Modeling Class Scheduling Using Linear Programming

Mir Tafseer Nayeem
Graduate Student, University of Lethbridge
mir.nayeem@uleth.ca

Problem Statement

This project addresses the problem of class scheduling to be used by students to choose their classes / courses according to their preferences. Each student has different preferences. In this project of class scheduling implements only the choices of a specific student called Sara. This project also deals with the possible extensions according to the choice of Sara. The Sara's course choices is listed in the project description provided by the instructor named as Course Data.
She made a list of the courses according to her preferences from the online class schedule and rate them between 1 and 5. The overall rating is the weighted average of the following parameters,

1. Content of the course based on her interest.
2. Reputation of the instructor from previous experience and word of mouth.
3. Timing of the course by avoiding morning classes.

The objective of this project is to make a schedule for Sara that will maximize the total rating.
Sara is a final year student of Takshashila University (TU), university imposes some specific requirements/constraints listed below.

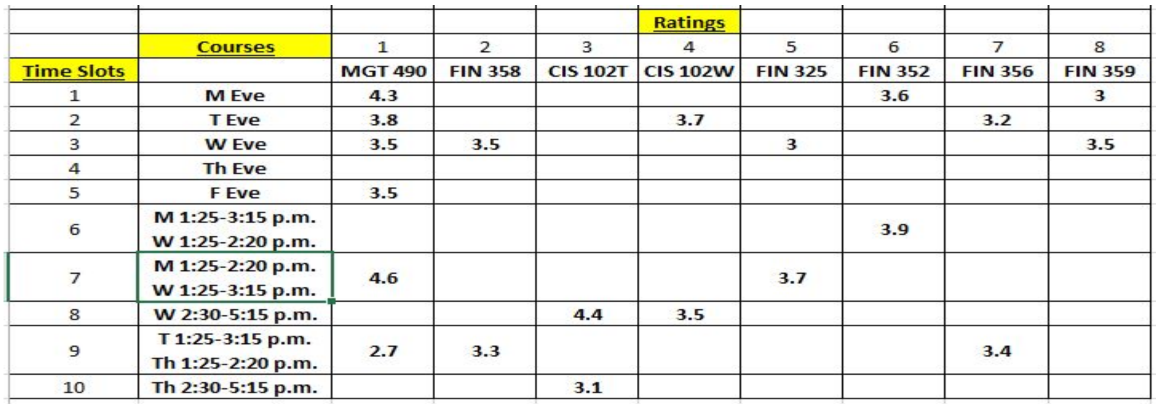
1. She needs to take exactly five courses.
2. She is a management major so she needs to take two compulsory courses Business Strategy (MGT 490), International Finance (FIN 358).
3. One of the service learning courses (Sara finds interesting CIS 102T & CIS 102W).
4. She needs to take exactly two courses out of FIN 325, FIN 352, FIN 356, and FIN 359.
5. She cannot take more than one section of each course.
6. She cannot take more than one class during any time slot or overlapping time slot.

The problem of scheduling the courses of Sara according to her preferences and university constraints can be solved using two possible methods,

1. Heuristic: A heuristic can be choosing the course with the highest rating.
2. Linear Program (LP): The problem can be expressed as a Linear Programming model.

Input Data

First we need to organize the data in a meaningful way that can easy to read off and will help us to implement our selected solution strategies. Such a data table is shown in Figure 1. The rows in the data table represent distinct time slots, the columns represent distinct courses. The entries corresponding to the specific time slot and course is the ratings. Doing this we discovered that there are 8 different courses are offered in 10 different time slots. The distinct courses and the time slots numbered accordingly. This enhances the understanding of the problem.



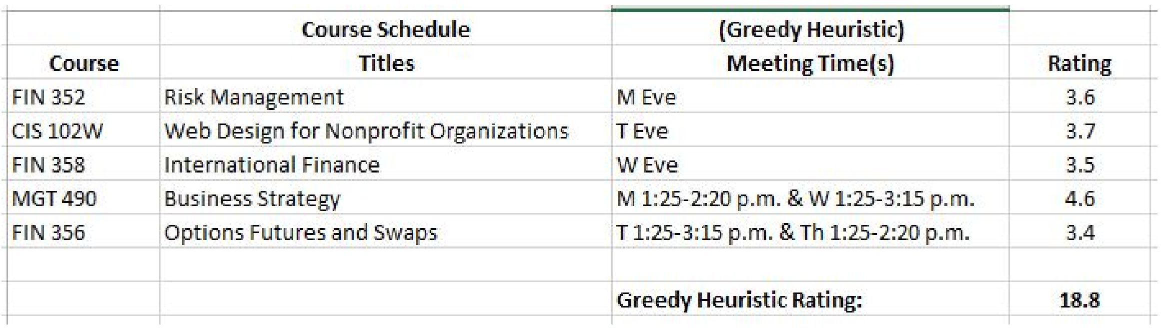
					Ratings				
	Courses	1	2	3	4	5	6	7	8
Time Slots		MGT 490	FIN 358	CIS 102T	CIS 102W	FIN 325	FIN 352	FIN 356	FIN 359
1	M Eve	4.3					3.6		3
2	T Eve	3.8			3.7			3.2	
3	W Eve	3.5	3.5			3			3.5
4	Th Eve								
5	F Eve	3.5							
6	M 1:25-3:15 p.m. W 1:25-2:20 p.m.						3.9		
7	M 1:25-2:20 p.m. W 1:25-3:15 p.m.	4.6				3.7			
8	W 2:30-5:15 p.m.			4.4	3.5				
9	T 1:25-3:15 p.m. Th 1:25-2:20 p.m.	2.7	3.3					3.4	
10	Th 2:30-5:15 p.m.			3.1					

Figure 01 : Data Table

Solution Strategies

Heuristic: A simple greedy heuristic has been implemented by choosing the highest rating choice at each step considering no overlap with the previously selected courses.

