12-2011

Office Hour Scheduling Tool

Tyler Dunlap
University of Arkansas, Fayetteville

Office Hour Scheduling Tool

An Undergraduate Honors Thesis submitted to the

University of Arkansas

College of Engineering

Department of Industrial Engineering

By

Tyler Royce Dunlap

Advisor: Dr. Chase Rainwater

Reader: Dr. Ed Pohl

Introduction

At the beginning of every semester, professors are tasked with scheduling office hours for each class they teach. Whether this class contains 10 people or 100, every student expects to be able to have access to their professor during their office hours. However, there are scheduling conflicts that make the problem of scheduling accommodating office hours non-trivial. It is common that students take up to 6 courses per semester, which eliminate 18 hours that they are available for office hours in a single course. Secondly, professors have numerous obligations outside of the classroom that include advising students, research and service activities. Each of these tasks eliminates feasible times that can be allocated to office hours for a specific course. At this time, *there is no comprehensive tool that seeks to maximize student-teacher interaction outside lecture by holistically considering both students and professor availability* across the numerous courses they are affiliated with. In order to address this need, *this work introduces Excel Visual Basic for Applications (VBA) decision support that can be used by administrators, faculty and students to better serve the educational needs of the student body*.

Our tool was developed with specific consideration given to the Department of Industrial Engineering at the University of Arkansas. However, the tool is applicable to any department at any collegiate institution. The tool has the ability to painlessly collect information regarding professor and student's schedules. This functionality of the tool allows this information to be quickly processed and organized into data files usable by the commercial optimization solver (CPLEX). When coupled with our optimization model, this information is used to make informed decision as to times in which office hours should take place for each class. With the office hour scheduling tool, teachers will now be able to skip the mundane and tedious task of

considering every student's schedule, and students will no longer have to prioritize over meeting with a professor during their office hour or going to a different class.

Optimization Model: Maximizing Satisfied Students

Before introducing the decision support tool, it is important for the reader to understand the mathematical framework used to make the office hour scheduling decisions. As we will discuss more elaborately in a later section, the optimization done in this work *was not implemented in the Excel VBA tool*, but was carried out by the CPLEX using C++ Concert Technology. However, our tool automatically generates the formatted input files required by CPLEX.

Our model depends on three main pieces of data that are essential for its objective:

1. Class Times
2. Professor Schedules (including meetings)
3. Student Schedules

As will be shown later, this data comes from the user input inside of the Excel tool which acts as the model's front end. The elements of our model can be broken into sets, parameters, decision variables, constraints and the objective. Each of these elements is described in detail in the following subsection.

Model Sets

Our model makes use of the following 4 main sets. The first set to take into consideration is the set of students. These students are represented numerically, such that:

S = set of all students s in the database $\in \{i = 1, 2, 3, \dots, |S|\}$. (1)

Similarly, the professors for each university as well as the classes offered at the university make up the following sets:

P = set of professors $p \in \{j = 1, 2, 3, \dots, |P|\}$ (2)

and

C = set of classes $c \in \{j = 1, 2, 3, \dots, |C|\}$. (3)

The last set in the model represents the different times that an office hour, class meeting, or professor's meeting, can occur:

T = set of time $t \in \{i = 1, 2, 3, \dots, |T|\}$ (4)

Set (4) assumes that the start time for each period has 3 corresponding end times associated with it, 50, 80, or 110 minutes later. Each different start time and end time combination, as well as the day or combination of days that it occur(s) on, is a unique time *t*. This was done to ensure the model captured all time periods of classes. However, it does not capture the classes that run over 110 minutes because of the rarity of longer class times that exceed this amount.

Model Parameters

The inputs required by our model are represented by the following 4 matrices.

A_{ct} = 1 if class *c* occurs during time period *t*; 0 otherwise (5)

B_{sc} = 1 if student *s* is in class *c*; 0 otherwise (6)

D_{pc} = 1 if professor *p* teaches class *c* (7)

E_{pt} = 1 if professor *p* is not available during time period *t*; 0 otherwise (8)

Parameters (5) are binary values that indicate whether or not class *c* occurs during time period *t*. Parameters (6) indicate which students are in which classes, while parameters (7) identify which

professor teaches each of the courses in *C*. Finally, parameters (8) distinguish the periods in which a professor is not available for office hours due to a commitment outside of teaching. Parameters (5), (7), and (8) are related to each other in the fact that if professor *p* teaches class *c* ($D_{pc} = 1$) then professor *p* is unavailable during the time period *t* in which class *c* occurs (i.e. the time period for which $A_{ct} = 1$).

