gruepr, an Open Source Tool for Creating Optimal Student Teams

Dr. Joshua L. Hertz, Northeastern University

Dr. Hertz earned a B.S. in Ceramic Engineering from Alfred University in 1999 and then a Ph.D. in Materials Science and Engineering from the Massachusetts Institute of Technology in 2006. Following this, he worked at the National Institute of Standards and Technology as a National Research Council postdoctoral fellow. He joined the Department of Mechanical Engineering at the University of Delaware as an Assistant Professor in September 2008, leading a lab that researched the effects of composition and nanostructure on ionic conduction and surface exchange in ceramic materials. In 2014, he moved to Northeastern University to focus on teaching and developing curriculum in the First Year Engineering program.

Dr. Susan F Freeman, Northeastern University

Susan Freeman, is a member of Northeastern University's first-year engineering facutly, a group of teaching faculty expressly devoted to the first-year Engineering Program at Northeastern University. The focus of this team is on providing a consistent, comprehensive, and constructive educational experience that endorses the student-centered, professional and practice-oriented mission of Northeastern University. Susan Freeman has moved into the Associate Dean, Undergraduate Education role, and continues as a teaching professor in the first-year program along with many new responsibilities.

gruepr, an Open Source Tool for Creating Optimal Student Teams

Abstract

This paper presents the latest updates and newest findings on gruepr, a recently released, open-source software tool that can be used to place students onto optimal project teams. This software was designed and written by one of the authors as a no-cost alternative to existing solutions, such as CATME. The software is written in C++, and currently comprises about 8000 lines of code. Executables for Windows and macOS are publicly available, as is the code itself, released under the GNU General Public License. Recent updates to the software include the use of the Qt libraries for a graphical user interface and other expectations of modern software applications, multi-threading some of the optimization task using OpenMP, adding flexibility to the set of data that can be used to define an optimal project team, and increased automation of the use of Google Forms to collect the student data. In this paper, the current design of the program is presented along with validation of its use by an increasing number of faculty. Validation is presented in the form of results from a survey of faculty to assess what student information they collected and used in forming teams, and what their perceptions were of faculty and student experience with the use of gruepr.

Background

When placing students onto project teams, instructors have several options. They can let students pick their own teammates, they can randomly assign students to teams, or they can assemble teams by selecting students based on their traits. Despite the added work for the instructor in creating intentionally assembled teams, this option is often chosen in order to maximize educational outcomes and/or project success. For example, instructors may choose to balance or, rather, to concentrate certain project skills or high-performing students among the teams depending on the instructor's goals. The instructor also may have preferences regarding placement of students based on gender or underrepresented minority status, or may want to group students based on out-of-class worktime availability. A robust body of literature is devoted to the benefits of intentional team formation, for example [1] – [4].

For relatively simple definitions of "optimal," the problem of finding an optimal partitioning of students into teams can be fairly quick for an instructor to solve by hand. As an example, if the instructor only wishes to ensure that a section of 16 students is partitioned into 4 teams of equal size and with each team having maximally homogeneous student majors, then the instructor can quickly find sets of 4 students that have the same major, followed by sets of 4 students with 2 majors represented, etc. On the other hand, this optimization problem quickly gets to be computationally very difficult as the number of students increases and/or the instructor's definition of "optimal" becomes more complex. Increasing the size of a section from 16 students to 40 students means that the number of possible ways to partition the section onto teams of size 4—i.e., the search space for the optimization algorithm—increases from 2.6 million to $3.5 \cdot 10^{27}$. Further, it is difficult

for an instructor to find suitable teams by hand if the instructor wishes to assemble teams so that each team has maximal homogeneity in academic major *and* maximal homogeneity in preference for or against weekend work *and* maximal heterogeneity in the preferred work role on the team. Preferences on, for example, each team having at least 3 but preferably 6 hours out-of-class each week where every team member is free would additionally require the instructor to look through complicated weekly schedule matrices for each student.

