Scheme for Cheating Prevention in Online Exams during Social Distancing

Navid Ibtehaj Nizam, Shan Gao, Mengzhou Li, Hisham Mohamed, Ge Wang*
Department of Biomedical Engineering, Rensselaer Polytechnic Institute, Troy, NY, USA

*Email: wangg6@rpi.edu

April 8, 2020

Abstract: In the current pandemic of COVID-19, students and faculty are subject to social distancing and online learning. How to test students in this unprecedented environment is a new educational challenge with immediate and global impacts. The main contribution of this paper is to establish the feasibility that by a clever design we can control the average gain (which is referred to as the g-factor) from cheating behaviors to a degree as small as pre-specified so that accurate and reliable online exams can be administered. It is underlined that even after the pandemic the methods and systems in the spirit of our proposal are still valuable for cost-effective exams to promote open courses and internet-based education.

Key Words: Online exam, social distancing, statistical analysis, game theory.

I. Introduction

In fundamental ways, measurement is necessary and instrumental in science, technology/engineering, education, and mathematics/statistics/data science (STEM). Measurement in education is commonly known as examination/testing, which can be open or close book, online or paper-based, subjective or objective (multiple choices). The ideal exam should sense a student's capabilities of capturing knowledge units, solving problems and thinking creatively, be cost-effective and widely accessible.

Along the above guideline, several comments can be made in order. Thanks to Google and other information retrieval engines, knowledge acquisition is never as easy as now. Hence, remembering facts and formulas becomes less important than understanding the subject matter thoroughly and mastering skills of finding and integrating needed pieces of information. In other words, open book tests seem in many cases more relevant than close book tests. Since individualized problem-solving problems lead to inhomogeneous answers which are generally time-consuming to be evaluated objectively and efficiently, multi-choice problems become increasingly popular.

During a traditional exam in a school or college, a proctor sits in a classroom to prevent students from cheating. In a conventional online exam such as TOEFL and GRE, students can be also put in the same room under human monitoring. Arguably, it could be easier to cheat in online exams than to do so in paper-based tests. In all these cases, students are in the same room under centralized monitoring. However, in the pandemic of COVID-19 students are subject to social distancing and remote learning. How to test students in this environment is a new educational challenge with immediate and global impacts. As an example, College Board "*canceled the May 2, 2020, SAT and SAT Subject Test administration. Makeup exams for the March 14 administration (scheduled for March 28) were also canceled*" [1], since cheating would invalidate the test results during social distancing.

In practice, several well-known means to suppress cheating behaviors are in use for management of online exams. First, testing software exists to prohibit students from copying/pasting. Also, the order of questions in an exam can be randomized so that neighboring students cannot peer each other's' results. Since students take different time lengths to solve the same problems, each of the questions can be timed to separate prepared and unprepared students. In addition to the above means, software can be developed to enable remote proctoring. For example, Proctortrack^{TM} [2] verifies the identity of test takers and detect when he/she leaves a test, if he/she does anything improper online such as searching for information. However, it relies on a webcam-based tracking which can be easily fooled. A user can easily cover the microphone of his/her computer and have the answers relayed to them from another device. Additionally, such software is expensive, and the cost is further enhanced by the requirement of additional equipment, such as webcams, speakers, etc. Thus, all these tricks, tips, programs and devices are insufficient or unsatisfactory to suppress cheating in online exams during social distancing.

With the modern Internet and mobile technology, every student is well equipped at home. Hence, an exam during social distancing must be open book so online searches are allowed. Then, the challenge is how to prevent students from real-time communication during an exam, such as on a secondary device to do multi-choice problems with friends through Zoom/WebEx/Skype/FaceTime or even simple audio communication, which can be set up with little effort. Therefore, it is difficult to eliminate such cheating completely during social distancing. In a typical scenario, a perfect student A and an unprepared student B are friends and agree to cheat in an online exam during social distancing. Student A knows all answers perfectly, while independently Student B can only do random guessing. In the online exam, Student A can effortlessly give his/her answers to B, and B can achieve the perfect score as A, instead of B's guessing-based score. If all questions are 4-choices with one and only one valid answer, the perfect student achieves 100%, while the statistical mean of Student B's performance is 25%. By cheating, B achieves 100% instead of 25%, and his/her average gain (which is formally defined as the g-factor below) is 75% via cheating!

Here we propose a scheme for cheating prevention in online exams during social distancing. With our scheme, even the two students can collaborate optimally, the unprepared student cannot achieve significantly better score than his/her random chance (which is 25% in the above example). Furthermore, with a more aggressive design potential cheating behaviors can be detected and penalized. The main contribution of this paper is to establish the feasibility that by a clever design we can control the average gain (the g-factor) from cheating behaviors to a degree as small as pre-specified in an online exam during social distancing. In the next section, we describe the general methodology. In the third section, we present representative results. In the last section, we discuss relevant issues and conclude the paper.

