A Time-Saving Algorithm for Team Assignment and Scheduling in a Large-Scale Unit Operations Laboratory Course

Dr. Andrew Maxson, Ohio State University

Andrew Maxson is an assistant professor of practice in chemical engineering at The Ohio State University where he teaches Chemical Engineering Unit Operations. He earned his B.S. in chemical engineering from Rose-Hulman Institute of Technology and his M.S. and Ph.D. in chemical engineering at Ohio State. Having worked as a manufacturing process engineer for ten years, his focus is on optimizing the process of teaching, as well as hands-on, practical engineering concepts relevant to chemical engineers entering industry.
Abstract

The chemical engineering program at The Ohio State University has been growing rapidly over the last decade, and it is now one of the largest in the United States. Students enroll in Unit Operations for two semesters, making the course twice the size of other courses in the department; enrollment is typically about 220 students in the combined lab/lecture course. To cope with the growing enrollment, the course instructors have adopted a number of time-saving strategies.

The most successful of these strategies has been the development and use of an evolutionary algorithm to assign students into groups and schedule those groups in the laboratory. The evolutionary algorithm makes random changes to the group assignments and lab schedule and keeps beneficial changes as measured by ten criteria. Data are presented for one recent semester in which the algorithm was successfully used for group assignment and scheduling with an enrollment of 225 students in 48 groups, each completing 2 of 15 experiments available in the lab.

A working example of the algorithm has been made publicly available for download. It was implemented in Microsoft Excel so that no special software installation or licensing is required, and so that anyone familiar with writing worksheet formulas in Excel can easily customize the algorithm for their own use.

The full algorithm (225 students, 48 groups, 12 experiments, and 10 criteria) reached an acceptable solution after about 9 hours of computing time, while the simpler working example provided by the author (72 students, 18 groups, 5 experiments, and 4 criteria) reaches a near optimal solution in about 10 seconds.

Introduction

Unit Operations Laboratory (Unit Ops) courses are staple components of most chemical engineering curricula. Unit Ops is viewed by many as an essential and defining experience for chemical engineers, but the laboratory setting poses different challenges for instructors from those encountered in the classroom setting.

Team-based work poses one such challenge and is common to nearly all Unit Operations Laboratory courses – in a recent survey, 69 out of 70 programs reported that their Unit Ops students work in teams [1]. This means that Unit Ops courses must have a strategy for placing
students into teams, which by itself is a difficult problem and an active area of research. Instructors have several options for assigning teams. One is team self-selection (allowing students to choose their own groups), which requires minimal effort on the part of the instructor. However, there are several drawbacks associated with self-selection, including bad student experiences, team homogeneity, clique behavior, and negative effects on students’ perception of many aspects of the course. These are well-summarized by Layton et al. in their 2010 paper [2]. Instructor-assigned teams are an alternative, but assigning teams can be a time-consuming task, especially if the class size is large and the instructor wishes to use evidence-based practices for team assignment.

Software-assisted team assignment has become a popular way to address this problem, the most popular of which is CATME Team-Maker. But in a laboratory course, team assignment is only half the problem; there will inevitably be many other logistical constraints in any Unit Ops course imposed by limited laboratory space and time. In other words, in addition to deciding which students will be in which groups, instructors must also decide which groups will conduct which experiments on which days. Many logistical factors add complexity to this problem, such as the need to manage the number of students in the lab at one time, schedule specific experiments for use by different teams at different times, ensure the availability of instructors and teaching assistants, etc. This complexity increases exponentially with course size.

The chemical engineering program at The Ohio State University has one of the largest Unit Ops courses in the United States, with an enrollment of more than 200 students every semester, so the adoption of time-saving strategies is essential. The most valuable of these has been the creation of a custom group assignment and laboratory scheduling algorithm.

**Evolutionary algorithms**

An evolutionary algorithm was chosen for the task of group assignment and laboratory scheduling because such algorithms have frequently been applied for that purpose [3, 4, 5]. CATME Team-Maker, the popular software-assisted team assignment tool, also works using an evolutionary algorithm [2].

In an evolutionary algorithm, a solution is first randomly generated. That solution is checked for fitness by a scoring method, and then random changes or “mutations” are made. After each change, the fitness is checked again. If an improvement has been made, the new solution is kept and serves as the starting point from which random changes are made in the next iteration.

In the algorithm described in this paper, the starting point is a random assignment of students into groups, and random assignment of groups to focus areas (experiment categories), laboratory experiments, and lab dates.