One very popular system to use computers to solve this optimization problem is the web-based Team-Maker, a component of the Comprehensive Assessment of Team-Member Effectiveness (CATME). CATME was originally funded by NSF and the Team-Maker portion was made available for public use in 2007 [5]. Unfortunately, the source code for Team-Maker is not open source, and CATME now operates on a fee-based model. Another option for web-based team formation that had been available until recently is called groupformation.org, described in [6]. This tool has since gone offline, unfortunately.

With no freely available option found for the computerized creation of teams, the author began work on creating an open source solution. The lead goals of the project were to create a solution that is maximally flexible in terms of how the instructor wishes to collect student data and define an optimal team and to remain user friendly for instructors, with the source code as well as Windows and Mac executables freely distributed.

The author's solution is named gruepr, and a version was first presented at the 2019 ASEE annual conference [7]. Gruepr solves the difficult optimization problem using a genetic algorithm, wherein an initial population of 30,000 randomized ways to partition the students into teams evolves to increasingly near-optimal partitionings. This process typically takes less than 1 minute on a modern laptop computer. Gruepr currently relies on Google Forms to survey the students and collect the information that will be used in forming the teams. An instructor can easily create a highly customized Google Form survey from within gruepr. After students submit their survey responses, gruepr can automatically interpret the downloaded file of survey results.

The instructor can choose how to define an optimal set of teams based on a wide variety of factors, such as having a certain number of overlapping free time blocks in the weekly schedules, or having either homogeneity or heterogeneity of up to 9 "attributes," which can be any categorical or numerical/ordered trait. Gruepr presents the possible teaming options to the instructor, and then automatically translates the chosen options into the genetic algorithm's "fitness function." A fitness function is an objective, mathematical function that summarizes how optimal a given solution attempt (a "genome") is. Details on the fitness function used in gruepr can be found in [7].

After finding an optimized set of teams, the results are displayed to the instructor. These results can be saved to disk as text or pdf files, or printed directly from gruepr. The saved results can include each student's survey data or instead, as would be appropriate for distribution to the students, can include solely the students' names, email addresses, and a table of the team's availability throughout the week.

Gruepr is being used now at an increasing number of faculty at a few institutions, including Northeastern University, University of Tennessee, Ohio State, Texas A&M, and elsewhere. The source code is freely available, released under the GNU General Public License. At the project homepage [8], both the full source code and the compiled Windows and Mac binaries are available. In this paper, the significant advances in gruepr since its initial announcement will be described along with feedback on its use from various faculty.

Improved functionality

The first versions of gruepr, detailed in [7], were text based, and ran in a console window. This manner of operation was easy to code, but made using the software difficult. To improve the user interface, gruepr has now been ported to make use of the cross-platform Qt application development framework [9]. Qt is not a programming language, rather it extends C++ with an extensive set of features, including pushbuttons, tooltips, dialog windows, and other GUI elements; drag and drop functionality; and interfaces to the operating system such as standard dialog boxes, and app preferences saved to the registry. Qt is developed actively by the Qt Group, who offer both licensed and open source versions. Figures 1 and 2 demonstrate the modern, graphical interface of the latest version of gruepr.

