Proceedings of Student-Faculty Research Day, CSIS, Pace University, May 5th, 2017

A Greedy Approach for Assignment of Student Groups to Projects

Monali Joshi and Jigar Jadav
Seidenberg School of CSIS, Pace University
{mj92314n, jj03171w}@pace.edu

Abstract - today, in the growing industry of technology, it is a critical task to assign teams/groups to projects that would be effective and productive in order to get a successful outcome. The assignment of teams becomes a more difficult task when it comes to forming group of students for projects because it is crucial to form a diverse group which includes students with distinct capabilities and interest areas who are definitely not biased on the basis of friendship. This paper describes an algorithm designed to bolster auto assignment of groups to the pre-determined projects that ensures the suitability of the students to the projects.

Keywords – greedy search, algorithm, top-down approach, category, skillset, backtracking

I. Introduction

Collaborative teamwork is trending and has replaced independent work. Collaborative work leads to easy management of time, money, and resources and also promotes less stress as compared to individual projects. In the era of education, each student comes with a distinct set of skills and knowledge, which may lead to excellent results with shared experience and knowledge. Difficult projects become easy to tackle because of pooled work [1]. Individuals may face difficulties due to limited experience while group projects enable the team members to expand subject knowledge and potential. Discussion in collaborative work leads to creative ideas and creativity. Independent work may be fun at times, but, some projects requires vast strength of distinct knowledge and it is rare that only a single person has all the strengths needed.

A good team is the first step towards an outstanding outcome. A bad assignment may lead to poor performance. A particular skill set is required for each project in order for it to be executed in an effective manner. It is critical to assign a group of students who combine all the necessary skills. It is not feasible to have all students with the same skill set in a single group. Diversity in the group may result in improved problem solving tactics, intensified creativity, and seamless execution of the project. In addition, the personal choice of students is also an important factor because they will not be able to put 100% efforts if the project is not of their interest.

Determining the strategy to form teams is a critical task. There are number of ways teams can be formed. Firstly, the students can choose their teams on their own. If the students are allowed to form the groups themselves, it is highly possible that the groups will be formed on the basis of internal connections and friendship. Friendship is sometimes based on gender, ethnicity, race or similar interests. So, this approach most probably will form biased groups and would eliminate the chances of learning from students who have distinct thinking patterns and different backgrounds. The students who are not very social might be left without any group. It is likely that the leftover students will have to form a group of their own, even if they don't want to, because they were not accepted by any of the groups formed originally. This situation can sometimes have an adverse effect on the confidence level of the leftover students.

Secondly, the professor can assign the teams manually. If the professor is to decide the groups manually, it will be very difficult to satisfy every student's choice and skill set. Also, the professor will have to spend a significant amount of time on the assignment and the students may not be happy with it. This approach is feasible if there are only a few students and the number of projects is in the single digits. It is not necessarily the case that all of the students will be happy with the assignment. The professor will wait to wait until all the students have submitted their choices to avoid the repetition of the assignment process. Also, students are likely to change their preferences. This creates stress at the end moment. There is a possibility that the students would feel that the groups are biased.

Lastly, teams can also be assigned in a random fashion. If the assignment is done in a random manner, it may ease the process. But, it is possible that certain projects may be assigned a good team and the outcome of the project is outstanding, but it is also possible that certain teams will not have the required skills to proceed. For instance, one or two teams might have sound developers, technical writers, and a leader while the other teams may lack one or more of the skills. So, the project may fail due to a lack of adequate skills. Even after spending a significant amount of time on the assignment, all of the approaches listed above may lead to a disappointing outcome of the projects [2]. So, random assignment is the worst option to assign the groups. Also, this approach will initially save a significant amount of time but, later on, the professor will probably spend more time switching students among groups because there is a good chance that more students will be unhappy if the assignment is random.