A C++ program was written to implement this heuristic. The program data described in Figure is stored into a text file. The program loads the data from this text file into a dynamic 2D array and perform the greedy operations. At first the conflicting time slots (e.g. 6, 7 and 8) are identified. The greedy heuristic algorithm chooses the highest rated cell at each time. For instance, the algorithm chooses the cell in row 7 and column 1 with 4.6, then the course MGT 490 and corresponding time slot MW 1:25-2:20 p.m. is added to the schedule. Then row 7 is checked in the conflicting time slots array if found then all conflicting rows are deleted as well as the column 1. Then this process is repeated for all compulsory and elective courses. The program has been developed in way that it can adapt in any situation and data given. Finally, it produces a feasible schedule with the total rating of 18.8 and store it in a CSV file in a readable format shown in Figure 2. The complete implementation is attached as an appendix.



	Course Schedule	(Greedy Heuristic)	
Course	Titles	Meeting Time(s)	Rating
FIN 352	Risk Management	M Eve	3.6
CIS 102W	Web Design for Nonprofit Organizations	T Eve	3.7
FIN 358	International Finance	W Eve	3.5
MGT 490	Business Strategy	M 1:25-2:20 p.m. & W 1:25-3:15 p.m.	4.6
FIN 356	Options Futures and Swaps	T 1:25-3:15 p.m. & Th 1:25-2:20 p.m.	3.4
		Greedy Heuristic Rating:	18.8

Figure 02 : Readable Output Format

Linear Programming (LP) Model: I used binary integer linear programming (BILP) to model the course scheduling problem. The model details are given below.

Decision Variables: The decision variables will contain binary values either the course is scheduled in a particular time slot not.

$$x_{ij} = \begin{cases} 1, & if\ the\ time\ slot\ i\ \ has\ been\ scheduled\ for\ course\ j \\ 0, & otherwise \end{cases}$$

Where $i = \{1, 2, \ldots\ldots 10\ \}\ and\ j = \{1, 2, \ldots\ldots 8\}$

In total we have 80 decision variables.

Parameters: The constant data for the model shown in Figure: 01

Course Rating Matrix:

$$c_{ij} = \begin{cases} +ve, & if\ the\ course\ j\ has\ a\ meeting\ time(s)\ on\ i \\ 0, & otherwise \end{cases}$$

Where $i = \{1, 2, \ldots\ldots 10\ \}\ and\ j = \{1, 2, \ldots\ldots 8\}$

Objective: The objective is to maximize the total rating for the courses scheduled.

$$\sum_{i=1}^{10}\sum_{j=1}^{8} c_{ij}\ x_{ij}$$

<u>**Course Constraints:**</u>

1. **Graduation Requirement:** She needs to take exactly five courses. So the sum of all the course selection variables has to be five.

$$\sum_{i=1}^{10}\sum_{j=1}^{8} x_{ij} = 5$$

2. **Compulsory Course Constraints:** She is a management major so she needs to take two compulsory courses Business Strategy (MGT 490), International Finance (FIN 358) they are the course IDs 1 and 2 accordingly.

$$\sum_{i=1}^{10} x_{ij} = 1 \quad , \quad j = 1\ , 2$$

3. **Elective Course Constraints:**

   One of the service learning courses (Sara finds interesting CIS 102T & CIS 102W with the IDs 3 and 4 accordingly).

$$\sum_{i=1}^{10} x_{i3} + \sum_{i=1}^{10} x_{i4} = 1$$

   She needs to take exactly two courses out of FIN 325, FIN 352, FIN 356, and FIN 359 with IDs 5, 6, 7 and 8 accordingly.

$$\sum_{i=1}^{10} x_{i5} + \sum_{i=1}^{10} x_{i6} + \sum_{i=1}^{10} x_{i7} + \sum_{i=1}^{10} x_{i8} = 2$$

4. **Course Section Constraints:**

   She cannot attend classes more than one section of each course.

$$\sum_{i=1}^{10} x_{ij} \leq \; 1 \quad , \quad j = 3, \; \ldots, \; 8$$

   Note: Compulsory courses are not added in this section constraints because those are already met previously. Adding them will be redundant here.

