 doi:10.20944/preprints202005.0502.v1

Anti-cheating Online Exams by Minimizing the Cheating Gain

Mengzhou Li[1], Sujoy Sikdar[2], Lirong Xia[3], and Ge Wang[1]*
[1]Department of Biomedical Engineering, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA
[2]Department of Computer Science and Engineering, Washington University in St. Louis, St. Louis, MO 63130, USA
[3]Department of Computer Science, Rensselaer Polytechnic Institute, Troy, NY, 12180, USA

***Abstract*—Cheating prevention in online exams is often hard and costly to tackle with proctoring, and it even sometimes involves privacy issues, especially in social distancing due to the pandemic of COVID-19. Here we propose a low-cost and privacy-preserving anti-cheating scheme by programmatically minimizing the cheating gain. A novel anti-cheating scheme we developed theoretically ensures that the cheating gain of all students can be controlled below a desired level aided by the prior knowledge of students' abilities and a proper assignment of question sequences. Furthermore, a heuristic greedy algorithm we developed can refine an assignment of questions from a cyclic pool of question sequences to efficiently reduce the cheating gain. Compared to the integer linear programming and min-max matching methods in a small-scale simulation, our heuristic algorithm provides results close to the optimal solutions offered by the two standard discrete optimization methods. Hence, our anti-cheating approach could potentially be a cost-effective solution to the well-known cheating problem even without proctoring.**

***Index Terms*—Online exam, cheating prevention, discrete optimization, social distancing**

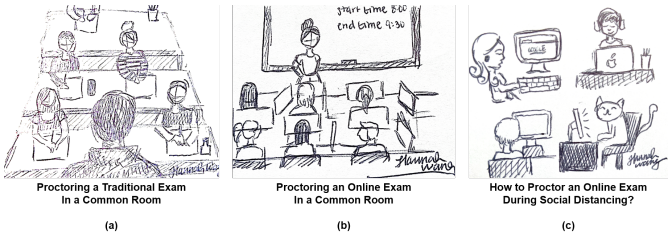


Fig. 1. Quality assurance of exams in increasingly challenging modes. (a) Proctoring a paper-based exam by separating students physically to avoid copying from a neighbor, (b) proctoring a conventional online exam by disabling some functions of students' computers, and (c) it is the most challenging setting when students take an online exam in their individual homes where modern internet/telecommunication tools are readily available.

I. Introduction

Exams have been widely used in education for measurement of students' mastery of knowledge, problem-solving ability, and skills. One of the most important criteria for good exams is fairness, and an important factor compromising fairness is cheating. Therefore, in traditional paper-based tests efforts have been made in creating, administering, and proctoring an exam to maintain academic integrity. Similar guidance has been practiced for computer-based exams to prevent cheating [1], largely relying on proctoring services, such as proctor supervision, camera monitoring, and specialized protocols. Such services are not only costly but also raise concerns on privacy leakage [2].

Conducting low-cost and privacy-preserving anti-cheating exams becomes a pressing issue in particular due to the COVID-19 pandemic. As shown in Fig. 1, students who are required to take online exams are now on a much larger scale than ever before. How to maintain a high quality test becomes a new educational challenge with immediate and global impacts. For example, American College Testing (ACT) "has rescheduled its April 4 national test date to June 13 across the U.S. in response to concerns about the spread of the coronavirus (COVID-19)" [3], and Educational Testing Service (ETS) has stopped its TOFLE and GRE testing from China mainland and Iran [4, 5], since inefficient measures to prevent cheating would undermine the intended assessment. Similarly, many graduate and undergraduate classes use web-camera-based proctoring tools, perform open-book tests, or rely on class project based assessments. Nevertheless, ineffective proctoring or no proctoring at all encourage cheating behaviors, and compromise the teaching outcomes.

With modern internet and mobile technology, the cheating barrier becomes much lower for well-equipped students at home. Remote proctoring itself can be insufficient for cheating prevention during social distancing, which monitors student with webcam and screen video and often comes with complex protocols to prevent students from cheating. The monitoring is often performed by human due to complexity of cheating identification, and it could be very troublesome for exams of large class. Thus, instead of using solutions for proctoring physically or virtually, our group proposed a new anti-cheating mechanism for online exams which could help prevent cheating by reducing the gain a student can obtain through cheating [6]. The general idea of our previous method is to present students questions one by one in a way such that no (or almost no) two students receive the same question within each allowed time slot/window for his/her current question, as demonstrated in Fig. 2 [6]. The average gain from cheating of a student is greatly reduced due to this synchronized different question sequences the students receive. However, this method is not perfect, e.g., the required number of questions for the test depends on the size of the class. The cheating gain of a student relates to his/her relative position to the target student, changes in a large range, and is beyond control.

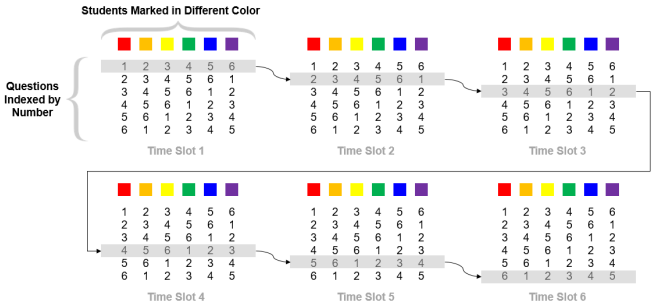


Fig. 2. Key idea of the anti-cheating scheme proposed in [6] illustrated with a simple example, in which 6 students take an exam, and 6 multiple choice questions are provided to each student one by one in a circular shift fashion, and each question must be finished within the corresponding time slot shown as the gray bar.

*To whom correspondence should be addressed; E-mail: wangg6@rpi.edu.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

To address these problems, here we propose a grouping-based anti-cheating scheme which is able to control the maximum cheating gain of any student to a desired level regardless of the class size by taking advantage of the prior information of the students ability profile (which can be inferred from their previous performance or extracted in an exam adaptively) and a proper assignment of question sequences. In addition to this general mechanism in the case the number of questions for each student and the size of the question pool are both fixed, we also provide a heuristic greedy optimization method which can further reduce the cheating gain of the class by searching for a better assignment of sequences with a fast speed. Due to the nature of this optimization problem, comparison was done with standard searching methods in the discrete optimization field, i.e., integer linear programming (ILP) and minimum weight maximum matching. Note that the goal of our heuristic optimization method is not to find the optimal solution under one setting but to find a practically optimal solution within a short time, hence to be able to give a quick answer with a reasonable small cheating gain to different settings. The standard methods often come up with an optimal or nearly optimal solution but usually take longer time for the large size of the permutation space. The comparison with them illustrates how much room left for further improvement of our heuristic method. A good way is to use our heuristic results to initialize the standard optimization procedure. In the following sections, we describe our methodology and results, and discuss issues which can be addressed for further improvement.

II. Definitions and assumptions

Before talking about the methods, let us first briefly describe the scenario with some notations and introduce our assumptions about the cheating mechanism and our goal.

In an online exam, M multiple-choice questions (MCQs) from a bank of M_p MCQs (assumed with equal difficulty without loss of generality) are provided to a class of N students, and there are Q choices per question with one and only one of them being correct. Thus, the size of the pool of all possible question sequences will be $M_p!/(M_p - M)!$.

With a proper method, we can extract N sets of questions from the pool and assign the question sequences to the students in the exam (some sets of questions can be repeated used if needed). Suppose that the ability profile of the students is known which can be inferred based on their previous performance or even during an exam, and the ability of a student represents the probability for the student to solve a problem correctly. Without loss of generality, we can rank the students i from 1 to N, and denote their corresponding abilities as y_i ($1 \geq y_1 \geq y_2 \geq \ldots \geq y_N \geq 1/Q$).

The assumptions and simplifications about cheating are as follows:

1) Cheating is unidirectional. If two students A and B collaborate on cheating and $y_A > y_B$, then only B will copy answers from A which is termed as B cheating from A while A helping with B;
2) B can get the answer from A if A has already answered the question before B or they are working on the problem at the same time. Thus, different relative question sequences for A and B will influence the number of questions that B can copy from A;
3) Each student can only cheat from no more than one student ("A helping B" model);
4) Helping others cheating does not affect the status of a student cheating from others;
5) B cheating from A does not influence C cheating from A; in other words, one student can help multiple students;
6) Answers from cheating are not to be spread through helping other students (This simplification can be justified that given the limited time of an exam and involved stresses B is unlikely to remember what he/she copied from A and to have the time to provide C with the answer).

Our goal is to develop a method to find an appropriate assignment of question sequences (i.e., sets of M questions) to the students which can reduce (hopefully minimize) the overall expected and/or maximum gain by cheating utilizing the information on student ability profiles. The problem has been discussed in two cases: the case to control the cheating gain without limits on M_p and M and the case to minimize the cheating gain with fixed M_p and M. Section IV mainly discussed the first case while the other sections mainly addressed the second case.

III. Model Restatement for Optimization

Based on the above assumptions and simplifications, an instance $([N], [M_p], M, Q, Y, P)$ of the distanced online testing (DOT) problem is defined by a set of $N \in \mathbb{N}$ *students* $[N] = \{1, 2, \cdots, N\}$, a set of $M_p \in \mathbb{N}$ *questions*, a positive integer $M \in [1, M_p]$ which is the number of questions to ask each student, a positive integer Q which is the number of choices in each question with only one correct answer, an *ability profile* $Y = \{y_i \in [1/Q, 1] | i = 1, 2, \cdots, N\}$ indexed in an descending order, and a *cheating matrix* $P = (p_{j,i})_{i,j\in[N]}$, where $p_{j,i}$ depicts the probability for student i to actively cheat from student j if $i \neq j$ and the first index indicates the source of the answers while the second index indicates the destination of the information. Based on the assumption (1) that no student copies from another student with lower ability, for every pair of students $i, j \in [N]$, if $j > i$, $p_{j,i} = 0$, and our cheating matrix is upper triangular. As a special case $p_{i,i}$ denotes the probability that student i does not commit active cheating in the exam as follows:

$$p_{i,i} = 1 - \sum_{j=1}^{i-1} p_{j,i}. \tag{1}$$

An *assignment* $A = \{a_i \in S_{QS} | i = 1, 2, \cdots, N\}$ is a mapping from $[N]$ to the pool of all possible question sequences S_{QS} (M-permutations of $[M_p]$), where for each student $i \in [N]$, a_i is the question sequence assigned to student i. The question sequence pool $S_{QS} = \{s_i | i = 1, 2, \cdots, n\}$ with each s_i representing a question sequence of length M, and the size of the pool $n = M_p!/(M_p - M)!$.

