NOVEL DISTANCE-LEARNING METHODS TO OVERCOME CHALLENGES CAUSED BY COVID-19 IN UNDERGRADUATE PROGRAMMING COURSES

Seyed Parsa Neshaei, Amirmahdi Namjoo, Parham Chavoshian, Parham Saremi, Mohammad Taha Jahani-Nezhad, AmirMahdi Kousheshi, MohammadAmin Fazli

Sharif University of Technology (IRAN)

Abstract

With the rise of COVID-19 cases, all Sharif University of Technology undergraduate courses shifted toward a completely online paradigm, which caused new challenges in the education and assessment of students. Two of the most important courses for Computer Engineering students with lots of challenges due to the pandemic were Fundamentals of Programming and Advanced Object-Oriented Programming. Various distance-learning techniques were used to enhance the quality of education to overcome these difficulties. These techniques include restructuring the teaching assistant selection process, automating the evaluation and grading of homework and projects, holding interactive online workshop sessions, assigning guides to each student group, and using modern code plagiarism detection systems. Some of them existed before the COVID-19 era, while others are new. This article presents an in-depth analysis of each technique and its effect on improving students' ability to efficiently design and develop programming projects. We also cover special considerations needed to enforce these methods competently. We will back the methods' effectiveness with quantitative data, including a statistical comparison between groups that implemented the mentioned tactics and those that did not.

Keywords: computer science education, e-learning, distance learning, covid-19, programming courses, plagiarism detection.

1 INTRODUCTION

The increase in daily infections during the COVID-19 pandemic has forced governments and institutions in many countries to practice social-distancing techniques [1]. Restrictions have been imposed on gatherings to reduce virus transmission. Traditionally, college classrooms have been an area for virus infection due to the numerous students sitting close together while attending lectures or participating in tutorials and workshops [2], so schools and universities were among the places closed after the pandemic started to exist [3].

Due to the newer variants of the virus, classes and lectures, including Fundamentals of Programming and Advanced Programming, continued to be held online for several semesters at Sharif University of Technology. Since the classes shifted to an online paradigm, instructors have tried to use various educational models and techniques to improve their courses [4]. The techniques were different in each case, based on the type of courses.

Fundamentals of Programming (FOP, also called Basic Programming / BP) is a course taught in the Computer Engineering department, aiming to familiarize students with basic programming concepts. Computer Science (CS) and Computer Engineering (CE) undergraduates should take FOP the first semester after they enter the university, as the course's material is deemed to be essential to understanding upcoming compulsory CS courses, such as Data Structures and Algorithms (DSA). The course's syllabus covers a range of CS-related subjects, starting with computer arithmetic and introduction of algorithms, continuing into an introduction to C and its syntax, and ending in an introduction to intermediate C topics such as structures, pointers, and networking. The lectures are accompanied by homework, tutorial workshops, and a course project. Due to the importance of being knowledgeable of FOP's material in the rest of the CE undergraduate program, the course needs to fulfill the criteria of teaching junior students successfully.

Advanced Programming (AP, also called Object-Oriented Programming or OOP) is another course from the same CE department. According to its syllabus, it introduces object-oriented programming and design concepts using the Java language. The course's material starts with an introduction to Java and Java's main data structures and containers; it then continues with object-oriented design principles, both in theory and in code, which takes a sizable portion of the class sessions. After OOP, the course introduces more advanced topics to the first-year students, such as concurrency, software testing, and designing GUIs and peer-to-peer networks. Advanced Programming, while not as fundamentally important as FOP, also contains necessary skills needed in most of the practical and project-based CS and CE courses, such as System Analysis and Design, and also contributes a lot to the introduction of tools used in the industry and technology companies to the students. Both AP and FOP are highly-demanded courses with many students registering for them (around 100 to 200), so selecting teaching assistants and managing them is very important.

Due to the nature of FOP and AP, these courses need a highly-interactive classroom environment to maximize students' learning. Besides, traditionally, TA one-to-one or one-to-group sessions for delivering homework handouts and course projects have contributed a lot to the absorption of these courses. Holding classes online eliminates many of the necessary features that made FOP and AP successful courses in educating students in the first place. This removal needs to be taken care of to minimize its impact [5].