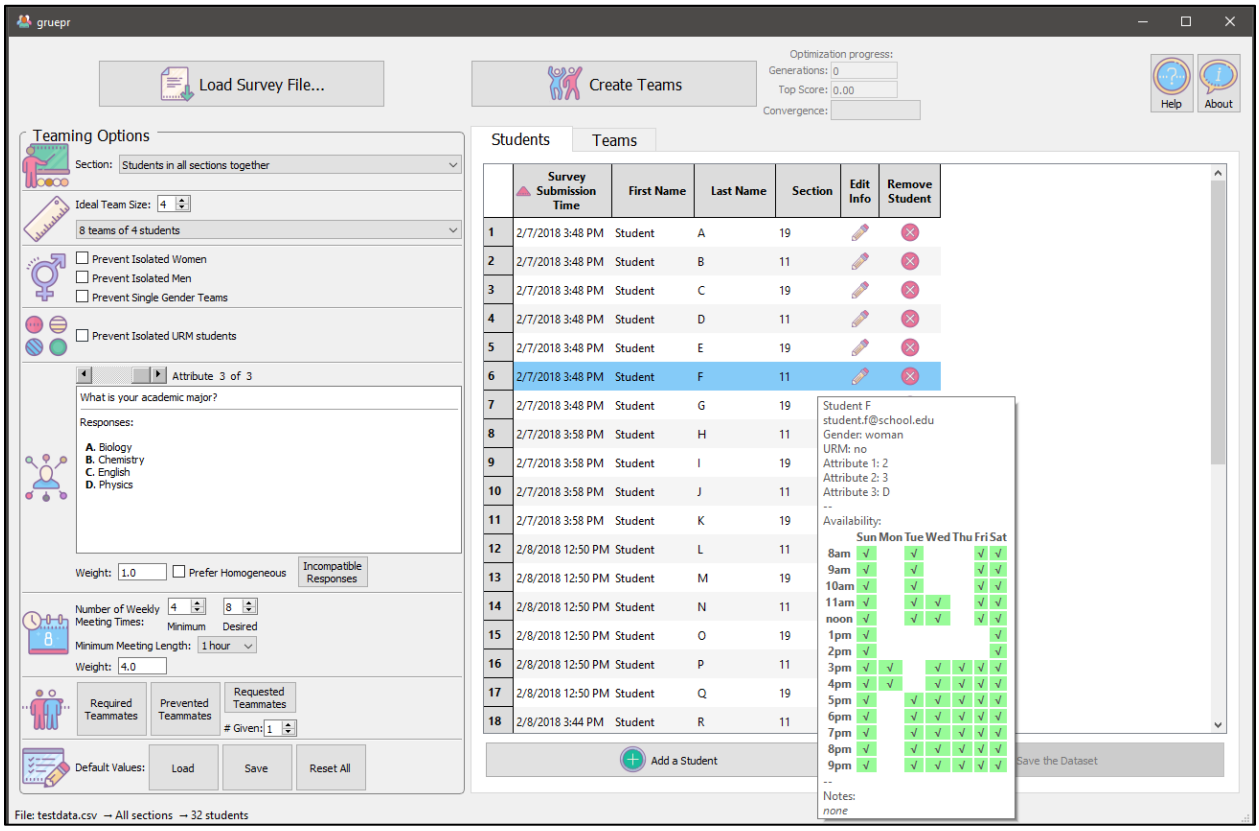

Figure 1. After loading the .csv file of student survey responses downloaded from Google Forms, gruepr shows the possible teaming options on the left and a list of the students on the right. Hovering the mouse over a student brings up a tooltip that summarizes the student's survey data.

