AC 2008-1961: A SEMI-AUTOMATIC APPROACH FOR PROJECT ASSIGNMENT IN A CAPSTONE COURSE

Mark Chang, Franklin W. Olin College of Engineering

Mark L. Chang is an Assistant Professor of Electrical and Computer Engineering at the Franklin W. Olin College of Engineering.

Allen Downey, Olin College of Engineering

Allen Downey is an Associate Professor of Computer Science at the Franklin W. Olin College of Engineering.

A Semi-Automatic Approach for Project Assignment in a Capstone Course

Abstract

This paper presents a semi-automatic approach to assigning students to project teams for a year-long, industry-sponsored senior capstone course. Successful assignment requires knowledge of at least individual project requirements, student skills, student personalities, and student project preferences. This mix of hard skills, soft skills, and interpersonal impressions requires human involvement to produce a high-quality assignment. The importance of faculty input often requires that the assignment process be labor- and time-intensive.

Our approach attempts to reduce the time required to perform this assignment by selectively automating parts of the task flow. An automated search uses a randomized greedy algorithm combined with local optimizations to explore a large space of solutions. Candidate "good" solutions are then presented to capstone faculty. Criteria such as skill set, student capability, and personality compatibility are applied by human evaluators to reduce the candidate solution set. These candidate solutions are then distributed to small groups of faculty to look for improvements using system-generated tables of options.

This approach leverages automation at appropriate stages while keeping the experts—the faculty—involved in the selection process. Our initial implementation has reduced the time needed to select an allocation by about a factor of three over previous manual approaches.

Introduction

As engineering programs at colleges and universities strive to make pedagogical reinventions, faculty are experimenting with active learning methods to bring authentic engineering experiences into the classroom. A prominent feature of many project-based learning approaches is the use of student teams to solve complex problems. One of the significant challenges is therefore the assignment of students to teams.

In an experience such as a final-year senior engineering capstone, the administrative burden of team formation is often exacerbated by the needs of a more complex, department- or college-wide capstone program. Issues may include larger teams, interdisciplinary needs of projects, satisfying external constituencies, budgeting, and more. These higher stakes make a high-quality team selection process even more important.

In this paper we present a semi-automatic approach for placing students onto project teams. The chief goals of using this approach are to save personnel time and increase the level of satisfaction for all users. The users for our system include students, faculty, and the capstone program.

The Team Formation Problem

The Franklin W. Olin College of Engineering requires all students to complete a two-semester,

senior-year, engineering capstone project course. This course is the culminating engineering design experience for our students, and strives to provide real-world engineering problems and experiences from paying industry sponsors. Each year, approximately 75 students participate in 13-15 projects, and it is the job of the capstone program to try and best assign these students to the projects.

Our capstone program began in the fall of 2005, and is heavily modeled upon the Harvey Mudd Clinic program.

Successful team formation requires knowledge of project requirements and student skills and preferences. Students are expected to work with external sponsors as well as faculty and each other, and they are required to manage a significant budget, so we need to consider "soft" skills like leadership, teamwork and communication along with more technical skills.

Many of the projects are multi-disciplinary. Most include mechanical, electrical and/or software components, and many involve areas such as industrial design, environmental studies, ethnographic studies, and business/entrepreneurship.

Effective team formation is important for the short- and long-term goals of the program. In the short term, if students feel that the allocation conforms to their preferences and interests, they are likely to have higher morale and motivation. In the long term, we expect a good match between projects and student skills to yield good project outcomes, which is important for sustaining external support for the program.

The data we use to form teams comes from several sources, including student transcripts and project descriptions from external sponsors. Information about student preferences comes from a survey we administer at the beginning of the academic year. Students are given a short description and presentation on each project. They have a few days to ask questions, investigate the projects and the sponsors, and to talk to each other. Then we ask them to complete an online survey.

The survey asks students to score each project on a scale from 1-5, where 1 indicates no interest, 5 indicates strong interest, and 3 indicates that the student would be willing, but not necessarily happy, to work on a project.

Students are also allowed to identify up to two other students they do not want to work with. To encourage students to use this option sparingly, we ask them to name another student only if they believe that being on the same team with that student would be detrimental to the success of the project.

We have used almost the same survey instrument for all three years of the program. The rationale for this simple preference survey is that students will self-select projects that they want to participate in, and have the capability to make an impact on.

From student transcripts, we have each student's major, courses taken and grades, and grade point average (GPA).

Some issues arise in using this data set in an automated fashion. Some projects are more popular than others; projects that attract few students constrain the space of feasible allocations. Also, there are usually a few students who are identified as an anti-preference by a disproportionate number of classmates. Assigning these students to a project also constrains the space of solutions. Nevertheless, in the three years of the program, it has always been possible to staff each project with interested students without violating any anti-preferences.

