, calculating the fingerprint of a code segment, using the winnowing algorithm. These tools implement mechanisms that allow the detection of plagiarism using the obfuscation techniques presented in Figure 1.

III. PLAGIARISM DETECTION IN ALGORITHMI

Algorithmi is an information system to support learning programming consisting of a multi-platform application (IDE) designed to help students take their first steps in programming and a server which provides various services that can be accessed directly from a web browser, Figure 2.

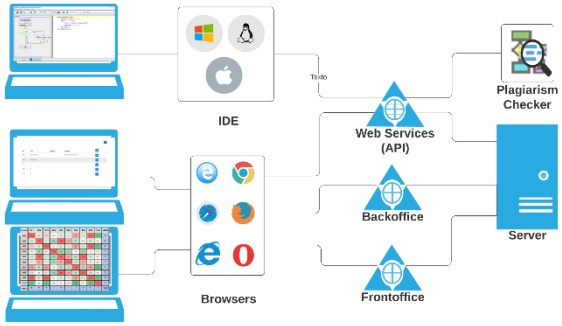


Figure 2 - Information system architecture of Algorithmi

The application is available for free at www.algorithmi.ipt.pt, and allows students to encode the algorithms using flowcharts or pseudocodes. The application internally uses a marker-based language, GAL (Generic Algorithm Language) [16], which is translated into the student's natural language for editing and viewing. This way the algorithm can be viewed and edited in different natural languages such as Portuguese, English or Chinese.

The GAL language supports the definition of simple and indexed variables of several types: integer, real, logical and text; it has I/O console instructions; it allows to control the execution flow through decisions and iterations and supports the definition of subalgorithms which can be recursive. This set of instructions allows coding complex algorithms through a set of tools designed for teaching and learning algorithms.

The algorithm can be edited in both languages and its visualisation can be displayed simultaneously as shown in Figure 3.

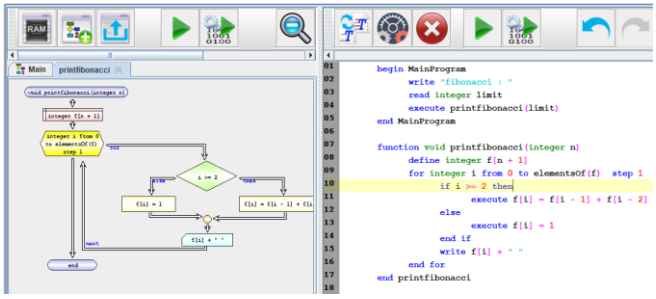


Figure 3 - Simultaneous editing and display of an algorithm.

The IDE not only shows and allows editing the flowchart and pseudocode, but also allows the display in several programming languages (Python, Java, C, C++, C#, PHP and JavaScript), Figure 4.

The server component is available through registration and authentication in the same domain. The server used was implemented with open-source technologies Apache, MySQL and PHP that allow the provision of various features to the information system. This server provides services that access a common database: Backoffice - a web application used by faculty, Frontoffice - a web application where students can find solved problems and their score, and an API - a set of REST web services ttha includes plagiarism checker.

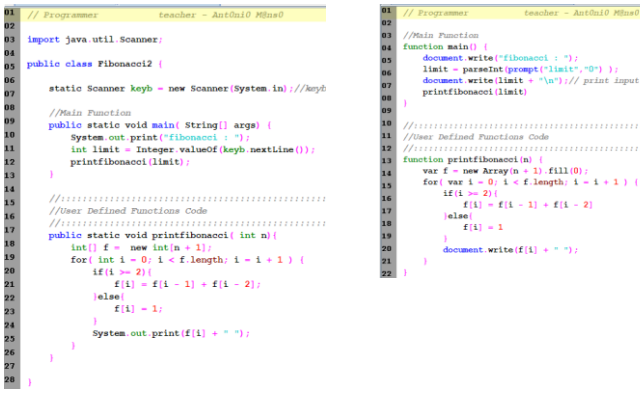
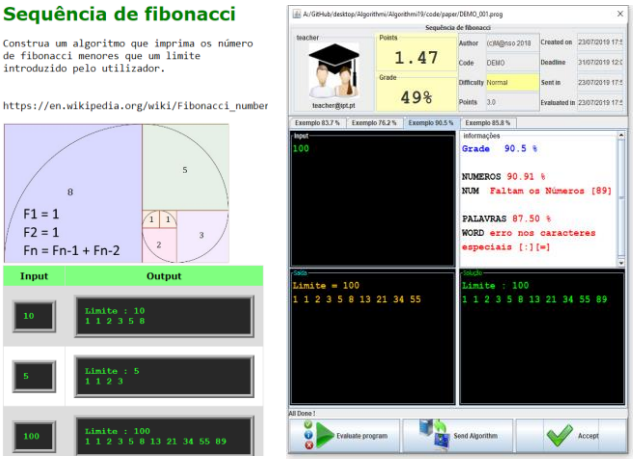