<u>**Time slot Constraints:**</u>

Non-overlapping time slots

The non-overlapping time slots are (MTWThF Eve, T 1:25-3:15 Th 1:25-2:20, Th 2:30-5:15 p.m.). Sara cannot take more than one class during any time slot.

$$\sum_{j=1}^{8} x_{ij} \leq \; 1 \quad , \quad i = 1, \; 2, \; 3, \; 4, \; 5, \; 9 \; \textit{and} \; 10$$

Overlapping time slots

Conflicting time slots are (M 1:25-3:15 W1:25-2:20, M 1:25-2:20 W1:25-3:15, W 2:30-5:15 p.m.) with the IDs [6, 7 and 8] where the pair time slots (M 1:25-3:15 W1:25-2:20, M 1:25-2:20 W1:25-3:15 p.m.) conflict with each other. On the other hand (M 1:25-2:20 W1:25-3:15, W 2:30-5:15 p.m.) conflicts with each other.

$$\sum_{j=1}^{8} x_{6j} + \sum_{j=1}^{8} x_{7j} \leq \; 1$$

$$\sum_{j=1}^{8} x_{7j} + \sum_{j=1}^{8} x_{8j} \leq \; 1$$

So, the complete Primal ILP becomes,

maximize

$$\sum_{i=1}^{10} \sum_{j=1}^{8} c_{ij} \; x_{ij}$$

Subject to,

$$\sum_{i=1}^{10} \sum_{j=1}^{8} x_{ij} = \; 5$$

$$\sum_{i=1}^{10} x_{ij} = \; 1 \quad , \quad j = 1 \; , 2$$

$$\sum_{i=1}^{10} x_{i3} + \sum_{i=1}^{10} x_{i4} = \; 1$$

$$\sum_{i=1}^{10} x_{i5} + \sum_{i=1}^{10} x_{i6} + \sum_{i=1}^{10} x_{i7} + \sum_{i=1}^{10} x_{i8} = 2$$

$$\sum_{i=1}^{10} x_{ij} \leq 1 \quad , \quad j = 3, \; \ldots., \; 8$$

$$\sum_{j=1}^{8} x_{ij} \leq 1 \quad , \quad i = 1, \; 2, \; 3, \; 4, \; 5, \; 9 \; and \; 10$$

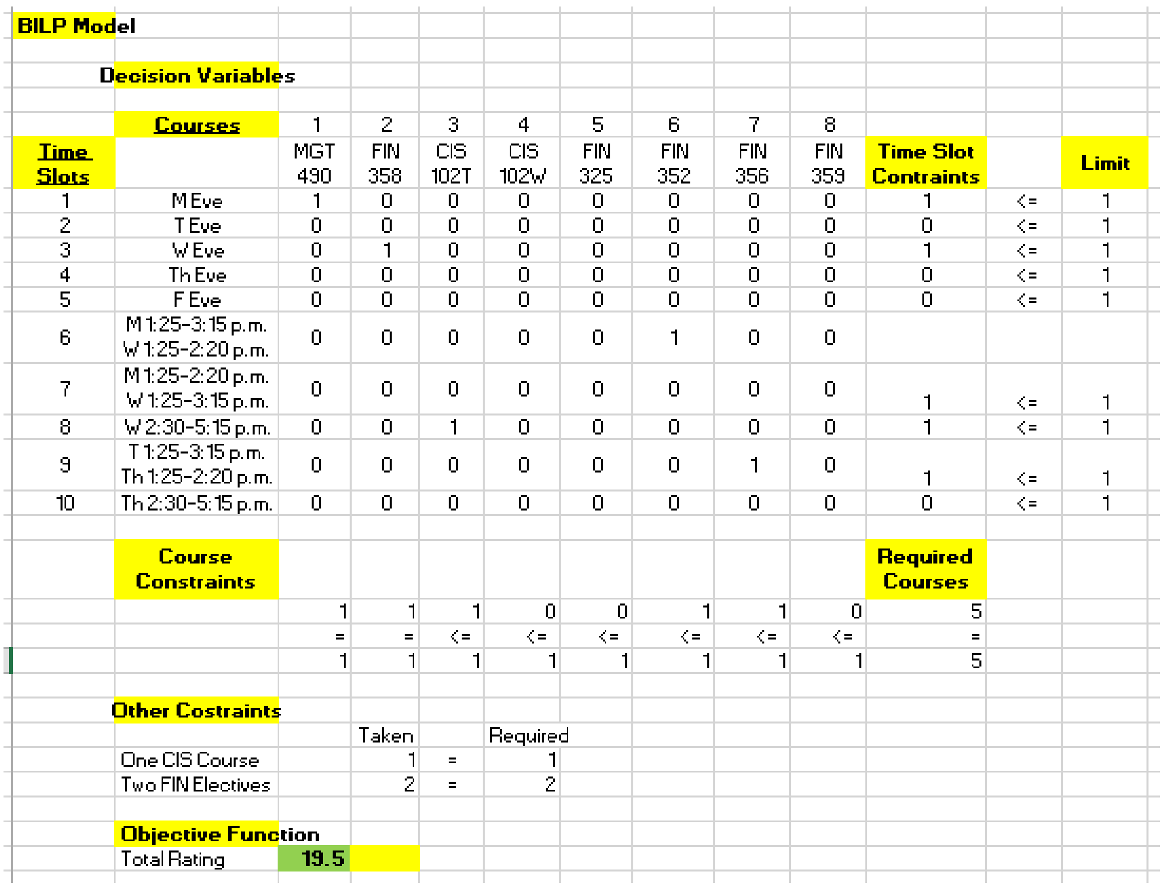
$$\sum_{j=1}^{8} x_{6j} + \sum_{j=1}^{8} x_{7j} \leq 1$$

$$\sum_{j=1}^{8} x_{7j} + \sum_{j=1}^{8} x_{8j} \leq 1$$

$$x_{ij} \geq 0 \quad for \; i = 1, \ldots., \; 10 \; and \; j = 1, \ldots.., \; 8$$

As the right hand side of each constraint is 1 along with the non-negativity $x_{ij} \geq 0$ forces the LP to become BILP (Binary Integer Linear Programming) implicitly. So our initial requirement $x_{ij} = 0$ or 1 has been met by default.We will see it later.

