Preference-Driven Assignment Decision-Support System for Multiple Sets of Pairs

By

STEPHEN J. GILMORE

A Senior Project submitted in partial fulfillment of the requirements

for the degree of Bachelor of Science in Industrial Engineering.

California Polytechnic State University

San Luis Obispo, CA

Graded by: ________________________________ Date of Submission: ________________________

Checked by: _______________________________ Approved By: ______________________________

Table of Contents

Table of Figures

Table of Tables

Abstract

This project focuses on designing a computerized decision support system (DSS) to assist the Cal Poly Alumni Association with scheduling their Mustang Mentoring day. The current scheduling process is time intensive, error prone, and does not consider student preferences. The decision support system optimally generates multiple sets of pairs of students and mentors and schedules them avoiding conflicts and duplicates. The system was developed in Microsoft Excel using its Solver add-in. Excel was selected as the system platform also because of its relative popularity compared to alternative platforms. The DSS utilizes integer linear programming to maximize the total student satisfaction with their mentor assignments. The DSS replaces a process that used to take over an hour to one that takes less than ten minutes. The original process would have required employing 6 workers, while the new process requires only 1 worker to help generate assignments.

Introduction and Background

The Cal Poly Alumni Center hosts an annual event called the Mustang Mentoring Day to connect students with Cal Poly alumni for a day of networking sessions. In previous years, the event has only been open to the schools of architecture and engineering, however, the Alumni Association would like to open the program up to all six of Cal Poly's colleges during the 2011-2012 school year.

The event consists of various activities to foster mentoring relationships between alumni and students. Alumni are pre-assigned into groups of 10 or less per group prior to the day of the event. As students check in for the day, they are randomly assigned to groups with the alumni. The beginning of the event consists of an introductory presentation and ice breaker activities. Students and alumni are then divided into their assigned groups for a "Speed Mentoring" session. It works in a similar fashion to speed dating, where each student gets to talk to each alumnus in their group for a few minutes. Students then rate the top five alumni that they want to meet with later in the day. During lunch, a volunteer from the Alumni Association must create assigned pairs of student and alumni for two separate one-on-one discussions. The volunteer must pair each student with a single alumnus for each of the sessions, avoiding conflicting assignments (assigning a student to two alumni or vice versa), as well as avoiding duplicate pairs (assigning the same pair twice).

The current assignment process can take over an hour per college. This means that for 6 colleges, the Alumni Association would need 5 to 7 labor hours to make all the assignments. To do this during lunch time, it would require at least 6 employees or volunteers. The alumni center does not currently have the resources to employ that many extra people. The alumni center looked into the available options they had and concluded that a computer program could be designed that would allow performing the task without hiring additional workers. This project covers the development of a computer based decision support system (DSS) which makes the assignment process much more efficient and less error prone, and is based on optimization methodology. The deliverables include:

1. A thorough explanation of the model behind the computer application
2. A digital copy of an assignment generating application
3. An illustrated manual with step-by-step instructions for operation of the program
4. This document, a final report that fully describes all aspects of this project

This project began with a review of all of the published documents which could have had any pertinence to the assignment solution. Detailed documentation of the research conducted for this project is in the following section titled "Literature Review."

The following is a list of desired outcomes and features determined by the Alumni Association for the final solution:

- Design the program to be robust and work under many situations
- Design the program to replace the current manual method
- Decrease the amount of labor hours required to generate assignments
- Design the program to consider student preferences for mentors
- Design a "guaranteed" error-free solution

Literature Review

This portion of the report contains an outline of all of the articles and documents that were found to be relevant to completing this project. Detailed bibliographical information is available at the end of the report.

Mustang Mentoring Day Schedule

The Cal Poly Mustang Mentoring program is an annual event that has been available for students in the colleges of Engineering and Architecture and Environmental Design for the past several years. The structure of the event works nearly the same way each year. Prior to the event students and alumni fill out a questionnaire and are invited to attend on a first come first serve basis. On the day of the event, students and alumni check in and attend a welcome presentation to prepare them for the event. After the introductory presentation is over, students and alumni are split into their respective colleges for the rest of the day. The college specific events kick off with an ice breaker activity to help facilitate communication between students and alumni. After that, students and alumni are split into smaller groups consisting of 8-10 alumni and 8-10 students. The students and alumni participate in a "Speed-Dating" type of exercise where the students are given a few minutes to meet each of their group's alumni one-on-one. Students are then instructed to write down the top 5 alumni they would like to meet with after lunch. While the students and alumni are at lunch, an employee from the Alumni Association quickly assigns student-mentor pairs for the next activity. Each student must be paired with two different alumni for the two one-on-one sessions in the afternoon. In 2011, the assignment process took nearly an hour for one employee to complete assignments for the college of engineering and caused a slight delay.

The assignment process is slow and tedious in its current form. The Alumni Association employee starts with a premade Excel table that contains all the mentors and students in each speed mentoring group. The employee then takes all the students alumni rating cards and inputs their selections into the spreadsheet. Finally, the employee must look at the table and by hand, and make student-alumni while taking care to avoid conflicts, duplicates, and incorrect numbers of pairs. The current process does not give any preference to the assignments by the number rank that the student assigned to each of the alumni they requested.

As soon as lunch is over, students and mentors are shown their assignment results and they break off to meet with each other for two separate sessions. Once those sessions are completed, a closing presentation is conducted.

Research

Operations research is defined as, “The application of advanced analytical methods to help make better decisions (1).” Operations research is used to create solutions for analyzing complex situations to aid in decision making, process improvement, and efficient data analysis (2).

There are five steps to conducting an operations research study (3):

1. Define the problem and gather any pertinent data
2. Formulate a mathematical model to represent the problem
3. Create a computer-based procedure for calculating the solutions to the model from step #2
4. Test and evaluate the model, make changes as necessary
5. Work with management to develop a plan for use of the model as prescribed by management
6. Execute

Before a math model can be selected or a computer procedure can be programmed, the type of math model must be decided upon as well as the computer application or program that will perform the calculations.

The Classical Assignment Problem: Overview and Applications

The type of operations research problem that will be solved is commonly referred to as the 'assignment problem.' Assignment problems are defined as, "A special type of linear programming problem where assignees are being assigned to perform tasks (3)." The classical assignment problem is defined as:

> "The personnel-assignment problem is the problem of choosing an optimal assignment of *n* men to *n* jobs, assuming that numerical ratings are given for each man's performance on each job." (4)

Translated to reflect this project:

> 'The mentor-assignment problem is the problem of choosing an optimal assignment of *n* students to *n* mentors, assuming that the students give numerical ratings for their preference for each mentor.'