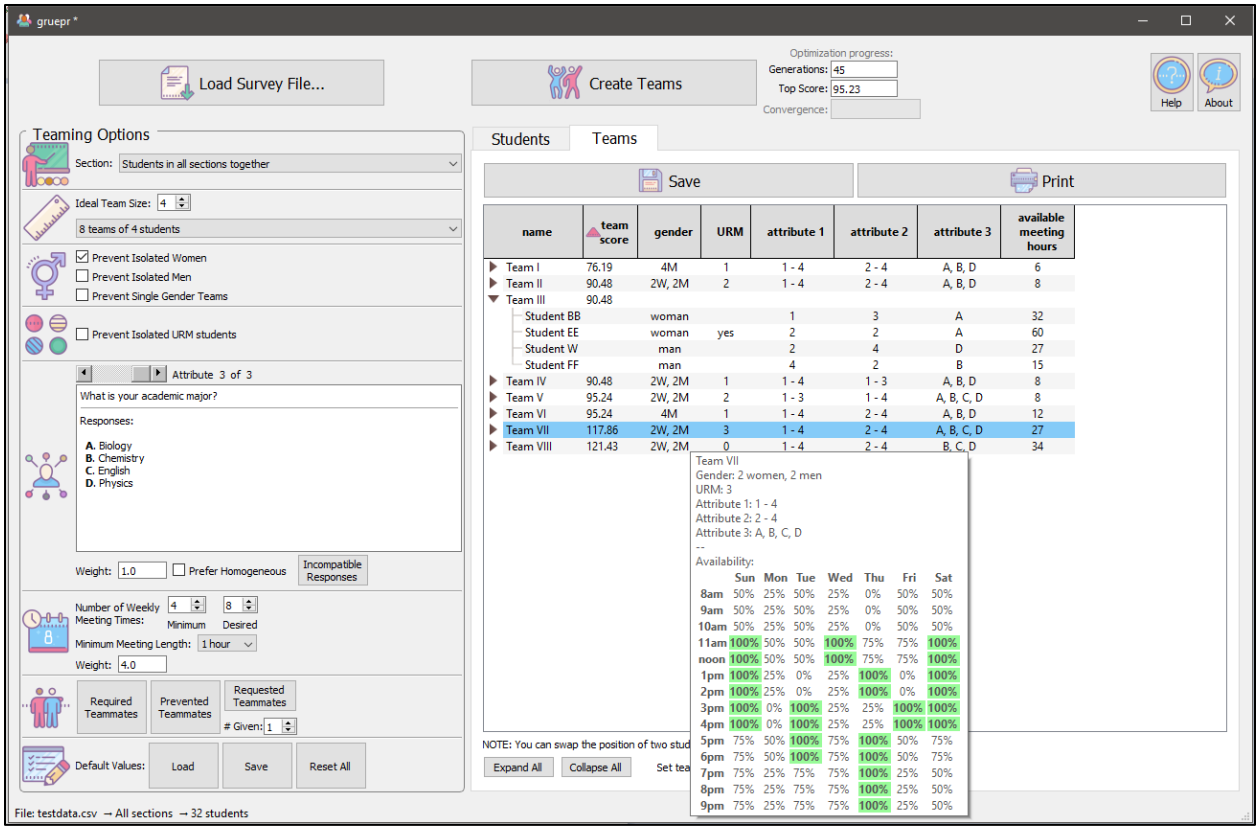

Figure 2. After the instructor selects their teaming options and then runs the optimization, the best set of teams found is shown on the right. A summary of the survey responses submitted by that team's students is shown, with greater detail displayed by hovering the mouse over the team name. In the tooltip, green boxes indicate when every teammate has an available opening in their weekly schedule.

In addition to the modern user interface, several significant features have been added to gruepr recently using Qt-based solutions. First, an instructor's desired set of default values for all of the teaming options can be saved or loaded. These preferences as well as the desired window size are saved to the registry on Windows or to a .plist file on macOS. Second, the opening, saving, and printing of files in gruepr are all handled by passing data to/from the operating system's standard dialog boxes. Additionally, when saving files, options to export to pdf are carried out using Qt's pdf exporter functions. Third, the use of drag-and-drop allows the instructor to easily switch students between teams and thereby manually tweak the results found from the algorithm. Finally, to help beautify gruepr and give a more unified appearance, a set of icons with common visual appearance was added. The icons come from icons8 [10], who allows free use of these icons with attribution.

Another recent improvement to the functionality of gruepr has come from significantly reducing the optimization time. When performing the genetic optimization algorithm in gruepr, most of the computational time is spent in calculating the fitness function for all 30,000 genomes in a generation. Each genome represents a single partitioning of the

students into teams. Since the genomes are independent from one another, the calculation of the 30,000 fitness function values is “embarrassingly parallel.”

The multiple cores on a modern processor should allow for a number of these calculations to occur in parallel across multiple threads. Qt provides some capabilities for multithreading, however after much time was spent unsuccessfully trying to implement these tools, these efforts were abandoned. Instead, the most recent version of gruepr uses the OpenMP C++ compiler directives and routines to parallelize the fitness function score calculations. The OpenMP specification is very high level, and required not many lines of additional coding to implement [11]. Using this parallelization tool, optimization occurs almost twice as quickly on a machine with a dual-core processor, and four times as fast on a machine with a quad-core processor.