I used the solver add-in of Microsoft Excel to test our integer programming model to test our objective from the given rating data. The advantage of using excel Solver is that it provides an easy to use interface to implement the model. One disadvantage of excel solver is that it can solve problems with up to 200 decision variables. Our problem luckily has 80 decision variables, the objective value obtained from the above BILP is 19.5. However, it's better than the greedy heuristics because of local optimum selection of the MGT 490 without having the complete knowledge of the rating matrix. Figure 3 displays the Excel implementation of our model with optimal solution.



BILP Model												
	Decision Variables											
	Courses	1	2	3	4	5	6	7	8			
Time Slots		MGT 490	FIN 358	CIS 102T	CIS 102W	FIN 325	FIN 352	FIN 356	FIN 359	Time Slot Contraints		Limit
1	M Eve	1	0	0	0	0	0	0	0	1	<=	1
2	T Eve	0	0	0	0	0	0	0	0	0	<=	1
3	W Eve	0	1	0	0	0	0	0	0	1	<=	1
4	Th Eve	0	0	0	0	0	0	0	0	0	<=	1
5	F Eve	0	0	0	0	0	0	0	0	0	<=	1
6	M 1:25-3:15 p.m. W 1:25-2:20 p.m.	0	0	0	0	0	1	0	0			
7	M 1:25-2:20 p.m. W 1:25-3:15 p.m.	0	0	0	0	0	0	0	0	1	<=	1
8	W 2:30-5:15 p.m.	0	0	1	0	0	0	0	0	1	<=	1
9	T 1:25-3:15 p.m. Th 1:25-2:20 p.m.	0	0	0	0	0	0	1	0	1	<=	1
10	Th 2:30-5:15 p.m.	0	0	0	0	0	0	0	0	0	<=	1
	Course Constraints									Required Courses		
		1	1	1	0	0	1	1	0	5		
		=	=	<=	<=	<=	<=	<=	<=	=		
		1	1	1	1	1	1	1	1	5		
	Other Costraints											
			Taken		Required							
	One CIS Course		1	=	1							
	Two FIN Electives		2	=	2							
	Objective Function											
	Total Rating	19.5										

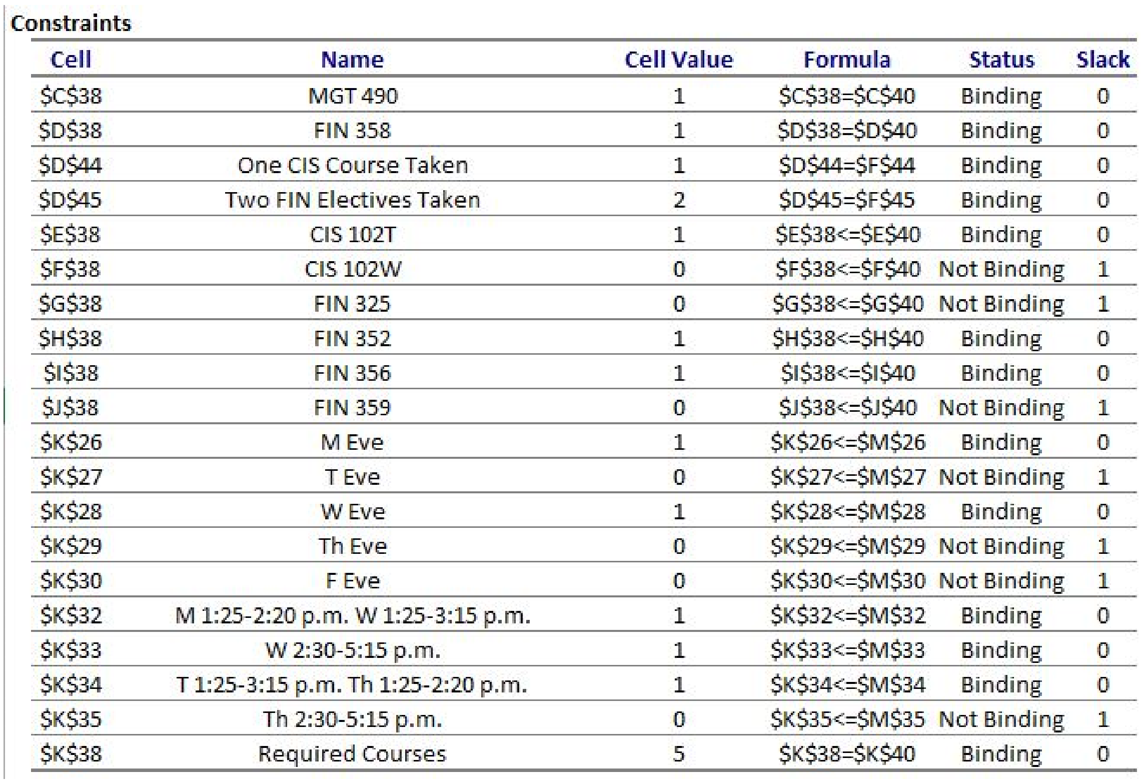
Figure 03 : Excel Implementation of the model with Optimum Solution

As I have previously said, even if I dont specify the binary constraint, this LP has binary constraint implicitly. While solving the problem in Excel Solver, I found that having the binary constraint set, the solver doesnt produce any sensitivity report. I will explain it later.