II. Key Concepts: ASC, ASF and g-Factor

In an online exam, *M1* multiple-choice questions (MCQs) from a pool of *M2* MCQs are provided to a class of *N* students, and there are *Q* choices per question with one and only one of them being correct. All the questions carry the same credit and take the same amount of time to answer. Without loss of generality, let us assume M1=M2=N for an initial analysis to illustrate the general idea. In this situation, we can present the questions to the students one by one in a way such that no two students receive the same question with each allowed time slot/window for his/her current question. For example, each student is provided *t* minutes before the question is replaced by another question from the set of the *M2* questions. To ensure the previous restriction (of no two students receiving the same question), we do a simple left circular shift (alternatively, we may do a right circular shift) in the manner shown in Figure 1. Without loss of generality, six students in different colors are examined with six MCQs.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 April 2020 doi:10.20944/preprints202004.0327.v1

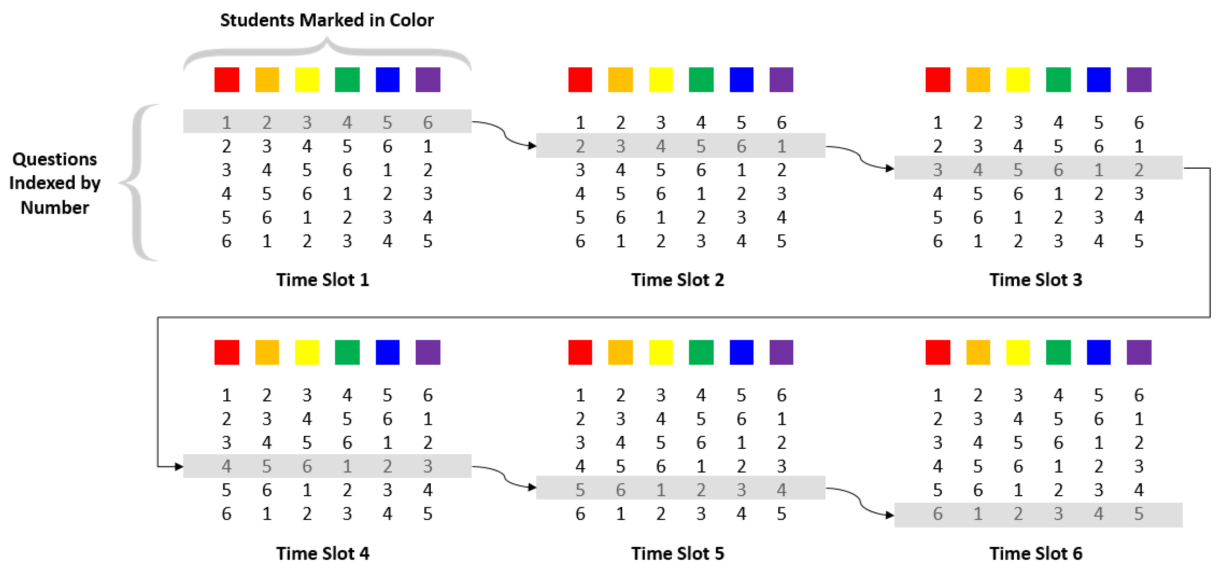

Figure 1. To six students, six MCQs are provided to each student one by one in a circular left shift scheme, and each question must be finished within the corresponding time slot in the grey bar.

More generally, we may design an exam with M2=K*M1>M1 as well, and K is a positive integer for convenience, and it is possible to have M2<N as well. In Figure 2, we present a flowchart of one way how to assign questions to students in an online exam in an orderly fashion (which can be shown to be essentially equivalent to random assignment).

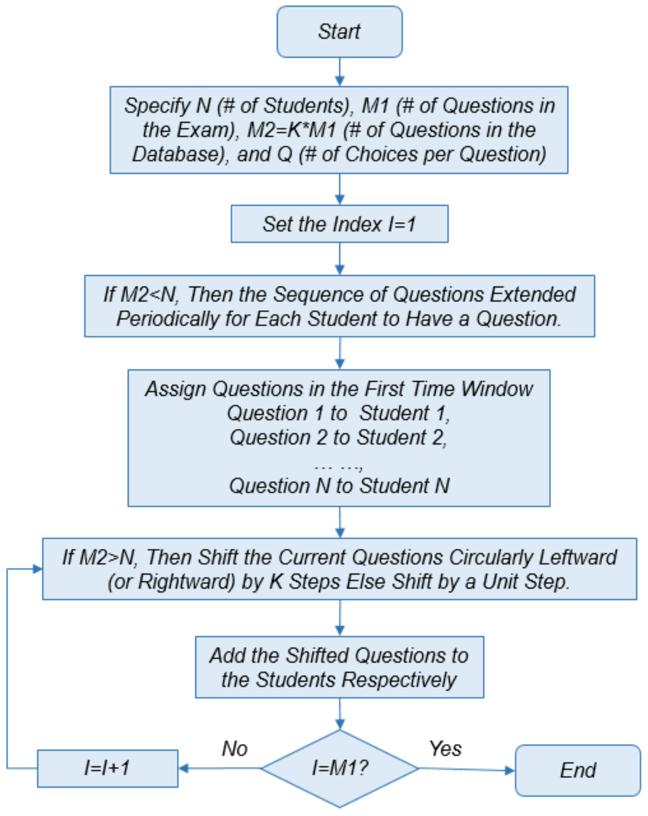
Start
Specify N (# of Students), M1 (# of Questions in the Exam), M2=K*M1 (# of Questions in the Database), and Q (# of Choices per Question)
Set the Index I=1
If M2<N, Then the Sequence of Questions Extended Periodically for Each Student to Have a Question.
Assign Questions in the First Time Window Question 1 to Student 1, Question 2 to Student 2, … …, Question N to Student N
If M2>N, Then Shift the Current Questions Circularly Leftward (or Rightward) by K Steps Else Shift by a Unit Step.
Add the Shifted Questions to the Students Respectively
No
I=I+1
I=M1?
Yes
End