Decision Variables

The model was formulated based on two sets of binary decision variables; one telling whether a class, *c*, holds office hours during a certain time, *t*, and the other representing whether a student is able to attend those office hours. That is, let:

$X_{ct} = 1$ if class *c* has an office hour during period *t* (9)

$Y_{sc} = 1$ if student *s* can attend office hour for class *c*. (10)

Objective

The goal of the model is to maximize the number of students able to attend the scheduled office hours. Using the sets and decision variables presented previously, our objective can be written as the following.

$$\max \quad \sum_{s \in S} \sum_{c \in C} Y_{sc} \tag{11}$$

It is important to note here that this objective function focuses on getting the most students access to office hours as possible. It does not ensure that the maximum number of students able to attend office hours *per class* will be reached. For areas of future study, it may be beneficial to consider the following objective function to maximize the minimum number of students who can

attend office hours for each class. Note that this would simply require a change of objective in the optimization model and would in no way impact the usability of the tool.

Constraints

Our model consists of three operational restrictions and two necessary binary constraints on the decision variables. Expanding on the first point, we have chosen to enforce that each class requires two separate office hour time periods, as is customary across the INEG faculty. This requirement can be modeled as follows.

$$\sum_{t \in T} X_{ct} = 2 \quad \forall\, c \in C \tag{12}$$

In other words, for every class, there must be exactly two time periods at which the class is scheduled to have office hours.

Obviously, for each class's office hour time period, the professor must be available for the entire duration of the period. To guarantee that this is always the case, we utilize parameters (7) and (8) to arrive at the following expression.

$$X_{ct} \leq 1 - max\left[D_{pc} \times A_{ct}, E_{pt}\right] \quad \forall\, c \in C, \forall\, p \in P, t \in T \tag{13}$$

This ensures that the office hour is only scheduled ($X_{ct} = 1$) during a time when the professor who teaches the class of concern is available. If the professor teaches the class ($D_{pc} = 1$) and the class is offered during that period ($A_{ct} = 1$), then the office hour cannot be scheduled during the time in which the class takes place. If the professor has a meeting or other obligation during that time ($E_{pt} = 1$), then the office hour also cannot be scheduled during the time in which the meeting or other obligation takes place. However, if the teacher either does not teach the

class ($D_{pc} = 0$) and is free during the time of the considered time period ($E_{pt} = 0$), then that particular time period is a feasible *option* in which to schedule the office hour for the class.

For the last logic-based constraint, we consider the fact that a student can only attend the office hours of a class if he is available during the time period in which the office hour is offered. This requirement can be expressed as the following.

$$Y_{sc} \leq \sum_{t \in T}\left\{X_{ct}\left(1 - \sum_{c' \in C, c' \neq C} B_{sc'} A_{c't}\right)\right\} \quad \forall\, s \in S, c \in C \tag{14}$$

This constraint serves the function of ensuring that if a student is enrolled in another class c', and that class is offered during the same time period as the office hour time period considered for class c, then the student cannot attend the office hour for the considered time (i.e. $B_{sc'} = A_{c't} = 1$).

Finally, the last two constraints are binary constraints on the decision variables, which can be written as follows.

$$X_{ct} \in \{0,1\}\ \forall\, c \in C, \forall\, t \in T \tag{15}$$

$$Y_{st} \in \{0,1\}\ \forall\, s \in S, \forall\, t \in T \tag{16}$$

Complete Model

Using the preceding decision variables, parameters, constraints and objective, our office hour scheduling optimization model is given as the following.

$$\text{Maximize } \sum_{s \in S} \sum_{c \in C} Y_{sc}$$

Subject to:

$$\sum_{t \in T} X_{ct} = 2 \quad \forall\, c \in C$$

$$X_{ct} \leq 1 - max\left[D_{pc} \times A_{ct}, E_{pt}\right] \quad \forall\, c \in C, \forall\, p \in P, \forall\, t \in T$$

$$Y_{sc} \leq \sum_{t \in T}\left\{X_{ct}\left(1 - \sum_{c' \in C, c' \neq C} B_{sc'} A_{c't}\right)\right\} \quad \forall\, s \in S, c \in C$$

$$X_{ct} \in \{0,1\}\ \forall\, c \in C, \forall\, t \in T$$

$$Y_{sc} \in \{0,1\}\ \forall\, s \in S, \forall\, c \in C$$

Office Hour Scheduling Tool

In order to obtain all of the necessary information on class schedules, professor's schedules, student's schedules, and possible time periods, a tool was developed with the ability to read in necessary information for the optimization model, output the number of students that would be available for a particular office hour time period, generate the required information for the model when the administrator is ready to optimize using CPLEX, and link the CPLEX results with the corresponding classes and time periods for their solved office hours. In the following sections, an in-depth overview of the tool is presented with screenshots for reference.