Applications of the final solution for this project could be used for other operations research problems. The final model will be adaptable for any problem which consists of *n* assignments to be divided amongst *n* individuals where each individual can only be assigned to one task.

- Assigning chores to roommates to maximize "fairness"
- Assigning the best combination of team members to events
- Transportation of a product from multiple distributors to multiple retailers to minimize cost
- Employee scheduling to minimize cost for tasks which require 1 employee at a time
- Assign Marriages (5)

Mathematical Approaches

There are two solution methods for the assignment type of problems in *Introduction to Operations Research* (3). The first is to modify a standard linear model used for finding optimality. The only special needs of this model are that each assignee-task combination possible is assigned a unique binary or discrete variable with a value of 1 for an optimum assigned pair and 0 for a pair which will not be assigned. Linear formulations can be used to find a maximizing or minimizing solution. Additionally, a linear formulation can easily and efficiently generate multiple sets of pairs which don't have any repeated pairs between sets.

The second method is called the Hungarian Method. The Hungarian method is used for finding a minimizing solution to an assignment problem. So, if each student in a group were to rank their potential mentor pairs from 1 to 5, with 1 being their preferred choice, then the minimizing solution found by the Hungarian method is optimal. One feature of the Hungarian method is that it has the ability to rank all the possible assignments solutions that can be created. The exact mathematical formulation for doing so is outlined in *An Algorithm for Ranking all the Assignments in Order of Increasing Cost* by Katta Murty (6) as well as in *Introduction to Operations Research* (3). The benefit of this method is that multiple sets of pairs can be created. The downside, however, is that each pair must be compared set to set to make sure that there are no conflicts between groups. This doesn't necessarily rule out the Hungarian method as a viable option, but it does mean that the Hungarian method may require more resources to find a multiple solutions than a properly formulated linear solution.

One limitation to both of the aforementioned methods is that if the number of assignees does not equal the number of tasks, a dummy task or assignee may be required to solve the problem without error. In the context of this project, there may be an issue of there is ever a time where there are more mentors than students or vice versa.

There are other advanced methods for solving assignment problems available. These methods are more challenging to formulate and use heuristic methods based on genetic algorithms. “A Genetic Algorithm for the Generalised [sic] Assignment Problem” (7) references a few of the intermediate solution methods and builds off of them to develop a more complex method to find the minimum cost with an ‘intelligent’ probabilistic search algorithm. “The Group Assignment Problem” is a journal article which discusses the clustering and group clustering methods for complex assignment problems with small differences in data (8). The clustering method would likely be too complicated for the solution required by this project.

Computer Program Selection

There are many computer applications available to solve assignment problems. For this project, the simplest program possible should be chosen. Calculation speed and efficiency will likely be negligible due to the small size of the assignments. The Alumni Office would prefer to keep all of the applications on a Microsoft Office application to reduce cost and complexity. Within the Microsoft Office Suite, the two most popular programs available that can be programmed are Access and Excel.

Excel is the simplest to use and the average Alumni Association employee is more familiar and adept with Excel over Access. The biggest benefit of using Excel and a linear formulation is that Excel includes a built in function to solve linear type programs called Solver (9). The downside though, is that the built in functionality of Solver is limited (10) to a maximum of 10 tasks to 10 assignees (2 groups * 10 assignees * 10 tasks). An extended version of the Solver add-in can be purchased for nearly $900 if necessary (11).

There are no pre-built functions for solving an assignment problem via the Hungarian method in Excel or Access. A program must be written in VBA to solve with the Hungarian method. There is a public VBA program available that uses the Hungarian method to solve an assignment type problem for a video game (12) that could be adapted for the purpose of this project, however, the author admitted that there are some bugs in the program that would need to be solved before it can be reliably implemented. There are methods in other programming languages which claim to solve assignment problems through a Hungarian based algorithm but do not have any test data available. A Stanford student developed a version programmed in C which suffers an error for "extreme" data inputs (13) and a professor from the University of Frelburg created another C variation that supposedly does not suffer from "extreme input" errors (14). Netlib.com also provides source code to square assignment problems (15).

Human Factors Considerations

A well-designed graphical user interface is important for customer satisfaction and ease of use. The human factors topics which are most pertinent to this project are display design and human computer interaction.

In designing the interface for the computer application, the critical components for a successful design are: to include user involvement throughout the design process to be sure their needs are met, incorporate principles of design, and to use iterative testing early in the design process (16).

Some of the principles of design include:

- *Provide Feedback*. Design the application to provide information to the user about what is happening. (16)
- *Build Consistency*. People learn faster and use less time and energy to find things that are familiar. (16)

- *User Freedom*. Allow users to undo, cancel, and redo to correct errors. Clearly label exits and controls (16).
- *Error Prevention*. Use clear messages to direct users (16).
- *Simplicity and Aesthetics*. The graphic interface should look good and any dialogue or text should be simple. Information should be logical (16).

Display Considerations:

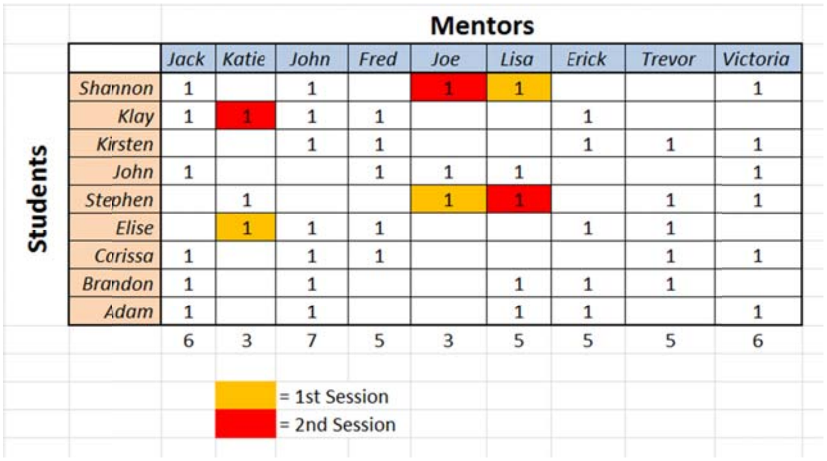
- Top-Down Processing
    - People generally perceive information based on the basis of their past experiences. People read left to right, top to bottom and the solution should be displayed similarly. Information that appears outside of its expected location can cause confusion and decrease the users speed at processing information (16).
- Minimize Information Access Cost
    - There is a cost in the form of time or effort to move from one part of the application to another. Frequently used sources such as the input and output parameters should be placed in a way which minimizes the access cost associated with them (16).