Figure 2. Flowchart of presenting questions in a spatially diversified temporally controlled fashion.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 April 2020 doi:10.20944/preprints202004.0327.v1

In an ideal scenario, such an online exam would effectively prevent the students from cheating. Indeed, if the students were taking the online exam on their personal computers, they may also use their smartphones to communicate with each other. Since each student will in general have a different question within a particular time slot, it becomes difficult for them to collaborate. Unfortunately, this does not totally eliminate the possibility of cheating. Let us assume that two students, Red (a perfect student who knows all correct answer) and Green (a totally unprepared student who can only guess), decided to cheat during the exam. Although Red cannot directly share his/her answer with Green on the very first question, it may be possible for them to cheat on a later question. For instance, Red finishes Question 2 on the 2nd cycle while Green gets Question 2 on the 5th cycle. Hence, if Green asks for help on the fifth cycle, Red may provide the answer.

in an MCQ exam, a perfect student can get the correct answer to any question with the probability of 1, while a totally unprepared student can find the correct answer with the probability of *1/Q*. Let us assume that Green did not collaborate with Red and answered all the questions randomly. Given that each MCQ has *Q* options, one and only one of which is correct, the probability of Green getting *x* questions correctly answer, $P_X(x)$, out of *M1* questions can be expressed as the binomial distribution:

$$P_X(x) = \frac{M1!}{x!(M1-x)!}\left(\frac{1}{Q}\right)^x\left(1-\frac{1}{Q}\right)^{M1-x} \qquad (1)$$

By the design of the exam, Red will not be able to relay the answers of all the questions to Green. In Figure 1, Red receives 3 questions before Green. Since Green has no competence about the exam, the source of information will be unidirectional; i.e., from Red to Green. Under such a circumstance, if Red receives *z* questions before Green, Green will get at least *z* questions right. Out of the remaining *M1-z* questions, the probability for Green to get *t* questions correct is computed as

$$P_T(t,z) = \frac{(M1-z)!}{(t)!(M1-z-t)!}\left(\frac{1}{Q}\right)^t\left(1-\frac{1}{Q}\right)^{M1-z-t} \qquad (2)$$

Therefore, the total probability of Green getting *x* out of *M1* questions correct becomes

$$P_X(x) = \begin{cases} 0, x < z \\ P_T(x-z,z),\ z \le x \le M1 \end{cases} \qquad (3)$$

where the zero probability for *x<z* is due to the assumption that Green copied Red on all the z questions correctly. In order to compare the probabilities of Green getting *x* questions right out of *M1* questions with and without cheating, for a case shown in Figure 1, each question with four options, we have the results in Table 1.

x (out of 6)	$P_f(x)$	$P_c(x)$
0	*0.1780*	*0*
1	*0.3560*	*0*
2	*0.2966*	*0*
3	*0.1318*	*0.4219*
4	*0.0330*	*0.4219*
5	*0.0044*	*0.1406*
6	*0.0002*	*0.0156*

Table 1. Comparison of the probabilities of Green's chances in the exam with and without cheating, where $P_f(x)$ and $P_c(x)$ stand for probabilities of Green getting x questions right out of M correct in the fair and cheated exams respectively (Q = 4, z = 3 as shown in Figure 1).

Closer looking at Figure 1, it is immediately realized that the relative positioning of the two students in cheating has a significant impact on the cheating performance/gain. For instance, if

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 April 2020 doi:10.20944/preprints202004.0327.v1

Red and Purple are to collaborate, with Purple having the same academic competence as Green, the results will be quite different. This is because Red now receives five out of the six questions before Purple, and we have Table 2 for Purple similar to Table 1 for Green.

x *(out of 6)*	*$P_f(x)$*	*$P_c(x)$*
0	*0.1780*	*0*
1	*0.3560*	*0*
2	*0.2966*	*0*
3	*0.1318*	*0*
4	*0.0330*	*0*
5	*0.0044*	*0.75*
6	*0.0002*	*0.25*

Table 2. Comparison of the probabilities of Purple's chances in the exam with and without cheating, similar to Table 1 for Green. Note that due to his/her favorable position, Purple gains more than Green from Red in the exam.