Auto assignment provides an effective solution to this problem. A computer program can ease the process and also take care of all the possible problems listed above. There is a distinct set of algorithms which can be used to address this problem. A greedy algorithm is the best fit for the assignment problem. This paper describes a computer program that was developed in core Java. The program implements a greedy algorithm to solve the problem of student assignment. The assignment is done on the basis of work experience, project preference, skill set, and location of the student. The program follows a top down approach and attempts to assign the most suitable student as the team leader first and then other students are assigned in the group. The algorithm focuses on combining students with programming skills, technical writing skills, presentation skills and local availability. Also, the program

II. Greedy Algorithm

A greedy algorithm has long been used for assignment problems both in academic and non-academic settings. In 1994, Chinneck, LaPorte, and Carter developed an algorithm called EXAMINE to schedule exams on the basis of room availability. The algorithm followed the heuristic greedy approach. The algorithm focused on assigning a number of available time slots and rooms to distinct exams. The algorithm was developed in such a way that no student will have more than one exam at the same time. [3] Also, a group of students at Rutgers-Camden implemented an algorithm based on a greedy approach in Pascal to assign students to projects. This algorithm's assignment was made on the basis of availability, preferences, and background knowledge of the subject of the students.

The program provided three possible solutions, out of which the best one can be chosen with little adjustment so that all groups will have at least two students with the same timings. Since there were courses of computer science with different schedules, the program focused on the timing compatibility. For collecting the data, a survey was performed, which gathered information on students' schedules, preferences, and levels of familiarity with the nature of the project they will work on, on a scale of 1 to 10. The only problem was that all the information provided by the students had to be computer readable. So, the information collected had to be entered into a computer readable file. The program helped with the formation of groups for four years [1].

Greedy algorithms work best with optimization problems. A greedy algorithm is effective in providing an optimal solution to a problem. The traditional dynamic programming implements a bottom-up approach while a greedy algorithm is implemented in top-down fashion according to which it choses greedy options one after another. Like any other algorithm, a greedy algorithm has advantages and dis-advantages both. It is very easy to analyze the runtime of a greedy algorithm as compared to other algorithms like divide and conquer. A greedy algorithm provides flexibility to choose the solution that looks best at the iteration and it is sometimes more efficient than dynamic programming. It is easy to calculate the time complexity with a greedy approach. In each stage or iteration, it chooses the best possible solution.

A greedy algorithm can be broken down into three steps, which are as follows.

1. Select – the greedy approach chooses the option from the criteria and returns its value in output.
2. Feasibility Check – the algorithm checks if the selected solution can result in the most feasible solution.
3. Solution – it checks if the solution is obtained.

The general pseudo code for a greedy approach can be written as follows.

```
FROM n inputs

{
    INITIALIZE solution

        WHILE n is NOT NULL
        {
                        SELECT element
            {
                                IF feasible
                GET element
            }
        }
}
```

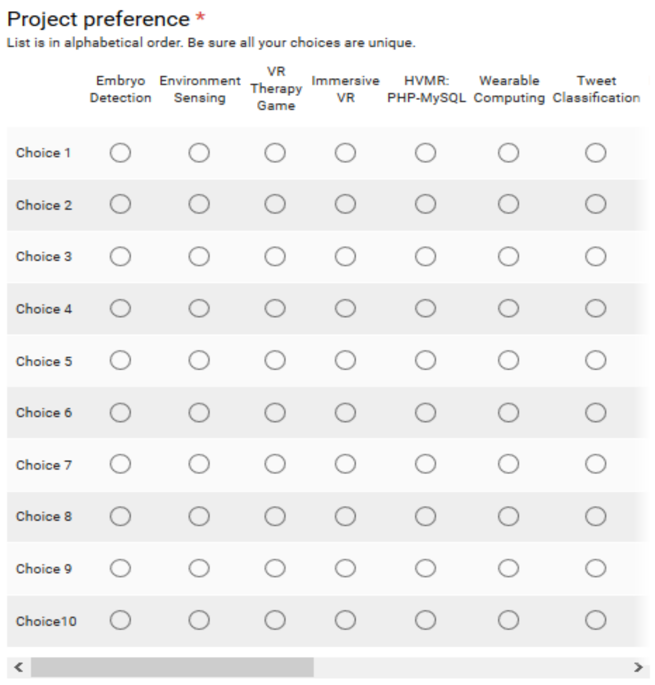
A greedy algorithm has several advantages over other algorithms, such as simplicity (because it is easy to code-up), and efficiency (because it is easy to implement a greedy algorithm more efficiently as compared to others). Also, most of the algorithms focus on making choices based on a global perspective while a greedy algorithm focuses on choosing the best local solution. But, it has some challenges too. Coding up a greedy algorithm is easy, but before the coding is done, determining the right design for the approach is a critical task. Despite of the solution not being globally optimal, but it may be an impressive one despite its being sub-optimal. Scheduling and assignment problems can be solved easily by utilizing a greedy approach.