Design (Theory)

The design of this project began with an analysis of the current methods of generating assignments. A short interview with Erica Janoff, an Alumni Association employee, provided a complete overview of the current methods of assignment generation.

Current Method

The current (old) method creating student-mentor pairs is completed entirely by hand. After the speed mentoring session, students turn in score cards with ratings of which mentors they'd like to meet with for the extended one-on-one sessions. The students are instructed to mark an "X" next to the names of the top 5 mentors they'd like to meet with. The score cards are then turned into the Alumni Association worker that creates all the assignments. The worker must sort all the students' cards by group and then create a table in Microsoft Excel for each group. A sample table can be seen to the right in Figure 1. A "1" represents that the student wanted to meet with that mentor. Once all the choices have been input from the score cards, the worker must make pairs by hand. This is generally done just by simply changing the color of the box. Orange was used to mark a pair for session 1 and red was used to mark a pair for session 2. For example, Shannon would meet with Lisa for the first session and Joe during the second. This process takes over an hour once applied to all the groups within a college and is very error prone. It also doesn't give students any preference for one mentor over another. These problems determined the following three goals for the final program:



	Mentors								
Students	Jack	Katie	John	Fred	Joe	Lisa	Erick	Trevor	Victoria
Shannon	1		1		1 🟥	1 🟧			1
Klay	1	1 🟥	1	1			1		
Kirsten			1	1			1	1	1
John	1			1	1	1			1
Stephen		1			1 🟧	1 🟥		1	1
Elise		1 🟧	1	1			1	1	
Carissa	1		1	1				1	1
Brandon	1		1			1	1	1	
Adam	1		1			1	1		1
	6	3	7	5	3	5	5	5	6

🟧 = 1st Session
🟥 = 2nd Session

1. The program should decrease the time required to make assignments
2. The program should eliminate errors in the assignment process
3. The program should allow students to assign preferences to potential mentors

Selecting a Software Program for the Solution

After researching how the assignment process works, the next step was to decide what language or program this project would be created in. The top two candidates were Microsoft Access or Microsoft Excel. These were chosen over other possibilities because they both allow for easy integration with a simple user interface. Table 1 illustrates a decision table that was used to aid in the decision to choose Microsoft Excel over Microsoft Access. The two biggest differences between the two programs are that Excel has a built in Solver add in that helps in solving operations research types of problems and generally, people tend to be more comfortable with Microsoft Excel, which boosts its rating for Ease of Use for the End User. All of the programming for each program would be written in Microsoft's Visual Basic for Applications (VBA) language.

	Ease of GUI Integration	Simplicity of Programming	Ease of Use for End User	Program Availability	Weighted Total
Excel	5	5	5	5	2.48
Access	5	2	2	4	1.52
Total	10	7	7	9	

Table 1: Program Decision Matrix

Selecting a Mathematical Model

The literature confirmed that the linear and Hungarian methods of solving an operations research problem were the most applicable to developing this DSS. More information can be found about each of these methods from the sources in the literature review portion of this paper. Both the Hungarian and integer linear programming (ILP) solve the same types of problems. The only downfall of the Hungarian method is that it can be only used for a minimizing type of solution. This means that the students' ratings would have to be from 1 to 5 where 1 is the most preferable and 5 is the least. To minimize the differences between solution methods, it was decided that students would rate mentors on a 1-5 scale with 1 as the most preferable.

While researching the feasibility of both methods, there were multiple Hungarian method based solvers available and free to use, however, all of these solvers had errors, leaving them unusable for this project since it needed to be completely error. Due to a lack of confidence in the feasibility of programming a fully-functional Hungarian solver in Excel, the ILP method was selected.

The biggest benefit of selecting the ILP method was that Excel's built in Solver add-in could be manipulated with VBA to create a custom, easy-to-use solution that is flexible in its number of inputs and easily customizable to the Alumni Association's situation.

Napkin Design

The design of the program went through more than ten different revisions prior to the final version. This portion of the report covers the initial design process. The first step was to define the constraint equations on paper. Before equations could be created, variables had to be defined. Looking back to Figure 1, each square that doesn't have a name on it would receive two variables. For example, the top left square would be represented by X_{111} and X_{112} where the first two subscript numbers denote the location within the table and the third number denotes whether it is a match for the first one-on-one session or the second. The variables would represent a binary number, either a 1 or 0. A 1 denotes a matched pair, and a 0 means that the mentor-student pair represented by that variable would not be used.

Once the variables were defined, constraint equations could be defined. Each of the constraint equations fit into one of the following categories (The total number of constraint equations for a 9x9 grid is listed as well):

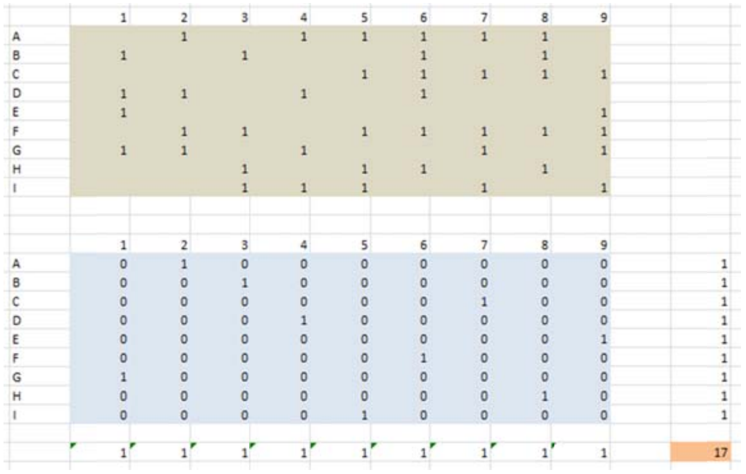
1. No student can be assigned to more than one mentor per session. (18 Equations)
2. No mentor can be assigned to more than one student per session. (18 Equations)
3. There cannot be any duplicate pairs between sessions. (81 Equations)
4. Each variable must be binary. (162 Equations)

In the end, the number of variables totaled 162 and the total number of constraint equations equaled 279.