It should be noted that although the probability of answering each question correctly by Purple, without the help from Red is 0.25 (in case of four options), the probability of answering multiple questions correctly decreases due to the binomial nature of such a distribution. Therefore, to better understand how much the probability of success for a cheating student improves, we define the average score in a fair exam (ASF, normalized with *M*) and the average score in a cheated exam (ASC, normalized with M) as follows:

$$ASF = \frac{\sum_{x=0}^{M1} xP_f\ f(x)}{M1} \tag{4}$$

$$ASC = \frac{\sum_{x=0}^{M1} xP_c\ (x)}{M1} \tag{5}$$

Using these two parameters, we further define a parameter called the gain-factor (g-factor) as follows:

$$g = ASC - ASF \tag{6}$$

Table 3 presents the ASF, ASC, and g-factor values for various cases as shown in Figure 1.

Scenario	*ASF*	*ASC*	*g-factor*
Red→Orange	*0.25*	*0.3750*	*0.1250*
Red→Yellow	*0.25*	*0.5000*	*0.2500*
Red→Green	*0.25*	*0.6250*	*0.3750*
Red→Blue	*0.25*	*0.7500*	*0.5000*
Red→ Purple	*0.25*	*0.8750*	*0.6250*

Table 3. ASF, ASC, and the g values for different scenarios in Figure 1 (N=M1=M2=6, Q=4).

It is evident that the ASF remains a fixed value (1/Q) irrespective of the position of the student but the ASC and g-factor depend on the relative position of the ideal student and the student who is copying from him/her.

In order to see how the g-factor changes with the position of the copying student when the perfect student is in the first position (like Red in Figure 1), we consider a class size of N=78 (which is the class size for our undergraduate course "Bio-Imaging and Bio-Instrumentation" in this semester). We set *M1=M2=N=78* and *Q=4.* With MatLab, we developed a program to compute the value of the g-factor from the second position to the 78th position, as plotted in Figure 3.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 April 2020 doi:10.20944/preprints202004.0327.v1

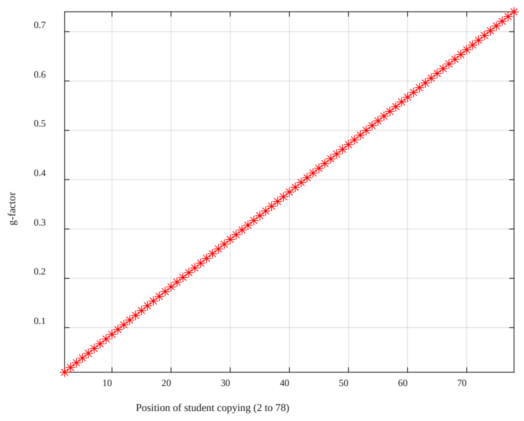

Figure 3. g-factor versus the position of the cheating student (M1=M2=N=78, Q=4).

It is observed in Figure 3 that there is a steady rise in the value of the g-factor with an overall average g-factor of 0.3750. It each student gets exactly the same question in each time slot, the value of the g-factor for a cheating student helped by a perfect student would be 0.75 (meaning he/she would get a perfect score too). Since 78 MCQs usually take a long time to answer, meaning a portion of the students may not be able to answer all the questions within the allotted time. This will reduce the value of the g-factor computed in the hypothetical situation. In Figure 3, the cheating student at the 78th position has a g-factor close to 0.75 (having a nearly perfect score). Therefore, a question arises is how we can control the maximum value of *g*. One way to address this issue would be to increase *M2* and use *M1* out of these *M2* questions. In such a case, it becomes feasible to reduce the information flow from the perfect student to the cheating student. This scenario is illustrated in Figure 4 where *M2=8* instead of *M2=6* in Figure 1.

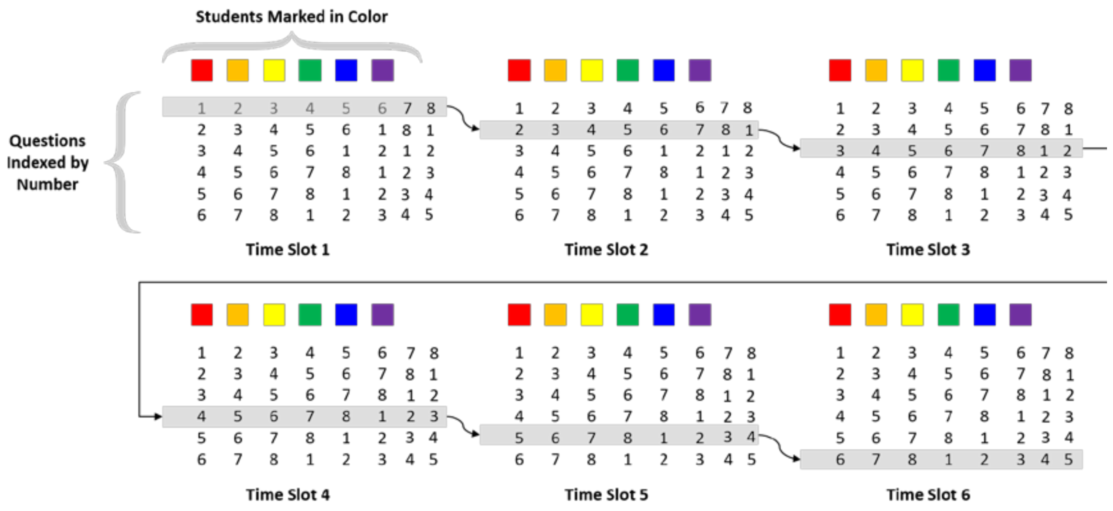

Figure 4. To six students, eight MCQs are provided to each student one by one in a circular left shift scheme, which an enhanced version of the exam shown in Figure 1.