Introductory Graphical User Interface

The office hour scheduling tool requires students, professors, and an administrator to input data concerning class schedules, courses taught, and course information, respectively. Because of this, a graphical user interface (*GUI*) was created that enables the user to come in contact only with the content that is applicable to him. In *Figure 1* below, the *GUI* of the tool is seen. The user, depending on their role, can press the appropriate button corresponding with their desired action (e.g. adding/deleting, adding a professor time conflict, etc.). The options available to the user are (i) complete administrative tasks, (ii) input student class information,

(iii) input professor class/conflict information and (iv) output individual student availability for a proposed office hour. Each of these options are detailed in the following sections.

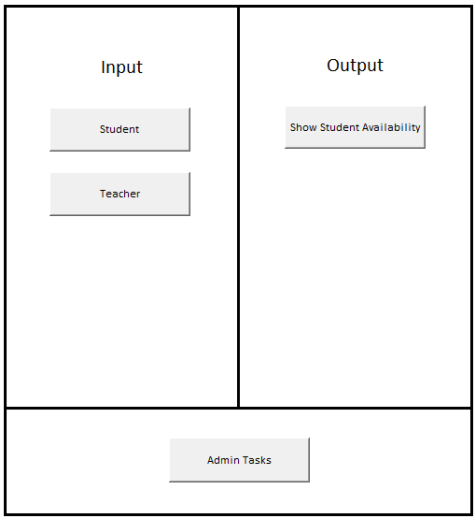

Figure 1. Graphical User Interface

Admin Tasks

Since the first thing the tool needs is the input of all of the classes offered by the university, the description of the tool will begin with the Admin Tasks section. It is recommended that a single dedicated staff member be responsible for the tool's raw data. Since much of this data is stored in ISIS, a representative of that organization would be the ideal candidate to have administrative control of this portion of the tool. This will help to ensure accuracy of entered information, as well as timely changing of information, when necessary. When the user clicks the "Admin Tasks" button, the form shown in *Figure 2* is displayed.

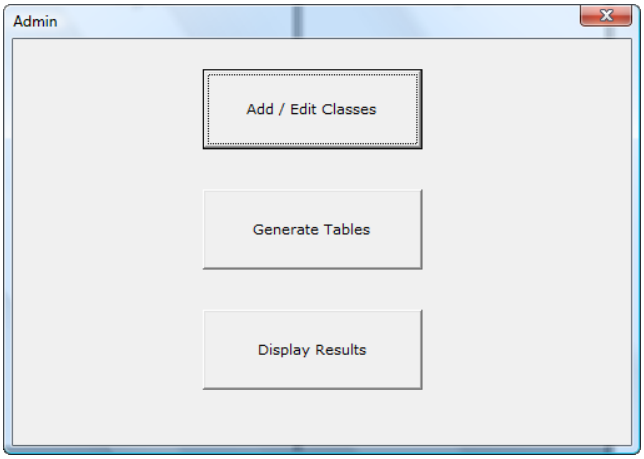

Figure 2 Admin Initial Form

If the administrator wishes to add / edit class(es), he will click on the "Add / Edit Classes" button, another form will appear, and he can add / edit the class of interest. *Figure 3* shows this form completed with an example class. Please note that this part of the tool simply adds the courses available to the students. It is not the portion of the tool in which students can enter in their individual schedules. In fact, it is important that information is correctly added into this segment of tool, as it is the information from which the teacher and student will both choose when inputting their class schedules. An alternative to entering the data via this form would be for the administrator to skip the user form altogether and navigate to the "INEG Classes" Sheet and manually create an entry. However, *the user must be aware that the tool it will not check to see if the class has already been added* if data is entered in outside of the provided user form.