The final major improvement to gruepr functionality comes in the increased automation of the use of Google Forms as the means to collect student survey data. A Google Script was written and deployed as a publicly accessible API so that a browser with a correctly formatted URL will create a Google Form that includes the desired questions in a format that will create responses readable by gruepr. Further, the script responds by displaying a webpage with further instructions for the user. Within this webpage is provided direct links for the instructor to edit the survey, to directly download the csv file of survey results, and a shortened URL to send to students for them to fill out the survey. Figure 3 shows how the SurveyMaker tool is accessed easily from within the single gruepr app, and Figure 4 shows the SurveyMaker tool itself.

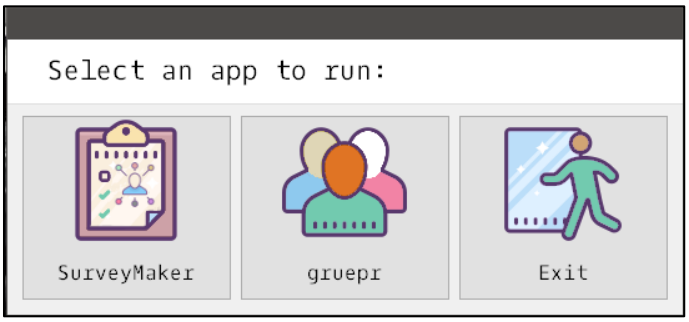

Figure 3. When first starting the gruepr app, the user can choose whether to run SurveyMaker or gruepr.