Using our *MatLab* simulator, keeping all other conditions the same (Q=4) we can compute the values of the ASF, ASC, and *g* values again. The results are shown in Table 4.

Scenario	ASF	ASC	g-factor
Red→Orange	0.25	0.25	0
Red→Yellow	0.25	0.25	0
Red→Green	0.25	0.3750	0.1250
Red→Blue	0.25	0.5000	0.2500
Red→ Purple	0.25	0.6250	0.3750

Table 4. ASF, ASC, and the g values for different scenarios in Figure 1 (N=M1=6, M2=8, and Q=4).

Thus, by increasing the pool of questions the g-factor can be reduced. Furthermore, we run a simulation similar to that shown in Figure 3 for *N=M1=78*, and *M2=100*. The g-factor is again plotted with respect to the position of the cheating student from the 2nd to the 78th position in Figure 5. This time a few students have zero g-factors, and the maximum g-factor is slightly more than 0.5.

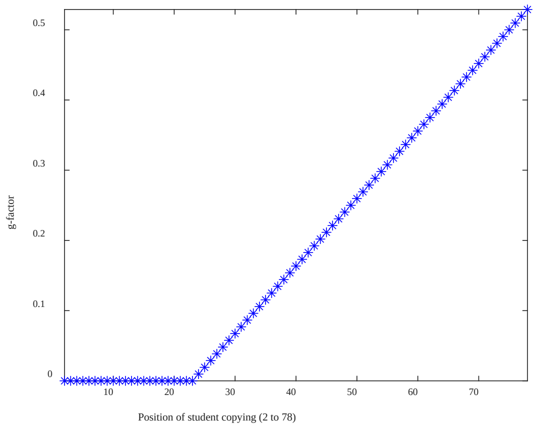

Figure 5. g-factor versus the position of the cheating student (M1=N=78, M2=100, and Q=4).

In practice, *M1* should be chosen smaller than *N*, in the case of large classes or national exams. The selection of *M1* mainly relies on the content to be tested and the period of time to be reasonable, and should not depend on *N*. Hence, a realistic scenario is that given *N* students, *Q*-choices MCQs of a pre-decided number *M1*, we can achieve a targeted small average g-factor of all positions a cheating student can take using a sufficient large MCQ database of size *M2* (we only consider *M2=K*M1* without loss of generality, where *K* is a positive integer for convenience).

III. Guiding Principle: Average g-Factor Reduction

To find good combinations of *M1* and *M2*, we can use the mean ASC or *g* scores as the criterion, defined as the average obtained at all student positions except the first one that is for the knowledge-wise perfect student who helps the cheating student. A flowchart for calculation of the mean ASC scores is shown in Figure 6.