Lets first see the answer report produced by solver in Figure 4.The report indicates that for any particular selection the corresponding time slot constraint and the course constraint are binding. That means their rating data has taken part in the optimal objective function value. One very important point is that, the other constraints are non-binding with 1 in the slack. That means they didnt take part in the optimal scheduling of courses.

In the Figure 5, the shadow prizes of all the binding constraints are non-zero. That means the objective function can be increased/decreased by this amount (shadow price) by one unit change in the right hand side. However, for this particular problem of BILP the allowable increases are 0 as this increase will make them non-binding or sometimes infeasible as previously mentioned about the course and slot constraints right hand side ¡= 1.

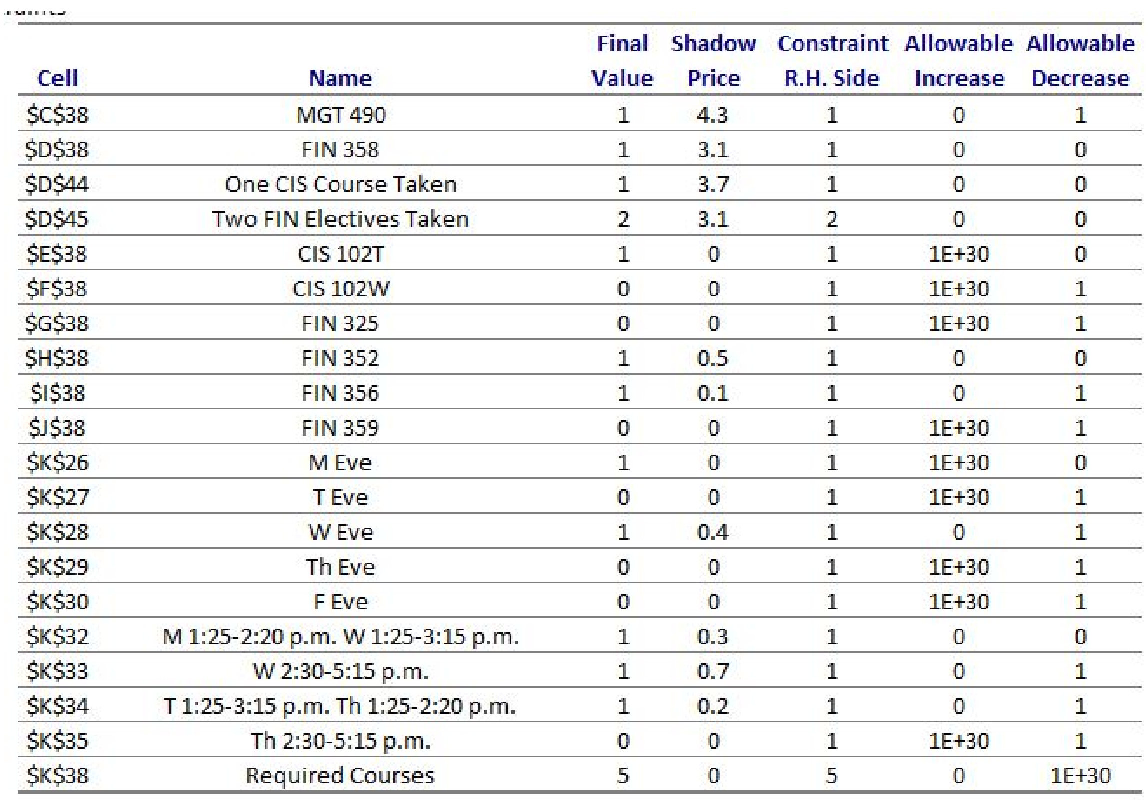
Some of the binding constraint in Figure can decrease up to a certain limit and can still remain binding. For instance the shadow price of constraint associated with T 1:25-3:15 p.m. Th 1:25-2:20 p.m is 0.2 and it can decrease 1 unit. So the new objective will be (19.5 (0.2 x 1)) = 19.3, this is exactly the value we will get in the extension of our LP. The value then becomes 0 from 1 in our extension for this cell.



Constraints

Cell	Name	Cell Value	Formula	Status	Slack
$C$38	MGT 490	1	$C$38=$C$40	Binding	0
$D$38	FIN 358	1	$D$38=$D$40	Binding	0
$D$44	One CIS Course Taken	1	$D$44=$F$44	Binding	0
$D$45	Two FIN Electives Taken	2	$D$45=$F$45	Binding	0
$E$38	CIS 102T	1	$E$38<=$E$40	Binding	0
$F$38	CIS 102W	0	$F$38<=$F$40	Not Binding	1
$G$38	FIN 325	0	$G$38<=$G$40	Not Binding	1
$H$38	FIN 352	1	$H$38<=$H$40	Binding	0
$I$38	FIN 356	1	$I$38<=$I$40	Binding	0
$J$38	FIN 359	0	$J$38<=$J$40	Not Binding	1
$K$26	M Eve	1	$K$26<=$M$26	Binding	0
$K$27	T Eve	0	$K$27<=$M$27	Not Binding	1
$K$28	W Eve	1	$K$28<=$M$28	Binding	0
$K$29	Th Eve	0	$K$29<=$M$29	Not Binding	1
$K$30	F Eve	0	$K$30<=$M$30	Not Binding	1
$K$32	M 1:25-2:20 p.m. W 1:25-3:15 p.m.	1	$K$32<=$M$32	Binding	0
$K$33	W 2:30-5:15 p.m.	1	$K$33<=$M$33	Binding	0
$K$34	T 1:25-3:15 p.m. Th 1:25-2:20 p.m.	1	$K$34<=$M$34	Binding	0
$K$35	Th 2:30-5:15 p.m.	0	$K$35<=$M$35	Not Binding	1
$K$38	Required Courses	5	$K$38=$K$40	Binding	0