**Overview of Ohio State’s Unit Operations Laboratory course**

To provide context for the logistics and criteria that went into the development of the algorithm, a brief overview of the course is given in this section.
• Enrollment of 200+ students each semester; in the last 3 years the lowest enrollment was 207, and the highest was 236
• 1 lab section (5 hours once weekly)
• 15 total experiments, with 12 operational each semester
• 8 lab days per semester (several weeks in the semester are devoted to safety training and other activities)
• Each group conducts two experiments per semester
• Group size 4–5 students
• Each student takes the course for 2 semesters, so approximately half of students are first-time students and half are second-time students.
• Experiments are categorized into three “focus areas”: Classical, Environmental, and Biological. Students complete a survey to identify their focus area preference based on their career interests.
• Students are surveyed to identify other students with whom they prefer not to be grouped (based on past group work experiences or personal conflicts) and dates when they would prefer not to attend lab (e.g. due to job interviews or other personal commitments).

A note on team assignment criteria

The development of strategies for team assignment that maximize the performance and learning experience of engineering students is an active area of research, and this paper is not intended to endorse any particular set of team assignment criteria. On the contrary, the algorithm described here and the sample workbook provided both take an impersonal approach to team assignment and instead focus mainly on logistical considerations. However, the algorithm is capable of forming teams based upon any criteria desired by the user.

In case the user prefers to rely on outside tools which specialize in assigning teams based upon evidence-based practices, a provision is made in the example workbook to disable team assignment. When team assignment is turned off, group numbers entered by the instructor will not be changed by the algorithm. This enables the use of the provided Unit Ops Scheduler algorithm to handle complex logistical constraints in conjunction with an external tool like CATME Team-Maker to handle team assignment.

Implementation in Microsoft Excel

The algorithm described here was written in Visual Basic for Applications (VBA) in Microsoft Excel. VBA is included with every version of Microsoft Excel since 1997, with the exception of Excel 2008 for Mac.

Although elegance and speed are compromised in VBA when compared with other programming languages, there are several advantages associated with this platform. First, Microsoft Office is
ubiquitous on college campuses in the United States, for both students and faculty. This means that faculty members can easily start using the algorithm in their own courses without the need for installation or licensing of new software, and when a solution is generated it can be easily disseminated to and viewed by students. Secondly, making manual changes to the group assignments and lab schedule is simple and familiar using Excel’s spreadsheet interface. Finally custom logistics and criteria can easily be added by users with no programming experience, through the use of Excel formulas.

**Mutations**

To try to improve the “fitness” of the solution, mutations are introduced into the group assignments and lab schedule. Several different types of mutations are needed to move through the space of solutions to find improved solutions. The types of mutations used in the algorithm are shown in Table 1. In the descriptions of the mutations, “swap” refers to exchanging two cell values between two groups, “switch” refers to exchanging two cell values within a group, and “change” refers to simply overwriting a cell value.

<table>
<thead>
<tr>
<th>Table 1: Mutations used in group assignment and lab scheduling algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
</tr>
<tr>
<td>2.</td>
</tr>
<tr>
<td>3.</td>
</tr>
<tr>
<td>4.</td>
</tr>
<tr>
<td>5.</td>
</tr>
<tr>
<td>6.</td>
</tr>
<tr>
<td>7.</td>
</tr>
<tr>
<td>8.</td>
</tr>
<tr>
<td>9.</td>
</tr>
<tr>
<td>10.</td>
</tr>
</tbody>
</table>

In each iteration of the algorithm, each of the 10 mutations is made individually. After each mutation, if the quality of the solution has decreased, the change is reverted. If the quality of the solution is the same or better, the change is preserved. The preservation of neutral mutations results in neutral “drift” in the solution, allowing the discovery of new, better solutions that may not be accessible from one solution but can be accessed from a different solution with the same score.

**Scoring method**

For the Unit Ops group assignment algorithm described here, the fitness of the solution is scored using 10 criteria. Deviations from each criterion’s target are counted and multiplied by the criterion’s weighting. For example, one of the criteria used for lab scheduling is that students should not be assigned to an experiment they have already completed. So, the number of instances of a student being assigned the same experiment more than once is counted using Excel
worksheet formulas and multiplied by a weighting. The resulting product is included in a sum along with the products from the other criteria. Reducing the sum of the products is the goal of the algorithm.

The weightings allow certain criteria to take priority over others. The weightings are powers of 10, so that a change reducing the number of deviations from the target of a higher priority criterion will be kept even if it increases the number of instances of a lower priority criterion by ten.

Several of the criteria are critical to the structure of the course, and no deviations from the target value can be permitted. For example, a student can never be assigned to the same experiment more than once. For other criteria, some deviation is acceptable or expected, for example students being assigned to their 2nd choice of focus area.

The criteria and weightings are shown together with the results of the final accepted solution in Table 2 in the next section.

