FALL 2019

MATH 412 – OPTIMIZATION

TERM PROJECT

Instructor Assignment Problem

Group members; Omran Al-Mofleh, 100045079

Begad ElHouty, 100045069

Ahmmad Saieed, 100045067

Submitted to Dr. Alip Mohammed Alifo

November 25th, 2019

Project Description

The Khalifa University Math department is looking to implement a program which will incorporate preference optimization in the assignment of instructors to courses. In the past, the assignment of professors was all done manually and with the continuous increase in the faculty number that process will be infeasible to do. Therefore, our project aims to build a program which would automate and facilitate the assignment of professors taking into account the preferences of professors, assignment constraints and other variables. This project is a type of an integer programming problem, more specifically an assignment problem, that can be optimized through many techniques and algorithms such as Branch & Bound, Tabu-search and more.

Optimization Model Setup

In order to setup an optimization model, we first list the main objective our project as well as constraints that we need to abide by. In addition, since we are taking the preferences of professors in mind, these preferences we will need to be taken into account in the model as parameters and not as constraints. Next, we formulate our model as an integer programming model in order to implement it in a program. This is how we set up our optimization model.

Main Objective:

Our main objective is to maximize instructors' happiness/fitness to their assigned courses.

Constraints:

a) Each course must be assigned to one instructor. A single course cannot be taught by 2 instructors. In certain circumstances, a course could be taught by more than 1 instructor, but ideally this does not happen.
b) Each instructor has minimum credit hours that they must teach in a semester.
c) Each instructor has maximum credit hours that they should not exceed in a semester.
d) Certain instructors whom are lecturers cannot be assigned a higher-level course.

Instructor Preferences:

a) Each instructor has courses they would prefer to teach each semester.
b) Each instructor prefers to teach on a certain campus. This is not important at the moment since all the Math faculty teach at the main campus. Nevertheless, we have taken into account this preference in our model.
c) Some instructors like to have more sections of the same course.
d) Professors whom have published in field has higher priority

Formulation

Decisions:

Let x_{ij} be $\begin{cases} 1\ if\ instructor\ j\ teaches\ course\ i \\ \quad 0\ otherwise \end{cases}$

Parameters:

Let c_{ij} be the happiness of instructor i with course j.

$$c_{ij} = \begin{cases} 100 & \text{if a course } \mathbf{j} \text{ is instructor } \mathbf{i}\text{'s 1st preference} \\ 90 & \text{if a course } \mathbf{j} \text{ is instructor } \mathbf{i}\text{'s 2nd preference} \\ 80 & \text{if a course } \mathbf{j} \text{ is instructor } \mathbf{i}\text{'s 3rd preference} \\ 0 & \text{Otherwise} \end{cases}$$

Let p_{ij} be whether an instructor i published in the field of course j.

$$p_{ij} = \begin{cases} 10 & \text{if instructor i published in the field of course j} \\ 0 & \text{Otherwise} \end{cases}$$

Let a_j be the course credit for course j.

Sets:

Let L be the set of all lecturers.

Let H be the set of all higher courses.

Let I be the set of all professors.

Let J be the set of all courses.

Then our linear integer programming problem is:

$$Max\ z = \sum_{j}^{n} \sum_{i}^{m} (c_{ij} + p_{ij}) x_{ij}$$

subject to

$$\sum_{i}^{m} x_{ij} = 1 \quad \forall j \in J$$

$$\sum_{j}^{n} a_j x_{ij} \geq 3 \quad \forall i \in I$$

$$\sum_{j}^{n} a_j x_{ij} \leq 7 \quad \forall i \in I$$

$$x_{ij} = 0 \ \ \forall i \in L, \forall j \in H$$

Formulation Discussion

- **Objective function**

We are looking to maximize the overall happiness of all instructors with their assigned course. Therefore, the objective function will sum up the happiness of instructors that are assigned a course. The p_{ij} coefficient is an added bonus to instructors whom published in the field of a course which explains the appearance of coefficient p_{ij}.