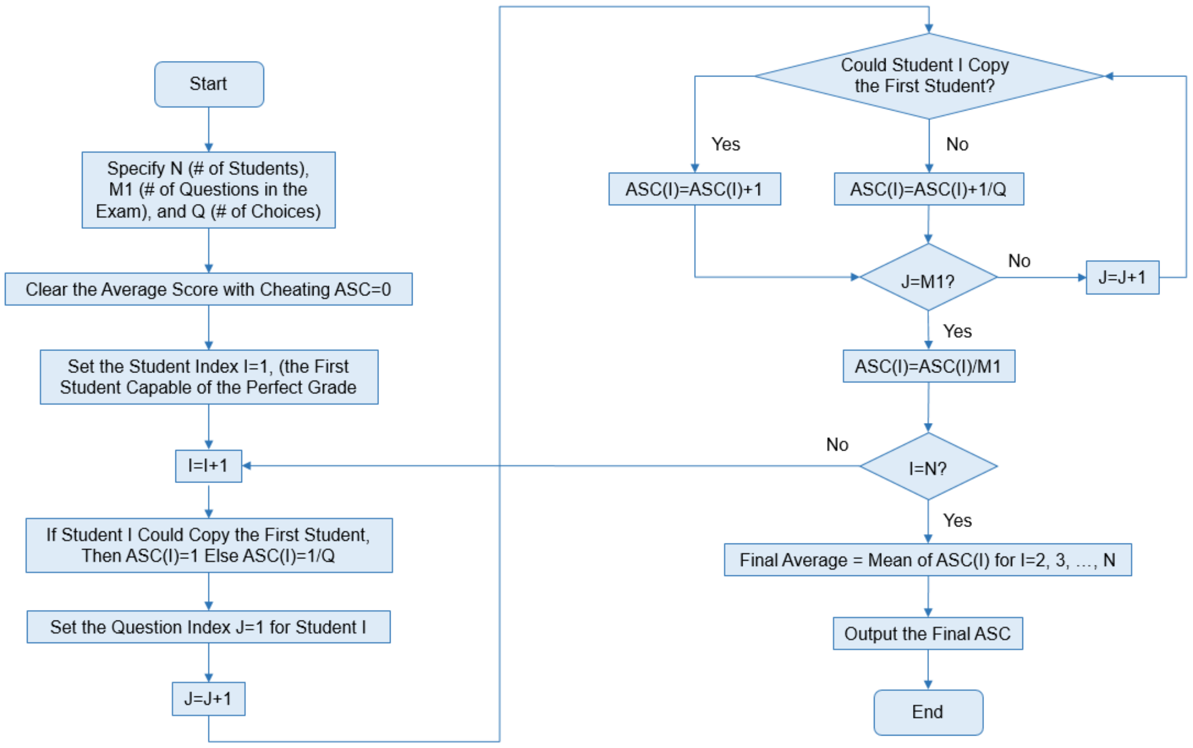
Start
Specify N (# of Students), M1 (# of Questions in the Exam), and Q (# of Choices)
Clear the Average Score with Cheating ASC=0
Set the Student Index I=1, (the First Student Capable of the Perfect Grade
I=I+1
If Student I Could Copy the First Student, Then ASC(I)=1 Else ASC(I)=1/Q
Set the Question Index J=1 for Student I
J=J+1
Could Student I Copy the First Student?
Yes
No
ASC(I)=ASC(I)+1
ASC(I)=ASC(I)+1/Q
J=M1?
No
J=J+1
Yes
ASC(I)=ASC(I)/M1
No
I=N?
Yes
Final Average = Mean of ASC(I) for I=2, 3, …, N
Output the Final ASC
End

Figure 6. Flowchart for calculation of the mean ASC scores.

We evaluated the combinations of *M1* from 10 to 100 with a step of 10, *K* from 1 to 15, and *N* of 80 and 40 respectively. The mean ASC scores in these cases are summarized in Tables 5 and 6. The mean g-factor can be easily calculated by subtracting the results by 0.25.

M1	***K = 1, 2, 3, …, 10***									
10	*0.6582*	*0.4494*	*0.3544*	*0.3449*	*0.3022*	*0.3022*	*0.3022*	*0.2927*	*0.2842*	*0.2766*
20	*0.6392*	*0.4399*	*0.3497*	*0.3402*	*0.3070*	*0.2932*	*0.2813*	*0.2714*	*0.2671*	*0.2633*
30	*0.6044*	*0.3972*	*0.3611*	*0.3101*	*0.2880*	*0.2788*	*0.2709*	*0.2642*	*0.2614*	*0.2589*
40	*0.6297*	*0.4351*	*0.3333*	*0.2951*	*0.2785*	*0.2716*	*0.2657*	*0.2607*	*0.2585*	*0.2566*
50	*0.5747*	*0.3981*	*0.3166*	*0.2861*	*0.2728*	*0.2673*	*0.2625*	*0.2585*	*0.2568*	*0.2553*
60	*0.5696*	*0.3734*	*0.3055*	*0.2801*	*0.2690*	*0.2644*	*0.2604*	*0.2571*	*0.2557*	*0.2544*
70	*0.5931*	*0.3558*	*0.2976*	*0.2758*	*0.2663*	*0.2623*	*0.2590*	*0.2561*	*0.2549*	*0.2538*
80	*0.6250*	*0.3426*	*0.2917*	*0.2725*	*0.2642*	*0.2608*	*0.2578*	*0.2553*	*0.2543*	*0.2533*
90	*0.5833*	*0.3323*	*0.2870*	*0.2700*	*0.2627*	*0.2596*	*0.2570*	*0.2547*	*0.2538*	*0.2530*
100	*0.5500*	*0.3241*	*0.2833*	*0.2680*	*0.2614*	*0.2586*	*0.2563*	*0.2543*	*0.2534*	*0.2527*

Table 5. Mean ASC scores under different combinations of M1 and K (N = 80, Q = 4).