**Performance**

In Autumn Semester 2018, 225 students were enrolled in the Unit Ops course. Student surveys garnered 250 schedule conflict dates and 37 personal conflicts. After randomly populating the group assignments and lab schedule, 63,943 iterations were required to reach an acceptable solution.

Each iteration took about 0.5 seconds using a 3.6 GHz 8-core AMD FX processor, for a total of over nine hours of processing time. However, it should be noted that the run time increases exponentially with the number of students and criteria, so solutions for smaller course sizes and fewer criteria will be reached much more quickly. In the working example file provided along with this paper, which includes 72 students, 18 groups, 5 experiments, and 4 criteria, the algorithm typically produces a near-optimal solution in fewer than 2,000 iterations and less than 20 seconds (fewer mutation types are also used in the provided example, so iterations complete more quickly). If faster run times are desired, certain mutations can be turned off (commented out) once the criteria relying upon them have been completely solved.

For the solution that was accepted, the target and actual values for each of the criteria are shown in Table 2.

Note that this was not necessarily the best possible solution, and running more iterations or restarting the algorithm from the beginning may result in further improvements. However, it is not always possible to obtain the target value for every criterion, because a higher priority criterion may prevent it. For example, in the results shown in Table 2, a second-time student may have been assigned to lab on a date when he or she had indicated a schedule conflict (violating Criterion 9) because other members of the same group indicated schedule conflicts on all other possible lab days. Criterion 6 (second-time students not grouped with previous group members) has a higher priority than Criterion 9, so a mutation that moved the student to another group would have been reverted, even if it resolved the scheduling conflict.

The number of deviations for the target values for Criteria 1–10 are shown for iterations 1–63,943 in Figure 1. Note that the scale on the x-axis in Figure 1 is logarithmic. This is because rapid
### Table 2: Criterion number, description, target values, actual values, and weightings for scoring criteria

<table>
<thead>
<tr>
<th>Crit. #</th>
<th>Description</th>
<th>Target Value</th>
<th>Actual Value</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Groups assigned to experiment outside focus area</td>
<td>0</td>
<td>0</td>
<td>$10^9$</td>
</tr>
<tr>
<td>2</td>
<td>Double-booked or unbooked experiments</td>
<td>0</td>
<td>0</td>
<td>$10^8$</td>
</tr>
<tr>
<td>3</td>
<td>Second-time students assigned outside previous focus area</td>
<td>0</td>
<td>0</td>
<td>$10^7$</td>
</tr>
<tr>
<td>4</td>
<td>Number of group members</td>
<td>0</td>
<td>0</td>
<td>$10^6$</td>
</tr>
<tr>
<td>5</td>
<td>Students repeating experiments</td>
<td>0</td>
<td>0</td>
<td>$10^5$</td>
</tr>
<tr>
<td>6</td>
<td>Second-time students not grouped with previous group members</td>
<td>0</td>
<td>0</td>
<td>$10^4$</td>
</tr>
<tr>
<td>7</td>
<td>Students assigned to 3rd preference for focus area</td>
<td>0</td>
<td>0</td>
<td>$10^3$</td>
</tr>
<tr>
<td>8</td>
<td>Students assigned with personal conflicts</td>
<td>0</td>
<td>0</td>
<td>$10^2$</td>
</tr>
<tr>
<td>9</td>
<td>Students assigned with date conflicts</td>
<td>0</td>
<td>18</td>
<td>$10^1$</td>
</tr>
<tr>
<td>10</td>
<td>Students assigned to 2nd preference for focus area</td>
<td>0</td>
<td>14</td>
<td>$10^0$</td>
</tr>
</tbody>
</table>

Progress is made at the start; the probability of finding a net beneficial mutation is higher when there are many deviations from the optimal solution and when improvements are still possible for the criteria with the highest weightings. Once the solution approaches the optimal solution, it becomes more and more difficult to find beneficial mutations.

**Availability and use of example workbook**

A simplified, working example of the algorithm has been made publicly available online. The link to the workbook is: [https://osu.box.com/v/unit-ops-scheduler](https://osu.box.com/v/unit-ops-scheduler). To use the example workbook, active content must be enabled in Excel. The interface is shown in Figure 2.

To run the example algorithm, first click the “Randomize” button to randomly populate the group assignments and lab schedule. Next, click the “Run” button, and the algorithm will run the number of iterations shown in cell B18 (the default number is 1,000) to try to improve the solution. This should take about 10 seconds and should nearly always obtain exactly the target value of zero for each of the first three criteria and should greatly decrease the score for the fourth criterion.