- **Constraints**

The first constraint is the assignment of a course to only one instructor. This is because if a course is only be taught by one instructor, then all instructors are not teaching the course implying that summing over the column (over instructors) of that course will be 1. It is exactly 1 since the course must be taught.

The second and third constraints are the minimum and maximum credit hour constraints respectively. They are quite similar to the first constraint but instead of summing over columns, we are summing over rows (over courses). We also multiply the credit hour parameter with the decision variable to get the sum of the credit hours an instructor will teach in a certain semester.

The last constraint is for lecturers that should not be assigned a course. It says that given a professor is a lecturer $\forall i \in L$ and a course is high-level $\forall j \in H$ then the decision variable is equal to zero i.e. the instructor i who is a lecturer will not teach course j.

- **Preferences**

The happiness coefficient c_{ij} is subjective to the purpose of the user and our main focus was to ensure that instructors get their preferred courses if possible. Each instructor was asked to give 3 course preferences and add additional comments if they had any more preferences. The values for c_{ij} were assigned as follows:

- If an instructor gets their first preference, they get a fitness of a 100.
- If they get their second preference, they get a fitness of 90.
- If they get their third preference or a course in the comments, they get a fitness of 80.
- Any other course that the instructor does not mention get a fitness of 0.

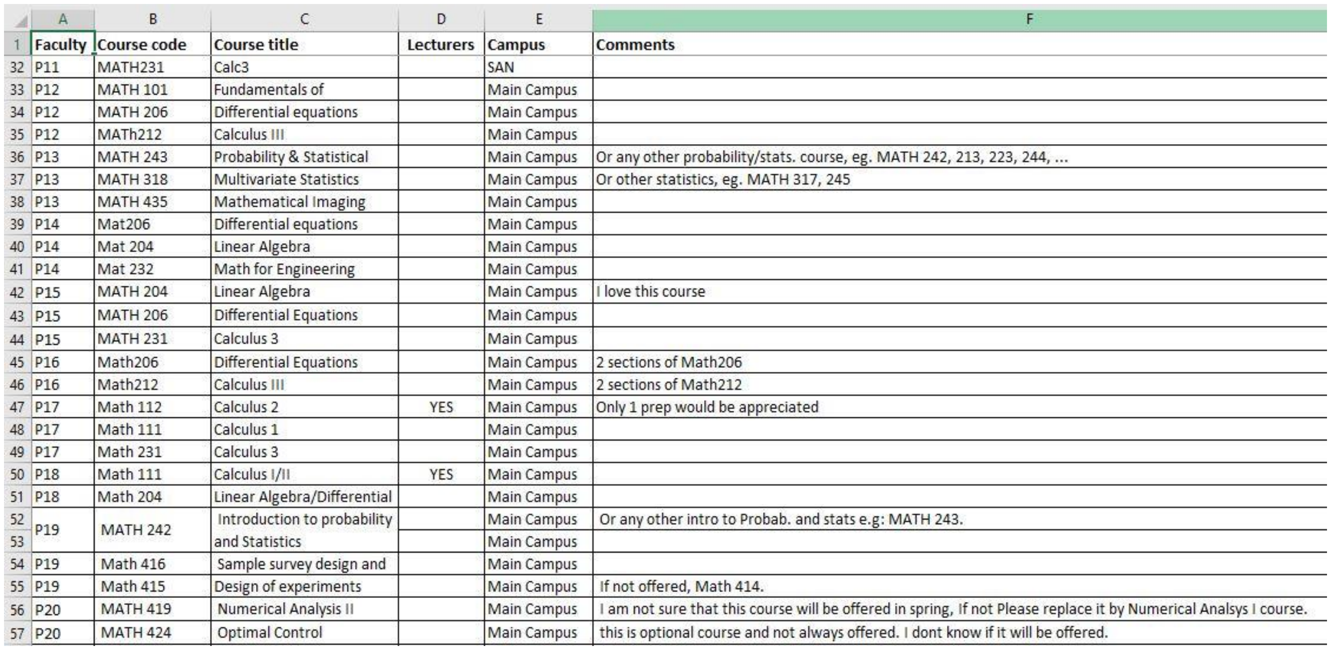
Moreover, if a course has sections in different campuses, they get different fitness depending on the instructor's preference of campus. However, sections of courses on the same campus get the same fitness from each instructor. Finally, if an instructor published in the field of a course, they get an extra 10 points added to their preference of that course.