Figure 4 – Automatic translation of the algorithm to Java language and Javascript language

The backoffice component is used by faculty to create and make available programming exercises that can be accessed through the IDE, provided that the student is authenticated in the information system. The programming exercises can be corrected automatically and, in case of errors, the IDE provides a description of the errors made by students. This feature is particularly important in order to promote autonomous learning by students, as the IDE helps students verify that their algorithms are correct, and provides clues for their resolution.

After solving the exercises, the students submit them to the server that stores them in the student's personal repository. The student can visualise the resolution of their exercises in a web environment using the same tools as the IDE: flowchart, pseudocodes or programming languages; and can also execute them by translating them into Javascript embedded in a web page.



a) b)

Figure 5 - Programming exercise: a) Problem Statement; b) Automatic correction of the exercise by the IDE.

A. *Plagiarism control in Algorithmi*

Algorithmi encourages stand-alone programming through the exercise assessment tool and the tutorial system that identifies errors. Problem repository is one of the forms of assessment that can be used to assign a score to students, and the provision of a roll of honour listing the names of the students with the highest scores. Gamification is a key component in learning environments to maintain students' motivation to solve more complex exercises [16] [17].

The pressure to get a good score in the repository and the opportunity to belong to the roll of honour make students plagiarise some coursework. The IDE provides some tools that limit source code plagiarism. The exercises provided by the server to the IDE are marked with the username which prevents other students from taking them as their own. The exercise delivery and assessment module is only available for authenticated users and the programs of that user. The other tool that prevents plagiarism is the cut and paste tool. Pasting from software programmes within and outside Algorithmi are only allowed within software programmes of the same user. This way we prevent the source code from being shared between users.

With the help of these two features we can ensure that the algorithm was built by the authenticated user, but we can not ensure that it was not copied from another user.

In order to inform the lecturer of this type of plagiarism, we introduced the plagiarism detection module in the information system to compare the similarity between the resolutions of the students submitted in their repository. The module was developed using JPlag (sources available at https://github.com/jplag/jplag) which communicates with the information system through a web service. Due to the Algorithmi's ability to convert the algorithm into various programming languages, we use the Java language to compare source codes and detect plagiarism.

The service receives as passing parameter two programmes in Java and returns a number in the interval [0.100] that represents the plagiarism prevalence rate. The results are presented through the backoffice with the results of the plagiarism prevalence rate for each programming exercise and with aggregate information for each worksheet and for each student.

IV. CASE STUDY

This section analyses the application of the plagiarism module to the exercises submitted in the course "Introduction to Programming" of the CteSP programme in Information Systems Technologies and Programming offered at the Escola Superior de Tecnologia de Tomar - Instituto Politécnico de Tomar. The course covers the basic programming concepts: variables, operators, functions and control flow. One of the assessment components of the course is made based on the portfolio of programming exercises submitted by students. The exercises are divided into 15 worksheets with a total of 480 different exercises which cover the various subject contents. In each worksheet, the exercises have difficulty levels:

- *Demonstration* - Solved exercises and problems that are provided to students to demonstrate a new concept. These exercises are crucial for stand-alone learning and students can edit them in order to check the outcome of these changes.
- *Easy* - Exercises that are easy to solve and are usually variants of demonstration exercises.
- *Normal* - Medium-difficulty exercises
- *Difficult* - High-difficulty exercises where algorithmic techniques are explored, especially in the limit cases.
- *Challenging* - Challenging exercises whose resolution depends on the combination of several complex algorithmic techniques

The greater the difficulty of resolution, the more points the student earns, improving his ongoing assessment component and consequent rise in the roll of honour position. Plagiarism in demonstration exercises is almost 100%. The lecturer-in-charge provides the resolution, the students can then execute and submit it and consequently the resolutions.