Initial Program Design

The first step in the program design began with laying out the solution design and using Excel's graphical Solver utilities. This was used to test the constraint equations and settings that would have to be later programmed in VBA.

Figure 2 is what the initial design looked like. Scores were entered in the table at the top. Then the solver settings would be set and the solution for the first session would be displayed in the bottom table. Originally there was a third table on the same worksheet for the second set of solutions that are needed for the mentoring event, however, Excel's free version of solver limits the number of constraints and variables that can be used. The limit is set at 100 constraints and 200 variables (17). The number of variables wasn't a problem, but the number of constraints was. At 279 constraints, solver could not handle this problem all at once.



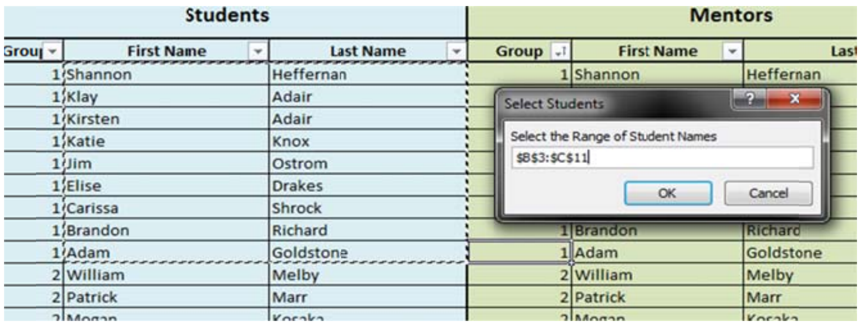
	1	2	3	4	5	6	7	8	9
A		1		1	1	1	1	1	
B	1		1			1		1	
C					1	1	1	1	1
D	1	1		1		1			
E	1								1
F		1	1		1	1	1	1	1
G	1	1		1			1		1
H			1		1	1		1	
I			1	1	1		1		1

	1	2	3	4	5	6	7	8	9	
A	0	1	0	0	0	0	0	0	0	1
B	0	0	1	0	0	0	0	0	0	1
C	0	0	0	0	0	0	1	0	0	1
D	0	0	0	1	0	0	0	0	0	1
E	0	0	0	0	0	0	0	0	1	1
F	0	0	0	0	0	1	0	0	0	1
G	1	0	0	0	0	0	0	0	0	1
H	0	0	0	0	0	0	0	1	0	1
I	0	0	0	0	1	0	0	0	0	1
	1	1	1	1	1	1	1	1	1	17

The solution to the above problems was to limit the group size to 9 mentors and 9 students and to break the solving process into two separate iterations. The first iteration would solve pairs for the first session and the second iteration would solve pairs for the second session. During the second session, the program had to also verify that no duplicate assignments were created.

Automatic Generation of a table for a Group – "Fill Table" Command

The first section of code written for the program was designed to simplify and speed up the process of inputting the students' and mentors' names into the score table. The table generation was designed so that the user would click a button and then be prompted to select the names of the students and mentors in the current group they are working on from a list on a separate worksheet. The list of names along with group number would be copied over from a master spreadsheet that the Alumni Association has. The list can be sorted by group to make the selections easier. Figure 3 shows the filter options and the selection process. When copying the names into the score table, the program automatically combines the first and last names. Pop up boxes were added throughout the process so the user could go back a step if something wrong happened and so that the user would understand what the program is doing without an instruction manual. This process successfully decreased the number of errors in misspelling names and inputting names from the wrong group.



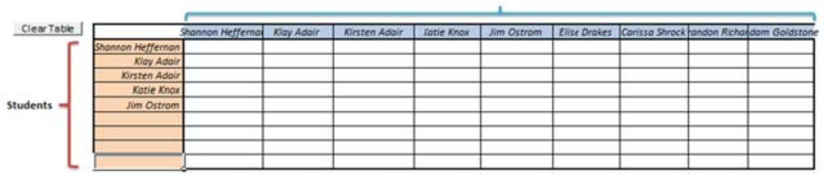
Students			Mentors		
Grou	First Name	Last Name	Group	First Name	Las
1	Shannon	Heffernan	1	Shannon	Heffernan
1	Klay	Adair			
1	Kirsten	Adair			
1	Katie	Knox			
1	Jim	Ostrom			
1	Elise	Drakes			
1	Carissa	Shrock			
1	Brandon	Richard	1	Brandon	Richard
1	Adam	Goldstone	1	Adam	Goldstone
2	William	Melby	2	William	Melby
2	Patrick	Marr	2	Patrick	Marr

Select Students

Select the Range of Student Names

$B$3:$C$11

This portion of the program is designed to erase all the old names before placing in the new ones. This way, if one group is smaller than a previous one, then none of the old group's names get left in the new group's table accidently.



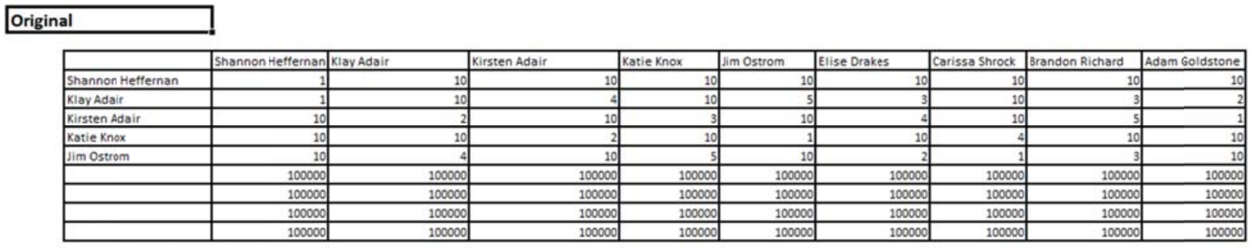
	Shannon Heffernai	Klay Adair	Kirsten Adair	Iatie Knox	Jim Ostrom	Elise Drakes	Carissa Shrock	randon Richa	dam Goldstone
Shannon Heffernan									
Klay Adair									
Kirsten Adair									
Katie Knox									
Jim Ostrom									

Figure 4 shows the score table filled with names after using the "Fill Table" command. At this point, the table is ready to have scores input so that a solution can be generated.