Given an assignment A, for each student $i \in [N]$, $q_i(A)$ denotes the *expected cheating score* of student i on the exam, when they cheat according to the cheating matrix P, and $q_i^*(A)$ denotes the *expected honest score* when they do not cheat, and $g_i(A) = q_i(A) - q_i^*(A)$ as the *expected gain* from cheating on assignment A. We simply use q_i, q_i^*, and g_i when the assignment A is clear in the context. Based on the ability definition, q_i^* only depends on the ability y_i of student i, hence we have $q_i^*(A) = y_i$ regardless of the assignment A. Our goal is to minimize g_i for $i = 1, 2, \ldots, N$. Due to the non-negativity of g_i and the independence between students, a reasonable metric (or loss function) can be defined as follows:

$$\min_A g = \sum_{i=1}^{N} g_i(A) \tag{2}$$

To calculate g_i for student i given an assignment A, we further define the *positional matrix* $Z = (z_{j,i})_{i,j\in[N]}$ where $z_{j,i}$ depicts the number of questions that student i can cheat from student j if $j \neq i$, and the special case $z_{i,i}$ is defined as M. Note that the values of positional matrix Z can be calculated from assignment A and are independent from the cheating matrix P and the student ability

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

profile Y. The cheating score q_i of student i consists of two parts, the cheating cases and the honest case, and is expressed as

$$q_i(A) = \frac{1}{M}\sum_{j=1}^{i-1} p_{j,i}[z_{j,i}(A)y_j + (M - z_{j,i}(A))y_i] + y_i p_{i,i}. \quad (3)$$

With Eqs. (1) and (3) the cheating gain g_i of student i is computed as

$$g_i(A) = q_i(A) - y_i = \sum_{j=1}^{i-1} \frac{z_{j,i}(A)}{M} p_{j,i}(y_j - y_i) \quad (4)$$

The overall cheating gain g becomes

$$g(A) = \sum_{i=1}^{N} g_i(A) = \sum_{i=1}^{N}\sum_{j=1}^{i-1} \frac{z_{j,i}(A)}{M} p_{j,i}(y_j - y_i) \quad (5)$$

and can be easily written in the matrix form due to the upper triangular nature of P,

$$g(A) = \frac{1}{M}\text{sum}\{Z(A) \circ P \circ D\} \quad (6)$$

where $\circ$ denotes Hadamard multiplication (element-wise), and the ability difference matrix D is defined as $(d_{j,i})_{i,j\in[N]}$ and $d_{j,i} = \max(y_j - y_i, 0)$.

In Eq. (6), $Z(A)$ is only determined by the relative question sequences of students defined by A while P is defined by the cheating mechanism, and D can be taken as constant given the ability profile Y. Thus, our goal is to construct or search a good mapping A between question sequences S_{QS} to students $[N]$ for a practically insignificant loss.

In the conventional scenario without cheating prevention, all students share the same question sequence, hence $z_{j,i} = M$ for $i, j \in [N]$ based on the assumption (2), and the overall cheating gain is computed as

$$g_0 = \sum_{i=2}^{N}\sum_{j=1}^{i-1} p_{j,i}(y_j - y_i) = \text{sum}\{P \circ D\}. \quad (7)$$

IV. Grouping-based Anti-cheating Scheme

We propose a general scheme for cheating prevention in online exams during social distancing with prior knowledge on students' abilities. Generally speaking, three strategies have been used to reduce the overall cheating gain: (1) Group students with similar abilities together and assign them the same question sequence; (2) Limit the number of questions that can be copied by a low-ability group from a high-ability group in a circular shifting manner; (3) Make the question bank size larger than the number of questions each student receives in the exam to further reduce the question leakage between groups.

The idea behind the scheme is as follows. Circular shifting is an easy yet effective way to generate different question sequences systematically. As an illustrative example shown in Fig. 3(a), the sequences in the topper part tend to have smaller question leakage from top to down, i.e., the maximum leakage in the top 3 sequences is 2 compared to the maximum leakage of 5 in the top 4 sequences. Moreover, if $M < M_p$, we can even make the maximum leakage to 0 in the top few sequences as shown in Fig. 3(b). We can actually make the top $M_p - M + 1$ sequences have the maximum leakage of 0. If the number of students $N < M_p - M + 1$, a good cheating prevention solution could be that we assign students (in a descending order of their abilities) with these top $M_p - M + 1$ sequences and they will theoretically have no cheating gains. But this is often impractical for a big class with hundreds of students which requires hundreds of questions in the bank. However, instead of increasing the question bank size, we can achieve the same goal by equivalently reducing the

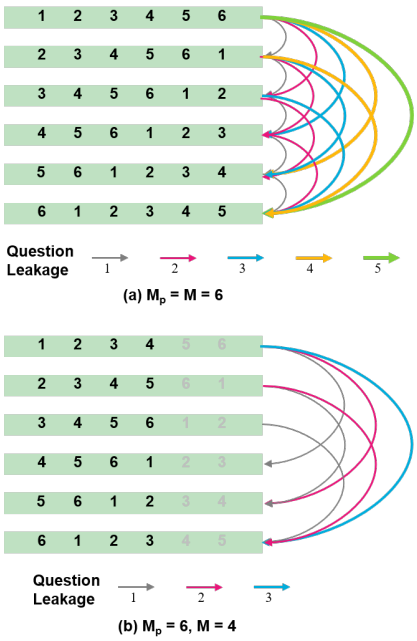


Fig. 3. Circular shifting demo: (a) $M_p = M = 6$; (b) $M_p = 6$, $M = 4$.

number of students with our first strategy. By grouping, the equivalent number of students (the number of groups) can be significantly reduced. As long as the group number is no larger than $M_p - M + 1$, we can eliminate the inter-group question leakage. On the other hand, students with similar abilities have small probabilities to cheat within their group due to the fact they can only obtain tiny cheating gains at risk, and the intra-group cheating gains are facilitated because of the same sequence shared, but this type of cheating can remain at a small level due to the small differences among their abilities. Furthermore, the maximum ability difference inside a group could be well controlled by increasing the group numbers.

Based on the idea above, we design a scheme which can control the maximum individual cheating gain as well as the average cheating gain to any desired level for any large-size class. The scheme can be designed as follows.

1) Specify the desired upper limit of the maximum individual cheating gain as $(1 - 1/Q)/C$, where C is a positive integer controlling the cheating gain;
2) First divide the ability range into C intervals of equal length;
3) Classify the students based on these intervals into the corresponding C groups;
4) Choose the question bank size M_p and the test length M to make $M_p - M + 1 \geq C$;
5) Generate different SQs by circularly shifting of questions with unit step to left;
6) Assign the top C sequences to the C groups in an descending order of their abilities.

The inter-group cheating gain is eliminated in this scheme due to the zero question leakage between groups, while the maximum individual intra-group cheating gain is no greater than $(1 - 1/Q)/C$ which is only achieved when there are two students who happen to be at the two end points of one interval as shown in Fig. 4. The cases achieve the maximum individual cheating gain are usually rare, so the average cheating gain should be much smaller than the maximum individual cheating gain, and a toy example illustrating the effectiveness of the scheme is shown in Fig.5. With this scheme, by increasing C to be sufficiently large we can control the maximum individual cheating

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

gain as well as the average cheating gain below any desired level, e.g., the maximum individual cheating gain is theoretically controlled below 3.6% for any large-size class with a reasonable test setting of $Mp = 60, M = 40, Q = 4$.

For better understanding of this general grouping-based anti-cheating scheme, we present the following theorem.

Theorem 1. *Given sequences of M questions from the bank of M_p MCQs and with only one correct choice out of Q choices for each question, the maximum individual cheating gain can be controlled to no larger than $(1-1/Q)/(M_p-M+1)$ (by following the grouping-based anti-cheating scheme).*

Proof. For any two same length question sequences s_1 and s_2, let $F_z(s_1, s_2)$ stand for the number of questions that can be copied from s_1 to s_2. Let us denote the M_p questions as $\{1, 2, 3, \ldots, M_p\}$.

Following the grouping-based anti-cheating scheme, (1) by circular shifting, we can easily create $M_p - M + 1$ sequences: $s_1 = [1, 2, 3, \ldots, M]$, $s_2 = [2, 3, \ldots, M, M+1]$, ..., $s_{M_p-M+1} = [M_p - M + 1, M_p - M + 2, \ldots, M_p]$. It is easy to check for any pair s_i and s_j out of them, we have $F_z(s_i, s_j) = 0$ if $i < j$. (2) Let us divide the maximum possible student ability score range $[1/Q, 1]$ into $M_p - M + 1$ intervals, and the length of each interval is $(1-1/Q)/(M_p-M+1)$. Then, we can group the students whose abilities are in the same interval, and obtain $M_p - M + 1$ groups. We assign the sequences $s_1, s_2, \ldots, s_{M_p-M+1}$ to the groups ranked in the descend order of their abilities. By doing so, we have achieved two goals: first, there is no cheating gain between groups ensured by (1); second, the individual maximum gain inside a group cannot be greater than $(1-1/Q)/(M_p-M+1)$ which is the interval length, ensured by (2), and regardless of how many students are in the group. Hence, we have proved the theorem. ■

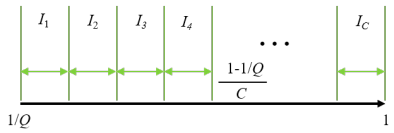


Fig. 4. A general scheme to control the maximum individual cheating gain to be below any desired level. By dividing the ability range into C intervals and grouping students into these intervals can controls the maximum individual gain below the length of the interval $(1-1/Q)/C$.

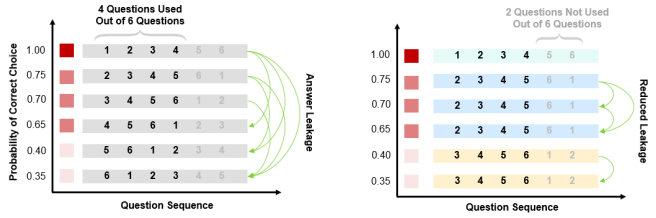


Fig. 5. A toy example of sequences of 4 questions from the pool of 6 questions to be given to 6 students for reduction of cheating gains. (a) A heuristic assignment of 4 questions (in black) to 6 students via circular shifting, yielding a maximum individual cheating gain 48.75% and an average cheating gain in the worst case $\approx 14.6\%$, and (b) an assignment follows the general grouping-based anti-cheating scheme, reducing the maximum individual cheating gain to 10% and the average cheating gain in the worst case $\approx 3\%$. Note that in (b) the students are properly grouped to decrease the inter-group cheating gain at a cost of a slightly increased intra-group cheating gain.

Alg-Gen: general grouping-based anti-cheating scheme. In light of Theorem 1, we know that for the exam design with predefined M_p and M, we are still able to control the maximum individual cheating gain of student to a level no greater than $(1-1/Q)/(M_p-M+1)$. Hence, we introduce an algorithm for this case (Algorithm 1). Note that if $M_p - M + 1 \geq N$ then the best solution will be each student to form his/her own group and there will be no cheating gain, but this case is trivial, and we do not include it in the algorithm.