Figure 6 gives the plagiarism prevalence rate of an easy exercise solved by 18 identified students from S01 to S018, and whose plagiarism indexes were provided by the plagiarism detection module. The numerical information is presented in tabular form with the plagiarism prevalence rate between each of the submissions for the same exercise, together with a colour that ranges between green and red for easier reading of the results. By the empirical analysis of the preliminary results we consider that two algorithms are plagiarised if their similarity is equal to or higher than 90%. This information is presented in the last row and in the last column of *Figure 6* and is useful to verify the degree of originality of the resolution of the exercise. In *Figure 6* we can see that there are 10 resolutions that the system identifies as plagiarised and one that differs from all the others. This plagiarism prevalence rate is natural, since it is an easy exercise based on a similar demonstration problem.

Alunos	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	>=90%
S01	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S02	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S03	76	76	100	76	100	76	69	76	100	76	76	98	100	98	76	76	100	76	7
S04	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S05	76	76	100	76	100	76	69	76	100	76	76	98	100	98	76	76	100	76	7
S06	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S07	72	72	69	72	69	72	100	72	69	72	72	68	69	68	72	72	69	72	1
S08	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S09	76	76	100	76	100	76	69	76	100	76	76	98	100	98	76	76	100	76	7
S10	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S11	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S12	74	74	98	74	98	74	68	74	98	74	74	100	98	100	74	74	98	74	7
S13	76	76	100	76	100	76	69	76	100	76	76	98	100	98	76	76	100	76	7
S14	74	74	98	74	98	74	68	74	98	74	74	100	98	100	74	74	98	74	7
S15	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S16	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
S17	76	76	100	76	100	76	69	76	100	76	76	98	100	98	76	76	100	76	7
S18	100	100	76	100	76	100	72	100	76	100	100	74	76	74	100	100	76	100	10
>=90%	10	10	7	10	7	10	1	10	7	10	10	7	7	7	10	10	7	10	8,3

Figure 6 – Plagiarism detection results in an easy exercise

Alunos	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	S11	S12	S13	S14	S15	S16	S17	S18	>=90%
S01	100	83	90	75	75	64	49	99	68	100	83	92	93	83	83	100	41	54	7
S02	83	100	63	69	69	54	47	52	41	83	100	91	72	100	100	83	59	59	5
S03	90	63	100	58	58	50	50	89	68	90	63	71	72	63	63	90	41	55	4
S04	75	69	58	100	100	64	38	74	50	75	69	76	73	69	69	75	41	42	2
S05	75	69	58	100	100	64	38	74	50	75	69	76	73	69	69	75	41	42	2
S06	64	78	50	64	64	100	69	74	46	64	78	78	72	78	78	64	57	49	1
S07	49	47	50	38	38	69	100	52	67	49	47	46	47	47	47	49	63	77	1
S08	99	52	89	74	74	74	52	100	69	99	52	89	92	52	52	99	44	52	5
S09	68	41	68	50	50	46	67	69	100	68	41	47	64	41	41	68	53	91	2
S10	100	83	90	75	75	64	49	99	68	100	83	92	93	83	83	100	41	54	7
S11	83	100	63	69	69	54	47	52	41	83	100	91	72	100	100	83	59	59	5
S12	92	91	71	76	76	78	46	89	47	92	91	100	81	91	91	92	61	66	8
S13	93	72	72	73	73	72	47	92	64	93	72	81	100	72	72	93	39	52	5
S14	83	100	63	69	69	54	47	52	41	83	100	91	72	100	100	83	59	59	5
S15	83	100	63	69	69	54	47	52	41	83	100	91	72	100	100	83	59	59	5
S16	100	83	90	75	75	64	49	99	68	100	83	92	93	83	83	100	41	54	7
S17	41	59	41	41	41	57	63	44	53	41	59	61	39	59	59	41	100	55	1
S18	54	59	55	42	42	49	77	52	91	54	59	66	52	59	59	54	55	100	2
>=90%	7	5	4	2	2	1	1	5	2	7	5	8	5	5	5	7	1	2	4,1

Figure 7 – Plagiarism detection results in a difficult exercise

Figure 7 gives plagiarism detection results in a difficult exercise. Due to the complexity of the algorithm that solves the exercise, the cases identified as plagiarism will be really plagiarism. In that figure we can identify a set of students who have a degree of similarity of 100% and a set of unique resolutions.