Utilizing VBA to Solve Pairs for Two Sessions

As previously mentioned, Excel limits the number of constraints and variables for each solution set. This required that two solution sets be programmed in. The code used to write the solutions is relatively straightforward with the use of the Solver articles in the Microsoft help database. The biggest problem encountered was with rearranging scores so that a solution could be created.

When choosing between the Hungarian and ILP methods, it was decided that students would use a 1 to 5 rating scale and the goal of the solution would be to minimize the preference level. For example, if there is a 9x9 grid of students and mentors, and each student got their first choice, the preference level would be 9. If everyone but one person got their first choice, the preference level would be a number higher than 9. If the program is trying to minimize the preference level, there becomes a problem with all of the empty squares. The worker can't be expected to place a number higher than 5 in each empty cell to solve this problem, so a new table must be created that automatically takes care of this problem. A second problem is that if the table is not filled with everyone's names, it's possible that someone's top 5 could be given to a "dummy" or empty cell, one without a name.



Original

	Shannon Heffernan	Klay Adair	Kirsten Adair	Katie Knox	Jim Ostrom	Elise Drakes	Carissa Shrock	Brandon Richard	Adam Goldstone
Shannon Heffernan	1	10	10	10	10	10	10	10	10
Klay Adair	1	10	4	10	5	3	10	3	2
Kirsten Adair	10	2	10	3	10	4	10	5	1
Katie Knox	10	10	2	10	1	10	4	10	10
Jim Ostrom	10	4	10	5	10	2	1	3	10
	100000	100000	100000	100000	100000	100000	100000	100000	100000
	100000	100000	100000	100000	100000	100000	100000	100000	100000
	100000	100000	100000	100000	100000	100000	100000	100000	100000
	100000	100000	100000	100000	100000	100000	100000	100000	100000

Figure 5: The Table Used for the Solution Process

The score table on the home page is copied to another worksheet where all of the tables used in the solving algorithm are located. This is the same table from Figure 4, but scores have been placed into the table. The first if statement replaces any blank cells with a number slightly higher than 5. This way, the solver will work to avoid giving a student a mentor that wasn't in their top 5, however if there are too many conflicts, the solver can assign a mentor that wasn't in a student's top 5 choices so that each student has an assigned mentor to meet with. The second if statement is to solve the potential problem that if there are more mentors in a group than students or if the group isn't full, a student could be assigned someone not on their top 5 because an empty or "dummy" assignment was made.

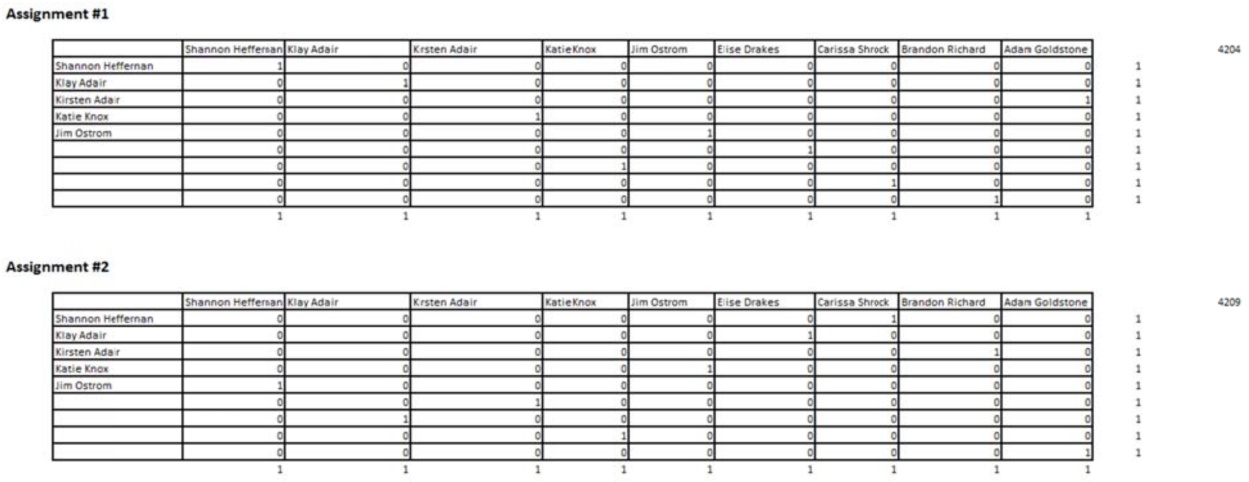
Dummy assignments are necessary because of the cases where there is not a full group or an equal number of students and mentors. Each space in the mentor row and the student column must receive an assignment as part of the solution. In cells that align with an empty cell or one with no name, its value is set much higher than all the others because the goal is to avoid the situation where someone can't meet with someone rated in their top 5 because a dummy cell was assigned to that mentor.

Once all of the bugs had been worked out, a Solve button was added to the home page that automatically ran the portion of the code that generates the solution pairs.

At this point in the program design, there were three worksheets within the program. The first was the Home Page. This page was designed to contain a "Solve Button" and the button that controls the "Fill Table" command in addition to the table where a worker inputs the students' scores from their score cards. As part of the Fill Table command, there was the List worksheet which stores a list of all the mentors and students. Finally, for the solver variables, constraints, and solution table, there was a Solver table that contains various sub tables required to generate and store the assignments.

Storing Solutions

Figure 6 shows what the solutions look like directly after the program has generated the pairs. The table stores either a 1 or a 0, a match, or not a match. The next step was to add a function to the program that would scan through each of the cells, and every time it found a cell with a 1 in it, save the student-mentor pair to a separate worksheet.