Algorithm 1 Alg-Gen

Input: A DOT instance $([N], [M_p], M, Y)$**;**
Generate $S_{CS} = \{s_1, s_2, \ldots, s_{M_p}\}$ from $[M_p]$ by left circular shifting
Sort Y in the descending order
Partition $[\min(Y), \max(Y)]$ into $M_p - M + 1$ intervals $I_1, I_2, \ldots, I_{M_p-M+1}$ with equal length;
Initialize a temporal index for interval $t = 1$;
for $i = 1, \ldots, N$, **do**
  if $y_i \in I_t$ **then**
    $a_i \leftarrow s_t$
      ▷ Students inside one interval receive the same sequence
  else
    $t = t + 1$;
    $a_i \leftarrow s_t$
return A

V. Heuristic Greedy-Searching Algorithm

Although the general scheme may work well in many cases, it is usually not optimal since it does not fully take advantage of the student ability profile. In addition, in many cases what we want is a small quiz instead of a big exam, hence we are not that motivated to generate a large question bank, and besides this, we may have a huge class. In this type of cases (limited or fixed Mp), the general scheme might fail to provide what we want, and an optimization-based method is needed.

In principle the optimal assignment to achieve the minimized cheating gain should be searched from the set of all possible assignments whose size is n^N and n is the size of the pool of question sequences S_{QS}. This searching problem belongs to the discrete optimization category which is often harder than the continuous optimization problems. As the optimal assignment appears hard to compute, we propose the following efficient algorithm. We first narrow the searching pool of question sequences to the sequences generated by circular shifting (let us denote the set as S_{CS}) from S_{QS} following the heuristic that S_{CS} is a good representative subspace of S_{QS}. S_{CS} contains all possible z values achieved by any two sequences from S_{QS}, and if we randomly choose two sequences from the two space, the expected z value of two sequences from S_{CS} is even smaller than that from S_{QS} (please refer the Appendix for the proof). Then, we choose to use a greedy-searching algorithm from a randomly initialized assignment or the assignment generated with the general scheme described in section IV, and repeat the searching process for multiple times until the loss does not decrease. Through this greedy-searching, relatively good results can be easily obtained in polynomial time. By changing the initialization and iterating the searching process, many results which are better than that obtained by the general scheme are often found in our study.

The core idea of our greedy-searching algorithm is that we iterate on each student and replace his/her question sequence with one from S_{CS} if the updated assignment achieves a smaller total cheating gain. The details of one cycle greedy-searching is shown in Fig. 6, and the loss function is depicted in Eq. (5). Several cycles of greedy-searching are needed to fulfill a complete search, and the output assignment

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

from the last cycle will be treated as the initialization for the next cycle during the iteration. The flow chart of one complete search is shown in Fig. 7. Conducting multiple complete searches with different initialization and choosing the best result is another strategy to improve the solution. The formal description of the algorithm is as follows with the pseudo code.

Alg-Cyclic: Greedy Searching from Cyclic Pool. We introduce Alg-Cyclic (Algorithm 2) to greedily search better assignments from a restricted set of *circular sequences* S_{CS} generated using left circular shifting as we illustrate in Fig. 3. Algorithm 2 fixes the students in a descending order of their abilities and proceeds in two phases as follows: In Phase 1, we use the result A_0 from Algorithm 1 as our preferred initialization, and together with other reasonable/random initializations we will find the best one among the respectively optimized results. In Phase 2, a student is selected according to the fixed order, and a sequence which minimizes the total gain is selected from S_{CS} to be assigned to the student(the assignment is updated only if the update reduces current average cheating gain to ensure convergence). For any assignment A, we will use (s, a_{-i}) to denote the assignment where student i's sequence a_i is replaced with a sequence s. The steps in Phase 2 will be repeated for a maximum of N_{rep} times or until a local minima is reached.

Algorithm 2 Alg-Cyclic

Input: A DOT instance $([N], [M_p], M, Y, P, A_0)$**.**
Generate $P_{CS} = \{s_1, s_2, \ldots, s_{M_p}\}$ from $[M_p]$ by left circular shifting
$A \leftarrow A_0$. ▷ Initialization A_0
$N_{rep} = 30$. ▷ Maximum repetition time
$tA = A_0$.
for $iter = 1, \ldots, N_{rep}$, **do**
  for $i = 1, \ldots, N$, **do** ▷ Greedy improvements
    for $j = 1, \ldots, M_p$, **do**
      if $g((s_j, a_{-i}), P) < g(A, P)$ **then**
        $a_i \leftarrow s_j$
  if $tA == A$ **then**
    break ▷ Assignment does not change, and stop
  else
    $tA = A$
return A.

VI. Simulation Study

A. Experimental Setup

1) Cheating Matrix Construction: To perform the optimization, we heuristically construct a cheating matrix P depicting the probability of one student cheating from another student, based on following reasonable assumptions about cheating mechanisms: (1) The probability of student i cheating is related to his/her ability y_i (due to the gain to risk ratio); Student 1 tends not to cheat since he/she could obtain no gain (risk greater than benefit), while student N will try all means to actively cheat since he/she will always gain (benefit greater than risk); (2) The probability of cheating happens between two students A and B is related to the difference of y_A and y_B. Student i will have the strongest willingness to actively cheat with student 1, but the least willingness to actively cheat with student j if $y_i = y_j$ since he/she cannot trust j more than himself/herself, and he/she will never actively cheat with j if $y_i > y_j$.

Based on the assumptions above, the cheating matrix P is heuristically constructed as follows:

$$p_{j,i} = \begin{cases} 0, & y_j \le y_i \\ \frac{y_j - y_i}{\sum_{k=1}^{n_f(i)} (y_k - y_i)} (1 - p_{i,i}), & y_j > y_i \end{cases} \tag{8}$$

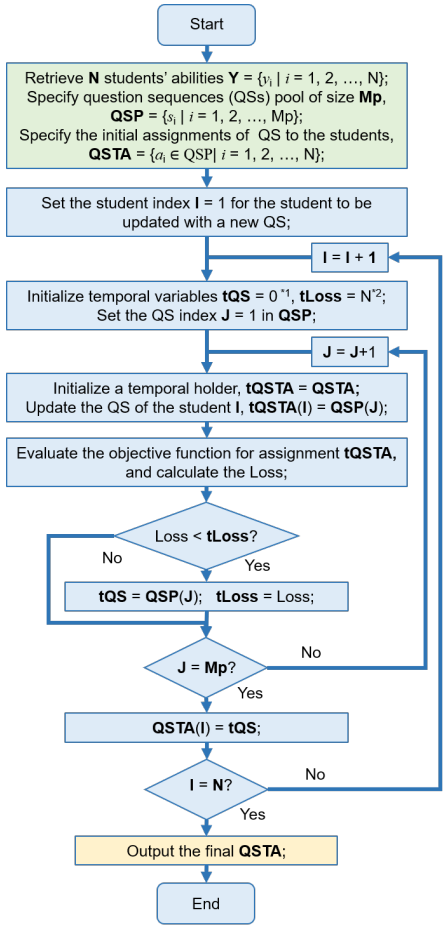


Fig. 6. Flow chart of the greedy-searching process, and the output is the question sequences assigned to students in the order of descending abilities.

$$p_{i,i} = \left[1 - \frac{\sum_{k=1}^{n_f(i)} (y_k - y_i)}{\sum_{k=1}^{N} (y_k - y_N)}\right]^{\eta} \tag{9}$$

where $n_f(i)$ is defined as the number of elements in Y that are greater than y_i, and η is a non-negative constant which can be used to adjust students' willingness to cheat. Larger η will increase the cheating probability, and students are supposed to always commit active cheating if $\eta = \infty$. Eqs. (9) and (8) define the probabilities of the cheating and non-cheating states of student i respectively, and in the cheating state, the possibility of student i will cheat from student j is proportional to their ability difference $y_j - y_i$ normalized by the sum of ability differences in all possible cases.

We further assume that students have different abilities ($y_1 > y_2 > \ldots > y_N$), without losing generality (due to the fact that adding tiny differences to two equal y negligibly affects the result of g), we simplify the expression of $n_f(i)$ as

$$n_f(i) = i - 1 \tag{10}$$

Hence, p_{ji} can be written more explicitly as follows:

$$p_{j,i} = \begin{cases} 0, & j < i \\ (1 - p_{i,i})(y_j - y_i)/(\sum_{k=1}^{i} y_k - i y_i), & j > i \\ \left[1 - \sum_{k=1}^{i} (y_k - y_i)/\sum_{k=1}^{N} (y_k - y_N)\right]^{\eta}, & j = i \end{cases} \tag{11}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

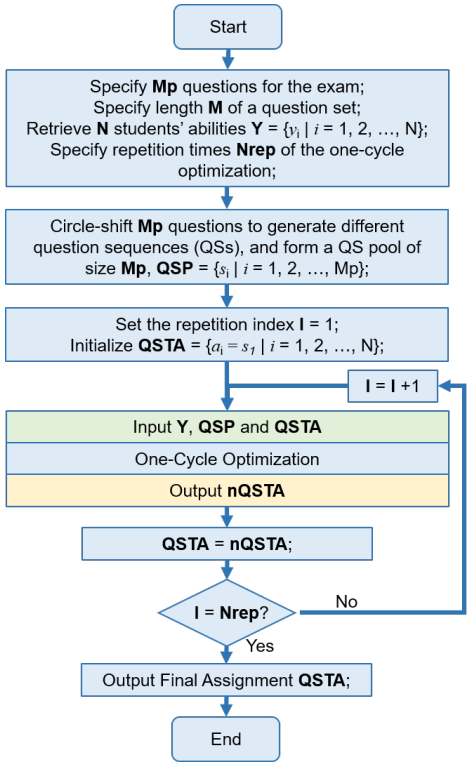


Fig. 7. Flow chart of one complete searching which iterates the one-cycle optimization multiple times, and the output is the question sequences assigned to students in the order of descending abilities. One example of initialization is provided as letting all students use the first question sequence in the cyclic sequence pool.

Note that this heuristic cheating matrix P may not exactly match real life but it is a reasonable start for optimization. In our construction, P puts a larger weight on the cheating between students with a larger ability difference than that with a small ability difference. This setting can provide effective constrains on the worst case cheating gain while limiting the total cheating gain (as shown in Fig. 8 and will be explained later in the subsection VI-B). Since mismatches are very likely to exist between the model and the practice, the worst-case analysis needs to be performed on the optimized result. If the cheating gain calculated in the worst situation for the output assignment is not acceptable, the result should be used with caution or just use different initializations to generate diverse solutions and pick the best one. P can also be constructed dynamically to achieve the optimization with the objective of minimizing the maximum individual cheating gain, which will be explained in the subsection VI-B.