Data Provided

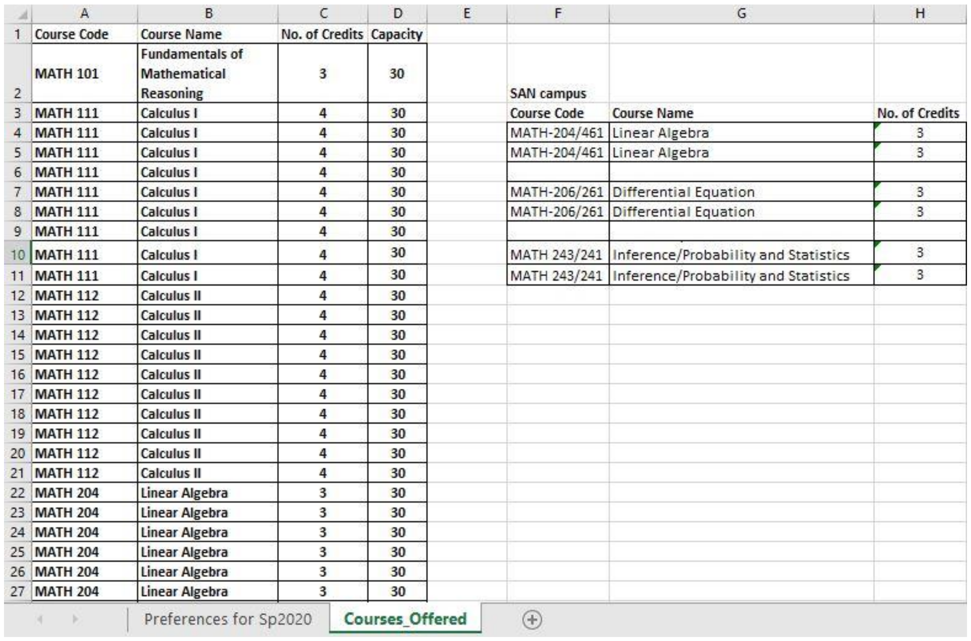
The data we worked with is provided by a faculty member and contains:

I. Set of teachers: For each teacher, you know whether he is lecturer or higher rank faculty
II. Set of courses to be offered (each course can have one or more sections)
III. Credit hours for each course
IV. Set of 3 course preferences for each teacher

And an assumption that must be satisfied is that all courses are covered by pre-optimization agreements, otherwise the problem can be infeasible. The following two figures show the original data table we received.