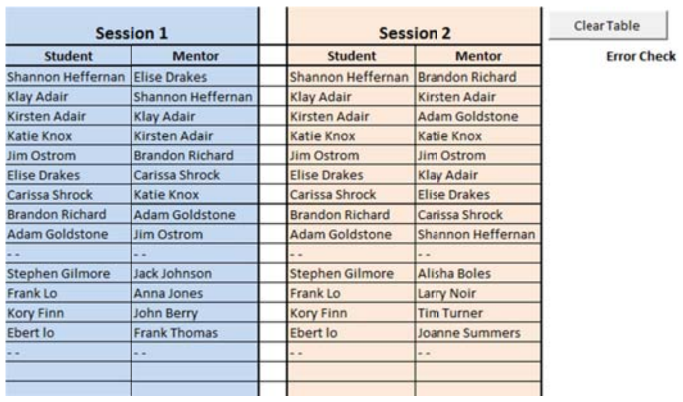
Assignment #1

	Shannon Heffernan	Klay Adair	Krsten Adair	KatieKnox	Jim Ostrom	Elise Drakes	Carissa Shrock	Brandon Richard	Adam Goldstone		4204
Shannon Heffernan	1	0	0	0	0	0	0	0	0	1	
Klay Adair	0	1	0	0	0	0	0	0	0	1	
Kirsten Adair	0	0	0	0	0	0	0	0	1	1	
Katie Knox	0	0	1	0	0	0	0	0	0	1	
Jim Ostrom	0	0	0	0	1	0	0	0	0	1	
	0	0	0	0	0	1	0	0	0	1	
	0	0	0	1	0	0	0	0	0	1	
	0	0	0	0	0	0	1	0	0	1	
	0	0	0	0	0	0	0	1	0	1	
	1	1	1	1	1	1	1	1	1		

Assignment #2

	Shannon Heffernan	Klay Adair	Krsten Adair	KatieKnox	Jim Ostrom	Elise Drakes	Carissa Shrock	Brandon Richard	Adam Goldstone		4209
Shannon Heffernan	0	0	0	0	0	0	1	0	0	1	
Klay Adair	0	0	0	0	0	1	0	0	0	1	
Kirsten Adair	0	0	0	0	0	0	0	1	0	1	
Katie Knox	0	0	0	0	1	0	0	0	0	1	
Jim Ostrom	1	0	0	0	0	0	0	0	0	1	
	0	0	1	0	0	0	0	0	0	1	
	0	1	0	0	0	0	0	0	0	1	
	0	0	0	1	0	0	0	0	0	1	
	0	0	0	0	0	0	0	0	1	1	
	1	1	1	1	1	1	1	1	1		

Figure 6: Solutions on the Solver Worksheet



Session 1		Session 2	
Student	Mentor	Student	Mentor
Shannon Heffernan	Elise Drakes	Shannon Heffernan	Brandon Richard
Klay Adair	Shannon Heffernan	Klay Adair	Kirsten Adair
Kirsten Adair	Klay Adair	Kirsten Adair	Adam Goldstone
Katie Knox	Kirsten Adair	Katie Knox	Katie Knox
Jim Ostrom	Brandon Richard	Jim Ostrom	Jim Ostrom
Elise Drakes	Carissa Shrock	Elise Drakes	Klay Adair
Carissa Shrock	Katie Knox	Carissa Shrock	Elise Drakes
Brandon Richard	Adam Goldstone	Brandon Richard	Carissa Shrock
Adam Goldstone	Jim Ostrom	Adam Goldstone	Shannon Heffernan
- -	- -	- -	- -
Stephen Gilmore	Jack Johnson	Stephen Gilmore	Alisha Boles
Frank Lo	Anna Jones	Frank Lo	Larry Noir
Kory Finn	John Berry	Kory Finn	Tim Turner
Ebert lo	Frank Thomas	Ebert lo	Joanne Summers
- -	- -	- -	- -

The worksheet creation is designed to be 100% automated. After the user clicks solve, the solutions are generated by the program and automatically placed in the Assignments worksheet. Assignments are divided between the first and second sessions as can be seen in

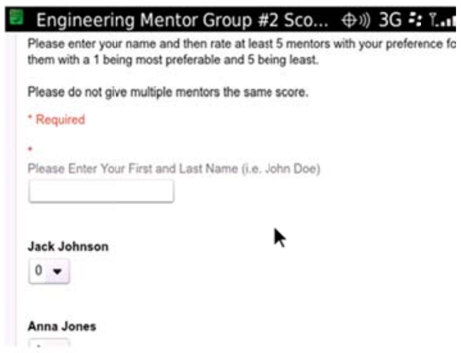
Figure 7. Additionally, each group is automatically separated by a line break to help differentiate between groups. Although not shown, groups can also be color coded to further differentiate them from each other. The goal was to neatly display the assignments generated in an organized manor so that once all the groups had been taken care of, this page could be displayed on a projector so everyone would be able to see where they are supposed to go.

In Figure 7, a column title Error Check is visible in the top right of the screen. This was inserted to help check for duplicate assignments. The Clear Table button was added to easily delete all the assignments from a previous event. Pressing that button prompts the user to ask if they are sure they would like to delete all the assignments. If the user clicks "Yes," then all the old assignments are deleted.

Further Process Improvement: Google Forms

Once the DSS was created and fully functional, a test was conducted to determine how much time the new process saved. The test determined that the program has 100% accuracy and decreases the hour that assignments once took down to about thirty minutes. It was found that most of the time was consumed by inputting the students' scores into the score table. If that process could be eliminated, the time required to generate all the pairs would be only 10 minutes.

The solution for removing the manual score entry was to utilize Google Forms. With Google Forms, a score card can be e-mailed to students and then filled out on a smart phone or computer. Google automatically compiles the data into a table which can be copy and pasted directly on top of the original score table. Then, the worker just has to click solve and the pairs are created automatically. Figure 8 shows what a Google Forms score card looks like on a mobile device. Creating the forms takes some extra preparation time before


Engineering Mentor Group #2 Sco... 3G

Please enter your name and then rate at least 5 mentors with your preference fo
them with a 1 being most preferable and 5 being least.

Please do not give multiple mentors the same score.

* Required

*
Please Enter Your First and Last Name (i.e. John Doe)

Jack Johnson
0

Anna Jones

e

the day of the event, but makes the assignment process, which can only take a maximum of an hour, go much quicker.

Implementation and Training

The program has instructions within itself so that a user does not have to flip through a manual or switch between windows on their computer. The key instructions to running the program are located on the Home worksheet and divided between the manual data entry and Google Forms data entry methods.

Included in Appendix A of this report is a copy of the first time set up manual. Since the program utilizes an outside add-in to assist in generating the solutions, it must be enabled before the program will be fully functional. The process is outlined in 11 simple steps.