2) Other settings: The simulations were conducted in the setting of 80 students ($N = 80$), 40 MCQs for each student ($M = 40$), 60 MCQs in the pool ($M_p = 60$), and 4 choices for each MCQ ($Q = 4$) with only one correct answer. The Y profile was randomly generated from the Gaussian distribution with $\mu_0 = (1 + 1/Q)/2$ and $\sigma_0 = (1 - 1/Q)/6$, truncated to be meaningful $[1/Q, 1]$. The elements in Y were ranked in the descending order, and then P was calculated from Y according to Eq. (11) with η set to infinity. We repeated these steps 500 times, and each time a new Y profile was randomly generated.

Totally four exam scenarios of interest have been tested: (1) The conventional scenario without cheating prevention (anti-cheating level: **None**) as a control group; (2) The scenario without student ability prior (anti-cheating level: **Bind**) and random question sequences (sequences can be reused) from the cyclic pool S_{CS} randomly assigned to the students (Situations without prior knowledge of student ability are also common in real life, such as standard ability testing exams like TOEFL, GRE, and etc. This blind anti-cheating scheme also serves as a control group); (3) The scenario with students' performance prior and using the grouping-based anti-cheating scheme described in Section IV as one of our experimental group (anti-cheating level: **General**, with algorithm 1); and (4) the scenario with student ability prior and with optimized assignment obtained via our heuristic greedy-searching is the other experimental group (anti-cheating level: **Optimized** with algorithm 2), the repetition times for one complete searching was set as 30, and the result of the general scheme was used as the initialization. The cheating gain of the conventional scenario was calculated by Eq. (7), while those of other three scenarios were calculated by Eq. (5). Note that, the calculated total cheating gain for conventional scenario should be an underestimated value since students would all tend to cheat from the student with highest ability given they have the same problem sequence rather than following our assumed distributed probabilities of cheating with other students.

TABLE I
THE CHEAT GAINS UNDER THE SCENARIOS EQUIPPED WITH DIFFERENT LEVEL ANTI-CHEATING STRATEGIES (BOLD INDICATES SIGNIFICANTLY BETTER THAN OTHERS)

Anti-cheating level		None	Blind	General	Optimized
g	mean	13.7466	4.6870	0.0316	**0.0107**
	std.	1.4733	0.5226	0.0139	0.0026
g_W	mean	25.7075	17.3970	1.0672	**0.8430**
	std.	3.8631	2.1465	0.0932	0.0984
g_{MI}	mean	0.6027	0.5000	0.0334	0.0318
	std.	0.0645	0.0637	0.0013	0.0047

B. Simulation Evaluation

The total cheating gains g of the 500 repetitions under four scenarios were recorded, and the mean and standard deviation were calculated and shown in Table I. In the table, four different level anti-cheating strategies (**none**, **blind**, **general scheme** with algorithm 1, **optimized assignment** with algorithm 2) correspond to the four aforementioned scenarios described. Looking at the rows of cheating gain g, we can observe that the scenario with optimized assignment performs the best among others, and it surprisingly reduced the mean to around only one third of that of the general scheme and also has much smaller standard deviation, which demonstrates the effectiveness of our heuristic optimization. Comparing the results of the other three, the general grouping-based anti-cheating scheme majorly outperforms the two control groups in orders of magnitude, evidencing the high-efficiency of the proposed grouping-based anti-cheating scheme in suppressing the cheating gain. The two control groups perform in a similar magnitude, but huge improvement can be found in the blind anti-cheating scenario which suggests that the cyclic question sequences could help reduce the cheating gain even without prior knowledge of student ability profile.

Similar to the average case analysis and worst-case analysis in computer science, we may want to revisit our results in a worst-case study. Hence, another two important metrics are introduced to assess the optimization results from the risk control angle, i.e., the worst case overall cheating gain g_W defined as the overall cheating gain in the situation where all students manage to achieve their maximum possible cheating gain (the maximum possible cheating gain of the student i is achieved by setting the probability of i cheats with the

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

student j to 1, from whom i will obtain the maximum gain among other choices of i),

$$g_W(A) = \frac{1}{M}\sum_{i=1}^{N}\max_{j\in[N]}\{Z(A)\circ D\} \tag{12}$$

and the maximum individual cheating gain g_{MI} which is the maximum of the maximum possible cheating gains for all students,

$$g_{MI}(A) = \frac{1}{M}\max_{i,j\in[N]}\{Z(A)\circ D\}. \tag{13}$$

g_W can be used to assess the performance of the optimized results under the worst situation and can be treated as a reliable upper limit estimation of the cheating gain under the given ability profile Y since the calculation of g_W does not involve the cheating matrix. g_{MI} is a metric can be used to estimate the fairness of the exam from the aspect of the maximum cheating gain any student can achieve.

The two metrics g_W and g_{MI} were also evaluated in our simulation, and the results are also shown in Table I for comparison. From the aspect of the worst overall cheating gain g_W, the result of blind anti-cheating scheme does not seem that impressive as that in the aspect of g although it indeed improves. In addition, the results from two control groups seem quite similar in terms of the maximum individual cheating gain g_{MI}. The two observations indicate that robustness is weak of the blind anti-cheating strategy, and the fairness cannot be guaranteed. The two experimental groups still outperform the control groups by an order of magnitude in terms of both g_W and g_{MI}, which suggests the strong robustness of two anti-cheating strategies. Between the two experimental groups, the scheme with optimized assignment yields significantly a smaller mean value with a similar standard deviation in terms of g_W which suggests that the optimized assignment can provide better results than the general grouping-based anti-cheating scheme in a stable manner. By g_{MI}, the two experimental groups show comparable results, and are only around 3% which implies that the maximum gain any student can obtained through cheating has been limited to quite a small level, hence the fairness of the exam can be well ensured using these two strategies.

With these two metrics at hand, a natural question is what if we directly optimize with these two metrics which do not rely on the cheating matrix at all? With these two metrics to measure the loss, in the optimization we actually assume a cheating student knows the order of other students and chooses the most profitable student to copy, and the cheating matrix only contains 0 and 1 and will dynamically change over the assignment. We also simulated this situation for 500 hundred times in the same settings with the same Y profile in each repetition as that used in the last experiment. The results are shown in Fig. 8. An interesting phenomenon is that the optimized results of minimizing g_W do not perform the best among others in terms of g_W. Actually, the optimized results of minimizing g perform the best in terms of both g and g_W (slightly better than others in g_W and majorly better in g), at the cost of slightly dispersed g_{MI}. The general scheme results and optimized results of minimizing g_{MI} and g_W are comparable in all three metrics. The results of general scheme and optimized with g_W are almost the same, suggesting that the latter almost does not change with the initialization during optimization, and are slightly better than the results of optimized with g_{MI} in terms of g and g_W. Thus, the general scheme and the optimization with g are recommended for controlling the cheating gain in the exam with the help of the heuristically defined cheating matrix, which can achieve relatively good results but with less efforts.

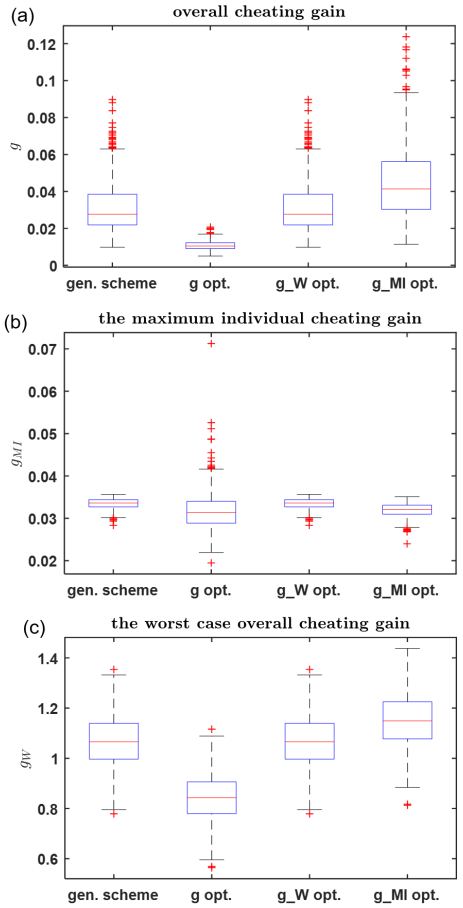


Fig. 8. Comparison between the general grouping-based scheme and greedy optimizations with respect to different objectives (the overall cheating gain g, the worst case overall cheating gain g_W, and the maximum individual cheating gain g_{MI}. (a) Overall cheating gain, (b) the maximum individual cheating gain, and (c) the worst case overall cheating gain. The settings are $N = 85, M_p = 60, M = 40$, and $Q = 4$. The number of instances is 500. The central red mark indicates the median, and the bottom and top edges of the box indicate the 25th and 75th percentiles, respectively. The whiskers extend to the most extreme data points not considered outliers, and the outliers are plotted individually using the '+' symbol.

VII. Optimization with State-of-the-Art Methods

As mentioned in Sec. V, the assignment optimization problem is in the category of discrete optimization. A number of standard algorithms have been developed in the field. Thus, it is of great interest to compare the performance of some state-of-the-art methods and our heuristic optimization method. Two representative methods as our benchmarks are the minimum weight maximum matching method (min-max matching method) and the ILP method.

Instead of searching in the cyclic pool S_{CS}, the two standard discrete optimization methods will search from the entire permutation pool S_{QS} and theoretically find the optimal assignment for a minimized cheating gain. Due to hardness of searching in a huge space when the problem scale is large, we implemented the ILP in the traditional form to find the global minimal and adapted the min-max matching method in the greedy searching form to work in polynomial time.

A. Min-Max Greedy Matching

Alg-mmGM: Min-Max Greedy Matching algorithm: A natural approach is to start with an initial random assignment and improve

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

it greedily by picking up one student at a time according to a certain order, and refining his/her question sequence that the total gain is minimized from the set of all possible M-permutations of M_p. We propose ALG-MMGM to greedily improve an assignment, and show that computing a sequence to replace a single student's sequence in an assignment that minimizes the total gain can be done in polynomial time by performing a minimum weight maximum matching, as we show in Algorithm 3. For convenience, we first introduce some notations. Given any $s \in S_{QS}$:

1) For each $j \in [M_p]$, we define $s(j) = l$ if j appears in the l-th position in s, and $s(j) = 0$ otherwise;
2) For each $j \in M_p$, $\alpha(s, j) = 1$ if $s(j) \geq 1$, and $\alpha(s, j) = 0$ otherwise, to indicate whether question j is on sequence s;
3) For each $j \in M_p$, each $l \leq M$, $\beta(s, j, l) = 1$ if $s(j) \geq 1$, $s(j) \leq l$, and $\beta(s, j, l) = 0$ otherwise, to indicate whether question j appears at or before position l on sequence s;
4) For each $j \in M_p$, each $l \leq M$, $\gamma(s, j, l) = 1$ if $s(j) \geq l$, and $\gamma(s, j, l) = 0$ otherwise, to indicate whether question j appears at or after position l on sequence s;
5) For any $s, s\prime \in S_{QS}$, and any $j \in M_p$, $\delta(s, s\prime, j) = 1$ if $s(j) > 1$, $s\prime(j) > 1$, and $s\prime(j) \leq s(j)$, and $\delta(s, s\prime, j) = 0$ otherwise to indicate whether a student assigned s can cheat on question j from a student assigned $s\prime$.