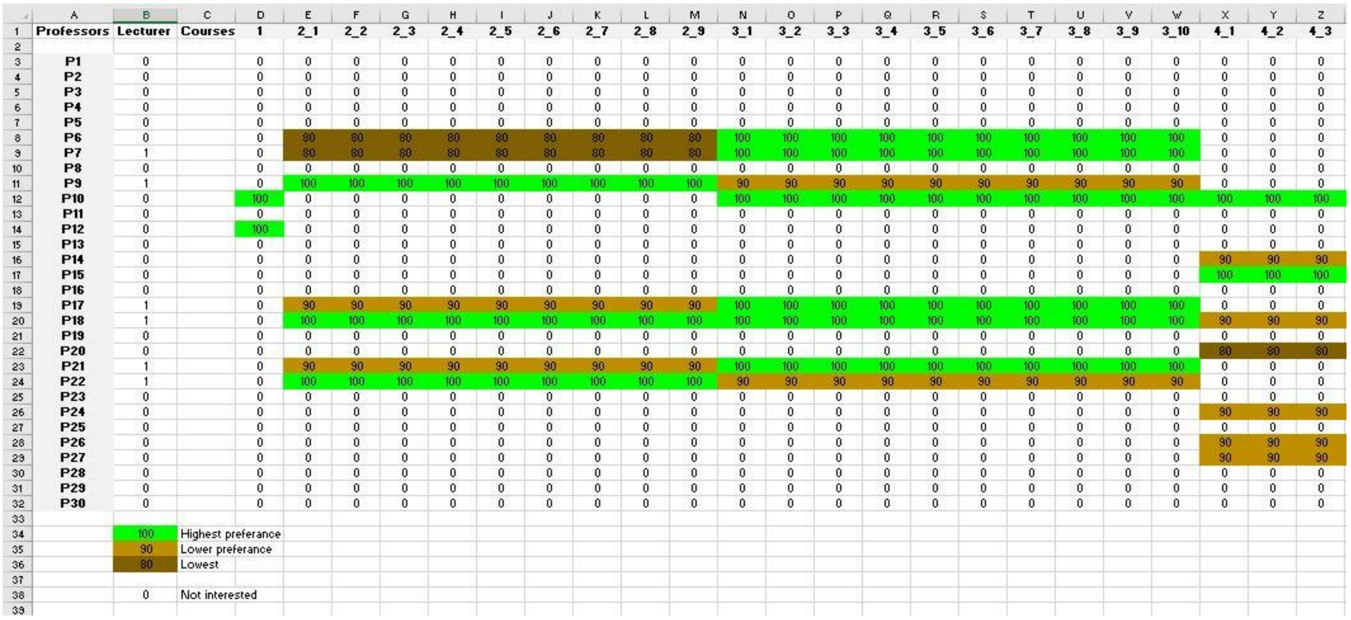
	A	B	C	D	E	F
1	Faculty	Course code	Course title	Lecturers	Campus	Comments
32	P11	MATH231	Calc3		SAN	
33	P12	MATH 101	Fundamentals of		Main Campus	
34	P12	MATH 206	Differential equations		Main Campus	
35	P12	MATh212	Calculus III		Main Campus	
36	P13	MATH 243	Probability & Statistical		Main Campus	Or any other probability/stats. course, eg. MATH 242, 213, 223, 244, ...
37	P13	MATH 318	Multivariate Statistics		Main Campus	Or other statistics, eg. MATH 317, 245
38	P13	MATH 435	Mathematical Imaging		Main Campus	
39	P14	Mat206	Differential equations		Main Campus	
40	P14	Mat 204	Linear Algebra		Main Campus	
41	P14	Mat 232	Math for Engineering		Main Campus	
42	P15	MATH 204	Linear Algebra		Main Campus	I love this course
43	P15	MATH 206	Differential Equations		Main Campus	
44	P15	MATH 231	Calculus 3		Main Campus	
45	P16	Math206	Differential Equations		Main Campus	2 sections of Math206
46	P16	Math212	Calculus III		Main Campus	2 sections of Math212
47	P17	Math 112	Calculus 2	YES	Main Campus	Only 1 prep would be appreciated
48	P17	Math 111	Calculus 1		Main Campus	
49	P17	Math 231	Calculus 3		Main Campus	
50	P18	Math 111	Calculus I/II	YES	Main Campus	
51	P18	Math 204	Linear Algebra/Differential		Main Campus	
52	P19	MATH 242	Introduction to probability		Main Campus	Or any other intro to Probab. and stats e.g: MATH 243.
53			and Statistics		Main Campus	
54	P19	Math 416	Sample survey design and		Main Campus	
55	P19	Math 415	Design of experiments		Main Campus	If not offered, Math 414.
56	P20	MATH 419	Numerical Analysis II		Main Campus	I am not sure that this course will be offered in spring, If not Please replace it by Numerical Analsys I course.
57	P20	MATH 424	Optimal Control		Main Campus	this is optional course and not always offered. I dont know if it will be offered.