Methods

The DSS was tested by running many trials of every imaginable scenario. All of the trials that have been conducted on the final program have experienced zero errors of any kind. The following are the situations which were tested for:

- A full 9x9 table.
- Equal numbers of mentors and students, but not a full table
- Fewer students than mentors, with 9 mentors.
- Fewer students than mentors, with fewer than 9 mentors.
- Fewer mentors than students – the program automatically detects an error and stops.

An error has been defined as any problem with the program itself, and any duplicate pairs that the program assigned. There have also been zero cases in which a blank cell has been copied over to the final Table of Assignments.

Results

The results are better than what was originally expected. The process to generate two sessions of student-mentor pairs has been decreased from over an hour to thirty minutes with manual data entry and ten to fifteen minutes with Google Forms. The DSS runs flawlessly and eliminates the chance of error in creating assignments.

The design is simple and straightforward to use. Human factors implications were considered at every step of the design phase to make the process as simple and straightforward as possible. Perhaps someone with a better understanding for aesthetics could make it look better, but it is still easy to follow and functionally sound.

The productivity estimates were better than expected. In the past, the alumni center has used one worker per college to make all the assignments. It is reasonable to expect that one worker can now make all the assignments within an hour with Google Forms.

In the future, Solver could be upgraded to allow groups larger than 9x9 to be solved for. For the purpose of this project, this isn't a problem at all, and most groups are 9 or less. If this program were applied elsewhere though, it may be beneficial to create a bigger grid. The necessary increase in size is ultimately dependent on its application.

Based upon the results, the Alumni Association will be much more efficient with their time during the lunch period where set up for the rest of the day occurs. This program will allow them to expand the program without incurring significant increases in labor hours or cost. The biggest source of problem is going to be human error with the program. The best way to combat this is to always have an unaltered back-up copy of the program so that if anything were to accidently go wrong, the original copy can quickly be pulled up. Another potential are problem is with the initial set up of the program. A guide is included in Appendix A, but it is very possible for someone who is not very comfortable in Excel to experience an error in that process.

The only limitation found was that the program only works in Excel 2010. In Excel 2007, there were errors in that the Solver Add-in would not accept some of the binary constraints, and thus provide an invalid solution. Research was conducted to look into a solution, but no solution was found.

Conclusion

In conclusion, the DSS was a success. It met or exceeded the Alumni Associations requirements. It would have been preferable had the solution worked with Excel 2007. Based upon the results, the solution is very successful, but the human factors design features could have been more developed. The following bullet points outline the key results of this project:

- The process time was decreased from over an hour to between 10-30 minutes depending on the method of data input used.
- The pairs are generated without any error.
- Student preferences are explicitly considered for mentor assignments.
- The experimental results indicate that this program will help the Alumni Association to save both time and resources as they expand the Mentoring Day program to include all 6 of Cal Poly's Colleges.
- Each initial objective was accomplished:
  - Develop a program to aid in generating pairs of assignments for the Mustang Mentoring Day Event
  - Provide adequate documentation and instructions for use of the program
  - Create a report that outlines the research, design, and approach to creating the program.

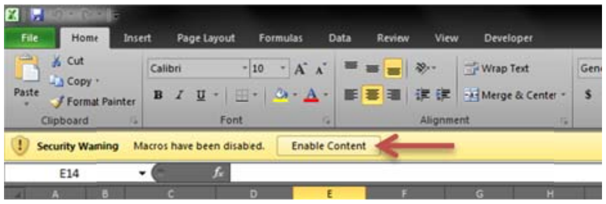
Appendix A: First-Time Set-Up Instructions

The following are instructions for the Alumni Association to set up the solver application.

Step 1: The Mentor Day program can only be ran on Microsoft Excel 2010. Earlier versions will not run properly.

Step 2: Open up the Solver Application

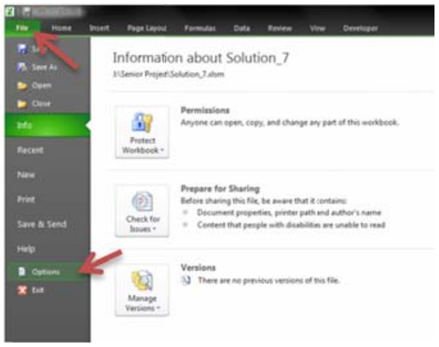
Step 3: Click "Enable Content"



Security Warning Macros have been disabled.

Figure 9: Enable Macro Content

Step 4: Click "File" then click "Options"



Information about Solution_7
I:\Senior Project\Solution_7.xlsm

Permissions
Anyone can open, copy, and change any part of this workbook.

Prepare for Sharing
Before sharing this file, be aware that it contains:
- Document properties, printer path and author's name
- Content that people with disabilities are unable to read

Versions
There are no previous versions of this file.

Figure 10: Excel Options

Step 5: Click "Customize Ribbon" and then check the box next to "Developer" under the Main Tabs Column on the Right, then select "OK"

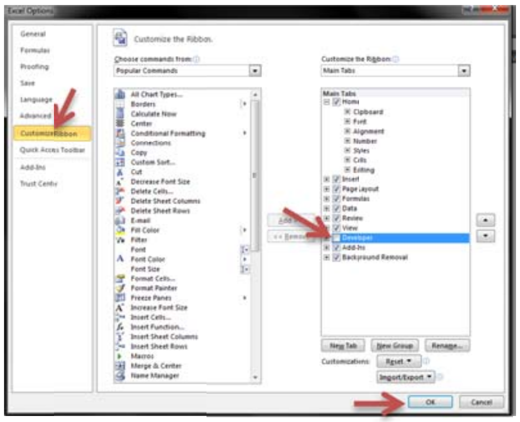


Figure 11: Enabling the Developer Tab

Step 6: Click the “Developer” tab at the top of the screen, then click on “Add-Ins”

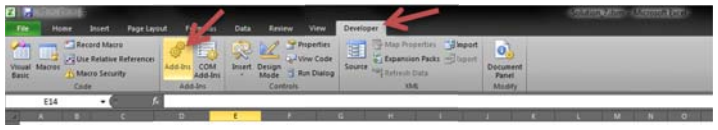


Figure 12: Viewing Add-Ins

Step 7: Check the box next to “Solver Add-in” then click “OK”

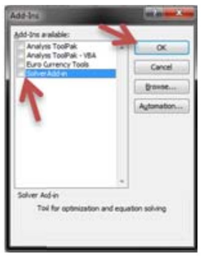


Figure 13: Enabling Solver Add-In

Step 8: While you are still on the Developer tab, click “View Code”