Algorithm 3 ALG-MMGM

1: **Input: A DOT instance** $([N], [M_p], [M], Y)$**;**
2: $A \leftarrow$ a random assignment;
3: **for** $i \leq N$, **do**
4: $G \leftarrow ([M] \cup M_p, E = \{(l, j) : l \leq M, j \leq M_p\})$; ▷ bipartite
5: **for** $l \leq M, j \leq M_p$, **do** ▷ Edge weights are the marginal gain over A by placing j as i's the l-th question;
6: $w_{(l,j)} \leftarrow 0$;
7: **for** $k \leq i$, **do**
8: $w_{(l,j)} \leftarrow w_{(l,j)} + p_{k,i}\big[[y_k\beta(a_k, j, l) + y_i(1 - \beta(a_k, j, l))] - [y_k\delta(a_i, a_k, j) + y_i(1 - \delta(a_i, a_k, j))]\big]$
9: **for** $h > i$, **do**
10: $w_{(l,j)} \leftarrow w_{(l,j)} + \sum_{h>i} p_{i,h}\big[[y_i\gamma(a_i, j, l) + y_h(1 - \gamma(a_i, j, l))] - [y_i\delta(a_h, a_i, j) + y_i(1 - \delta(a_h, a_i, j))]\big]$
11: $Matching \leftarrow$ minimum weight maximum matching on G;
12: For each $(l, j) \in Matching$, $a_i(l) \leftarrow j$.
13: **return** A

B. Min-Max Greedy Matching with Cyclic Initialization

Alg-mmGM-Cyclic: Alg-mmGM initialized by Alg-Cyclic. We define Alg-mmGM-Cyclic as an extension of Alg-mmGM and Alg-Cyclic by modifying line 2 in Alg-mmGM (Algorithm 3) by setting the initial assignment to the output of Alg-Cyclic and improving it greedily in the same manner as in Algorithm 3. This ensures that we will improve solutions from the Alg-Cyclic (at least no worse than), which implies a potential improvement of our heuristic optimization method Alg-Cyclic.

C. ILP

We cast this problem into an integer linear programming problem to find an optimal assignment in the permutation space, as shown below in Algorithm 4.

D. Comparative Analysis

To compare the relative performance of our algorithms in terms of the expected total gain from cheating, we evaluated the algorithms on multiple synthetic datasets with different settings corresponding to the choice of values for N, M_p, and M, over 100 instances for each setting. Different from the settings in the Section VI, we increased randomness to the cheating matrix in this experiment for a better evaluation of the relative performance of different algorithms. For each setting, we draw the abilities of N students i.i.d. uniformly from the $[0.25, 1)$ interval this time, we drew the cheating probabilities for each student $i \in \{2, \ldots, N\}$ to cheat from the students $k < i$ ($i - 1$ in total) from the $i - 1$-variate $Dirichlet$ distribution with a concentration parameter of $\alpha = 10$, meaning that $\sum_{k<i} p_{k,i} = 1$ and $p_{1,1} = 1$ in our experiments.

In addition to random assignments, and assignments obtained directly by circular shifting, we considered the four algorithms: **Alg-mmGM** (Algorithm 3), **Alg-Cyclic** (Algorithm 2), **Alg-mmGM-Cyclic**, and **ILP** (Algorithm 4). It is worth noting that Alg-Cyclic was first initialized with the assignment generated from Alg-Gen which provides a proven upper bound of the cheating gain, and then was randomly initialized from the cyclic pool S_{CS} for 9 times, and the best result was selected for comparison and the initialization of Alg-mmGM-Cyclic. Experiments were performed on a computer equipped with a AMD Ryzen 7 2700X processor running at 4.0GHz and 16GB memory. Due to practical considerations of running time and system memory, we evaluated against the ILP on instances with at most $N = 10$, $M_p = 5$, and $M = 3$. We further evaluated the relative performance of our greedy heuristic algorithms on larger instances with $N = 100$, $M_p = 30$, and different values of $M \in \{10, 20\}$.

Figure 9 presents our experimental results. For each of the four algorithms and one additionally random method (from the permutation pool) (Y-axis), Figure 9 shows the gain over the honest score (X-axis) computed using Equation 5 over 100 instances, with a box representing the upper and lower quartiles, an orange line within the box representing the median, a green arrowhead representing the mean, and whiskers extending from the box on either end representing the range of values observed. Statistic outliers are plotted individually using the 'o' symbol.

Comparing different algorithms, we observe immediately from Figure 9 that every one of our greedy algorithms displays significantly lower gain than the random assignments on average demonstrating the usefulness of our anti-cheating schemes in reducing the cheating gains. Second, the low total gain exhibited by the optimal solution computed by ILP on average validates our approach for minimizing the gain from cheating. Third, Alg-mmGM-Cyclic and Alg-Cyclic approximated the minimum gain computed by ILP, highlighting the effectiveness of the greedy algorithms in practice.

Comparing the optimized solutions computed under different settings, it is easy to find that optimized total cheating gain (with the optimal ILP algorithm for example) does not necessarily correlate to the permutation space size; e.g., (5, 3, 2)[1] and (5, 3, 3) have the permutation space of the same size but different optimal gains; from (10, 3, 2) to (10, 5, 3), the size of the permutation space increased but the optimal gains decreased; and from (10, 5, 3) to (10, 5, 5), the size of the permutation space increased but the optimal gains increased. However, the optimized cheating gains do correlate to the difference between $M_p - M$ (which has been implied by Theorem 1); i.e., an increased $M_p - M$ value significantly reduces the cheating gain as demonstrated by (5, 3 ,3) versus (5, 3, 2), as well as (10, 5, 5) and (10, 3, 2) versus (10, 5, 3). In addition, increasing the number of students can also raise the cheating gain as shown in the case of (5, 3, 2) versus (10, 3, 2).

[1]* $N = 5, M_p = 3, M = 2$ is noted as (5, 3, 2) for brevity.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

Algorithm 4 ILP

1: **Input: A DOT instance** $([N], [M_p], M, Y)$**.**
2: Solve the ILP in Fig. S4 below and set assignment A as follows: for each $i \in [N], j \in [M_p]$, such that $s_{i,j} > 0$, $a_i(j) \leftarrow s_{i,j}$;
3: **return** A.

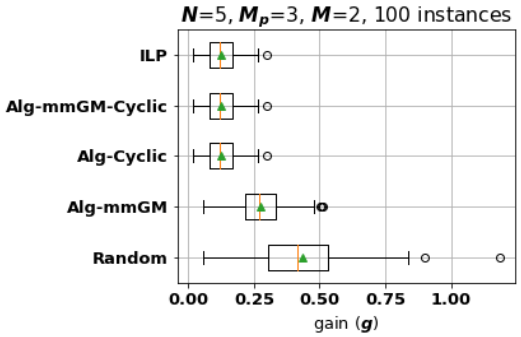
$\min_z \sum_{i\in[N],j\in M_p} s_{i,j}$,		objective
Variables		
$q_{i,j} \in [0,1]$,	$\forall i \in [N], j \in [M_p]$	expected scores
$s_{i,j} \in \{0,1,\ldots,M\}$,	$\forall i \in [N], j \in [M_p]$	rank of question
$x_{i,j} \in \{0,1\}$,	$\forall i \in [N], j \in [M_p]$	indicate assignment
$e_{i,j,l} \in \{0,1\}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	indicate $s_{i,j} \geq l$
$f_{i,j,l} \in \{0,1\}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	indicate $s_{i,j} \leq l$
$m_{i,j,l} \in \{0,1\}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	indicate $s_{i,j} = l$
$c_{i,k,j} \in \{0,1\}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	indicate i can copy k on j
$u_{i,k,j} \in \{0,1\}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	indicate both i,k assigned j
$v_{i,k,j} \in \{0,1\}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	indicate k sees j before i
Constraints		
$\sum_{j\in[M_p]} s_{i,j} = \sum_{l\in\{1,\ldots,M\}} l$,	$\forall i \in [N]$	feasible assignment
$x_{i,j} \geq \frac{s_{i,j}}{M_p}$,	$\forall i \in [N], j \in [M_p]$	feasible assignment
$x_{i,j} \leq s_{i,j}$,	$\forall i \in [N], j \in [M_p]$	feasible assignment
$e_{i,j,l} \geq \frac{1+s_{i,j}-l}{M_p+1}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$e_{i,j,l} \leq \frac{M_p+s_{i,j}-l}{M_p}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$f_{i,j,l} \geq \frac{1+l-s_{i,j}}{M_p+1}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$f_{i,j,l} \leq \frac{M_p+l-s_{i,j}}{M_p}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$m_{i,j,l} \leq e_{i,j,l}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$m_{i,j,l} \leq f_{i,j,l}$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$m_{i,j,l} \geq e_{i,j,l} + f_{i,j,l} - 1$,	$\forall i \in [N], j \in [M_p], l \in [M]$	feasible assignment
$\sum_{j\in[M_p]} m_{i,j,l} = 1$,	$\forall i \in [N], l \in \{1,\ldots,M\}$	feasible assignment
$u_{i,k,j} \leq x_{i,j}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	both assigned j
$u_{i,k,j} \leq x_{k,j}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	both assigned j
$u_{i,k,j} \geq x_{i,j} + x_{k,j} - 1$,	$\forall i \in [N], k \in [N], j \in [M_p]$	both assigned j
$v_{i,k,j} \geq \frac{1+s_{i,j}-s_{k,j}}{M_p+1}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	k has seen j
$v_{i,k,j} \leq \frac{M_p+s_{i,j}-s_{k,j}}{M_p}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	k has seen j
$c_{i,k,j} \leq u_{i,k,j}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	i can copy k on j
$c_{i,k,j} \leq v_{i,k,j}$,	$\forall i \in [N], k \in [N], j \in [M_p]$	i can copy k on j
$c_{i,k,j} \geq u_{i,k,j} + v_{i,k,j} - 1$,	$\forall i \in [N], k \in [N], j \in [M_p]$	i can copy k on j
$q_{i,j} = \sum_{k\in[N]} y_k p_{k,i} c_{i,k,j} + y_i(1 - \sum_{k\in[N]} p_{k,i} c_{i,k,j})$,	$\forall i \in [N], j \in M_p$	expected score

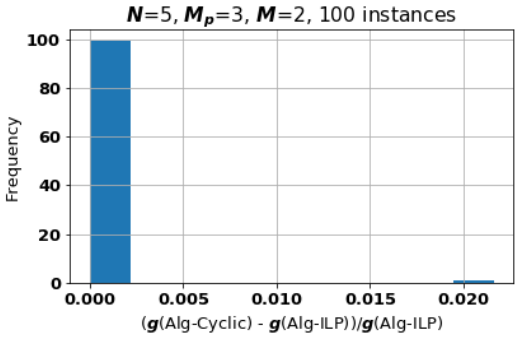
For the settings of (10, 3, 2), (10, 5, 3) and (10, 5, 5), besides the observations mentioned above, larger discrepancies between the Alg-Cyclic and Alg-mmGM in terms of cheating gains appear as the optimal cheating gains reduce, suggesting the efficiency of our heuristic greedy approach (Alg-Cyclic) in reducing the cheating gains even searching in the size-limited cyclic pool, while Alg-mmGM could potentially provide a better solution but its real performance is limited in practice due to the local minima distributed in large permutation space.