To customize the example spreadsheet for a particular Unit Ops course, edit the values in the “Course Setup” area to reflect the desired number of groups, the number of laboratory experiments available, and the total number of available lab days. A second worksheet includes a list of generic placeholder values for student names. Replace the generic placeholder name data with names of the students in the course. A third worksheet contains a list of groups, the first and second experiment each group is assigned to perform, and the lab session number to which that group is assigned.

If the number of groups is changed, the data in columns A through G for those groups must be populated on the Group Data sheet. If the number of experiments or lab days is changed, the data
Figure 1: Number of deviations from ideal solution for Criteria 1–10 through 63,943 iterations

Figure 2: Interface for working example workbook
in columns I through N on the Group Data sheet and the corresponding formulas on the Run
Algorithm sheet must be modified accordingly. Once all data have been entered, enter the desired
number of iterations in cell B18 and click the “iterate” button. The current iteration number is
displayed in cell B19.

The example workbook assigns teams and optimizes the lab schedule based upon four
criteria:

1. **Students repeating experiments** – minimize the number of times students are assigned to
complete experiments they have already completed in the past.

2. **Groups with too many or too few members** – minimize the sum of the absolute values of the
differences between the target number of students in each group (4) and the actual number
of students in each group

3. **Double-booked experiments** - minimize the number of times experiments are booked more
than once on the same lab day

4. **Group's average birthday difference from Jul 1** – attempt to create groups in which the
average birthday of the group members is equal to July 1st. This criterion is included as an
example of a team assignment criterion; it has no effect on lab scheduling or logistics. Of
course, it doesn’t make sense to target a particular average birthday for a group, but the
same code could be used to target group average GPAs that are equal to the class average
GPA, or to target a particular value for any characteristic of the groups that can be
calculated numerically. This is the most time-intensive criterion included in the example
sheet, but group average birthdays are typically solved to within a week of July 1 within
about 20 seconds.

Certain assumptions are built into the example spreadsheet and cannot be changed via the
interface, but by editing the VBA code and/or introducing new criteria any and all custom course
logistics can be accommodated. One assumption built into the provided spreadsheet is that each
group completes two experiments each term. Another assumption is that every experiment can be
operated every session. A session is simply a number used to define a window of time in which
students conduct an experiment. It could refer to a class meeting, a day, a week, or any other type
of time slot. The sheet assumes that the second round of experiments will be completed in the
same order as the first round, so if there are five sessions, a group assigned to Session 1 will
conduct their two experiments in the first and sixth time slots.

**Compatibility with Other Team Assignment Tools**

By changing the value in cell B8 from 1 to 0, team assignments can be turned off. This will
disable mutations that would change a student’s group number, so the algorithm will only try to
optimize lab scheduling and other logistical criteria. This allows the use of another tool such as
CATME Team-Maker for team assignment.

**Example addition of a criterion**

New criteria can be added without modifying the VBA code if the deviation from the target can
be calculated using Excel’s worksheet formulas and the mutations needed are already included in
the algorithm. For example, to add a constraint that students may be assigned to either
Experiment 1 or Experiment 2 but not both (e.g. if the two experiments are very similar), the following formula could simply be added to cell B15:

```
=COUNTIFS('Group Data'!B3:B20,="1",'Group Data'!C3:C20,="2") + COUNTIFS('Group Data'!B3:B20,="2",'Group Data'!C3:C20,="1")
```

This formula will count the number of groups that have been assigned Experiment 1 for their first experiment and Experiment 2 for their second experiment and add it to the count of the number of groups that have been assigned Experiment 2 for their first experiment and Experiment 1 for their second experiment. This total can then be given a target value of zero in C15, a weighting in D15, and the product of the deviation from the target value and the weighting can be calculated in E15 and included in the sum in E16. At this point, the algorithm will keep mutations that prevent groups from conducting both Experiment 1 and Experiment 2.

**Conclusions**

Team assignments and lab scheduling were completed using a custom evolutionary algorithm for a Chemical Engineering Unit Operations Course with 225 students enrolled for Autumn Semester 2018. Fitness of team assignment and lab scheduling were assessed using 10 criteria, and the first 8 criteria were all optimized to exactly their target values. For the remaining two criteria, deviations from the targets were acceptable or expected.

The algorithm took approximately nine hours to reach the final solution. The processing time is much shorter with fewer criteria or with smaller class sizes, and can take as little as a few seconds. Given the capability to easily optimize lab scheduling around multiple criteria, there is a temptation to add more and more complexity, so users must weigh the benefits of introducing additional criteria against the increased processing time required.

A simplified, working example of the algorithm has been provided in a publicly available Excel spreadsheet. Customizations for course-specific logistics or more sophisticated team assignment criteria can easily be added using simple Excel worksheet formulas. An option to disable team assignment has been provided to enable the use of other software-assisted team assignment tools.

**References**