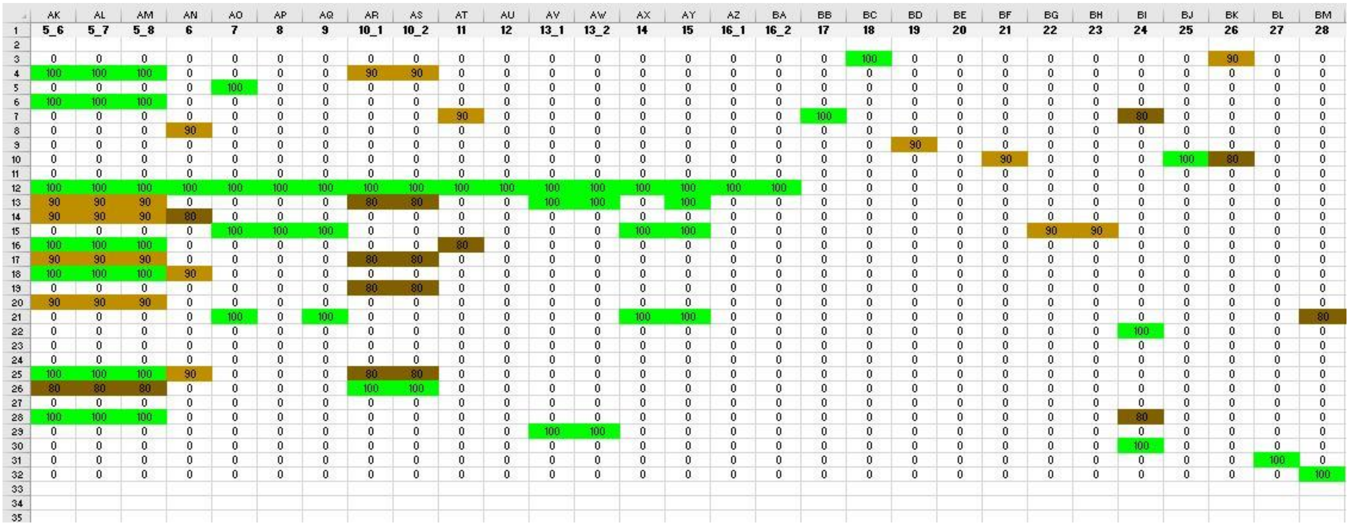
	A	B	C	D	E	F	G	H
1	Course Code	Course Name	No. of Credits	Capacity				
2	MATH 101	Fundamentals of Mathematical Reasoning	3	30		SAN campus		
3	MATH 111	Calculus I	4	30		Course Code	Course Name	No. of Credits
4	MATH 111	Calculus I	4	30		MATH-204/461	Linear Algebra	3
5	MATH 111	Calculus I	4	30		MATH-204/461	Linear Algebra	3
6	MATH 111	Calculus I	4	30				
7	MATH 111	Calculus I	4	30		MATH-206/261	Differential Equation	3
8	MATH 111	Calculus I	4	30		MATH-206/261	Differential Equation	3
9	MATH 111	Calculus I	4	30				
10	MATH 111	Calculus I	4	30		MATH 243/241	Inference/Probability and Statistics	3
11	MATH 111	Calculus I	4	30		MATH 243/241	Inference/Probability and Statistics	3
12	MATH 112	Calculus II	4	30				
13	MATH 112	Calculus II	4	30				
14	MATH 112	Calculus II	4	30				
15	MATH 112	Calculus II	4	30				
16	MATH 112	Calculus II	4	30				
17	MATH 112	Calculus II	4	30				
18	MATH 112	Calculus II	4	30				
19	MATH 112	Calculus II	4	30				
20	MATH 112	Calculus II	4	30				
21	MATH 112	Calculus II	4	30				
22	MATH 204	Linear Algebra	3	30				
23	MATH 204	Linear Algebra	3	30				
24	MATH 204	Linear Algebra	3	30				
25	MATH 204	Linear Algebra	3	30				
26	MATH 204	Linear Algebra	3	30				
27	MATH 204	Linear Algebra	3	30				