Dis-advantage of greedy algorithms is sometimes a greedy approach may not lead to the optimal solution. Instead, it may lead to the worst possible solution as well. Also, it is a critical task to define the correctness of the algorithm and when the solution is correct, it is still difficult to prove that it is correct. But above all the dis-advantages, a greedy algorithm is an effective solution to any optimization problem.

III. BACKGROUND WORK

Computer Science and Information Technology students at Pace University have a semester long capstone project. It takes the form of a web assisted course. The students enrolled in this course are required to attend three in-class meetings during the entire semester and give two presentations, one at the mid-semester and the other at the end of the semester.

An online survey tool allows the students to register the information. Collecting the data directly into a Google form saves a significant amount of time because no manual data entry is involved. The information includes student information such as name, expected year of degree completion, work experience, location, project preferences, and expected number of meetings the student will attend. The students will register their preference on a scale of 1 to 10, where 1 is the first preference and 10 is the last preference.[4] This is represented in the following figure.

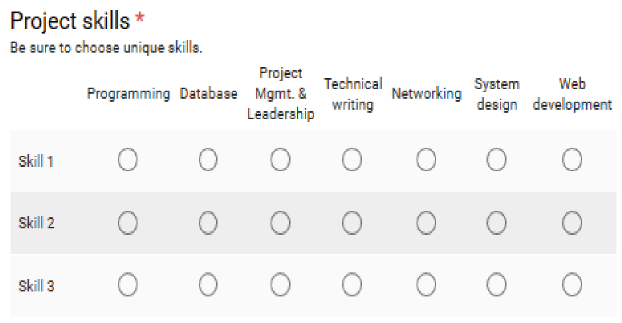

Project preference *
List is in alphabetical order. Be sure all your choices are unique.

	Embryo Detection	Environment Sensing	VR Therapy Game	Immersive VR	HVMR: PHP-MySQL	Wearable Computing	Tweet Classification
Choice 1	○	○	○	○	○	○	○
Choice 2	○	○	○	○	○	○	○
Choice 3	○	○	○	○	○	○	○
Choice 4	○	○	○	○	○	○	○
Choice 5	○	○	○	○	○	○	○
Choice 6	○	○	○	○	○	○	○
Choice 7	○	○	○	○	○	○	○
Choice 8	○	○	○	○	○	○	○
Choice 9	○	○	○	○	○	○	○
Choice10	○	○	○	○	○	○	○

Figure 1. Project preferences in Google Form

The form also collects information on students' skills. The list of skills is provided by the customer on basis of their needs in relation to the project. The customers are given a list of nine programming languages and twelve other expertise options. They are required to scale each language and each option on a scale of 1 to 10 based on their requirements. The customers are required to indicate the number of students they want in their project. It is not necessarily the case that all the projects require a team of 4-5 students. Some projects are a continuation from previous semesters. Such projects can be completed by a team of two students, or even just one student. Students may select 3 most suitable skills from a list of 7 skills, which are Programming, Database, Technical Writing, Networking, System Design, Web Development, and Project Mgmt. & Leadership. This information makes it easy to categorize the student as a leader, developer or a technical writer.

The students are provided with predetermined options. This makes the algorithm development easy because the developer does not need to worry about spelling mistakes and acronyms. This is represented in the following figure.



Project skills *
Be sure to choose unique skills.

	Programming	Database	Project Mgmt. & Leadership	Technical writing	Networking	System design	Web development
Skill 1	○	○	○	○	○	○	○
Skill 2	○	○	○	○	○	○	○
Skill 3	○	○	○	○	○	○	○

Figure 2. Project Skills in Google Form

The student may enter an additional skill. This information serves as the basis of the algorithm which assigns the most suitable students to the project.

The same information is converted into an Excel sheet. This Excel sheet has columns that represent each question in the form. It has two worksheets, Form Responses and Projects. The Form Responses sheet contains 21 columns with the information which the students have entered while the Projects sheet contains 2 columns which provide information on project names and the number of team members who can be accommodated in the project. The Form Responses sheet is auto generated while the user is supposed to create the Projects sheet. This excel sheet serves as the input to the algorithm.