An important source of soft data is the knowledge of the people participating in the allocation process. The people "in the room" for our process include the capstone faculty, the Dean of Students, the Dean of Faculty, and the capstone program director. From classes and other interactions, the faculty have personal knowledge about each student. The Dean of Students often knows about student life histories and interpersonal relationships. The Dean of Faculty provides oversight of the process from the perspective of educational outcomes. The Program Director provides oversight from a programmatic perspective.

The next section explains how we use data from the survey and other sources to allocate students to projects.

A Semi-Automated Approach

Motivation

In a fully-automated process, we would formulate allocation as an optimization problem and use a program to search for an optimal solution, where "optimal" means that the solution satisfies all constraints (like the number of students on each project) while minimizing a cost, like violations of student preferences, or maximizing a complementary benefit.

There are several problems with this approach:

- It requires all relevant information to be encoded in a way that can be manipulated by the program. This is easy with "hard" data, like the results of the student survey, but impossible with "soft" data, like our knowledge of students' personalities.
- It requires all constraints and costs to be quantified. Again, this is easy for some of our goals, like assigning students to projects they are interested in, but either complicated or impossible for other goals, like assuring an appropriate mix of skills for each project.
- It requires all tradeoffs between conflicting goals to be quantified. There is no general procedure for making these kinds of tradeoffs; it is only possible to consider, and reach a consensus about, specific cases.
- It requires all participants to trust the system and believe that the outcome is optimal. This is only possible if the process gives participants a mental model of the space of possible solutions.

The faculty and students involved in the capstone have different and sometimes conflicting goals and values. There is no "optimal" allocation that will satisfy everyone. What is needed

is an allocation *process* that assures everyone involved that the outcome is an appropriate compromise that is as good as possible.

Our Approach

To achieve this goal, we propose a semi-automated approach that tries to combine the efficiency of an automated search with the user satisfaction of a human-centered approach.

Our approach iterates between two phases:

1. The first phase uses an automated search to find a pool of allocations that are good candidates according to a coarse cost function.
2. In the second phase the capstone faculty evaluate proposed allocations and either accept one of them or adjust the parameters of the automated search and generate a different set of candidates.

There are a number of advantages to this two-phase approach:

- It uses computers to do what computers do best, and humans to do what humans do best.
- It takes advantage of both the hard data, which can be encoded and used during the automated phase, and soft data, which may exist only in the heads of the users.
- It can deal with both hard constraints, which cannot be violated, and soft constraints, which can be violated at a "cost."
- It gives the users opportunities to fine tune the allocation to balance the needs of students, faculty, and the program as a whole.
- It allows the users to discover and modify constraints and costs as the process goes along.
- It helps the people involved understand what the space of possible solutions looks like so that their expectations are calibrated appropriately.

Regarding this last point, we have observed that different people approach this kind of problem with different mindsets. One classification divides people into "maximizers" and "satisficers." Satisficers generally try to find an *acceptable* solution; maximizers try to find the *optimal* solution. The approach we are proposing can help avoid conflicts between these mindsets.

For satisficers, the process takes significantly less time and produces a large number of high-quality candidate solutions to choose from. For maximizers, the use of a computer algorithm

to search a very large space of solutions can alleviate the anxiety of a fully-human approach by providing an overall view of the landscape of solutions. Also, starting with a small pool of candidate solutions, maximizers can look for local optimizations until they are convinced that no further improvements are possible.

Our implementation

The following is a more detailed description of the steps we followed.

- Automated search for feasible solutions: Using information from the survey of student preferences, we use a greedy algorithm with local optimization to generate a large number of good quality feasible solutions.

  The automated search is guided by a cost function that assigns a score to each candidate allocation. Undesirable features are assigned a "cost"; the total score for an allocation is the sum of all costs.

  Most constraints are enforced by assigning high costs for violations. For example, understaffing or overstaffing a project costs 10000 points; in practice this means that no solution with this feature will win, but the program can consider allocations that violate this constraint as it moves from one local optimum to another.

  Assigning a student to a project with a preference of 5 ("high level of interest and strong match of skill") is free; a preference of 4 costs 1 point; a preference of 3 costs 5 points; a preference of 2 costs 1000 points and a preference of 1 ("not interested, or no match of skills") costs 10000. An allocation that assigns a student to a project with preference 2 or 1 is considered unacceptable, but can be used to move from one local optimum to another. The program tries to avoid assigning students to projects with preference 3, but in our experience so far, it has not been possible to avoid putting a small number of students in that position.

  If two students who conflict are assigned to the same project, that costs 100 points. This weight reflects our desire to accommodate anti-preferences almost absolutely while still considering that violating an anti-preference might allow the program to explore a part of the solution space that yields a better global allocation.