Improvements on the Data Provided

After looking at the data, we decided to create a whole new table that contains all the necessary information in a way that we could use it in mathematical models, i.e. programming algorithms. See an example of the first few classes in the following sample figures.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
1	Professors	Lecturer	Courses	1	2_1	2_2	2_3	2_4	2_5	2_6	2_7	2_8	2_9	3_1	3_2	3_3	3_4	3_5	3_6	3_7	3_8	3_9	3_10	4_1	4_2	4_3
2																										
3	P1	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	P2	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	P3	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	P4	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	P5	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	P6	0		0	80	80	80	80	80	80	80	80	80	100	100	100	100	100	100	100	100	100	100	0	0	0
9	P7	1		0	80	80	80	80	80	80	80	80	80	100	100	100	100	100	100	100	100	100	100	0	0	0
10	P8	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
11	P9	1		0	100	100	100	100	100	100	100	100	100	90	90	90	90	90	90	90	90	90	90	0	0	0
12	P10	0		100	0	0	0	0	0	0	0	0	0	100	100	100	100	100	100	100	100	100	100	100	100	100
13	P11	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	P12	0		100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	P13	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	P14	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	90	90
17	P15	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	100	100
18	P16	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	P17	1		0	90	90	90	90	90	90	90	90	90	100	100	100	100	100	100	100	100	100	100	0	0	0
20	P18	1		0	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	90	90	90
21	P19	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
22	P20	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	80	80
23	P21	1		0	90	90	90	90	90	90	90	90	90	100	100	100	100	100	100	100	100	100	100	0	0	0
24	P22	1		0	100	100	100	100	100	100	100	100	100	90	90	90	90	90	90	90	90	90	90	0	0	0
25	P23	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	P24	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	90	90
27	P25	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	P26	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	90	90
29	P27	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	90	90
30	P28	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
31	P29	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
32	P30	0		0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
33																										
34		100	Highest preferance																							
35		90	Lower preferance																							
36		80	Lowest																							
37																										
38		0	Not interested																							
39																										



	AK	AL	AM	AN	AO	AP	AQ	AR	AS	AT	AU	AV	AW	AX	AY	AZ	BA	BB	BC	BD	BE	BF	BG	BH	BI	BJ	BK	BL	BM
1	5_6	5_7	5_8	6	7	8	9	10_1	10_2	11	12	13_1	13_2	14	15	16_1	16_2	17	18	19	20	21	22	23	24	25	26	27	28
2																													
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0	0	0	0	90	0	0
4	100	100	100	0	0	0	0	90	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	100	0	0	0	0	0	0	80	0	0	0	0
8	0	0	0	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	90	0	0	0	100	80	0	0
11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
12	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0
13	90	90	90	0	0	0	0	80	80	0	0	100	100	0	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0
14	90	90	90	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	100	100	100	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	90	90	0	0	0	0	0
16	100	100	100	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	90	90	90	0	0	0	0	80	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
18	100	100	100	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
19	0	0	0	0	0	0	0	80	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
20	90	90	90	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	0	0	0	0	100	0	100	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	80
22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
24	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
25	100	100	100	90	0	0	0	80	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
26	80	80	80	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
28	100	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	80	0	0	0	0
29	0	0	0	0	0	0	0	0	0	0	0	100	100	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
30	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0	0	0	0
31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100	0
32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	100
33																													
34																													
35																													

As mentioned before, the numbers represent how happy each instructor is with the corresponding class. The column 'Lecturer' simply shows if an instructor can teach high-level courses or not, for instance, they can if the value is 1 and 0 otherwise. Also, courses and their sections are all considered as classes, so if an instructor is interested in teaching course i, all the sections of that course will have the same level of happiness according to the preference priority list of that instructor. Moreover, different campuses have different course codes and consequently we consider them as completely different classes.