The previous figures present various information about the rate of plagiarism of worksheet exercises. The course has a strong component of stand-alone work and the great majority of the exercises are completed outside the contact hours, however the students continue to maintain contact with each other. Since Algorithmi does not allow you to make full copies of exercises, even if students exchange solutions with each other, they will have to edit the algorithms themselves. For these reasons, the plagiarism rates of the exercises in the worksheets are high.

Figure 8 gives the plagiarism prevalence rate in an assessed practical test. Assessed practical tests are done individually during the contact hours and in the presence of the lecturer. The class is divided in shifts to increase the physical distance between students and the time for solving the exercises is limited. From the results presented in *Figure 8* we can see that the plagiarism prevalence rate has decreased, however we can see that there are at least three resolutions that the system has identified as 100% plagiarism.

Alunos	S01	S02	S03	S04	S05	S06	S07	S08	S09	S10	>=90%
S01	100	51	70	60	51	52	84	51	100	41	2
S02	51	100	51	46	100	81	49	60	51	61	2
S03	70	51	100	60	51	52	84	51	100	41	2
S04	60	46	60	100	46	47	58	46	60	53	1
S05	51	100	51	46	100	81	49	100	51	61	3
S06	52	81	52	47	81	100	50	81	52	63	1
S07	84	49	84	58	49	50	100	49	84	39	1
S08	51	60	51	46	100	81	49	100	51	61	2
S09	100	51	100	60	51	52	84	51	100	41	3
S10	41	61	41	53	61	63	39	61	41	100	1
>=90%	2	2	2	1	3	1	1	2	3	1	1,8

Figure 8 – Plagiarism detection results in an assessed practical test.

Student portfolios consists of more than 7,000 exercises evaluated by the Algorithmi module to obtain a grade and by the plagiarism module to obtain the similarity index.
Figure 9 gives an overview of average plagiarism in each student's exercises and the percentage of exercises solved.
The overwhelming majority of students have solved more than 50% of the proposed exercises and more than half have solved more than 90%. This statistic shows the work and commitment of the students in solving the exercises and because "*you only learn to program by programming*" the result was reflected in the score of the repository.

Another important conclusion is that the plagiarism prevalence rate varies according to the number of exercises solved by the students. Students with fewer solved exercises tend to plagiarise more, due to their learning difficulties.

Since Algorithmi doesn't let you make direct copies and deliver them as your own, plagiarising takes work and students replace the work of thinking and creating with the work of copyists that is far less rewarding.

Average levels of plagiarism have always remained high, in the order of 80%, which suggests a high rate of plagiarism.

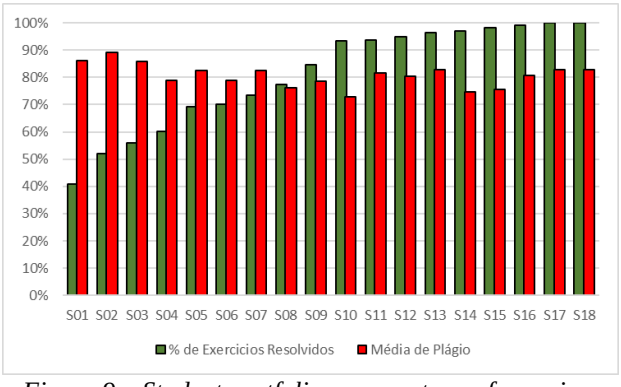


Figure 9 – Student portfolios: percentage of exercises solved and rate of plagiarism.

V. CONCLUSIONS

This paper presents the plagiarism detection module introduced in the information system of Algorithmi. This work was carried out with the support of a student of the Computer Engineering degree as part of his final project and

was designed to increase the effectiveness of this system for teaching/learning computer programming.

This module was applied to the course which was taught in the first semester of the school year 2019/2020 to check the rates of plagiarism in the students repository.

It was found that the rates of plagiarism are high, especially in the worksheets completed outside contact hours. In assessed practical exercises, whose objective is to assess knowledge, the plagiarism rates are significantly lower. This is because the exercises are more difficult, the students have a limited time for their resolution and were solved during contact hours. Nevertheless, the system detected several cases of plagiarism.

With the COVID-19 forced confinement of students and staff, in which the majority of teaching was delivered remotely, the module now developed has assumed greater significance. The work developed corresponded to expectations and above all opened new horizons that will be explored in future versions.