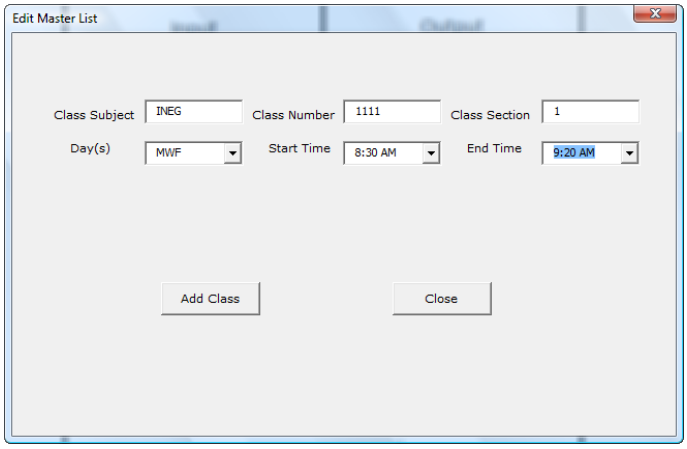

Figure 3 Add / Edit Classes

It is anticipated that users will make input errors. Therefore, the tool contains numerous built-in safeguards to protect from providing duplicate or incorrect information. For example, *Figure 4* shows the message box that is displayed if the administrator enters different information on a particular class that has already been added to the list. In the example shown in *Figure 4*, the user is told that information already exists for INEG 2102. The user is asked to verify that they want to override that entry with the newly entered information.

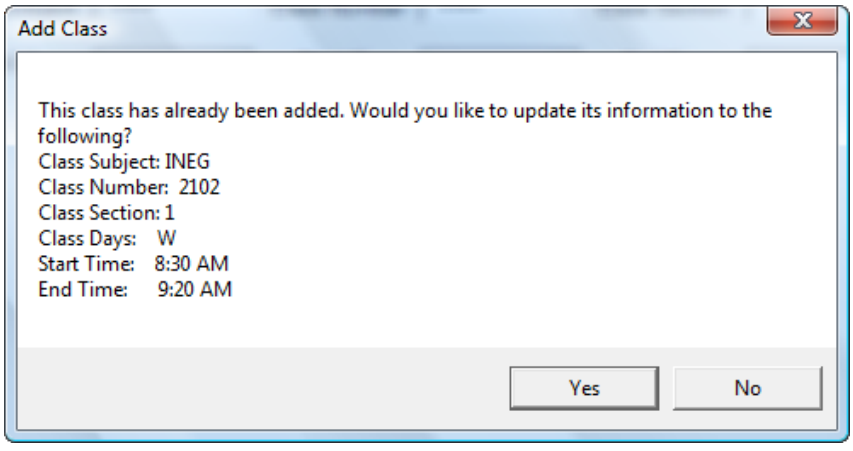

Figure 4 Edit Class

Within the “Admin Tasks” section, the administrator can also find the “Generate Tables” button. This button should only be pressed when *all information* (i.e. courses, professors, times) *has been entered.* Once this button is pressed, the model parameters discussed previously will be constructed as binary matrices inside of the Excel tool, and each will be saved as a .txt file inside of the same folder in which the tool is saved. These .txt files are then treated as input into the optimization model developed in the C++ Concert tool. The optimization results from the C++ tool are accessible by the decision support tool and can be displayed through the “Display Results” results button. *Table 1* shows the results that will displayed once the office hour times have been scheduled for each class.

Table 1 Example Results

Dept	Class	Section	Day	Start Time	End Time
INEG	2102	1	MW	7:30 AM	9:20 AM
INEG	2313	1	MW	8:00 AM	8:50 AM
INEG	2333	1	F	8:00 AM	9:20 AM
INEG	2403	1	TTH	8:30 AM	9:20 AM
INEG	2413	1	F	8:30 AM	10:20 AM
INEG	2513	1	F	9:00 AM	9:50 AM
INEG	3613	1	MWF	9:30 AM	10:20 AM
INEG	3623	1	MW	9:30 AM	10:50 AM
INEG	3713	1	TTH	10:30 AM	11:20 AM
INEG	3833	1	W	10:30 AM	11:50 AM
INEG	4433	1	MWF	10:30 AM	12:20 PM
INEG	4443	1	F	11:00 AM	11:30 AM
INEG	4543	1	TTH	11:00 AM	12:20 PM
INEG	4553	1	MWF	11:30 AM	12:20 PM
INEG	4563	1	M	12:30 PM	1:20 PM
INEG	4583	1	M	12:30 PM	1:50 PM
INEG	4633	1	TTH	1:30 PM	2:20 PM
INEG	4683	1	TH	1:30 PM	3:20 PM
INEG	4723	1	MWF	2:00 PM	3:20 PM
INEG	4904	1	MW	2:30 PM	3:20 PM
INEG	410V	2	TTH	3:30 PM	4:20 PM
INEG	2403	L001	TTH	3:30 PM	4:50 PM
INEG	2513	L002	W	4:30 PM	5:20 PM
INEG	4533	1	MW	5:00 PM	6:20 PM
MEEG	2003	1	MW	7:30 AM	9:20 AM
MEEG	2303	1	F	8:00 AM	8:50 AM

Input Section

The input section of the introductory *GUI* is divided into two parts: student input and teacher input. Within each type of input are data entry user forms. However, as stated before, these user forms cannot be populated with the class options until the administrator has added all of the classes in for each semester. The forms contain drilldown logic that prevents user error. In order for this drilldown logic to work, it must first have an accurate source of data.