This paper focuses on the methods that were used to effectively utilize the online educational environment and overcome the challenges imposed by the COVID-19 pandemic. In the subsequent sections, we discuss the changes made to the traditional way the two aforementioned courses were held.

While not all changes were done for the first time - and many had been experimentally tried even before the pandemic - these practices have led to students' better understanding of course topics. The point of this paper is to back the claims by performing a statistical comparison between groups that used this methods and those that did not.

2 METHODOLOGY

2.1 Workshops

Traditionally, in addition to the course lectures provided by the instructor and focusing more on the theoretical aspects of programming, such as learning computer arithmetic and fundamentals and design patterns, workshops have also been held on various topics all regarding the courses' syllabi. Attending lectures is necessary for students to be able to absorb the main concepts and succeed in the exams [6], but solving homework and doing the course project needs a more practical introduction to code structures and programming tactics and methods, which workshops provide as a form of "active learning" [7]. To make online workshops a better experience, we have enforced some practices when running workshops.

First, all workshop instructors enter the online room with at least two devices, which they code, write or present necessary slides - which are kept to a minimum - in the first display and open the online classroom link as the student sees in the second display. This enables the instructor to see the questions students ask without a noticeable banner on their main screen over the code and allows the instructor and the students to focus better on the workshop's contents. Frequent disconnection of the Internet due to the server or instructor's client connection also happens to be another significant problem. The problem is solved when the instructor connects to two different Internet connections, preferably one using cellular data and one using Wi-Fi or Ethernet. The instructor will observe a disconnection, or a notice of connection problems, from the notifications and messages the online platform or students send if the screen share video stream freezes.

Second, to further mitigate the inability of social interaction with workshop mentors when the workshop is not held in person, the idea of a "parallel mentor" has been practiced in some workshops, in which while the instructor is presenting the material, some of the teaching assistants called "parallel mentors" are online in voice channels in some other platform (different from and parallel to the platform the main workshop is being held), so the students can ask any question they have from a TA in an available voice channel, while the instructor keeps going through the material. This technique ensures the instructor is not paused frequently, as happens in many live workshops. The student can also share their screen, and the TA could assist them in debugging while no interruption happens to the primary instructor of the workshop.

2.2 Cheating and Academic Integrity

Cheating in exams and homework have always been a challenge for universities. Some research shows that academic cheating is prevalent [8]. There is also evidence of online cheating after the emergence of the COVID-19 pandemic [9]. As it was predictable that some students may resort to cheating, especially in virtual courses, we used multiple solutions to discourage and detect academic dishonesty.

To handle this problem efficiently, we first analyzed the main cheating methods. As most of the assignments and exams in FOP and AP are programming problems, there could be three main methods for cheating:

1 Getting the answer from other students and copying them.

2 Copying code from online resources.

3 Outsourcing the problems to third parties that solve problems in exchange for money.

To discourage academic dishonesty, we increased the penalty for cheating. Before the transition to online, cheating in homework resulted in a zero grade for that particular assignment. We increased the penalty for cheating to a negative mark for that whole section of the course, and also, cheating for the second time resulted in a fail grade for the student. Also, we noted that despite the in-person cases in which there could be possible situations where the cheatee is not aware of the cheating process, the cheatee is almost always aware in the online courses, so the penalty for cheating is applied to both the cheater and the cheatee.

Because the process of using online resources (such as Stack Overflow or online documentation) is an integral part of programming, we allowed the students to use online resources. Still, they were strictly prohibited from copying any code directly from the Internet. Also, they were expected to cite every online resource they used as a comment in the first lines of their submitted codes.

To detect cheating, we leveraged two different methods. The first one was using software to check for code similarity. Quera, the platform we used for grading the submitted codes, has a code similarity feature that internally uses Stanford's MOSS. MOSS is a widely used tool to detect plagiarism in programming classes. This tool compares students' answers pairwise and outputs the similarity percentage and mark-up of similar parts in each code. It is resistant to most bypassing techniques that could fool human graders. After getting the percentages, we manually checked similarity percentages above 50% and determined the probable candidates for cheating based on the problem and the answer.