IV. Program

In order to run the algorithm, the first requirement is the data. Input to the program is an Excel sheet described above, which contains the data extracted by the Google Form which collects information from students beforehand.

The program is modular and divided into 12 classes. It contains distinct functions. It manages a configuration file, which allows the user to manage the number of project team members, and the category of the students, i.e. developer, leader or technical writer. Students with skills in Programming, Database, Networking, System Design, and Web Development are assigned to the developer category. Project Mgmt. & Leadership skill falls into the category of leader and Technical writing skill falls into the category of technical writer. It also manages information on local cities. The configuration file makes the program very easy to use because the user can change the information in it very easily. This does not require much technical knowledge. Change in the configuration file does not change the main code at all. So, even a person with very little technical knowledge can use the program very easily.

The program also manages the properties file. When the end user runs the program for the first time, the information provided is saved in this properties file. So, the user does not have to enter the details again and again if the same file is going to be used for the next execution.

When the program is executed, it checks for the skills first (focus is on assigning the leader first on the basis of experience and location) and assigns the category to the student, which is either leader, developer, or technical writer. It then checks the preferences of the student and if the project is available then the project is assigned and if not, it will check for the next choice of the student. After the entire loop has iterated, if there is no project available for the student then it will change the category of the student and repeat the process until the student is assigned to a project.

The program produces more effective results when each team is required to be assigned more than four students. Clearly, when more students can be accommodated in a team, more students will be assigned to the projects listed among their first three preferences. When the team leader is not assigned to a team, the program assigns one fewer student than the total number of team members expected for that team. This makes the manual adjustments smoother.

A. Reading Data

Input to the program is the Excel sheet. The program requires the user to choose the location of the excel file and the names of both work sheets in the selected excel file as input. Initially on execution, the program starts reading all the configurations from the configuration properties file and saves them into the Java collection object named HashMap. HashMap works faster when iteration through the complete list is required. Unlike lists, HashMap is more effective in locating the elements. It then reads the Excel file and creates a student object for each student based on individual rows. For reading the data from the Excel file, the Apache poi library is used. It is an open source library, which allows the programmers to integrate java programs with MS Office. This library allows the display, modification, and creation of MS Office files. The pseudo code for reading the data is as follows.

```
GET location of the excel FILE

    READ EACH row
    CREATE object of EACH row

PASS student object to ALGORITHM
```

B. Algorithm

As soon as the object of the student is created, it is passed to the algorithm, which processes the object based on the configuration defined in the configuration file. In this process, the program initially gets all the skills of the student and assigns the category to the student. For instance, if the student has development skills then the category will be developer. If the student has mentioned leadership as a skill then the category will be leader, but this does not assign the student as leader or developer of any project. This is just category assignment.

By default, the program first fetches the category “leader” and checks if the student has the skill ‘Project Mgt. & Leadership’. If not, then it checks for technical writing skills and finally for developer skills. This preference can be easily changed in the configuration properties file. A good team will function very well if it has a leader assigned to it. Without a team leader, the team may fail to work in an organized way. This is why the program assigns a team leader first. But, it is quite possible that there are fewer student profiles with leadership capacities than the number of projects. For instance, there is a total of 14-15 projects, but only 9-10 students have work experience and have entered leadership skill in the skill set. In this case, the instructor can manually choose the most suitable student from the team as the leader. The following pseudo code explains the category assignment.

```
WHILE Category list is NOT EMPTY

    Fetch Category
        Check SKILLS list
        IF match
            ASSIGN the Category
            Break;
        ELSE
            CHANGE Category;

END WHILE
```

The skill set of the student is stored in a list which is iterated through the list of preferences of the student. The program checks for the space available in the project. If yes then the same object is added to that project. If the project has no space then the iteration moves to the next choice of the student. This process is continued until the student has been assigned to a project or until the choice list is empty. The same iteration is performed for all the student objects.