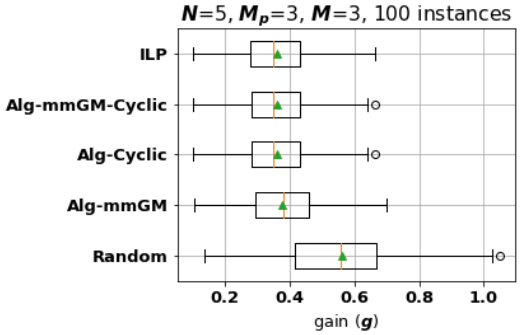
For better understanding of the relative performance, the distribution of the paired case-wise difference between the results of Alg-Cyclic, Alg-mmGM-Cyclic and ILP were compared. Interestingly, we find that the results differences between Alg-Cyclic and Alg-mmGM-Cyclic are all zeros for all 100 cases and under all these settings (including the settings of $N = 100$ described in Fig.11). These results suggest that the local minima reached by Alg-Cyclic could also be a local minima of the Alg-mmGM, which means searching beyond the result from Alg-Cyclic with Alg-mmGM did provide further improvement. The differences between Alg-Cyclic (and Alg-mmGM-Cyclic) and ILP are summarized as Fig. 10, in which over 95% cases of Alg-Cyclic reached the optimal solutions calculated with ILP for small permutation spaces (first three subfigures), and over 65% cases reached the optimal solutions when the permutation space increased by or over 10 times (the last two subfigures) while the maximum mismatches were smaller than 35%. The results suggest that our heuristic method would not be able to achieve the optimal solutions especially in a large-scale case, but the maximum mismatch could be very small ($< 35\%$ of the optimal result). Together with the hardly-distinguishable differences between the distributions of the results of Alg-Cyclic and ILP shown in Fig. 9, it demonstrates that our heuristic approach could provide a practically good enough solution, which is close to the optimal at a small computational cost.

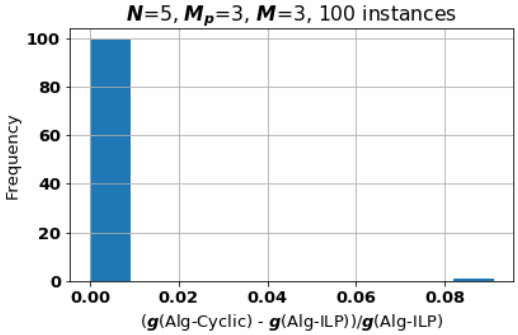
For each of the settings presented in Figure 9, the ILP solution has a very low total gain on average, validating our approach of suppressing cheating by an optimal assignment without proctoring. It appears that Alg-mmGM initialized by a random assignment is sensitive to the initial assignment, and is prone to being trapped at locally optimal solutions. On the other hand, although Alg-Cyclic is limited by the reduced search space to the set of circularly

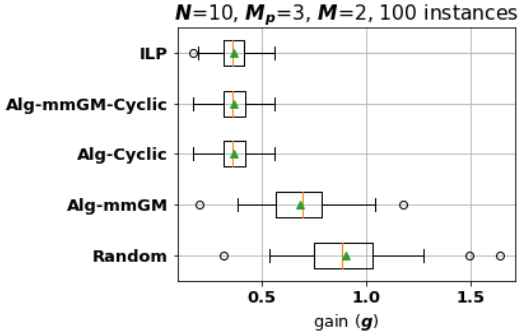
Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

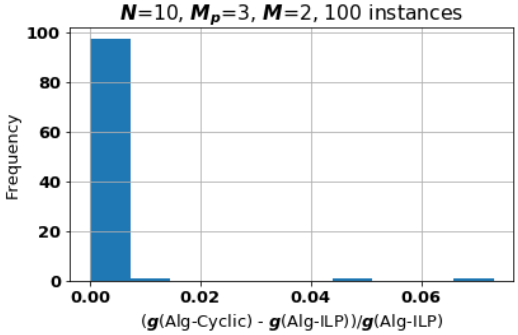


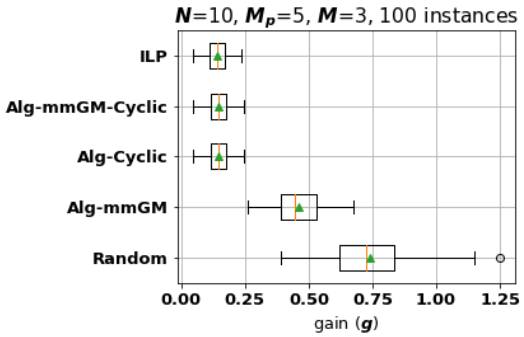


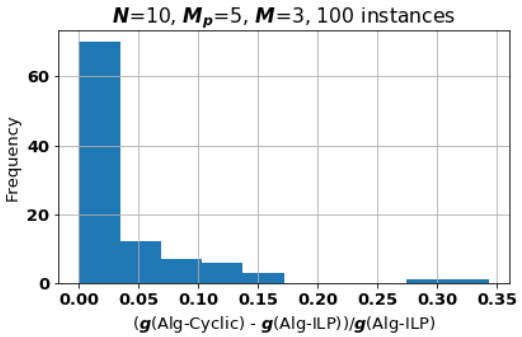


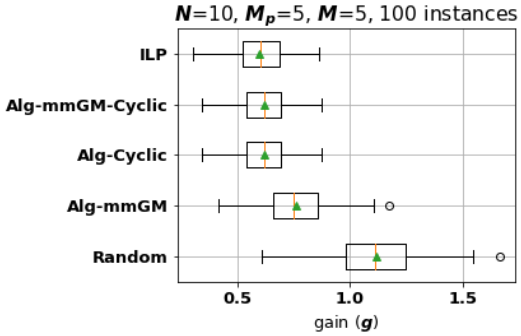












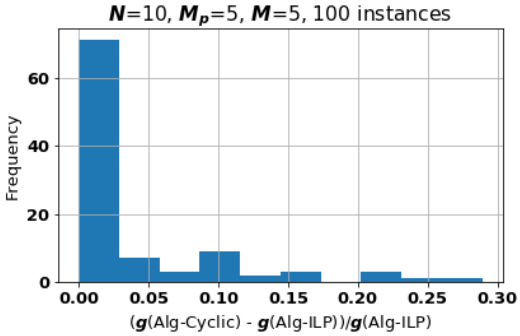


Fig. 9. Comparison of the greedy algorithms with the ILP method in terms of the overall cheating gains

Fig. 10. Distribution of the paired case-wise differences between results of Alg-Cyclic and ILP.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

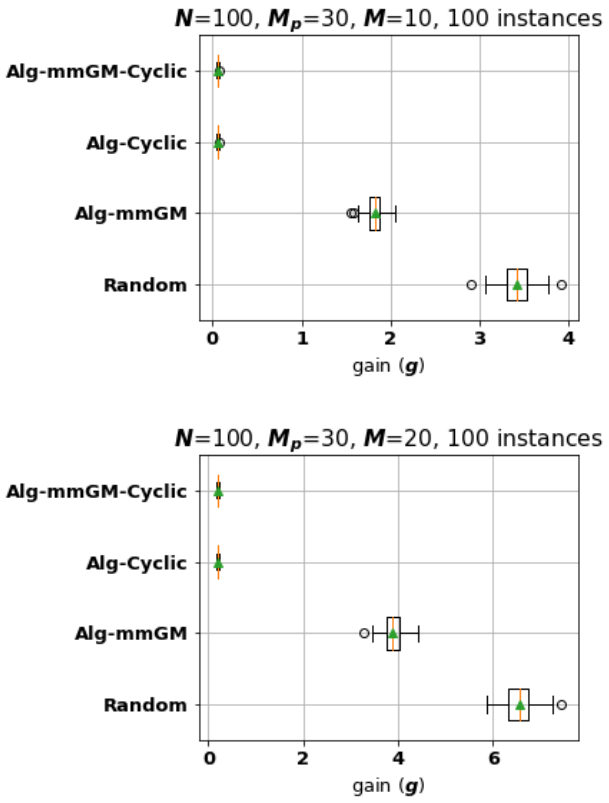


Fig. 11. Demonstration of the total gain of greedy algorithms on large-scale instances.

shifted sequences, the circular assignments can be close to the globally optimal solutions, and interestingly Alg-mmGM-Cyclic did not provide further improvement over Alg-Cyclic in our experiments.

Our experiments offer two key takeaways: For a given number of students N: (1) The minimum total cheating gain that can be obtained by our approach (e.g., the ILP solution) heavily relies on the value of $M_p - M$ which is essential to provide sufficiently different sets of questions for the assignment to minimize the cheating gain (controlled by either increasing the size M_p of the pool of questions or decreasing the number of questions M. (2) Greedy algorithms significantly outperform random assignments, and our heuristic greedy method Alg-Cyclic and the adapted Alg-mmGM-Cyclic provide practically good enough solutions at a small computational cost compared to the ILP method and may even approach the minimum total gain in the case of a small permutation space for question assignment.

In Figure 11, we demonstrate our greedy algorithms on relatively large instances with $N = 100$ students, a pool of $M_p = 30$ questions, and between $M = 10$ and 20 questions per student. Our experiments demonstrate again that (1) our greedy algorithms significantly outperformed random schemes, (2) Alg-Cyclic (and Alg-mmGM-Cyclic) significantly outperformed the Alg-mmGM algorithm initialized with random assignments, and (3) the total cheating gains were controlled under a very small level even with a large number of students.