  The program that generates solutions works in three phases:

  1. During the first phase, the program uses one of two probabilistic greedy algorithms to generate an initial allocation. One algorithm enumerates the students in random order and assigns each student to the available project with the highest preference. The other algorithm enumerates the projects in random order and chooses the student with the highest preference for the project, repeating until all students are allocated.

     Because the algorithms enumerate the students and projects in random order, and break ties at random, they generate different initial allocations each time they run.

Both algorithms generate allocations with an acceptable number of students on each project, and they tend to assign students to projects with high preference, but they make no attempt to avoid violating anti-preferences.

2. In the second stage, the program considers all possible trades (swapping two students) and moves (moving a student from one project to another) in random order. Any trade or move that improves the overall score for the allocation is accepted.

   Stage 2 is repeated until there are no moves or trades that improve the allocation.

3. In the third stage, the program identifies the students who are most unhappy, chooses one at random, and takes "desperate measures" to make the student happy, even at the cost of violating an anti-preference or making another student unhappy. The effect is to move the allocation out of a local optimum; the program then repeats Stage 2 to find the new local optimum.

   Stage 3 is repeated as long as it continues to find improvements. If a desperate measure fails to find a new optimum, the program discards the allocation and starts again with Stage 1.

As the program runs, it records all solutions with a global cost below a certain threshold. The longer the program runs, the more low-cost solutions it finds. In our experience, the program tends to find a few good solutions in 10-20 minutes; after a few hours it seems unlikely to do any better. Of course, we are making no effort to find a true optimum, and we wouldn't know if we found it.

Among the lowest-cost solutions, there tend to be repeating patterns: one project might be assigned the same team in many solutions, or a set of students might tend to be on the same team.

After the program has run for a while, we print about 20 of the best solutions and bring them to the next stage.

- Scoring candidate solutions: Capstone faculty evaluate possible allocations. If an acceptable solution emerges at this stage, the process can stop, but it is likely that additional constraints will be identified.

  We distribute hardcopies of the top-20 allocations to the capstone faculty for evaluation. Faculty advisers tend to evaluate their own teams, taking into account factors like the major, skillset and GPA of the students as well as personal information based on prior experience. For each allocation, faculty assign a score to the proposed team on a 5-point scale: 5 indicates a very strong team, 4 is strong, 3 is acceptable but borderline, 2 is unacceptable but possibly reparable, and 1 is unacceptable.

  During this process, faculty identify recurring patterns and problems. New constraints might be identified that eliminate some allocations from consideration.

  This stage can be decentralized; that is, the faculty involved don't have to meet to evaluate the candidate allocations. The faculty in charge of the allocation process (the authors of this paper) can aggregate responses from faculty advisers and other participants.

- Generation of more solutions: If additional constraints are identified, they can be enforced automatically by modifying the search code, or candidate solutions can be filtered by hand.

  We run the search program again with new constraints and select a new set of candidate allocations.

- Selection of best candidates: In a second round of scoring, capstone faculty identify a small number of candidates allocations that are either acceptable or nearly acceptable.

  During this stage, for the first time, faculty are all in the same room. By this time, they have a sense of what the search space looks like and an idea of how good an allocation is likely to be found. We have identified allocation features that are "showstoppers" and generated solutions that avoid obvious problems.

- Search for local optimizations: Capstone faculty are divided into teams; each team is given a candidate allocation to work with. For a given allocation there is often a particular problem they are asked to resolve. Their search is facilitated by automatically-generated tables of possible moves and trades. This stage ends when all teams decide that they cannot find additional improvements.

  The teams present their final allocations to the group. The faculty discuss the features of the proposed allocations with the goal of choosing one by consensus.

Evaluation

Our capstone program started in the 2005-6 academic year so we are offering it for the third time this year.

In the first two years we used a computer program to collate and print the data from the survey, but the allocation was performed manually by capstone faculty. We implemented the approach described in this paper for the first time this year.

The following sections describe our experiences in the first two years.

Year 1

In the inaugural year, the allocation process was completely manual. Without significant forethought we implemented a greedy algorithm based on student preferences, and then searched for local optimizations.

With a single piece of paper representing the survey results for each student, we allocated each student to the project with the highest score. At the end of this round, many projects were understaffed and a few were overstaffed.

Next we started an ad-hoc parallel search for better solutions. This often resulted in faculty making local trades between projects to form teams of 4-5 students per project. Other times faculty would ask the entire room for a student that met certain criteria, such as ranking their

project highly and having a particular major. Conflict resolution was done by individual faculty.

This approach yielded a relatively good allocation of students: of approximately 66 students, all but one was placed on a team they ranked 5 or 4 (highest and second highest). The remaining student was placed on a team ranked 3. However, the allocation process took a long time—approximately 96 person hours (8 hours by 12 people). The process was also prone to human error; at one point we thought we were done, then discovered that an anti-preference was accidentally violated. It took another 90 minutes to find a set of trades and moves that solved this problem.