Student Input

The student input section is the only part of the tool that the students will need to use. Within it, the student will have the option of adding a class or dropping a class he has already input into the tool. *Figure 5* shows the initial form that the student sees when he clicks on the "Student" button of the *GUI*.

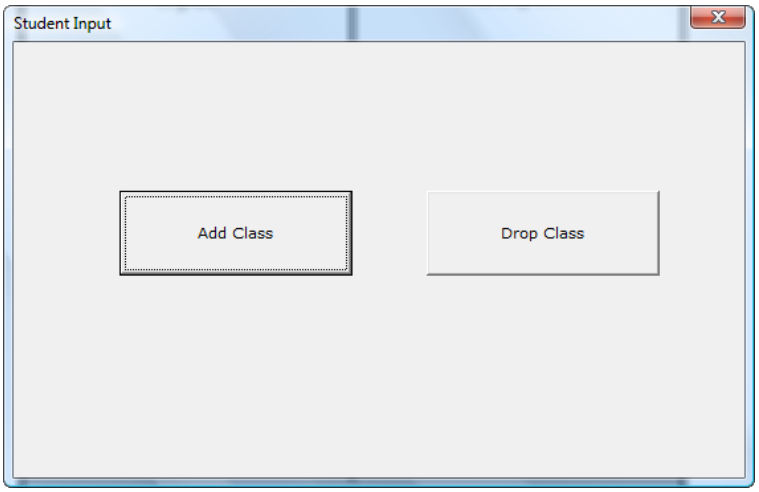

Figure 5 Initial Student Form

Once the student selects the “Add Class” option, another user form is displayed. The student can then populate the fields of the form with the necessary information pertaining to the class he wishes to add. *Figure 3* shows this form populated with an example student.

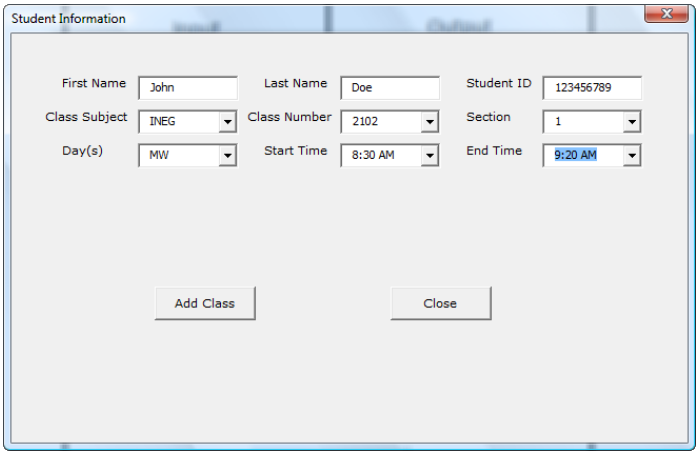

Figure 6 Example Student Input

In this example, John Doe seeks to add INEG 2102 Section 1 into his schedule. Since INEG 2101 Section 1 occurs on MW from 8:30AM – 9:20AM, those were the only choice available to John Doe as he filled out the remainder of the form. This was done to prevent any error that may result from manually inputting data.

Similarly, if a student wishes to drop a class, he will only have the option to drop the classes that he has already input using the “Add Class” button. *Figure 7* shows the form that the “Drop Class” button produces as well as the same information for John Doe.