Integer-Programming (IP) Approaches

Since our data table only deals with integer variables and coefficients, we decided to look at Integer-programming methods and find the appropriate algorithm to utilize to solve our problem. There are two main reasons for using integer variables when modeling problems as a linear program. Firstly, the integer variables represent quantities that can only be integer. For example, it is not possible to assign 1.5 instructors to a class. And secondly, the integer variables represent decisions (e.g. whether to include an edge in a graph) and so should only take on the value 0 or 1. These considerations occur frequently in practice and so integer linear programming can be used in many applications areas. And this why the first thing that came to mind having our problem is to use one of the many integer linear programming algorithms.

This problem is called the linear integer-programming problem. It is called a pure integer program when all decision variables must be integers. In general, variables will be fractional in the linear-programming solution, and further steps must be taken to determine the integer-programming solution [1]. In this report, we considered two integer-programming algorithms that we thought we could use for our problem, the first one is called Branch & Bound Algorithm, and the second is called Tabu-search Algorithm.

1. Branch and Bound Algorithm:

It is usually applied to those problems that have finite solutions, in which the solutions can be represented as a sequence of options. The first part of branch-and-bound, branching, requires several choices to be made so that the choices branch out into the solution space. In this method, the solution space is organized as a treelike structure. The two basic stages of a general Branch and Bound method are *Branching*; splitting the problem into subproblems, and *Bounding*: calculating lower and/or upper bounds for the objective function value of the subproblem [2].

In this method, the best IP-solution given by the solved subproblems is stored as an incumbent solution, a record holder. The incumbent objective value is an upper bound for the minimum value. A list of candidates subproblems is maintained and updated. A subproblem is fathomed (totally solved) and removed from the list, when it has an integer solution that is best so far and becomes the new incumbent solution, or, its optimum LP-solution objective is worse than the current incumbent value, or, the LP-problem is infeasible.

The steps of the branch and bound method for a maximization problem are very similar to the steps for the minimization problem. The only difference is that the relaxed solutions are rounded up, and upper and lower bounds are reserved. Now, since our objective is to maximize instructors' happiness, we are going to explain the steps for a maximization problem. The steps (with <= constraints) are described as the following:

1. Using the linear programming relaxation, we get the first optimal solution to the linear model while relaxing the restrictions on the integers.
2. Starting with node 1, the relaxed solution we get from step 1 is the new upper bound and the rounded-down integer solution will be the lower bound.
3. We select the variable with the biggest fractional part to branch from. then, we create two new constraints for this variable that reflect the partitioned integer values. We will create two separate nodes for each of the less then or bigger than constraints.
4. Solve the problem again with each of these two new constraints at each node.

5. The relaxed solution will be the upper bound at each node, and the existing maximum integer solution is the new lower bound.
6. We keep updating the lower and upper bounds if we produce better integer solutions. If we do not get better integer solutions, we branch from the node with the biggest upper bound.
7. Repeat from 3 until we reach to the only possible integer solutions from the nodes.

Generally, when the upper bound at a node is greater than or equal to the upper bound at any other ending node and this node has generated a feasible integer solution, then the optimal integer solution is achieved [3].

This method is used in the programming software AMPL as a built-in function, and we are going to explain the results we got from AMPL in the results section.

2. Tabu-Search Algorithm

Another algorithm that we took into consideration is the Tabu search. Tabu search is basically a heuristics method is functioned to find optimal solutions for various discrete problems, such as travelling salesperson problem (TSP), which aims for getting shortest way of going from the base town to the desired city and vice-versa. This method has a strategy to ignore prior solutions for a certain time [4].