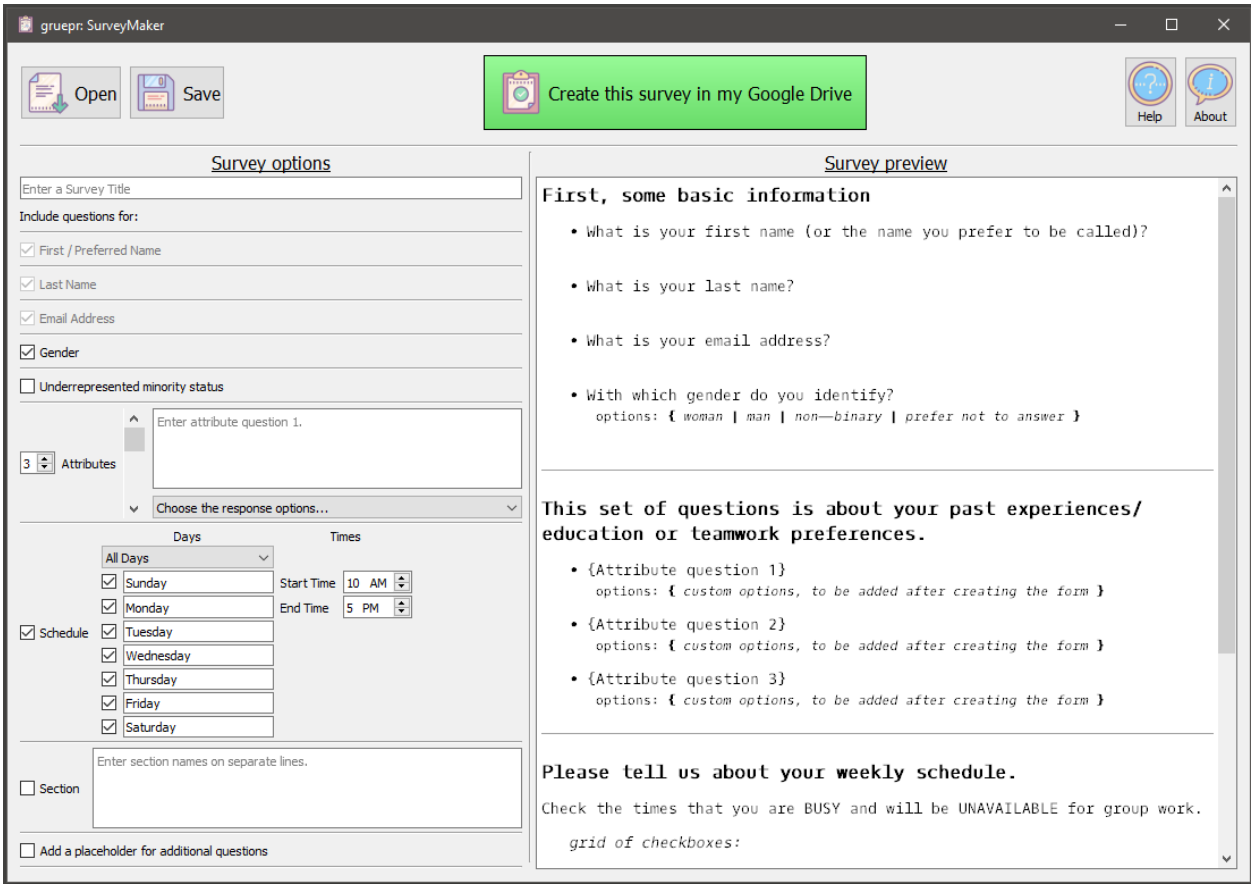

Figure 4. The SurveyMaker tool in gruepr allows the creation of a customized Google Form on the instructor's Google Drive account. Considerable flexibility is provided in which questions will be asked of the students and used to form teams in gruepr.

More flexible optimization options

One of the factors used in defining an optimal team can be based on homogeneity or heterogeneity of a student attribute. This attribute can be any categorical or Likert-scale descriptor of the student. Examples include the student's academic major, GPA, level of experience or comfort with a particular skill, and the student's preferred leadership role on a team. In gruepr's original formulation, an attribute score for a team is based on the range of attribute values that the teammates have relative to the total range of possible values found among all students. For example, perhaps there are 10 possible response values to this question and the students on a particular team have among them response values 2 and 5. In this case, since the range of values represented on this team is 3, the score for this attribute would be 3/10 if the instructor wanted heterogeneity or 7/10 if the instructor wanted homogeneity.

The issue with this original scoring mechanism is that it only makes sense for cases where the response values have a natural, linear ordering. The score would remain the same if the

students on the team have among them response values 2, 3, 4, and 5 instead of just values 2 and 5. For cases of purely categorical responses, such as academic major, this scoring does not capture the meaning of heterogeneity or homogeneity. It makes a difference if two majors are represented vs. four majors. Thus, gruepr now recognizes when the attribute questions are categorical or have a numerical meaning to their order by looking for whether the responses begin with a numerical value. Where the question is categorical, the score is no longer based on the range of values but rather on the count of how many response values there are.

Another feature added to gruepr was based on request from a user. This feature allows the instructor to specify "incompatible responses" to an attribute question. When using this feature, teams will be prevented wherein students on the team have particular response values to one of the attribute questions. An example use case of this feature is to have an attribute question asking students which topic they want to work on for a project, allowing students to select "robot", "catapult", or "either". The "robot" and "catapult" responses could be made incompatible. In this way, some teams will consist of students that chose "robot" and/or "either" and the other teams will consist of students that chose "catapult" and/or "either", but no teams will have students that chose "robot" with students that chose "catapult". The mechanism by which students with incompatible responses are prevented from being on the same team is the fitness function penalty described in [7] for preventing specific students from being on the same team.