For any program that involves accessing all the elements and choosing one, backtracking is the most critical part. Backtracking allows the re-evaluation of the previous solutions if an optimal solution is not reached. Non-chronological backtracking is an effective approach because it retracts the most suitable solution while chronological backtracking only retracts the most recent solution which may not be a possible solution. Most of the optimization problems implement non-chronological backtracking mechanism. There are different algorithms that can be integrated into the program for backtracking. A general pseudo code for backtracking is as follows.

```
WHILE there are un-attempted solutions

    CHECK the solution
    IF the solution can be implemented
        SELECT the solution
    End IF

END WHILE
```

The program implements a recursive backtracking mechanism on the category in order to assign the maximum number of students to the projects. The first category fetched is leader. If the student does not fit into any projects despite having leadership skills, the category is changed to technical writer. If the student does not fit into any project then the category is changed to developer. The following pseudo code explains this.

```
READ student OBJECT

WHILE Record is NOT NULL
    READ Skill Set
        ASSIGN Category

    WHILE Category is NOT NULL
                 WHILE Choice is NOT NULL
                    IF SkillPosition = EMPTY
               ASSIGN project to student
               BREAK
              ELSE
               GET NEXT Choice
                    END IF
                 END WHILE
                        GET NEXT Category
    END WHILE

END WHILE
```

C. Output

Finally, after iteration of all the rows, the program generates a Json object through HashMap, which has a list of projects and teams. For output, the program uses html and angular js. The following figure demonstrates the user interface of the program.

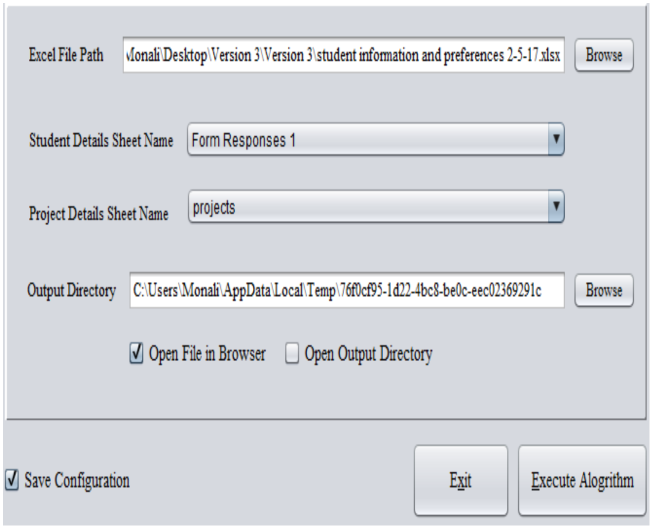

Excel File Path: Monali\Desktop\Version 3\Version 3\student information and preferences 2-5-17.xlsx
Student Details Sheet Name: Form Responses 1
Project Details Sheet Name: projects
Output Directory: C:\Users\Monali\AppData\Local\Temp\76f0cf95-1d22-4bc8-be0c-eec02369291c
☑ Open File in Browser ☐ Open Output Directory
☑ Save Configuration

Figure 3. User Interface

The user is required to enter the location of the Excel sheet, the sheet names, and the location where the user would want to save the output file. The location of the file and the names of both the worksheets are mandatory input. The defalult output directory location is displayed and the user can change it. The user is also given an option to open the output directory. The configuration which is entered for the first time can be saved by checking the save configuration option.

The program iterates through the HashMap list, which contains projects and teams, and copies all required files in the temp folder and finally runs an html page with the default configuration browser of the system. The output page shows the name of the projects, the number of team members on the project, and the students assigned to that project. The following figure shows how the team members on a particular project are displayed in the output.