VIII. Discussions

The proposed grouping-based anti-cheating scheme works very well in terms of minimizing the maximum individual cheating gain. To be mentioned, this does not rely on an accurate student ability profiling. It can work well even with only the rank of the students' abilities, and control the cheating gain to a low level, as shown in the proof of Theorem 1. In principle, it should be robust even if there are noises in the ability data. This can be readily understood that small noises will only make few students across the interval boundaries. The down-dropping student (DDS) could increase the maximum individual cheating gain g_{MI} since other students in the augmented group can cheat from the DDS, and the increment in g_{MI} will be no larger than the noise magnitude. Clearly, this will increase the worst case g_W due to the fact that all students in the same group will gain benefits. On the other hand, students in the upper group but with lower abilities than the DDS can potentially cheat with the DDS which creates the inter-group cheating gains but this cheating gain is negligible in terms of g_W and g_{MI} since this inter-group gain should be much smaller than their maximum intra-group gains. As for the Up-floating case, only the up-floating student (UFS) will obtain an increased cheating gain through intra-group cheating but again the increment in g_{MI} should be smaller than the noise magnitude. The UFS will also benefit from inter-group cheating but the gain is much smaller than his/her intra-group cheating gain. The increment in g_{MI} will be smaller than the magnitude of noise. Thus, the grouping-based anti-cheating scheme should be robust against noise in students' abilities. In addition to MCQs the general grouping-based anti-cheating scheme can deal with short-answer questions, too. More generally, the questions can have different time-lengths based on its difficulty, weights or credits.

Beyond the general grouping-based anti-cheating scheme, our heuristic greedy optimization algorithm is to search for a better assignment of questions in the cyclic pool. Besides using the assignment from the grouping-based anti-cheating scheme as the initial assignment, a random initialization from the circularly shifting pool can be used as well, and we can then pick up the best among the results associated with various initializations, getting closer to the optimal solution.

It is worth noting that besides the anti-cheating feature, the nature of circularly shifted sequences ensures that every student can receive the same number of questions on the same topic. This is easily achieved in the question bank.

A soft-window strategy may be designed to make this scheme and the traditional online test more similar so that students feel more comfortable. Technically, we can group three (or other numbers) questions together to make them a block, inside each block students can go back and forth to solve the questions in the block. Hence, students can work at their own pace to finish the exam while the anti-cheating function is not significantly compromised. Students with high chances to cheat between each other (have large ability differences) are typically assigned with different sequences of questions. Based on a significant accuracy discrepancy between the portion of questions that can be copied from other students and the other portion a student has to rely on himself/herself, together with the correlated incorrect answer patterns with others, the cheating behavior can be detected via data analysis, and potentially ultilized for better quality control of the online exam. With MCQ-based exams, this detection can be done in real-time. As long as a suspicious cheating signal is detected, the system could react in proper ways such as dynamically scheduling sequences of questions..

Our anti-cheating scheme relies on the prior knowledge of students' abilities. However, such prior information is not always available; e.g., for TOFLE and GRE. In this situation, using circularly shifted sequences can reduce the cheating gain but it is not nearly optimal as suggested by the simulation results in Table I. To address this problem, one solution is to dynamically make assignment of questions during the exam. A rough estimation of students' abilities can be obtained based on the accuracy in his/her first few questions, and then this prior can be in turn used for assignment optimization.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

More adaptively, we can continually improve the estimation of the students abilities during the test, and hence do an increasingly better assignment job. With this dynamic adjusting strategy, combined with real-time cheating detection functionalities, the suspicious cheating behaviors can be handled (such as by adjusting his/her question sequence timely, or optionally to increase his/her length of the exam with questions different from those students who might have offered help in cheating).

IX. Conclusion

In conclusion, we have proposed a grouping-based anti-cheating scheme for online exams, and theoretically proved its power to control the maximum cheating gain of all students to any desired low level regardless of the class size. Also, we have developed a heuristic greedy optimization method which can efficiently search for an better assignment of questions to achieve a practically insignificant cheating gain. Our experiments on exams of a small scale suggest that our heuristic results are close to the optimal solutions found by two standard discrete optimization methods (ILP and min-max matching). Thus, our proposed anti-cheating technology could potentially be a low-cost and privacy-preserving solution to the cheating problem in online exams during social distancing. Our results can further serve as the initial guess for the advanced discrete optimization algorithms to produce superior results. Further work is actively under way.

X. Appendix: Randomly sampling from S_{QS} or S_{CS}?

As mentioned in the Section VI, in the scenario without prior knowledge of students' competences, we prefer randomly assigning the students with random question sequences from the cyclic pool P_{CS} rather than from the permutation pool P_{SQ}. In this section, we calculate the expected cheating gain under randomly sampling from S_{CS} and random sampling from S_{QS}, and prove that the former is more desirable (i.e. smaller expected cheating gain) than the latter.

Let us define an operation $EZ(\cdot)$ on a question sequence (QS) pool S which calculates the expectation of $Fz(s_1, s_2)$ where s_1 and s_2 are two randomly selected elements from S. Actually, $EZ(S) = \text{mean}\{Z(S)\}$, where Z(S) is the positional matrix of S with diagonals set to the length of a sequence element from S, if replacement is allowed; otherwise, $EZ(S)$ equals to the mean of non-diagonal elements of $Z(S)$.

Theorem 2. *If we randomly sample m different sequences from a QS pool S and form $V = \{v_i \in S|, i = 1, 2, \ldots, m\}$ where $m \geq 2$ and is smaller than the size of S, and then $EZ(V) = EZ(S)$.*

Proof. If we first randomly select two sequences v_1 and v_2 from S, by definition, we have

$$E(Fz(s_1, s_2)) = E(Fz(s_2, s_1)) = EZ(S). \tag{14}$$

Then we can continue to randomly select the rest sequences from the rest of S, but this does not effect the result we already have shown in Eq. (14). On the other hand, these two steps can be taken as one step that we randomly select m different sequences from S, hence, we cannot differentiate the m sequences from each other. In that sense, the expectation of the Fz number between any two of them should be the same. Combining with Eq. (14), we have

$$E(Fz(s_i, s_j)) = E(Fz(s_j, s_i)) = EZ(S), \tag{15}$$

for $i, j = 1, 2, \ldots, m$, and $i \neq j$. Hence, the mean of the non-diagonal elements of the positional matrix of V equals to $EZ(S)$, and the theorem has been proved. ■

Theorem 3. *If we randomly sample m sequences from a QS pool S with replacement and form $V = \{v_i \in S|, i = 1, 2, \ldots, m\}$ where $m \geq 2$ and is smaller than the size of S, and then $EZ(V) = EZ(S)$.*

Proof. Follow the same idea of the proof of Theorem 2, but in one by one manner. ■

Theorem 4. *For any two M-length sequences s_1 and s_2 composed from the same M questions, $Fz(s_1, s_2) + Fz(s_2, s_1) = M + n_{spos}$ where n_{spos} equals to the number of the questions appearing at the the same positions in two sequences.*

Proof. Let us denote the M questions as $[M] = \{1, 2, \ldots, M\}$. For the case that a question $i \in [M]$ appears at the same positions in s_1 and s_2, i contribute 1 to both $Fz(s_1, s_2)$ and $Fz(s_1, s_2)$. For the other case that a question $i \in [M]$ does not appears at the same positions in s_1 and s_2, i contribute 1 to either $Fz(s_1, s_2)$ or $Fz(s_1, s_2)$. There are M different questions in the sequences, hence, $Fz(s_1, s_2) + Fz(s_2, s_1) = M + n_{spos}$. ■

Theorem 5. *For any two M-length sequences s_1 and s_2 share n_{com} common questions, $Fz(s_1, s_2) + Fz(s_2, s_1) = n_{com} + n_{spos}$ where n_{spos} equals to the number of the questions appearing at the the same positions in two sequences.*

Proof. For unique questions only in s_1 or s_2, they do not contribute to neither $Fz(s_1, s_2)$ or $Fz(s_1, s_2)$. For the other two cases, we have the same conclusions as Theorem 4. There are only n_{com} questions contribute to $Fz(s_1, s_2)+Fz(s_2, s_1)$, hence, $Fz(s_1, s_2)+Fz(s_2, s_1) = n_{com} + n_{spos}$.

Now let us look at our problem, the notations remain the same as that in Section II. The expectation of cheating gain without prior knowledge of students' abilities is actually the mean of the off-diagonal elements of the positional matrix of the assignment A, since all students are taken as identical and all cheating cases share the same weight. Theorems 2 and 3 tell us that the $EZ(\cdot)$ value of an assignments A generated by randomly sampling from a pool S should equal to $EZ(S)$, no matter of replacement is allowed or not. Hence, we have

$$E(g(A)) = EZ(S)N \tag{16}$$

■

A. circle-shifting pool

Now let us look at the cyclic sequence pool S_{CS} first, there are no questions that appear at the same positions in any two sequences due to the nature of circular shifting.

1) $M = M_p$: For the case replacement is allowed, we have

$$EZ(S_{CS}) = \frac{1}{M_p^2} \sum_{i,j=1}^{M_p} Z_{i,j}(S_{CS}) \tag{17}$$

where $Z(S_{CS})$ is the positional matrix of S_{CS} and $Z_{i,i}(S_{CS}) = M$ for $i = 1, 2, \ldots, M_p$. Based on Theorem 4, it is easy to know that

$$Z_{i,j}(S_{CS}) + Z_{j,i}(S_{CS}) = M, \tag{18}$$

holds for the off-diagonal elements. Hence, it is easy to obtain

$$EZ(S_{CS}) = \frac{1}{M_p^2}\left(\frac{M_p(M_p-1)}{2}M + M_p M\right) = \frac{M+1}{2} \tag{19}$$

Similarly, for the case replacement is not allowed, we have

$$EZ(S_{CS}) = \frac{1}{M_p(M_p-1)}\left[\sum_{i,j=1}^{M_p} Z_{i,j}(S_{CS}) - M_p M\right] = \frac{M}{2} \tag{20}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

2) $M < M_p$: The definition of $EZ(\cdot)$ reminds us that the mean of the positional matrix of a sequence pool can be calculated by the expectation of Fz number between two randomly sampled sequences from the pool. Without the loss of generality, we can assume the first sequence is $s_{ref} = [1, 2, \ldots, M]$ for convenience, because for other cases, i.e., $[k_1, k_2, \ldots, k_M]$, we can relabel tag 1 to k_1, 2 to k_2, ..., M to k_M. To be noted, by re-indexing, we have not changed the questions themselves.