When students are placed on teams more than once in a course, it is common that the instructor wants to place students with all new teammates each time. For this reason, a feature was added to gruepr to allow this. A feature had already been in place where the instructor could manually enter a set of students that should be prevented from being teammates. Now, added to that feature, is a button that reads a text file containing a previous set of teams and then automatically adds all previous teammates as prevented teammates for this next run of gruepr. Crucial to this feature is an algorithm to match the students' names from the text file to the students' names in the current data file, regardless of typos, usage of nicknames, or other differences in text values. The solution to this problem came in the use of the Levenshtein distance algorithm [12], which can search through a list of text strings and rank them from most to least likely to be a match to the current string. In gruepr, when using this separate-all-previous-teammates feature, if an exact match for the student name is not found, an ordered list of likely matches is presented to the instructor to choose.

Validation of gruepr usage by faculty

In this section, the emphasis is on the various ways that faculty use gruepr in their classes, and their feedback on the use of the tool, its helpfulness and ease of use. Since gruepr can be easily setup in a variety of configuration and customizations, it is valuable to see how one group actually has put it into practice. A survey instrument was created and distributed to 11 faculty members using gruepr at one institution. Feedback solicited in this survey regarded what questions each instructor asked of the students to form teams, what

the perceived quality of the created teams were, and what the quality of the user experience was for each instructor.

When instructors decided what questions to ask the students, the first standard question regarded the students' gender identity. Of the respondents, 100% prevent isolated women, 12.5% prevent isolated men, and no one prevents single gender teams. The next three questions are generally about previous experience and role on a team. 100% of the faculty use 2 questions, especially in semester 1 of the course, if they have a 2 course sequence and the second in the sequence has the same students. The 2 questions are: "How would you rate your programming experience?" (or similar) and "What is your level of hands-on building experience?" 60% of the faculty use a question about the students preferred role in their group with responses being "leader, mix of leader and follower, follower". Other attribute questions are:

- *"I have prior experience with long-term (Several weeks even months) group projects."*
- *"I am highly organized and comfortable handling multiple due dates."*
- *"I am comfortable speaking English."*
- *"I am available to work weekends."*
- *" What is your level of personal experience with graphic design or CAD programs?"*
- *"Are you available to work on weekends?"*
- *"My last semester GPA was?"*

As you can see, there is a great variety of ways to use the attribute questions, and they can be customized in any way, and most faculty take advantage of that.

The key feature is trying to maximize the time where students can meet. Most instructors select a fairly standard set of meeting time desired parameters, such as one hour meetings, 1-3 times a week. A few instructors make more and longer meetings, then scale it back if the program has trouble finding common meeting times for the groups.

For preventing or requiring teammates, the majority of the faculty do not allow it (70% say no on requiring, 80% say no on preventing). The course surveyed is frequently taught in a two semester series, though sometimes as 2 courses stacked up in one semester. Since the same students are taking the course in the second semester, it is interesting to see the effect of that. The most common difference for a number of faculty is not asking about experience, since the students are coming from the same experience in C++ for example. Some allow teammate choices in the second semester only. There is also some manual adjustments made, preventing personality conflicts after the first semester.

All of the faculty found that gruepr makes either great team matches or good team matches. Most of the team reported that is easy to learn and setup (87.5%), one person was more neutral. All of the faculty found that it is easy to use the results. Most of the faculty rate gruepr as much better or better than other teaming methods (77.7%), with 2 faculty more neutral. All would be very likely to recommend this to others.

The comments sum up the faculty use of gruepr, it makes great teams, saves time and effort, easy to use and is effective for solving the challenge. There were zero negative comments, here are a few typical comments.

> *"I love that I can customize or edit the sorting questions in gruepr. It's just as easy or easier to use than CATME once I get the hang of it."*
>
> *"Great, easy to load students, works really well."*
>
> *"It is a fantastic tool and time saver."*

In a detailed discussion with one user, here is a summary of their comments and usage.