The other side of the cheating was outsourcing the code. To detect these types of cheats, we scheduled sessions with each student after each assignment in which we asked a question about different parts of the code to verify the code is their work. In these sessions, we also talked with students that were suspicious of cheating based on the code similarity results to make the false-positive ratio of our process near zero.

We also found that the way we handle exceptional cases could increase or decrease the number of unethical actions done by the students. In courses where we publicly announced the accommodations for students who get COVID-19 or teams that disassemble, we saw an increase in students who tried to claim that they were eligible for these accommodations, which led to difficulties for the TA team to distinguish fake and confirmed cases. On the other hand, in courses that we did not publicly announce the accommodations, only students that got into serious problems approached us for help.

It is also noteworthy that we did not use aggressive techniques like an always-on webcam for exams. Although it could potentially decrease the cheating ratio, since there are problems regarding Internet connectivity for students in less-developed cities, this technique could lead to tremendous stress. It is shown that stress is a factor that can decrease students' performance [10]. Also, Internet connectivity problems could sometimes be interpreted as cheating when applying these methods, increasing the false-positive ratio by a significant and unacceptable amount.

2.3 TA Selection Process

Many students apply for being a TA in FOP and AP every semester. So, one of the challenges in these courses is choosing the TAs. A method is needed to examine the applicants' technical knowledge and responsibility. The following method was used in two semesters of FOP, an evolved version of the practice also used in the previous years of TA selection for FOP at Sharif University of Technology.

The applicants were given three challenges and were asked to participate in at least one of them. These three challenges consisted of designing some programming questions with specified topics and appropriate difficulty, choosing an instructive subject for the project and writing a short document about it, and producing educational content for a specific topic.

After the deadline for challenge submission passes, each head TA investigates the submitted answers and grades them. The grades are then sorted in a decreasing manner. After the grading process is done, all head TAs participate in a meeting and choose the final TAs considering their achieved grades, resume, motivation, and free time.

Using this method for choosing TAs will make the selected TAs more responsible, knowledgeable, and motivated. This leads to designing better content and improving the quality of representation of the courses.

2.4 Project Guides

The course project is an essential part of both FOP and AP courses. It is also more important in the AP course than FOP because the AP project at Sharif University of Technology is done in groups of three. Moreover, it is the first profound encounter of students with teamwork in the university.

To help students do their projects as best as possible and improve the overall learning process, a TA is assigned to each student group to act as their guide. The guides have to help their groups in both hard and soft skill aspects. In this case, hard-skill guidance includes showing the programming best practices to implement each part of the project in the best way and introducing different programming tools to help the group satisfy the different needs of the project. On the soft side of the matter, the TA should somehow play the role of a scrum-master, helping the team resolve misunderstandings and increase productivity.

The guides also monitor the overall progress of each team. Suppose they find out that a team was inactive for an extended period or has severe problems with the project. In that case, they could speak with them and inform the central teaching assistant team and the course lecturer of their circumstances to find the best way to help them overcome the difficulty. Also, as the team members could have been in different cities, the guide introduces different remote pair programming and meeting tools to their assigned teams to increase remote work productivity.

The use of guides proved to be very effective. According to the surveys given to students at the end of the semester, it improved the quality of projects and students' overall satisfaction. It also significantly decreased internal disputes between team members.

2.5 Code Judgement

Due to the large population of students, grading assignments was a challenge. If this process had been done manually, it would have taken a lot of the teaching assistants' time, and the grades would not be accurate enough. Also, in that case, the students could not be notified of the correctness of their program prior to correction. Because of these problems, we needed an alternative way of grading the assignments.

The best way to grade the assignments during the online education era was through online judging, traditionally used by many course instructors and programming contests. Every student could be notified of their grade when they uploaded their codes in this method.

We used two types of online judging. One of them was "program judge." This type of auto-grader only takes one code file as input. After that, it gives the input test case to the program and writes its output in another file. Then, if the input has only one correct answer, the output is compared with the correct answer; otherwise, the output is given to a checker code, and the code validates the output.