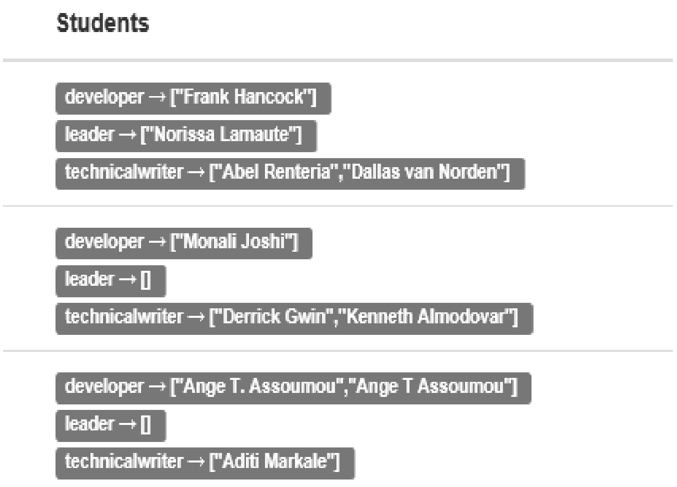

Students
developer → ["Frank Hancock"]
leader → ["Norissa Lamaute"]
technicalwriter → ["Abel Renteria","Dallas van Norden"]
developer → ["Monali Joshi"]
leader → []
technicalwriter → ["Derrick Gwin","Kenneth Almodovar"]
developer → ["Ange T. Assoumou","Ange T Assoumou"]
leader → []
technicalwriter → ["Aditi Markale"]

Figure 4. Output screenshot

V. RESULT COMPARISION

Manual Assignment of students to the capstone project is a critical task which involved a considerable amount of time. Manual assignment allowed 74% of the students to be assigned to a project out of the first three preferred projects. 31% of the students were assigned to their first preferred project. 20% and 12% of the students were assigned to first, second and third preferred projects, respectively. Also, taking care of the preferences of almost 50 students is tiresome.

This percentage is increased with the program. Also, it takes only few seconds to generate the output. The program assigned 79% of the students to the projects out of the first three preferences. The algorithm assigned 57% of the students to their first preferred project. 19 % and 3% of the students were assigned to their second and third preferred projects, respectively. Each team was assigned five students except for the team that required only one or two students.

With a little change in the configuration, the algorithm was able to assign 85 % of the students to the projects out of the first three preferences. 51 % of the students were assigned to their first preferred project. 18% and 14% of the students were assigned to their second and third preferred projects, respectively. Each team was assigned five students except the team that required only one or two students. In addition, the program saves a significant amount of time that the professor had to spend on assigning the teams to the projects at the starting of every semester.

The program works more effectively as number of projects and the number of students to be assigned to the projects increases.

VI. CONCLUSION

The output of this project is a computer program developed in core Java. It aids the student assignment to the projects. It attempts to assign the students based on their choices but the final solution may need a little revision [to adjust the compatibility of the group. The program assigns the groups effectively. Depending on the configuration, sometimes there are few students who are not assigned to any projects. The list of the leftover students and their project preferences is amended at the end of output, which makes the manual assignment of those students an easy task.

VII. FUTURE WORK

The current algorithm does not take the timestamp variable into consideration. The future versions of the algorithm can incoporate a logic on timestamp variable and make the algorithm more refined, which may result into an effective outcome. Also, it will be fair to assign a project to the student who has given the preferences earlier. The current link to the preferences form allows the students to give preferences more then one time. The algorithm can include a mechanism that eliminates duplicate students on the basis of timestamp.

Also, the program does not implement any backtracking mechanism on leader assignment. The future version of the algorithm can incorporate refined logic for the assignment of leader. The future version can focus on eliminating the little manual efforts which are needed with the current version of the algorithm.

.

REFERENCES

[1] Michael, A. Redmond “A Computer Program to Aid Assignment of Student Project Group.” Proceedings of the 32rd SIGCSE Technical Symposium on Computer Science Education, 2001, Charlotte, North Carolina, USA, 2001

[2] Douglas, Allison M., "A modified greedy algorithm for the task assignment problem." (2007).Electronic Theses ad Dissertations.Paper369.http://dx.doi.org/10.18297/etd/369

[3] Carter M.W., Laporte G. and Chinneck J.W. (1994). A General Examination Scheduling System. Interfaces, 24, 109– 120

[4] Robert D. Plumley, Charles C. Tappert. “A greedy Algorithm Assignment of Capstone Course Students to Teams and Projects Using Skill Heuristics. Seidenberg School of CSIS at Pace University.

Augustine, D. et al, Cooperation works! Cooperative Learning can benefit all students. Educational Leadership 7 (1989)

Cooper, J. Group formation in cooperative learning: what the experts say. Cooperative Learning and College Teaching 7 (1996), 14-15.

DePuy, G., Whitehouse, G., 2000. Applying the COMSOAL computer heuristic to the constrained resource allocation problem, Computers and Industrial Engineering, 38, 413-422.

DePuy, G., Whitehouse, G., 2001. A simple and effective heuristic for the multiple resource allocation problem, International Journal of Production Research, 39 (14), 3275-3287