Figure 04 : Answer Report Produced by Excel Solver



		Final	Shadow	Constraint	Allowable	Allowable
Cell	Name	Value	Price	R.H. Side	Increase	Decrease
$C$38	MGT 490	1	4.3	1	0	1
$D$38	FIN 358	1	3.1	1	0	0
$D$44	One CIS Course Taken	1	3.7	1	0	0
$D$45	Two FIN Electives Taken	2	3.1	2	0	0
$E$38	CIS 102T	1	0	1	1E+30	0
$F$38	CIS 102W	0	0	1	1E+30	1
$G$38	FIN 325	0	0	1	1E+30	1
$H$38	FIN 352	1	0.5	1	0	0
$I$38	FIN 356	1	0.1	1	0	1
$J$38	FIN 359	0	0	1	1E+30	1
$K$26	M Eve	1	0	1	1E+30	0
$K$27	T Eve	0	0	1	1E+30	1
$K$28	W Eve	1	0.4	1	0	1
$K$29	Th Eve	0	0	1	1E+30	1
$K$30	F Eve	0	0	1	1E+30	1
$K$32	M 1:25-2:20 p.m. W 1:25-3:15 p.m.	1	0.3	1	0	0
$K$33	W 2:30-5:15 p.m.	1	0.7	1	0	1
$K$34	T 1:25-3:15 p.m. Th 1:25-2:20 p.m.	1	0.2	1	0	1
$K$35	Th 2:30-5:15 p.m.	0	0	1	1E+30	1
$K$38	Required Courses	5	0	5	0	1E+30

Figure 05 : Sensitivity Report Produced by Excel Solver

The sensitivity analysis for a nonbinding constraint is different. At the optimal solution, changes to the right hand side do not affect the total rating. This means that the shadow price is 0 (as shown in Figure). The Allowable Increase for these constraints are show as 1E+30. This is Excels way of showing infinity. This means that the right hand side can be increased any amount without changing the shadow price and will not change the feasible region also the optimum objective.

Extension

In this section, I am going to extend my previous LP by limiting the number of days students have to be at school.

$$y_k = \begin{cases} 1, & if\ any\ course\ taken\ on\ day\ K \\ 0, & otherwise \end{cases}$$

$$(where,\ \ k = \ M,\ T,\ W,\ Th,\ F)$$

Sarah want to attend class only three days a week to allocate some time for relaxation, so the corresponding constraint becomes,

$$y_M + y_T + y_W \ \ + y_{Th} + y_F = 3$$

$Y = 1$ means Sarah can take classes on this day, on the other hand $y = 0$ means no class on this day hence the off day reserved for relaxation. On a given day she cannot take more than 5 courses (the total number of course she should take in her last semester). Therefore, the additional constraints are added,
For Monday,

$$\sum_{j=1}^{8} x_{1j} + \sum_{j=1}^{8} x_{6j} + \sum_{j=1}^{8} x_{7j} \leq \ 5\ y_M$$

For Tuesday,

$$\sum_{j=1}^{8} x_{2j} + \sum_{j=1}^{8} x_{9j} \leq \ 5\ y_T$$

For Wednesday,

$$\sum_{j=1}^{8} x_{3j} + \sum_{j=1}^{8} x_{6j} + \sum_{j=1}^{8} x_{7j} + \sum_{j=1}^{8} x_{8j} \leq \ 5\ y_W$$

For Thursday,

$$\sum_{j=1}^{8} x_{4j} + \sum_{j=1}^{8} x_{9j} + \sum_{j=1}^{8} x_{10j} \leq \ 5\ y_{Th}$$