The other type of online judging is "project judge." This type of auto-grader takes a compressed file as input. The uploaded compressed file must contain all the code files specified in the entry point file name. The project judge runs a bash script at first. The bash script extracts the uploaded file and compiles the program. After compiling is done, the script runs the complied files, gives the input tests to the program, and writes its output in another file. Then, the bash script runs a checker code, which validates the outputs using previously-written unit tests. Finally, the judge announces the grade considering the results of the unit tests.

For both judging types, we used a platform named Quera. In this platform, we can judge programs that have been written in various languages like C, C++, and Java. Also, we can set time limits and memory limits for running programs to check the optimization of the algorithms used in solving the problem. When online judging is used, grading each program takes only a few moments. In addition to that, every student can upload their program several times and check their grade, and if the result is not equal to the maximum possible score, they can try again and correct their code.

3 RESULTS

An experiment was performed to examine and find out if the claims are correct and if the proposed measures have a meaningful impact on the computer science education of the students.

The experiment consisted of two groups, A and B. Students in group A were those who were assigned by the university to a class teaching FOP as before and without the proposed methods, but students in group B learned the course material using the methods mentioned in the paper. Both groups entered the Computer Engineering department of the Sharif University of Technology at the same time. They took FOP in the first semester after being accepted to the university. Due to the details of the selection process for attending SUT, the two groups had an almost similar background and students were randomly assigned to the two groups by the university.

Then, in the next semester, both groups took the same AP course, and their performance in AP was compared. It goes without saying that being a successful student in AP requires comprehensive knowledge of FOP and proficiency in basic programming since FOP is a vital prerequisite for AP. The performance of the two groups in the AP course is summarized in Table 1.

Table 1. Comparison of groups A and B.

	Number of students in the group	*Mean of Grade (out of 20)*	*Standard Deviation of Grade*	*Number of failed students*	*Number of late-dropped students*
Group A	88	17.37	3.54	5	9
Group B	94	18.35	2.85	4	2

The total number of students does not differ significantly, so the comparison makes sense. All scores are calculated from 20, the maximum grade one may obtain at Sharif University of Technology.

As can be seen in Table 1, the average grade of group B students is almost one point more than that of group A. As a clarification, a difference of one point in grades is considered high in the Sharif University of Technology since grades are calculated from 20, not in letters or percentages as in many other institutes. Also, since Sharif University of Technology is considered the top-ranked university in the region, there is considerable competition among students, leading to the importance of every single point in the grades.

The number of students who failed the course was almost similar in the two groups since a failure in FOP and AP is rare in the Computer Engineering Department of Sharif University of Technology. Nevertheless, the number of students who late-dropped the course in group A was almost four times more than that in group B. To provide a context in case the reader is not familiar with Sharif University of Technology's education office rules, late-drop (also known as emergency withdrawal) is a privilege granted to any student once a semester in a specific timespan before the start of the finals. In the late-drop timespan, any student who believes they cannot pass a course with a good grade can late-drop that course. As a result, the number of students who late-drop a course resembles the number of students who struggled the most with the course's material. Table 1 indicates that group A students had more problems studying AP in general, leading to a higher count of late-drops.

To find out if the difference in the average grade of the two groups is meaningful or not, we have performed a t-test. Here, the null hypothesis indicates that the performance of the two groups in AP in terms of grade was similar. In contrast, the alternative hypothesis states that group B performed better in AP than group A. The code we used to conclude the results, calculated a p-value of around 0.0208, less than 0.05, and signals a significant difference. The result indicates that group B, who practiced the proposed methods in FOP, passed the AP course with meaningfully higher grades and better results. The raw data of the grades is available at https://github.com/spneshaei/fop-statistical-comparison without the name or ID of the students to preserve privacy.

4 CONCLUSIONS AND FUTURE WORK

In this paper, we examined the provided methods in the context of instructing Fundamentals of Programming and Advanced Programming courses to first-year computer engineering students at Sharif University of Technology. Our results show that the changes in how homework answers are graded, how workshops are held, and how teaching assistants are selected have mitigated the inevitable consequences of teaching Fundamentals of Programming and Advanced Programming, during a hard time of the pandemic, up to a reasonable extent.