M1	K = 1, 2, 3, …, 10									
10	0.6538	0.4423	0.3558	0.3365	0.3038	0.2904	0.2788	0.2692	0.2692	0.2615
20	0.6346	0.4327	0.3375	0.2933	0.2769	0.2702	0.2644	0.2596	0.2596	0.2558
30	0.5769	0.3718	0.3083	0.2788	0.2679	0.2635	0.2596	0.2564	0.2564	0.2538
40	0.6250	0.3413	0.2938	0.2716	0.2635	0.2601	0.2572	0.2548	0.2548	0.2529
50	0.5500	0.3231	0.2850	0.2673	0.2608	0.2581	0.2558	0.2538	0.2538	0.2523
60	0.5000	0.3109	0.2792	0.2644	0.2590	0.2567	0.2548	0.2532	0.2532	0.2519
70	0.4643	0.3022	0.2750	0.2624	0.2577	0.2558	0.2541	0.2527	0.2527	0.2516
80	0.4375	0.2957	0.2719	0.2608	0.2567	0.2550	0.2536	0.2524	0.2524	0.2514
90	0.4167	0.2906	0.2694	0.2596	0.2560	0.2545	0.2532	0.2521	0.2521	0.2513
100	0.4000	0.2865	0.2675	0.2587	0.2554	0.2540	0.2529	0.2519	0.2519	0.2512

Table 6. Mean ASC scores under different combinations of M1 and K (N = 40, Q = 4).

As demonstrated along the row direction of Tables 5 and 6, increasing *K* will decrease the *g*-factor, which is expected since we are increasing *M2*. Interestingly, going down the first column in Table 5, the mean ASC score decreases with jumps, which can be explained as follows. If *M2* is smaller than N, then some positions will inevitably receive the same sequence of MCQs as that of the first position receives, and those positions will have a large *z* equal to *M1*. In the first column, *M2=M1*, hence if *M1* is much smaller than *N*, we will periodically introduce more such positions and raise the mean ASC scores. Similar effects are also observed in the first column in Table 6. Comparing Tables 5 and 6, it is seen that the values in Table 6 are generally smaller than those in Table 5. This indicates that the case with a smaller *N* is easier to handle (to reduce the mean ASC). For the situation of our interest, with *N* around 80 and *Q=4*, a good and reasonable practice will be *M1=30*, *K=3*, which will lead to an average *g*-factor about 0.111 according to Table 5. In addition, if *N* is decreased to 40, the average *g*-factor can be reduced to 0.058 according to Table 6.

IV. Discussions and Conclusion

The above initial analysis is basically focused on the scenario where a perfect student A helps a totally unprepared student B, which is the extreme case that maximizes the gain of cheating. Given the limited time of an exam and involved stresses, the student B needs ideally only one perfect helper, and is unlikely to team up with more than one students. Since B is totally unprepared, he/she is unable to judge among inputs from multiple helpers. Multiple helpers might not really help B better than a single perfect helper, and are harder to coordinate than working with only Student A. Hence, the "*A helping B*" model is well justified.

How much Student A could help Student B depends on the orders in which the multiple choice questions are presented to each of them. Student B can only benefit from the answers Student A has finished before his/her allotted time slots are over, and in the best case B can see A's all answers (B may not have time to benefit from A's first answer) while in the worst case B can only get one correct answer from A. Practically, the average gain is what matters in our exam design, which is the *g*-factor we have analyzed above. Because a sufficient small *g*-factor can be always achieved, the cheating behavior in the "*A helping B*" model will not improve B's exam score significantly. This demonstrates that online exams during social distancing can be effectively conducted by a good design.

While the *g*-factor analysis is a good way to suppress cheating behaviors so that their effects are insignificant, with data analysis such cheating behaviors can be actually detected and potentially penalized for even better quality of online exams during social distancing. This is particularly possible if the value of *M2* is greater than *M1*. Since positioning of the student may have a significant impact on the performance, we may do a presumptive placement based on the performance of the students in their previous courses or their cumulative grade point averages (CGPAs). While it is no guarantee that a student with a low CGPA would copy from a student with a high CGPA, it can be one of the strategies that we employ to combat cheating during the exam. Other strategies include giving exams by shuffling the positions of the students and have data that we can analyze to optimize the orders of their questions in the exam. When a prior distribution of students' scores is available, their competence levels can be readily estimated to minimize the average *g* factor. Moreover, using machine learning techniques we can adaptively identify potential cheating episodes in real-time during an online exam in the following steps as an example. First, high and low performers can be easily identified in an early stage of the exam, and the answers of a high performer in an early stage can be correlated to the answers of any low performer in a later stage. Any significant discrepancy between the hit rates of a student in early and late stages may suggest a potential cheating incident. The significance of the discrepancy can be tested in reference of the prior distribution of students' scores. That is, an exam may be divided into two parts unknown to students, one part is used to capture potential cheating, while the other part is used to produce his/her score. Optionally, the length of an exam for Student B can be extended if the likelihood appears high to indicate that A is helping B. In the extended portion, B will get questions that A has no answers. An additional note that *M1* questions can have different weights/credits/time-lengths, as shown in Figure 7, and our methods can be generalized in this and other ways. More generally, the relationship between anti-cheating schemes and cheating strategies can be casted in the game theory framework [3], and deserves further investigation, even using the block chain technology [4] to keep the database of questions confidential (fully accessible to faculty only) and students' individual credits well managed (for them to earn academic credits).