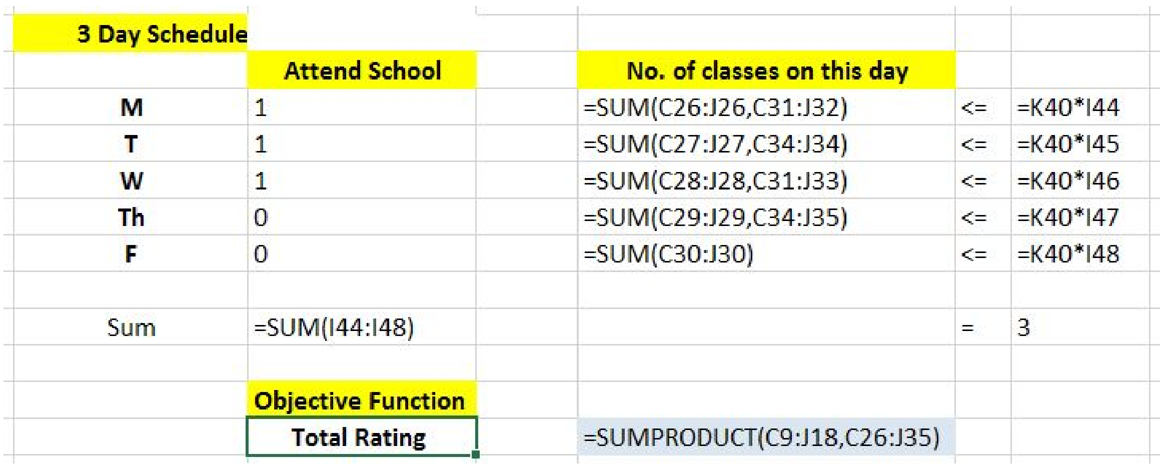
For Friday,

$$\sum_{j=1}^{8} x_{5j} \ \ \leq \ 5\ y_F$$

$$y_M + y_T + y_W \ \ + y_{Th} + y_F = 3$$

$$y_M \ \ , y_T, y_W \ \ , y_{Th}, y_F = 0\ or\ 1$$

The extension also implemented using excel solver shown in Figure 6 , and found that the overall rating becomes 19.3. Previous optimal was 19.5, by extending the LP the overall rating is not that much lower. Sarah will be happy to grant this 3 day schedule.



3 Day Schedule					
	Attend School		No. of classes on this day		
M	1		=SUM(C26:J26,C31:J32)	<=	=K40*I44
T	1		=SUM(C27:J27,C34:J34)	<=	=K40*I45
W	1		=SUM(C28:J28,C31:J33)	<=	=K40*I46
Th	0		=SUM(C29:J29,C34:J35)	<=	=K40*I47
F	0		=SUM(C30:J30)	<=	=K40*I48
Sum	=SUM(I44:I48)			=	3
	Objective Function				
	Total Rating		=SUMPRODUCT(C9:J18,C26:J35)		