> *The first step of building the survey for the students is well explained and set up clearly, so that with a few customizations, there is a link to send out. No negative feedback on the survey, students respond quickly and clearly. The .csv file is right there, once saved, it is time to run gruepr. Simple, clear questions to put in the weights, these are the questions the instructor created for the class and it runs. As an engineer, it is easy to try different combinations and experiment, very popular activity. This creates options to consider, but not so many that it is hard to select. Oftentimes, there is a steady state solution, that even with tweaking a few things, the result is still almost the same. Even if there is a problem once the students see it, a simple switch usually solves it. Allowing the students to select who they might want on the team, or who they do not want on the team can cause poor or even infeasible solutions, student a wants student b, b wants c, but c does not want a. Therefore, best use is to limit this, and create the rule that both must select each other for example. Unbeknownst to the students, the instructor can also match or unmatch students if they know them well enough, very helpful. The output shows where the student schedules overlap, which provides evidence that they can meet based on their input. Overall, the teams have been great.*

User feedback has helped make gruepr better and easier to use. This tool is free and a real boon to setting up teams. This group of faculty highly recommend this tool. In order to measure the usage of gruepr, a "software registration" feature has been added, and faculty from five institutions have registered their copy thus far.

Continuing Development

Two specific areas are being developed to improve the functionality of gruepr. One is the manner in which data about underrepresented minority status is collected in the survey. Currently, the survey question simply asks students to self-identify as member of an underrepresented group. In an upcoming version of gruepr, this question will be replaced with a question asking students to self-identify their racial, ethnic, or cultural identity. This question will have a free-response answer so that students can identify themselves however they like. When running gruepr, if an instructor wishes to use this data in forming the teams (e.g., preventing a student from an underrepresented group being the only such

student on a team) then the entire list of answers provided by students will be presented. The instructor will then be asked to select which of these responses should be considered as an underrepresented minority group. Duplicate answers will not be duplicated in this list of options.

The other specific area being developed is in allowing questions where students can fill in the names of other students that they request to be placed with or not placed with. Matching a name typed in by a student's peers with the name as given by the student themself can be performed with the Levenshtein algorithm discussed previously.

Conclusions

Gruepr is being used by an increasing number of instructors at a range of institutions. Recent updates have made the software faster, easier to use, and similar in feel to other desktop apps. Educators wishing to develop apps are strongly recommended to look into the Qt framework for cross-platform GUI development as well as OpenMP for parallelization of computation. While development of gruepr continues, we found here that instructors appreciate the current functionality and ease of use.

References

[1] M.-I. Sanchez-Segura, M. Hadzikadic, G.-L. Dugarte-Peña, and F. Medina-Dominguez, "Team Formation Using a Systems Thinking Approach," *Systems Research and Behavioral Science*, vol. 35, no. 4, 2018.

[2] M. Borrego, J. Karlin, L. D. McNair and K. Beddoes, "Team Effectiveness Theory from Industrial and Organizational Psychology Applied to Engineering Student Project Teams: A Research Review," *Journal of Engineering Education*, vol. 102, no. 4, 2013.

[3] F. Morgeson, M. Reider, & M. Campion, "Selecting individuals in team settings: The importance of social skills, personality characteristics, and teamwork knowledge," *Personnel Psychology*, vol. 58, no.3, 2005.

[4] N. Dasgupta, M. M. Scircle, and M. Hunsinger, "Female peers in small work groups enhance women's motivation, verbal participation, and career aspirations in engineering*," Proceedings of the National Academy of Sciences*, vol. 112, no. 16, 2015.

[5] https://info.catme.org/about/who-are-we/

[6] T. Henry, "Creating effective student groups: an introduction to groupformation.org," in *Proceeding of the 44th ACM technical symposium on Computer science education (SIGCSE '13)*, 2013.

[7] J.L. Hertz, D. Davis, B.P. O'Connell, C. Mukasa, "gruepr: An Open Source Program for Creating Student Project Teams," *2019 ASEE Annual Conference and Exposition*, paper ID# 26537 (2019).

[8] http://bit.ly/Gruepr

[9] https://www.qt.io/

[10] https://icons8.com/

[11] https://www.openmp.org/

[12] A. Levenshtein, "Binary Codes Capable of Correcting Deletions Insertions and Reversals", *Soviet Physics Doklady*, vol. 10, no. 8, 1966.