At the end of the allocation, the faculty overwhelmingly agreed that a better process was needed.

Year 2

In the second year of the program, in an effort to provide more global guidance and a structured method for allocating students, we implemented a process similar to professional sports draft.

In round-robin fashion, each faculty member chose the best student for their team. This process, while seemingly appropriate, was quickly found to be flawed.

After a few rounds we reached an incomplete allocation where many projects were understaffed, but the remaining students could not be assigned to the understaffed projects without violating student preferences.

The draft algorithm fails because it solves the easy part of the problem first and leaves the hard part for last. This violates a basic heuristic of allocation: "Pack the big rocks first."

Students with good academic records, interest in several projects, and few anti-preferences are "small rocks;" in a draft algorithm, they are likely to be chosen first. Students with weaker records, interest in few projects, or more anti-preferences are "big rocks." If they are left until the end, there will be nowhere for them to go.

To make things worse, this process puts the faculty in a competitive mindset, with each one trying to select the best teams for their projects. This makes trading more difficult.

After a few rounds, we were stuck in a locally-optimal solution that was far from globally optimal. There were few small, local changes that made things better; we needed to make big, non-local moves to get to another part of the solution space.

But in the competitive atmosphere of a draft algorithm, faculty are less likely to accept trades that involve giving up a "good" student (greedily selected early in the process) for a weaker student (left until the end). This problem is compounded because in the early rounds, faculty associated with a popular project are able to assemble a "dream team," which raises

their expectations unreasonably.

The result was a qualitatively worse solution than the previous year. Eight students were placed on a project they ranked 3 (compared to 1 in the previous year). The process took as long or longer (about 8 hours) and seemed more frustrating.

We left the room with the feeling that there were better solutions we could not reach. This feeling was confirmed when we developed the algorithm proposed here; using the data from Year 2 for development and testing, we were quickly able to find solutions that allocated only 3 students to projects ranked 3. Faculty who evaluated these solutions agreed that they were qualitatively better than the allocation we generated manually.

From this experience, it was clear that we needed to refine the process and introduce some degree of automation.

Year 3

For the third year, the authors prepared the approach detailed in the previous section.

The process took much less time—fewer than 40 person hours. Capstone faculty had an initial one hour meeting to perform the parallel scoring, then met again after new candidate solutions were produced to determine the final allocation. The number of person-hours is still large, but the process was perceived as less stressful and more satisfying.

More importantly, the allocation we generated was met with high satisfaction from both faculty and students. Almost all students were assigned to projects they ranked 5 or 4; only one was assigned to a project ranked 3.

Exportability

The authors expect this approach to be useful in other contexts where assigning students to teams is necessary. In exporting this approach, we must take into account scalability. Our program is small—we assign about 75 students to about 15 teams—and our process depends on the detailed familiarity of the faculty with the students.

If the ratio of students to faculty were much larger, the first phase of manual optimization might be less valuable for improving solutions, because the faculty would have less "soft" data about students. But this phase might still be helpful for giving the faculty an overview of the space of possible solutions, which makes it easier to achieve consensus and the feeling that the chosen solution is about as good as possible.

If the number of projects were much larger, the second phase of manual optimization might be less effective because of the increased number of possible moves and trades. For example, in our process, 10 faculty are divided into 3 teams, each attempting to optimize a candidate allocation of 75 students onto 15 teams. If the number of projects, students and faculty were much larger, it would be difficult for the teams to find local improvements.

One possible solution would be to introduce hierarchy into the human optimization of candidate allocations. Instead of groups of faculty assessing all teams, one could assign mechanical engineering projects to mechanical engineering faculty to optimize. However, this would restrict the faculty to moving students between a subset of teams, and might not work as well for multidisciplinary students and projects. The resulting solutions may then be less globally optimal, as it would be difficult for any group of faculty to have a global view and understanding of the full allocation.

In either case, scaling the problem size up makes the use of a semi-automated system more attractive. The use of a faculty-driven parallel search process (as in Year 1) becomes more difficult as the size of the problem increases. The use of a draft model (as in Year 2) also becomes more difficult as it gets harder to find, and get faculty to accept, the big moves that are necessary, toward the end of the process, to move from a local optimum to a more globally optimal solution.

Conclusions

The authors present a semi-automated approach to allocating students onto project teams for a senior capstone experience. The allocation is a high-stakes endeavor for all constituents, as increased student agency and happiness can lead to strong motivation and a successful project. Our approach is novel as it selectively automates parts of the process while keeping the human operators in the optimization loop at appropriate times. Using this approach, we have demonstrated initial success not only in reducing the time required to perform the allocation, but also improving the quality of the allocation.