In the future, we intend to apply the online module where the system registers and informs the student about the originality of the submitted code. In the offline version, whose results we present here, we only know which exercises are plagiarised, but we do not know the original source. With the online version, the first exercise to be submitted has a maximum degree of originality while copies submitted later will see their originality diminished. This way we intend to stimulate the development of different algorithms for the same problem, increasing the gamification already present in Algorithmi.

The fact that students realise that the system has a system that detects plagiarism may serve as a deterrent for them to avoid less ethical behaviour and channel their energy to the development of algorithmic reasoning, because in Algorithmi plagiarism also takes work.

REFERENCES

[1] Derus, S., & Ali, A. Z. M. (2012). Difficulties in learning programming: Views of Students, In 1st International Conference on Current Issues in Education (ICCIE 2012).

[2] Butler, M. & Morgan, M, (2007). “Learning challenges faced by novice programming students studying high level and low feedback concepts”, ASCILATE 2007 Singapore, pp. 99-107.

[3] Tan, P.-H., Ting, C-Y, Ling, S-W (2009). Learning Difficulties in Programming Courses: Undergraduates' Perspective and Perception, In Computer Technology and Development (ICCTD '09).

[4] Jian, H. L., Sandnes, F. E., Huang, Y. P., Cai, L., & Law, K. M. Y. (2008). On students' strategy-preferences for managing difficult course work. IEEE Transactions on Education, 51(2), 157-165.

[5] H. Maurer, F. Kappe e B. Zaka, Plagiarism - A Survey, Journal of Universal Computer Science, 2006, 12(8), pp. 1050-1084. Available: http://jucs.org/jucs_12_8/plagiarism_a_survey/jucs_12_08_1050_1084_maurer.pdf

[6] O. P. Rajasree, A. H. Mangala e U. T. Sunil Kumar, An Overview of Plagiarism and Intellectual Property Issues, National Conference on Academic Libraries in E-learning Environment: Role and Prospect, 2020, ISSN: 0474-9030, 68.

[7] I. Glendinning, T. Foltynek, C. Demoliou, K. Joswik, L. Stabingis e A. Stulginkis, Comparison of policies for Academic Integrity in Higher Education across the European Union, 2013.

[8] I. Glendinning, K. Joswik e A. Michałowska-Dutkiewicz, Plagiarism Policies in Portugal, full report , 2014.

[9] I. K. Tindall e G. J, Curtis, Negative Emotionality Predicts Attitudes Toward Plagiarism, Journal of Academic Ethics, 2020, 18, pp. 89-102.

[10] Vogts, D. Plagiarising of source code by novice programmers a “cry for help”? 2009 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists, 141-149.

[11] Vamplew, P., & Dermoudy, J. An anti-plagiarism editor for software development curses. 7th Australasian Computing Education Conference. Retrieved March 21, 2007, from http:// crpit.com/confpapers/CRPITV42Vamplew.pdf

[12] A. G. Liaqat & A. Ahmad. Plagiarism Detection in Java Code. Linnaeus University. School of Computer Science, Physics and Mathematics. Retrieved June 26, 2011.

[13] Sheard, Judy & Dick, Martin & Markham, Selby & MacDonald, Ian & Walsh, Meaghan. (2002). Cheating and plagiarism: Perceptions and practices of first year IT students. ACM Sigcse Bulletin. 34. 183-187. 10.1145/637610.544468.

[14] Schleimer ,S.; Wilkerson,S., Aiken, A. - Winnowing: local algorithms for document Fingerprinting. In Proceedings of the 2003 ACM SIGMOD international conference on Management of data. ACM, 76–85, 2003

[15] Prechelt, L, Malpohl, G, Philippsen , M. Finding plagiarisms among a set of programs with JPlag. J. UCS 8, 11 (2002), 1016, . 2002

[16] Manso, A., Marques, C.G., Santos, P., Lopes, L. & Guedes, R. (2019). Algorithmi: Bridging the Algorithms to Natural and Programming Languages., in 15th China-Europe International Symposium on Software Engineering Education. Lisbon, Portugal, May 30-31, 2019

[16] Manso, A., Marques, C.G., Santos, P. (2019). Algorithmi IDE – Integrated learning environment for the teaching and learning algorithms, in 21º International Symposium on Computers in Education (SIIE). Tomar, Portugal, 2019

[17] Manso, A., Marques, C.G., Santos, P. (2019). Teaching and Learning How to Program with Algorithmi., in 21º International Symposium on Computers in Education (SIIE). Tomar, Portugal, 2019