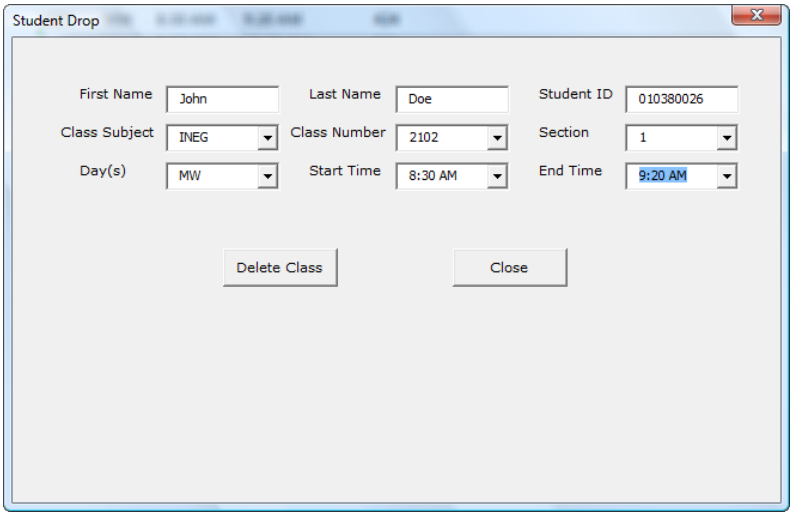

Figure 7 Example Student Drop

Teacher Input

The Teacher Input section holds the same opportunities as the Student Input section. In addition, an extra form is available for professors to provide conflicts that result from meetings, etc. Again, it is important that the tool take into account all professor unavailability, rather than assuming that a teacher is available during any other time they are not teaching a class. *Figure 8*, *Figure 9*, and *Figure 10* show the initial Professor Input screen, the professor's "Add Class" form, and "Delete Class" forms, respectively.