Figure 06 : Solver Solution for the extended LP

APPENDIX

```
//*******************************************************************************
// CPSC 5110
// Term Project for Computational Optimization
// File: Heuristic.cpp
// Purpose: Program to produce a feasible class schedule using Greedy
   Heuristic
// Written by: Mir Tafseer Nayeem
// Instructed By: Dr. Shahadat Hossain
//*******************************************************************************

#include <iostream>
#include <iomanip>
#include <fstream>

using namespace std;

//Constant Declarations
const int COLS=8;
const int ROWS=10;
const int conflictingTimeslots[] = {5,6,7};

//Define structure
struct course
{
  double rating;
  int row;
  int col;
};
```

```
//Function Prototypes
void loadData(double **);

void writeToFile(double **);

void writeToConsole(double **);

void deallocateArray(double **);

void chooseCompulsoryCourse(double **, int);

void chooseOptionalCourse(double **, int, int, int);

int loadCourses(double **, int, int);

void sortCourses(course [], int);

void crossOutSlots(double **, int, int);

double determineHeuristic(double **);

//*************************************************************************************
//Function: main function
//Purpose: Controling the overall problem flow.
//        This Program has been designed in a way that in can be easily
   reconfigarable
//        This can be more dynamic easily using user inputs
//*************************************************************************************

int main() {

   double** dataArray = new double*[ROWS];
   for(int i = 0; i < ROWS; ++i) {
      dataArray[i] = new double[COLS];
   }

   loadData(dataArray);


   chooseCompulsoryCourse(dataArray,0);

   chooseCompulsoryCourse(dataArray,1);

   chooseOptionalCourse(dataArray, 2,3,1);

   chooseOptionalCourse(dataArray, 4,7,2);


   writeToFile(dataArray);

   deallocateArray(dataArray);

}
```

```
//*******************************************************************
//Function: loadData
//Purpose: To load data from an input to data Array
//*******************************************************************

void loadData(double **dataArray){

   ifstream fin;
   fin.open("input.txt");

   while (!fin.eof()){
   for(int i = 0; i < ROWS; ++i) {
      for(int j = 0; j < COLS; ++j) {
         fin >> dataArray[i][j];
      }
   }
   fin.close();
   }
}

//*******************************************************************
//Function: writeToFile
//Purpose: To write into a CSV file according to a readable froamt
//*******************************************************************

void writeToFile(double **dataArray){

   ofstream fout;
   fout.open("output.CSV");
   double heuristicRating=0.0;

   char coursesCodes[][50] = {"MGT 490","FIN 358","CIS 102T",
                             "CIS 102W", "FIN 325","FIN 352",
                        "FIN 356","FIN 359"};

   char coursesTitles[][50] = {
                             "Business Strategy","International Finance",
                             "Intergenerational Computing",
                             "Web Design for Nonprofit Organizations",
                             "Data Analysis in Finance","Risk Management",
                        "Options Futures and Swaps","Fixed Instruments and
                           Markets"
                        };

   char timeSlots[][50] = {
                         "M Eve","T Eve","W Eve", "Th Eve", "F Eve",
                         "M 1:25-3:15 p.m. & W 1:25-2:20 p.m.",
                    "M 1:25-2:20 p.m. & W 1:25-3:15 p.m.",
                    "W 2:30-5:15 p.m.",
                    "T 1:25-3:15 p.m. & Th 1:25-2:20 p.m.",
                    "Th 2:30-5:15 p.m."};
   fout <<""<< ","
       <<"Course Schedule"<< ","
       <<"(Greedy Heuristic)"<< ","
       <<""<<","
       <<endl;

    fout << "Course"<< ","
       << "Titles"<< ","
       <<"Meeting Time(s)"<< ","
       <<"Rating";

   fout << endl;

    for(int i = 0; i < ROWS; ++i)
      for(int j = 0; j < COLS; ++j) {
         if(dataArray[i][j] > 0){
            fout << coursesCodes[j]<< ",";
            fout << coursesTitles[j]<< ",";
            fout << timeSlots[i]<< ",";
            fout << dataArray[i][j];
            fout << endl;
         }
      }

   heuristicRating = determineHeuristic(dataArray);

   fout << endl;

   fout << ""<< ","
        << ""<< ","
        << "Greedy Heuristic Rating:"<< ","
        <<heuristicRating;

   fout.close();
}


//*****************************************************************
//Function: determineHeuristic
//Purpose: Determine the Total Heuristic Rating
//*****************************************************************

double determineHeuristic(double **dataArray){

   double heuristicRating=0.0;

   for(int i = 0; i < ROWS; ++i)
      for(int j = 0; j < COLS; ++j)
         if(dataArray[i][j] > 0.0)
            heuristicRating = heuristicRating + dataArray[i][j];

   return heuristicRating;
}
```

```
//*****************************************************************
//Function: deallocateArray
//Purpose: Deallocates the array used in this program
//*****************************************************************

void deallocateArray(double **dataArray){

   for(int i = 0; i < ROWS; ++i) {
      delete [] dataArray[i];
   }
   delete [] dataArray;
}

//*****************************************************************
//Function: chooseCompulsoryCourse
//Purpose: Choose compulsory course with Highest Rating
//*****************************************************************

void chooseCompulsoryCourse(double **dataArray, int j){

   double maxRating=0.0;
   int timeSlotIndex;

   for(int i = 0; i < ROWS; ++i) {
      if(dataArray[i][j] > maxRating ){
         maxRating = dataArray[i][j];
         timeSlotIndex = i;
      }
   }

   crossOutSlots(dataArray, timeSlotIndex, j);
}

//*****************************************************************
//Function: loadCourses
//Purpose: Loads Optional courses in a Structure
//*****************************************************************

int loadCourses(double **dataArray, course optionalCourse[], int
    startCourseIndex, int endCourseIndex){

   int k=0;

   for(int i = 0; i < ROWS; ++i)
      for(int j = startCourseIndex; j <= endCourseIndex; ++j)
         if(dataArray[i][j] > 0){
            optionalCourse[k].rating=dataArray[i][j];
            optionalCourse[k].row=i;
            optionalCourse[k].col=j;
            k++;
         }
     return k;
}
```

```
//****************************************************************
//Function: chooseOptionalCourse
//Purpose: Choose Optional courses with Highest Rating
//****************************************************************

void chooseOptionalCourse(double **dataArray, int startCourseIndex, int
    endCourseIndex, int numberOfCourses){

   int length = ((endCourseIndex - startCourseIndex) + 1 ) * ROWS;
   course optionalCourse[length];

   int k;

    k = loadCourses(dataArray, optionalCourse, startCourseIndex,
        endCourseIndex);
   sortCourses(optionalCourse,k);

   for(int i=0; i<numberOfCourses; ++i){
      crossOutSlots(dataArray, optionalCourse[i].row,
          optionalCourse[i].col);
   }

   for(int j=numberOfCourses; j<k; ++j){
      if(optionalCourse[j].rating > 0){
             int m=optionalCourse[j].row;
             int n=optionalCourse[j].col;
         dataArray[m][n]=0;
      }
   }

}

//****************************************************************
//Function: sortCourses
//Purpose: Sorts Optional courses according to Highest Rating
//****************************************************************
void sortCourses(course optionalCourse[], int numberOfCourses){

   course temp;

   for(int i=0; i<numberOfCourses; i++){
      for(int j=i+1; j<numberOfCourses; j++)
      if(optionalCourse[i].rating < optionalCourse[j].rating){
         temp = optionalCourse[i];
         optionalCourse[i]= optionalCourse[j];
         optionalCourse[j] = temp;
      }
  }
}
```

```
//****************************************************************************
//Function: crossOutSlots
//Purpose: Crossing out Overlaping Schedules
//        General Program used to overlaping schedule
//          for both optional and compulsory courses
//****************************************************************************

void crossOutSlots(double **dataArray, int slotIndex, int courseIndex){

   int inConflictingSlots=0;
   int k;

   for(int i = 0; i < ROWS; ++i){
      if(i ==slotIndex)
         continue;
      dataArray[i][courseIndex]=0;
   }


   for(int i = 0; i < ROWS; ++i)
      if(slotIndex == conflictingTimeslots[i]){
         inConflictingSlots=1;
         break;
      }

   if(inConflictingSlots){
   for(int i = 0; i < (sizeof conflictingTimeslots/sizeof(int)) ; ++i){
      k = conflictingTimeslots[i];
      for(int j = 0; j < COLS; ++j){
          if(j == courseIndex)
             continue;
         dataArray[k][j] = 0;
      }
   }
   }
   else {
   for(int j =0; j < COLS; ++j){
       if(j == courseIndex)
             continue;
      dataArray[slotIndex][j]=0;
   }
   }
}

//end
```