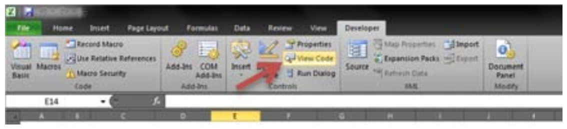


Figure 14: Navigating to the Code Window

Step 9: In the code window, click on “Tools” at the top of the screen, then click “References”

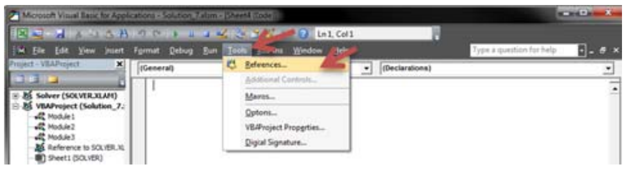


Figure 15: Viewing References

Step 10: In the references pane, scroll down until you’ve found the “Solver” reference. Check the box next to “Solver” and select “OK.”

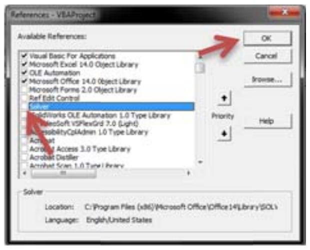

References - VBAProject
Available References:
✓ Visual Basic For Applications
✓ Microsoft Excel 14.0 Object Library
✓ OLE Automation
✓ Microsoft Office 14.0 Object Library
Microsoft Forms 2.0 Object Library
Ref Edit Control
Solver
Acrobat Access 3.0 Type Library
Acrobat Distiller
OK
Cancel
Browse...
Priority
Help
Solver
Location: C:\Program Files (x86)\Microsoft Office\Office14\Library\SOL
Language: English/United States

Figure 16: Enabling the Solver Reference

Step 11: Click on the red “X” button on the top right of the Code window to exit back to the solver.

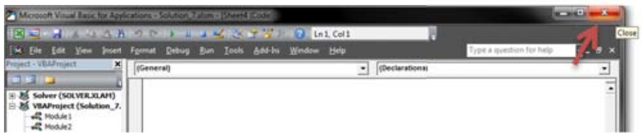


Figure 17: Exit the Code Window

The program is now ready to run. Make sure the List worksheet is filled out properly if you are manually entering the data in and that the Assignments Worksheet has been emptied of all old assignments.

References

1. **Institute For Operations Research and the Management Sciences.** About Operations Research. *informs online.* [Online] Institute for Opearations Research and the Management Sciences. [Cited: 2 13, 2011.] http://www.informs.org/About-INFORMS/About-Operations-Research.

2. **Institute for Operations Research and the Management Sciences.** Operations Research: The Science of Better. *ScienceOfBetter.org.* [Online] [Cited: February 15, 2011.] http://www.scienceofbetter.org/can_do/value_prop.htm.

3. **Hillier, F. S. and Lieberman, G. J.** *Introduction to Operations Research.* 8th. New York : McGraw Hill, 2005. pp. 350,. 0-07-252744-7.

4. *Algorithms for the Assignment and Transportation Problems.* **Munkres, J.** 1, s.l. : Society for Industrial and Applied Mathematics, 1957, Vol. 5.

5. **Swann, J.** The Marrieage Theorem. [Online] GA Tech. [Cited: February 16, 2011.] http://www2.isye.gatech.edu/~jswann/casestudy/marriage.html.

6. **Murty, K. G.** An Algorithm for Tanking all the Assignments in Order of Increasing Cost. *JSTOR.org.* [Online] May-Jun. 1968. [Cited: February 15, 2011.] http://www.jstor.org/stable/168595.

7. *A Genetic Algorithm for the Generalised Assignment Problem.* **Chu, P. C. and Beasley, J. E.** 1, London : Pergamon, 1996, Computer Ops Res., Vol. 24, pp. 17-23. S0305-0548(96)00032-9.

8. *The Group Assignment Problem.* **Poor, A. B.** Fort Collins : Numerica Corporation, 2004, Proceedings of SPIE, International Society for Optical Engineering, Vol. 5204, pp. 595-607. 0277-786x.

9. **Microsoft.** Introduction to Optimization with the Excel Solver tool. [Online] Microsoft. [Cited: February 15, 2011.] http://office.microsoft.com/en-us/excel-help/introduction-to-optimization-with-the-excel-solver-tool-HA001124595.aspx.

10. —. Excel specifications and limits. *Office.Microsoft.com.* [Online] Microsoft. [Cited: February 15, 2011.] http://office.microsoft.com/en-us/excel-help/excel-specifications-and-limits-HP005199291.aspx.

11. **Frontline Solvers.** Premium Solver for Excel - Optimization Software Overview. [Online] FrontlineSolvers. [Cited: February 16, 2011.] http://www.solver.com/xlspremsolv.htm.

12. **"capibarbaroja".** Hungarian algorithm for Excel/VBA. *Tribal Wars Forum.* [Online] Jelsoft Enterprises Ltd., July 2, 2009. [Cited: January 31, 2011.] http://forum.tribalwars.net/showthread.php?t=153055.

13. **Gerkey, B. P.** A C implementation of the Hungarian Method. *Robotics.Stanford.edu.* [Online] August 12, 2004. [Cited: February 15, 2011.] http://robotics.stanford.edu/~gerkey/tools/hungarian.html.

14. **Stachniss, C.l.** Cyrill Stachniss. [Online] University of Frelburg, September 9, 2004. [Cited: February 15, 2011.] http://www.informatik.uni-freiburg.de/~stachnis/misc.html.

15. **NetLib.** NetLib. *NetLib.com.* [Online] [Cited: February 16, 2011.] http://www.netlib.no/netlib/toms/548.

16. **Wikens, C. D., et al., et al.** *An Introduction to Human Factors Engineering.* [ed.] 2nd. Upper Saddle River : Pearson Prentice Hall, 2004. pp. 184-217, 383-418, 418-435. 0-13-18373-2.

17. Standard Excel Solver - Dealing with Problem Size Limits. *FrontlineSolvers.* [Online] FrontlineSolvers. [Cited: May 27, 2011.] http://www.solver.com/suppstdsizelim.htm.