LupSeat: A Randomized Seating Chart Generator to Prevent Exam Cheating

Joël Porquet-Lupine
jporquet@ucdavis.edu
Department of Computer Science
University of California, Davis
Davis, California, USA

Hiroya Gojo
Department of Computer Science
University of California, Davis
Davis, California, USA

Philip Breault
Department of Computer Science
University of California, Davis
Davis, California, USA

ABSTRACT

Randomly assigning seating is an effective way to prevent exam cheating. Our software tool, **LupSeat**, is a seating chart generator which is packed with features and can handle large classrooms. Based on a simple, textual representation of the classroom and a CSV-formatted student roster, the tool algorithmically ensures that students are assigned to random seats and get evenly spaced out. It can also account for other factors in the seating assignment, such as accommodating accessible seating needs or hand dominance, and it offers highly customizable, graphical output charts.

LupSeat provides both command line and graphical user interfaces for all major platforms and is available at https://gitlab.com/luplab/lupseat.

KEYWORDS

Random seating chart; Cheating prevention; Large classes

1 BACKGROUND

Cheating has been a concern for CS instructors for decades [1]. In their 2015 study [3], Levitt and Lin found strong evidence of cheating by at least 10 percent of the students in the exams of a general science course at a top university. However, they also found that when seating locations were randomly assigned –and monitoring was increased– cheating would virtually disappear.

Randomly assigning seating locations by hand can be a time-consuming task, usually reserved for very small classes. Existing software-assisted solutions are not meant to support large classes either, as they often involve significant human intervention or offer only basic features [2]. For example, instructors may be required to manually draw the classroom or the software tool may handle only grid-shaped classrooms, both of which are too limiting for real-life lecture halls at large universities. Besides, no tools are currently able to take into account special cases, such as automatically assigning left-handed seats to left-handed students.

2 TOOL OVERVIEW

Our tool, LupSeat, aims to bridge this gap. Based on a simple, textual representation of a classroom, and a CSV-formatted student roster, the tool efficiently automates the creation of a randomized seating chart where students are evenly spaced out when possible.

In the student roster input file, students are defined by at least their name (first and last) and their student ID number (SID). The textual representation of a classroom allows any configuration of contiguous seats and aisles to be described, making it easy to represent even large lecture halls. As a result, the tool produces a CSV-formatted output file which, by default, gives the list of seat number to SID number associations. It also produces two graphical output files which can be printed for students to see. The first one is a graphical version of the seat number to SID number associations, while the other is a visual representation of the classroom.

The tool supports many advanced features. For example, it offers a graphical user interface; it supports all major platforms (Linux, MacOS, and Windows); it allows the possibility to specify which seats can accommodate students who need accessible seating or left-handed students, as well as the dominant hand for each student and if they require accessible seating; it allows the instructor to provide a third input file that lists pairs of students who should not be seated next to one another (e.g., because they have partnered in a class project before or have belonged to the same study group). This tool is also highly customizable as, for example, the generation of the output files can be configured to show the associations between seat number and any part of the students' information. This feature can be useful to protect the confidentiality of SID numbers by only listing their last few digits in the seating chart.

3 FUTURE WORK

We are currently working on a string of improvements for LupSeat, which include smoothing out the installation process for the different platforms and revamping the design of the graphical user interface, among others. In order to increase the accessibility of the tool, we are also planning to start a database containing the textual descriptions of all the lecture halls on our university campus, and we will engage in outreach efforts at other university campuses to have their lecture halls mapped as well.

REFERENCES

[1] William Dodrill, Doris K. Lidtke, Cynthia Brown, Michael Shamos, Mary Dee Harris Fosberg, and Philip L. Miller. 1981. Plagiarism in Computer Sciences Courses(Panel Discussion). In *Proceedings of the Twelfth SIGCSE Technical Symposium on Computer Science Education* (St. Louis, Missouri, USA) *(SIGCSE '81)*. Association for Computing Machinery, New York, NY, USA, 26–27. https://doi.org/10.1145/800037.800956

[2] Instructure Inc. 2021. Canvas LMS. https://www.instructure.com/canvas

[3] Steven D Levitt and Ming-Jen Lin. 2015. *Catching Cheating Students*. Working Paper 21628. National Bureau of Economic Research. https://doi.org/10.3386/w21628

SIGCSE 2022, March 3–5, 2022, Providence, RI, USA

ACM ISBN 978-1-4503-9071-2/22/03.
https://doi.org/10.1145/3478432.3499139

4 ADDITIONAL CONTENT

4.1 Tool usage example

Let us consider a realistic (but fictitious) example with a 25-student class and a 38-seat classroom.