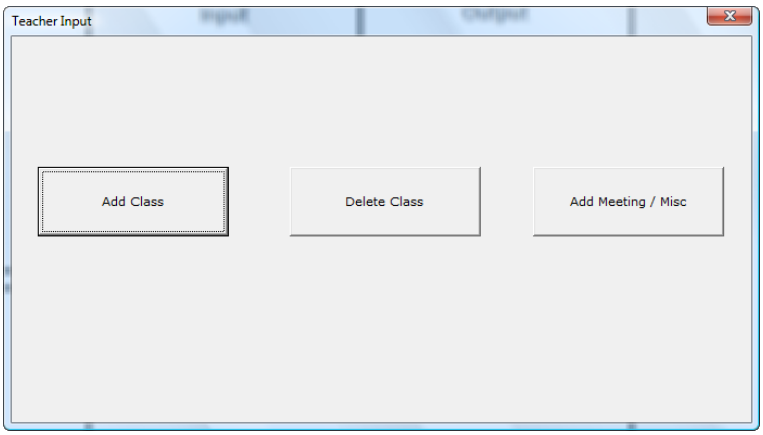

Figure 8 Teacher Initial Input Form

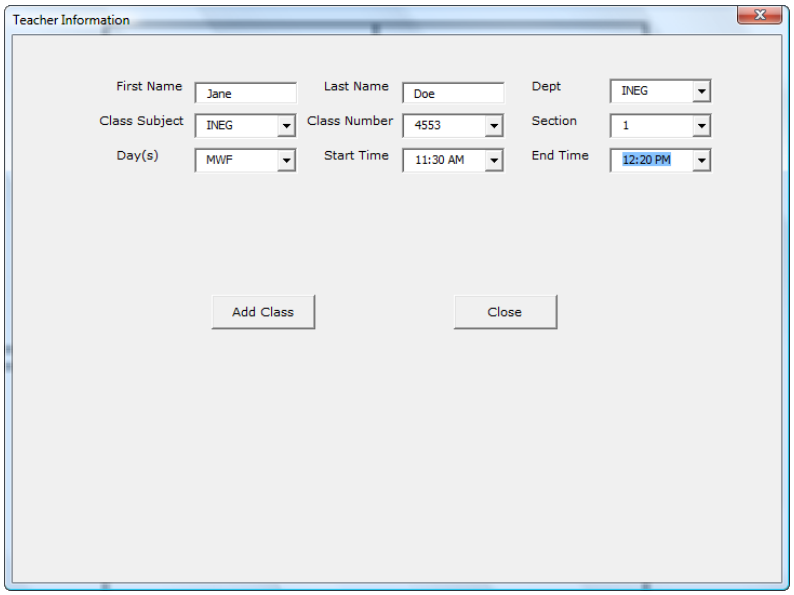

Figure 9 Teacher Add Class

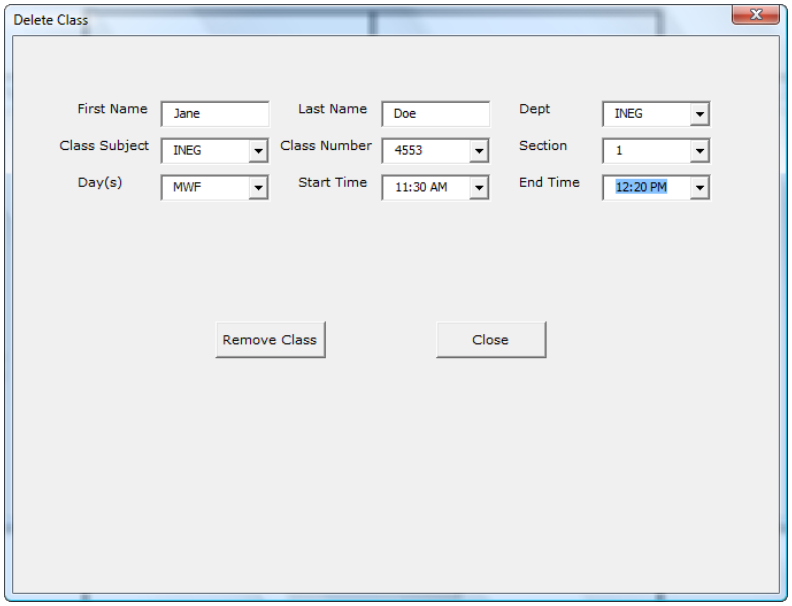

Figure 10 Teacher Delete Class

Finally, the tool possesses another input for the teacher for their weekly or monthly meetings, lunch breaks, or anything that may get in the way of a time period for office hours. *Figure 11* shows the form the teacher must fill out to have these times accounted for in the optimization model.

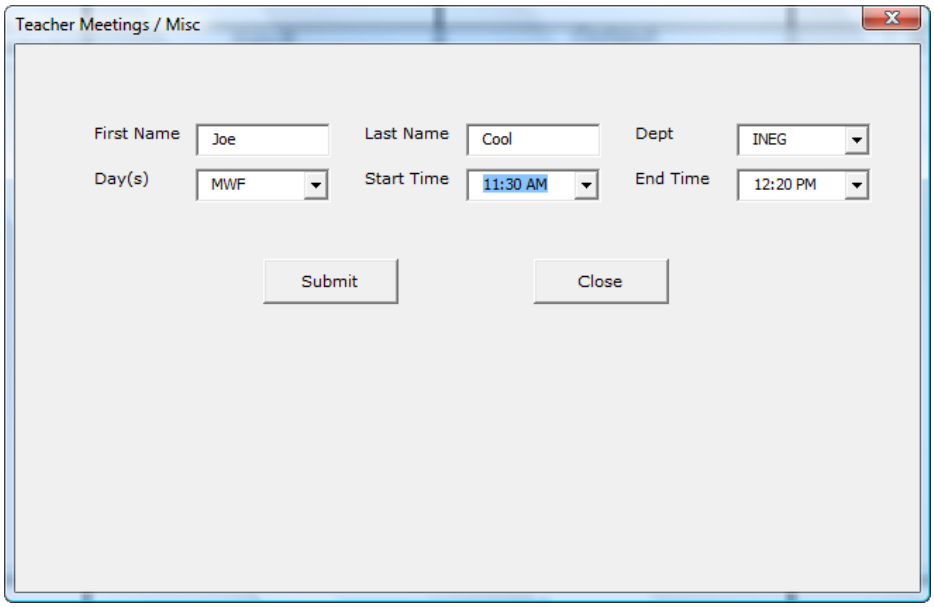

Figure 11 Teachers Meetings / Misc

Output Section

The final function of the office hour scheduling tool is the ability to check the availability of students in a theoretical office hour time period. This would allow teachers, before or after getting the results back, to experiment with various office hours for their specific classes. The output shows the number of students in the class as well as the total number that would be able to attend office hours because of no conflicting schedules with the proposed office hour. *Figure 12* shows the discussed tool ability, as well as the output for the professor's request.