We plan to broaden the methods and make more statistical comparisons in future work. Especially in addition to online classes, massive online open courses (MOOCs), provided either by the university or by independent instructors, have become popular among the students since the pandemic started [11]. Our focus for future research is to analyze the student's performance in attending MOOCs and including online practice tutorials, with and without using the methods described in the workshop and TA selection section. Research suggests that online and flipped courses can predict student success early on [12], which helps us recommend attending practical online sessions, especially for the group of students at risk.

ACKNOWLEDGEMENTS

We sincerely thank Mohammad Hossein Gheisarieh and Arshia Akhavan who made a valuable amount of contribution in running the aforementioned methods. We also thank more than a hundred teaching assistants throughout the semesters, and heads of individual sections of the course, including but not limited to Alireza Tajmirriahi, Alireza Shateri, Sayeh Jarollahi, Soroush Jahanzad, Mohammad Hossein Bahmani, Javad Hezareh, Mohammad Mahdi Barghi, Sina Elahimanesh, Nima Fathi, Alireza Babazadeh, Seyed Alireza Hashemi, Milad Saadat, Hadi Esnaashari, Fateme Khashei Varnamkhasti, Sara Khosravi, Mohammadamin Aryan, Mehrdad Mirzaei, Saber Zafarpour, Reza Amini, Ali Hatami Tajik, Ehsan Movaffagh, Zahra Sadat Bahri, Alireza Eiji, Siavash Rahimi Shateranloo, Arman Babaei, and Ali Shirmohammadi.

REFERENCES

[1] P. Sahu, “Closure of Universities Due to Coronavirus Disease 2019 (COVID-19): Impact on Education and Mental Health of Students and Academic Staff,” *Cureus. 12. 10.7759/cureus.7541.*

[2] P. Tupper, C. Colijn., “COVID-19 in schools: Mitigating classroom clusters in the context of variable transmission,” *Plos Computational Biology*, 2021.

[3] “The impact of coronavirus on higher education,” Retrieved from https://www.timeshighereducation.com/hub/keystone-academic-solutions/p/impact-coronavirus-higher-education

[4] Y. Denisova, V. Sopin., “The enhancement of student satisfaction in remote learning organization quality,” *EDULEARN21 Proceedings*, pp. 118-124, 2021.

[5] I. C. Gil Garcia, M. S. Garsia Cascales, A. Molina Garcia., “Master teaching in the COVID 19 era: interactive activities vs traditional activities in virtual environments,” *EDULEARN21 Proceedings*, pp. 125-131, 2021.

[6] W. Chin Hsu, S. W. Plunkett., “Attendance and grades in learning programming classes,” *ACSW ‘16*, no. 4, pp. 1-6, 2016.

[7] G. Buskes, B. O. Shen, J. S. Evans, A. Ooi., “Using active teaching workshops to enhance the lecture experience,” 2009, Retrieved from https://www.researchgate.net/publication/228573341_Using_active_teaching_workshops_to_enhance_the_lecture_experience

[8] D. L. McCabe., “Cheating in Academic Institutions: A Decade of Research, Ethics & Behavior,” pp. 219-232, DOI: 10.1207/S15327019EB1103_2.

[9] E. Bilen, A. Matros., “Online cheating amid COVID-19,” *Journal of Economic Behavior & Organization*, vol. 182, pp. 196-211, 2021.

[10] M. C. Pascoe, S. E. Hetrick, A. G. Parker., “The impact of stress on students in secondary school and higher education,” *International Journal of Adolescence and Youth*, pp. 104-112, 2020.

[11] C. Impey, M. Formanek., "MOOCS and 100 Days of COVID: Enrollment surges in massive open online astronomy classes during the coronavirus pandemic," *Social Sciences & Humanities Open*, vol. 4, no. 1.

[12] M. Marras, T. Tu, J. Vignoud, T. Käser., "Can Feature Predictive Power Generalize? Benchmarking Early Predictors of Student Success across Flipped and Online Courses," *14th International Conference on Educational Data Mining (EDM 2021)*, Paris, pp. 150-160, 2021.