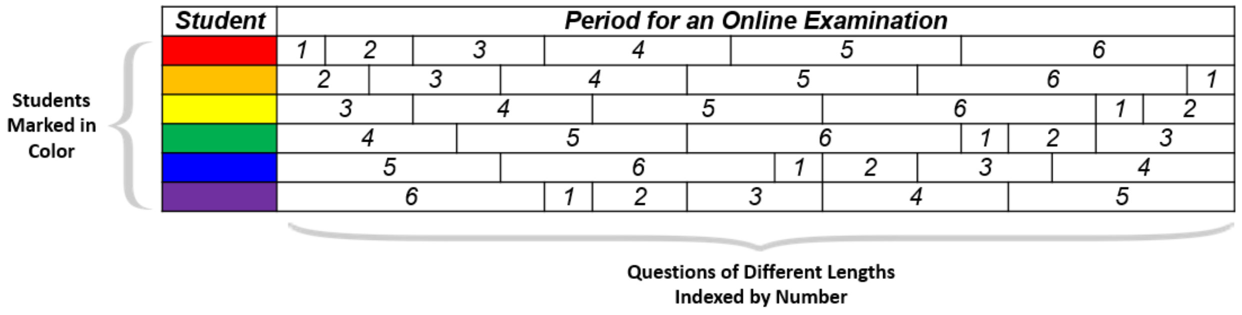

Figure 7. To six students, six MCQs of different lengths are provided to each student one by one in a circular left shift scheme.

Social distancing due to COVID-19 has a positive aspect that online learning and testing practice becomes immediately popular, thanks to the Internet and mobile technologies. A high-quality online examination system is a key to offer uncompromised educational outcomes. Anticipating further development of online learning and testing platforms, it is perhaps unique time to revisit traditional teaching practice. If both teaching and testing can be conveniently and professionally done during social distancing, why not keep doing in this mode even after the pandemic? This will reduce needs for conventional infrastructures, share and integrate optimal teaching and testing resources, and deliver high-quality education equally well globally.

In conclusion, we have proposed a novel scheme for online exams during social distancing, illustrated the design theory and examples, and produced guidelines for our own final exam in our undergraduate course "*Bio-Imaging and Bio-Instrumentation*" in this semester. Further investigation and implementation are actively under way. We welcome feedback and collaboration.

References

1. *http://www.act.org/content/act/en/covid19.html*
2. *https://www.proctortrack.com*
3. *Spaniel W: Game Theory 101: The Complete Textbook, 278 pages, Kindle Edition, Copyright William Spaniel, 2011-2013, http://gametheory101.com*
4. *https://www.guru99.com/blockchain-tutorial.html*

Appendix. Sample MATLAB Code for Our Numerical Study of Online Exams

```
close all
clear all
clc

%% taking input and getting rid of invalid inputs
while 1
    N=input('No of students:');
    M=input('No of questions:');
    P=input('Input pool of questions:');
    N=floor(N);
    M=floor(M);
    P=floor(P);
    if M~=N
        disp('Not possible, input again');
    elseif P<M
        disp ('Not possible, input again');
    else
        while 1
            C=input('Choices per question (at least 2):');
            C=floor(C);
            if C<2
                disp('Invalid no. of choices, input again');
            else
                break
            end
        end
        break
    end
end
% while 1
    ideal_student=input(['Select ID of perfect student (1 to
',num2str(N),'):']);
%     copy_student=input(['Select ID of copy student (1 to
',num2str(N),'):']);
    ideal_student=floor(ideal_student);
    copy_student=floor(copy_student);
    if ideal_student==copy_student
        disp('Not possible, give selection again');

    elseif ideal_student<1||ideal_student>N||copy_student<1||copy_student>N
        disp('Invalid input, give selection again');

    else
        break
    end
end

%%
student_id=1:N;
question_id=1:P;

if P>N
    student_id=[student_id zeros(1,P-N)]; % append the additional P-M
questions at the end
```

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 19 April 2020 doi:10.20944/preprints202004.0327.v1

```
end
master_array=student_id;
z=-1; % -1 for Left shift
for i=1:M
    master_array=[master_array;question_id]%assigns q_id to each student_id
    question_id=circshift(question_id,z);%tracks the questions as the exam
progresses
end
q_order_ideal=master_array(:,ideal_student);%question order for ideal stdnt
q_order_copy=master_array(:,copy_student);%question order for copy stdnt

count=0;
for m=2:length(q_order_ideal)%check for copy
    for n=m+1:length(q_order_copy)
        if q_order_copy(n)==q_order_ideal(m)
            count=count+1;
            break
        end
    end
end

%% calculate probabilities
prob_sure=zeros(1,count);%matching with ideal
if count %if no overlap with ideal
    prob_sure(count)=1;
else
    prob_sure=[];
end
rem_q=M-count;%non-matching
prob_array=0:rem_q;
prob_binomial=binopdf(prob_array,rem_q,1/C);%binomial distribution
prob_copy=[prob_sure prob_binomial];%probabilites for cheating
prob_fair=binopdf(0:M,M,1/C);%probabilities for fair exam
ASF=1/M*sum((0:M).*prob_fair);%ASF
ASC=(count+sum((0:M-count).*prob_binomial))/M;%ASC
g_factor=ASC-ASF;%g-factor
```