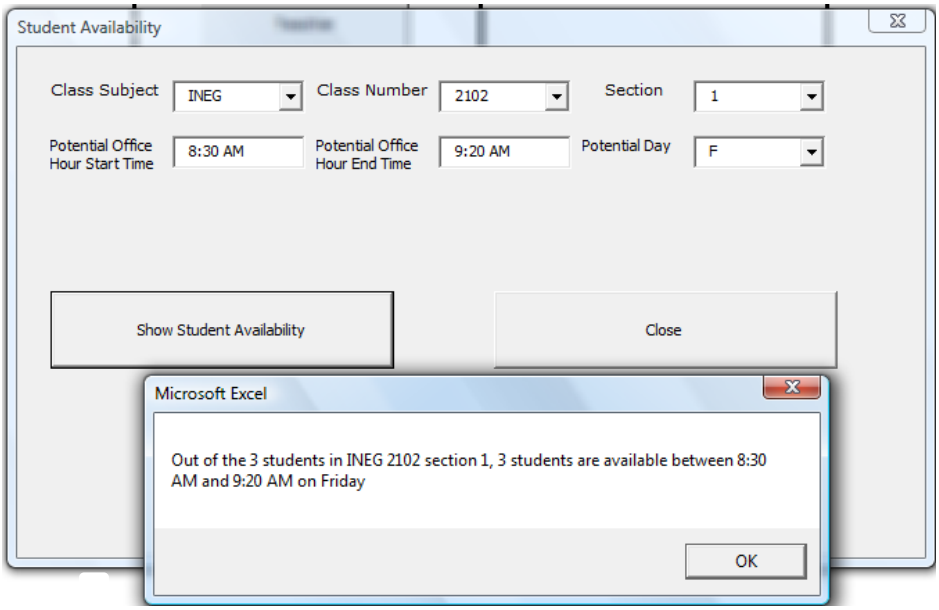

Figure 12 Student Availability Output

In this example, the professor of INEG 2102 is interested in determining how many students could attend an office hour that begin at 9:30am and ended at 10:20am. In this case, 2 out of the 3 students in the class would have no conflicts with that time. In general, this portion of our tool provides post-processing capability to the professor. For example, if the professor has specific times in mind for office hours, this additional functionality will allow them to quantify how many fewer students would be served if they only considered a limited set of office hour possibilities as opposed to choosing the optimal hours that may be at less convenient (but feasible) times.

Implementation and Sustainability

The first step in implementing the tool will be finding an administrator of the tool who is dedicated to it and its success. This will include tool management, tool troubleshooting, and answering questions from students and fellow faculty members. In order to save time in transferring in all of the class data at the beginning, the administrator may check with the ISIS

team to see if there is a copy of current classes and times that can easily be imported into Excel. If not, start with the Industrial Engineering classes first, followed by the university and engineering core classes. These will cover the majority of the classes that current student will be taking from semester to semester.

Once the classes are added, the tool will need to be made available (with security) on something similar to an Engineering SharePoint site, making it readily accessible for students and professors. The tool will need to be reloaded with all of the classes before the start of each semester, and incoming freshman and new students would need an overview of the tool in one of their classes. This may be a good idea for a GNEG class time or even an Intro to IE class. The largest remaining concern is the need for students to simultaneously access the tool. This issue could of course be dealt with by assigning a single staff member to enter all student schedules for the IE department. Alternatively, students could be assigned a time to access the file, if they would like to be considered in the office hour scheduling effort. A final idea would be to expand this tool to include web-based capabilities. This concept is explored further in the future work section that follows.

Future Work

For areas of future study, it may be beneficial to consider enhancing the functionality of the tool itself. The more user-friendly a tool is, the more exposure it will get, and in this case, the more students that will be able to attend office hours. One possible area for future development is the addition of “drill-ups” in the menus that currently only consists of “drill-down” drop boxes. That is, if the user chooses a class number, but realizes he chose the wrong one, he may

not be able to choose the correct one without the "drill-downs" for the subsequent fields including the options for his first choice.

A bigger modification that can be made in the future, if the need arises, is adding an unavailability option for students, similar to the current professor's meetings / misc option. The pros and cons of this would have to be weighed. Introducing new constraints would only drive your objective function down, but this may be a suitable option, especially for the "non-traditional" college students who are becoming more common with each passing year.

A final consideration, worthy of a separate study, would be the development of a web-based version of this decision support tool. Specifically, the shell of the tool presented in this work could be migrated into an ASP.net framework. Developers could take the *GUI* design and implementation offered in the VBA tool and easily portray the same interface to the user on a webpage. The information entered into the ASP.net tool would be stored through a link with an Microsoft Access database, rather than the Excel worksheets utilized in the current version of the tool. The benefits of the web-based effort would be that users could access it from any device connected to the internet and multiple users could easily make use of the tool at the same time.

Conclusions

With the current processes of taking surveys and arbitrarily choosing when an office hour will be for a class, not enough students are getting enough to one-on-one time with teachers that many need to be a successful graduate of the College of Engineering. The Office Hour Scheduling Tool seeks to minimize the number of students that are unavailable for their professor's office hours every week by collecting information from students and faculty, and

using an optimization model designed to give the best times to schedule each office hour. With the help of the Industrial Engineering department, this tool will be a valuable source for professors and students for years to come.