3.6 Documentation of Optimization Tool

Setup

To run our optimization tool, you will need a few pieces of software.

Python

We recommend the Anaconda distribution of python which can be downloaded at https://www.anaconda.com/products/individual. Included in this distribution are numpy and pandas, which allow python to work with excel files and other forms of data.

Gurobi

This powerful optimization tool can be downloaded from https://www.gurobi.com and is licensed for use by USC students and faculty.

Inputs and Outputs

Input File

Your input file should be an .xlsx file with two sheets named "UG_Master_Distribute" and "base_schedule" (no quotes).

UG_Master_Distribute should have the following rows and columns

Columns

- The first two rows of the first column should be blank.
- The following rows of the first column should be the start time of a two hour block of classroom allocation. For example MW, 8:00, 10:00, 12:00, 2:00, 4:00, 6:00, 8:00, TH, 8:00, 10:00, etc...
- The rows marked MW, TH, and F should not have any entries in columns other than the first one.
- All following columns are headed by the name of a Marshall Classroom (ex. ACC201).
- Directly below the name of each Marshall classroom is the seating capacity of that classroom (ex. 48 for ACC201).

Rows

- Each row represents the department allocations of that classroom at a given time.

The Following is a snapshot of How UG_Master_Distribute should look. This is essentially Hal's initial department allocations sheet for undergraduate courses. Cell coloring is not necessary for the optimization to run but can be helpful to the user.

	A	B	C	D	E	F	G
1		ACC201	ACC205	ACC236	ACC303	ACC310	BRI5
2		48	36	39	46	54	42
3	MW						
4	8:00	ACCT	BUCO		ACCT	ACCT	ACCT
5	10:00	ACCT	BUCO		ACCT	ACCT	ACCT
6	12:00	ACCT	BUCO	ACCT	ACCT	ACCT	ACCT
7	2:00	ACCT	BUCO	ACCT	ACCT	FBE	ACCT
8	4:00	ACCT	BUCO	ACCT	ACCT	FBE	ACCT
9	6:00	ACCT	BUCO	ACCT			
10	8:00	ACCT		ACCT			
11	TH						
12	8:00	ACCT	DSO		ACCT	ACCT	ACCT
13	10:00	ACCT	DSO		ACCT	ACCT	ACCT

base _schedule represents the courses offered by each department should have the following columns. Each row represents a section of a course, where each section has the following properties.

Columns

- section : the section code of the course (to identify unique course sections)
- department: the department of the course in question
- course: the course name - necessary for
- level : the level (UG for undergraduate and G for graduate)
- seats_offered: The number of seats offered in this section of the course
- first_days: the days on which the course is offered, for example MW or TH
- first_begin_time: the start time of the course (can be the same as past years)
- first_end_time: the end time of the course (can be the same as past years)

The columns first_start_time and first_end_time are used to calculate the length of the class in minutes, and do not necessarily indicate the proposed times of the course. For new courses with no precedent for start and end times, the user can input 08:00 as the start time and select the end time based on the desired duration of the proposed course. Note that all times use a 24 hour clock. Below is an example of the base_schedule sheet of your input file.

	A	B	C	D	E	F	G	H
1	section	department	course	level	seats_offered	first_days	first_begin_time	first_end_time
2	14010	ACCT	ACCT-380	UG	25	TH	12:00:00	13:50:00
3	14012	ACCT	ACCT-385	UG	20	H	18:00:00	21:30:00
4	14025	ACCT	ACCT-370	UG	38	MW	8:00:00	9:50:00
5	14026	ACCT	ACCT-370	UG	39	MW	10:00:00	11:50:00
6	14027	ACCT	ACCT-370	UG	39	MW	12:00:00	13:50:00
7	14029	ACCT	ACCT-370	UG	109	F	10:00:00	11:50:00
8	14040	ACCT	ACCT-371	UG	39	TH	12:00:00	13:50:00

Output File

The output of the optimization will be a .xlsx file with a sheet for each department listing the suggested room for each course section. Each is titled for the department in question. For example 'DSO Class Allocations'. Each of the excel sheets has the following columns.

- Class_Section: The section number of the course
- Room: The suggested room - available to the department according to the classroom allocations
- Course: The course name (there will be many repeating values as courses have multiple sections)
- Average Empty Seats per Class: The average number of empty seats across all rooms for this department (smaller is better)

Below is an example of the output for the DSO department titled 'DSO Class Allocations'

	A	B	C	D
1	Class_Section	Room	Course	Average Empty Seats per Class
2	14882	JFF105	BUAD-310	6.448275862
3	14883	JFF 240	BUAD-310	
4	14884	JFF 331	BUAD-310	
5	14885	JFF 331	BUAD-310	
6	14886	JFF 331	BUAD-310	
7	14892	JFF 331	BUAD-310	
8	14893	JFF 331	BUAD-310	

Using The Optimization Tool

Before you run the optimization tool, make sure your input file is located in the same folder as the optimize.py file. This will allow python to access your input file without any additional steps. Once the gurobi and the anaconda distribution of python have been installed, your input file is formatted correctly, and your input file is in the same folder as optimize.py, you are ready to begin optimizing.

1. Navigate to the folder containing your input file and optimize.py.
   Windows

   a. Open an Anaconda Command Prompt
   b. Make sure to change to the directory path where the files are located, for example, if your current directory is C:\\Users\\USC, and the files are located in C:\\Users\\USC\\Downloads, simply type 'cd Downloads'.

   Mac

   a. Right-click on the folder where optimize.py and your input file are located, then select the very last option "New Terminal at Folder". This will open a new command line window where you can run the optimization tool.
   b. Alternatively, you can open "Terminal": an application located in the utilities folder that allows you to run programs using the command line.
      i. Terminal will start in your home directory by default. To see the list of folders in a directory, type **ls** and press enter.
      ii. To navigate into your folder of choice, type **cd** followed by the name of the folder. For example **cd Desktop** followed by enter.
      iii. If your desired folder is deep in your system, you may have to perform step b multiple times. To autocomplete the name of your folder after you have started typing it, press tab.

2. Decide what you would like to call your input and output files. For this example, suppose your input file is called **input.xlsx** and your output file will be called **output.xlsx**
3. Once you have navigated to the folder where both optimize.py and your input file are located, type **python optimize.py input.xlsx output.xlsx**
   a. If the program returns **File {Filename} not Found** or a similar output, make sure all the files are named exactly as in the command line above.
   b. If the program returns another error, it is likely because one of the classroom allocations is not feasible and does not have enough space for the proposed courses.
4. After the optimizer has finished running (it should take no more than a few seconds), check your output file for the results, the program will have created a new file with the name you specified, or overwritten a file with the same name if one existed previously.

Congratulations! You are ready to use the results of our optimization tool and continue the process of course scheduling.

Fight On!