For the case with replacement, s_{ref} can be combined with any sequence from S_{CS} with equal chance. If we list the sequences in S_{CS} in a right circular shifting manner started with s_{ref}, it is easy to find that for the second sequence s_i to be chosen from S_{CS}, where integer i indicates the position of the sequence in the sorted list of S_{CS},

$$Fz(s_{ref}, s_i) = \begin{cases} M - i + 1, & i < M + 1 \\ 0, & M + 1 \leq\leq M_p \end{cases} \tag{21}$$

To be mentioned, $s_i = s_{ref}$ for $i = 1$. Thus, we have the values of $Fz(s_{ref}, s_i)$ as $[M, M-1, \ldots, 1, 0, 0, \ldots, 0]$ for $i = 1, 2, \ldots, M_p$, with $M_p - M$ zeros in total. Hence,

$$EZ(S_{CS}) = \frac{1}{M_p} \sum_{i=1}^{Mp} Fz(s_{ref}, s_i) = \frac{M(M+1)}{2M_p} \tag{22}$$

Similarly for the case without replacement, the second choice can be only chosen from the rest of the pool, resulting $M_p - 1$ choices (no s_{ref}). We have the values of $Fz(s_{ref}, s_i)$ as $[M-1, M-2, \ldots, 1, 0, 0, \ldots, 0]$ for $i = 2, 3, \ldots, M_p$, with $M_p - M$ zeros in total. Hence,

$$EZ(S_{CS}) = \frac{1}{M_p - 1} \sum_{i=2}^{Mp} Fz(s_{ref}, s_i) = \frac{M(M-1)}{2(M_p - 1)} \tag{23}$$

B. permutation pool

Now let us look at the permutation pool S_{QS}. Similar ideas can be followed to the calculation of $EZ(S_{CS})$ with the $M < M_p$ case. $EZ(S_{QS})$ can be calculated by the expectation of Fz number between two randomly sampled sequences from the pool. Similarly, we can assume the first sequence is $s_{ref} = [1, 2, \ldots, M]$ for convenience. Now we are going to calculate the mean Fz value of all the combinations of s_{ref} and any sequence in S_{QS} for the case allowing replacement, and the mean Fz value of all the combinations of s_{ref} and the other sequences in S_{QS} for the case without replacement.

Suppose the second sequence s_i has been chosen from S_{QS}, and $s_i = [v_1, v_2, \ldots, v_M]$ where v_j for $j = 1, 2, \ldots, M$ are the question tags. Now we have $s_{ref} = [1, 2, \ldots, M]$ and $s_i = [v_1, v_2, \ldots, v_M]$, it is easy to find that s_i can copy the problem v_j if $v_j \leq j$, hence an easy criterion can be formed to judge whether a question $v_j \in s_i$ contributes to $Fz(s_{ref}, s_i)$,

$$f(v_j) = \begin{cases} 1, & v_j \leq j \\ 0, & v_j > j \end{cases} \tag{24}$$

1) $M = M_p$: Noted that, by random permutation, we can generate a much more question sequences than that obtained through circular shifting. S_{QS} actually contains $M_p!$ elements. Suppose the s_i is just a placeholder for a sequence, the sum of all $Fz(s_{ref}, s_i)$ values can be calculated by question positions instead of by sequences, and for the case allowing replacement, it is

$$\text{sum}_{s_i \in S_{QS}} Fz(s_{ref}, s_i) = \sum_{s_i \in S_{QS}} \left[\sum_{j=1}^{M} f(v_j) \right] \tag{25}$$

$$= \sum_{j=1}^{M} \left[\sum_{s_i \in S_{QS}} f(v_j) \right]. \tag{26}$$

and for the case not allowing replacement, it is

$$\text{sum}_{s_i \in S_{QS}, i \neq 1} Fz(s_{ref}, s_i) = \sum_{s_i \in S_{QS}, i \neq 1} \left[\sum_{j=1}^{M} f(v_j) \right] \tag{27}$$

$$= \sum_{j=1}^{M} \left[\sum_{s_i \in S_{QS}, i \neq 1} f(v_j) \right]. \tag{28}$$

Each question from $\{1, 2, \ldots, M_p\}$ has an equal possibility to be v_j (the jth question in a sequence), and the frequency of each question to be the jth question all equals n/M_p in S_{QS}, where n is the number of sequences in S_{QS} which is equal to $M_p!$. Thus, based on the criterion Eq. (24), Eq. (25) can be easily calculated with

$$\sum_{s_i \in S_{QS}} f(v_j) = \frac{j}{M_p} n, \text{ for } j = 1, 2, \ldots, M \tag{29}$$

$$\text{sum}_{s_i \in S_{QS}} Fz(s_{ref}, s_i) = n \sum_{j=1}^{M} \frac{j}{M_p} = \frac{nM(M+1)}{2M_p} = \frac{n(M+1)}{2} \tag{30}$$

and Eq. 27 with

$$\text{sum}_{s_i \in S_{QS}, i \neq 1} Fz(s_{ref}, s_i) = \text{sum}_{s_i \in S_{QS}} Fz(s_{ref}, s_i) - Fz(s_{ref}, s_1) \tag{31}$$

$$= \frac{n(M+1)}{2} - M \tag{32}$$

Hence, we can calculate the mean of all $Fz(s_{ref}, s_i)$ values by normalizing the results in Eqs. 30 and 31 with their corresponding number of cases, and obtain $EZ(S_{QS})$ with and without replacement as follows:

$$EZ(S_{QS}) = \frac{n(M+1)/2}{n} = \frac{M+1}{2} \tag{33}$$

$$EZ(S_{QS}) = \frac{n(M+1)/2 - M}{n-1} = \frac{M+1}{2} - \frac{M-1}{2(n-1)} \tag{34}$$

2) $M < M_p$: Similar calculations can be done with the case of $M < M_p$, and the only changes are the size of S_{QS} which $n = M_p!/(M_p - M)!$ now and $M_p \neq M$ now. Thus, we can easily obtain the results for the cases with and without replacement as

$$EZ(S_{QS}) = \frac{nM(M+1)/2M_p}{n} = \frac{M(M+1)}{2M_p} \tag{35}$$

$$EZ(S_{QS}) = \frac{\frac{nM(M+1)}{2M_p} - M}{n-1} = \frac{n}{n-1} \frac{M(M+1)}{2M_p} - \frac{M}{n-1} \tag{36}$$

C. Comparison between cyclic pool and permutation pool

For the ease of comparison, the $EZ(\cdot)$ values of random assignments in those cases have been put in a Table II. Based on Eq. (16), we know the expected average cheating gain of a random assignment actually equals to the $EZ(\cdot)$ value of the pool that the assignment has been sampled from, i.e., $E(g(A_{CS}))/N = EZ(S_{CS})$ and $E(g(A_{QS}))/N = EZ(S_{QS})$. Comparing the results in Table II,

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 31 May 2020 doi:10.20944/preprints202005.0502.v1

we can find for the random sampling with replacement, there is no difference between $EZ(S_{CS})$ and $EZ(S_{QS})$ in both cases of $M = M_p$ and $M < M_p$. But for the random sampling without replacement, the results are little tricky. For the case $M = M_p$ without replacement, it is easy to find that $EZ(S_{QS})$ is always greater than $M/2$ except the equality at $M = 2$. Since $n = M_p!$ increases rapidly, we have $EZ(S_{QS}) \approx (M+1)/2$ and approximates to the value with replacement. Hence, in this case ($M = M_p$ without replacement), random sampling from S_{CS} is better than from S_{QS}. For the other case without replacement, we scaling the $EZ(S_{QS})$ a little bit for easier analysis,

$$EZ(S_{QS}) = \frac{n}{n-1}\frac{M(M+1)}{2M_p} - \frac{M}{n-1} \tag{37}$$

$$= \frac{n}{n-1}\frac{M}{2}\left(\frac{M+1}{M_p} - \frac{2}{n}\right) \tag{38}$$

$$> \frac{M}{2}\left(\frac{M+1}{M_p} - \frac{2}{n}\right) \tag{39}$$

$$= \frac{M}{2}\left(\frac{M-1}{M_p-1} + \frac{M+1}{M_p} - \frac{2}{n} - \frac{M-1}{M_p-1}\right) \tag{40}$$

$$= EZ(S_{CS}) + \frac{M}{2}\left(\frac{M+1}{M_p} - \frac{2}{n} - \frac{M-1}{M_p-1}\right) \tag{41}$$

The residual part in Eq. (41) can be proved to be non-negative when $M_p \geq 3$ and $M \geq 2$ as

$$\frac{M+1}{M_p} - \frac{2}{n} - \frac{M-1}{M_p-1} = \frac{2M_p - M - 1}{M_p(M_p-1)} - \frac{2}{n} \tag{42}$$

$$= \frac{2M_p - M - 1}{M_p(M_p-1)} - \frac{2(M_p-M)!}{M_p!} \tag{43}$$

$$\geq \frac{2M_p - M - 1}{M_p(M_p-1)} - \frac{2(M_p-2)!}{M_p!} \tag{44}$$

$$= \frac{1}{M_p(M_p-1)}[2M_p - M - 1 - 2] \tag{45}$$

$$= \frac{1}{M_p(M_p-1)}[M_p - M + M_p - 3] \geq 0 \tag{46}$$

Thus, we have $EZ(S_{QS}) > EZ(S_{CS})$ when $M_p \geq 3$ and $M \geq 2$. It is easy to check that for the case $M = 1$, $EZ(S_{QS}) = EZ(S_{CS}) = 0$. Since $M_p > M$, if $M \geq 2$ then $M_p \geq 3$. Therefore, we have $EZ(S_{QS}) \geq EZ(S_{CS})$, and equality only holds on the situation $M = 1$.

Overall, we always have $EZ(S_{QS}) \geq EZ(S_{CS})$, the equality only holds in the case with replacement or in the case with $M_p = M = 2$ or $M_p > M = 1$. Thus, sampling from the cyclic pool for random assignment tends to have a smaller expectation of the average cheating gain than sampling from the permutation pool, and should be a preferred choice if without sequence size issues.

TABLE II
COMPARISON BETWEEN $EZ(S_{CS})$ AND $EZ(S_{QS})$

Conditions		$M = M_p$	$M < M_p$, $n = \frac{M_p!}{(M_p-M)!}$
$EZ(S_{CS})$	w.r.	$\frac{M+1}{2}$	$\frac{M(M+1)}{2M_p}$
	wo.r.	$\frac{M}{2}$	$\frac{M(M-1)}{2(M_p-1)}$
$EZ(S_{QS})$	w.r.	$\frac{M+1}{2}$	$\frac{M(M+1)}{2M_p}$
	wo.r.	$\frac{M+1}{2} - \frac{M-1}{2(n-1)}$	$\frac{n}{n-1}\frac{M(M+1)}{2M_p} - \frac{M}{n-1}$

REFERENCES

[1] M. S. Medina and A. N. Castleberry, "Proctoring strategies for computer-based and paper-based tests," *American Journal of Health-System Pharmacy*, vol. 73, no. 5, pp. 274–277, 2016.

[2] "Mass school closures in the wake of the coronavirus are driving a new wave of student surveillance," https://www.washingtonpost.com/technology/2020/04/01/online-proctoring-college-exams-coronavirus/, accessed: 2020-05-03.

[3] "April 2020 national exam covid-19," https://www.act.org/content/act/en/covid-19.html, accessed: 2020-05-01.

[4] "TOEFL iBT® Special Home Edition: Am i eligible for this test?" https://www.ets.org/s/cv/toefl/at-home/, accessed: 2020-05-01.

[5] "GRE® General Test at Home: Am i eligible for this test?" https://www.ets.org/s/cv/gre/at-home/, accessed: 2020-05-01.

[6] N. I. Nizam, S. Gao, M. Li, H. Mohamed, and G. Wang, "Scheme for cheating prevention in online exams during social distancing," 2020.