Figure 1 shows an excerpt of the CSV-formatted student roster file (on the left side), in which the first student –*"Carree Heggs"*– is specified to have a preference for a left-handed seat and the second student –*"Ludvig Quinane"*– is specified to require a special needs seat. The Figure also shows the textual class description file (on the right side), which makes it easy to describe blocks of adjacent seats within each row, as well as qualifiers for certain seats or ranges of seats (`b` for broken seats, `l` for left-handed seats, and `s` for special needs seats).

```
$ cat students.csv
Carree,Heggs,1378,left
Ludvig,Quinane,1821,,special
Charisse,Scemp,1924
Hazlett,Michie,2218
...
```

```
$ cat classroom.txt
Seats:
,a3-a14,
b1-b2,b4-b12,b15
c1-c3,c5-c11,c13-c16

Specifiers:
b:a3
l:a14,b2,b12,c3,c11,c16
s:c5-c7
```

Figure 1: Example of student roster and classroom description

Running the tool as shown in Figure 2 will produce three output files that show the seating chart in various ways. The format argument `--fmt` on the command line allows the instructor to customize how the association between seat numbers and students is shown. In the example below, only the students' initials and the last three digits of their SID number will be displayed.

```
$ <hidden> --student students.csv --seats classroom.txt \
          --fmt "{fname|0}.{lname|0}.{sid|1,4}"
```

Figure 2: Running the tool

Figures 3 and 4 shows excerpts of the two graphical output files generated by the tool. As we can see in Figure 3, the left-handed student was assigned to a left-handed seat (C3), and the special needs student was assigned to a special needs seat (C5); Figure 4 visually shows that the students have evenly spaced out in the classroom and the broken seat (A3) has not been assigned, .

4.2 Seat assignment algorithms

An interesting part of the tool is the seating assignment algorithm. Two different algorithms were implemented and tested, which we called *chunk increase* and *consecutive divide*.

Chunk increase is a bottom-up algorithm, where the initial step is to try and place students in such a way that they all have an empty seat (or an aisle or a wall) on either side. This initial configuration represents a chunk size of one. If the configuration does not fit the number of students in the class, then the chunk size is incremented, which now allows two students to sit next to one another. The chunk size is successively incremented until all students fit within the classroom.

Chunk divide is a top-bottom algorithm, where the initial step is to consider all the available seats in the classroom. As long as the number of available seats exceeds the number of students, then empty seats are successively introduced in order to split the biggest chunks of adjacent seats. To avoid simply splitting the biggest chunk into two smaller chunks (with a empty seat in the middle), which could otherwise lead to unbalanced chunks within the same row of seats, a backtracking phase is used. Before an additional empty seat is introduced to split the biggest chunk, the entire row is first reconstituted and then re-divided evenly.

Regardless of the algorithm, special needs students and left-handed students are assigned to the appropriate seats first.

Thorough testing on 500 randomly generated rosters and classroom configurations showed that the *chunk divide* algorithm consistently performed better, which is why we chose it as the default seating assignment method.

A10	G.B.629	B1	M.F.122	C1	C.S.924
A12	D.C.266	B10	H.M.218	C11	E.P.795
A14	R.M.675	B12	R.B.509	C13	C.M.345
A4	I.K.345	B15	N.T.694	C14	A.B.794
A5	L.F.072	B2	L.J.829	C16	B.J.198
A7	F.N.251	B4	J.L.871	C3	C.M.378
A8	C.M.127	B5	I.J.360	C5	L.Q.821
		B7	N.L.052	C7	B.B.127
		B8	G.K.849	C9	F.S.873

Figure 3: Seating list

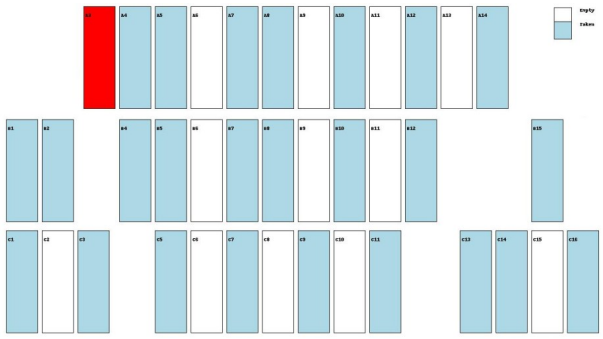

Figure 4: Classroom view