The Tabu search algorithm should involves p finite iterations to find local best solutions, for i = 0,1,…., p, and it starts from choosing a nonempty set of x values, let us name it $X := \{x_0, x_1, \ldots, x_p\}$, and picking any x value in X, plus the initial tabu list L_i should be empty at the iteration i = 0, then we need to construct another set $N_i = S_i - L_i$, where $S_i := \{x_i - 4, \ldots, x_i - 1, x_i + 1, \ldots, x_i + 4\}$, if the local x_i in S is better than x_i in the iteration before, therefore the x value in the previous step should be in the Tabu list L_i for i = 1,…..,p [5]. We continue processing steps till the finite number of steps has been finished, and hence the local optimum solution is found [5]. The figure 1 shows this algorithm in an easy flow chart structure [6].

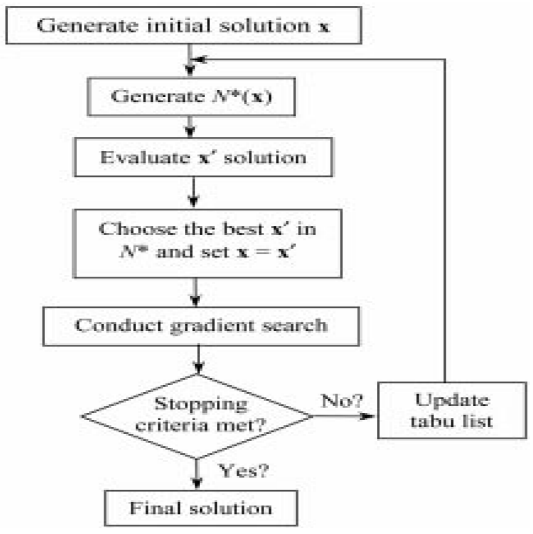


Figure 1. Tabu search algorithm

To relate to our problem, the Tabu search considers four things to find optimal number of professors that can teach a course or a course section: the amount of professors (divided to full-time and part-time groups), the amount of courses, the amount of sections, and the initial maximum number of instructors who are assigned to a section of a course [7]. Briefly speaking, there is a numerical example of that scheduling problem. Given a random set contains 20 and 5 of two categories of full-time professors respectively, in addition to 5 part-time instructors, plus it is assumed that there are 40 courses offered, and largest values of sections and assigned professors to a section are 5 and 2 respectively [7]. Using Tabu search, we have to put a certain amount of courses that could be illustrated by a professor, let us denote it C, in order to make feasible solutions by not having instructors who have to teach more than the certain number of courses that could be taught by one professor to avoid getting penalty values of not teaching overabundant courses are assigned for those full-time instructors who are able to teach more than C and therefore to get the suitable balance of load for each instructor [7].

It is kindly noted that due to the difficulty of using Tabu search for representing the problem with matrices in our data or the expected long time to solve or code the model using that algorithm, we have decided to consider the branch-and-bound algorithm, and we will see that in the solutions part below.

Solutions

In this project, we had used two programming software to solve our problem: AMPL and Gurobi solvers.

1. AMPL Solution

We implemented three separate codes in AMPL IDE software in order to generate the optimal solution for instructors scheduling problem.

First, we identified the corresponding parameters and sets, plus the objective functions and constraints as from project description above on prefdata.mod:

```
param m;
param n;
set I := {1..m};#set of teachers
set J := {1..n};#set of courses
set L := {7,9,17,18,21,22};#set of lecturers
set H := {48,49,50,51,52,53,54,55,56,57,58,59,60};#set of higher courses
param c{I,J}>=0;#coefficient 1 of objective function(happiness of professor i with some
course section j)
param p{I,J}>=0;#coefficient 2 of objective function(publication of professor i in the
field of course section j)
param A{J}>=0;#row vector of coefficients of some constraints below(number of credits for
each course section j)
var x{I,J}>=0, binary;#decision variables(it is 1 if prof. i teaches the course section j
or 0 o.w.)
maximize z: sum{j in J,i in I} (c[i,j]+p[i,j])*x[i,j];
subject to
c1 {j in J}: sum {i in I} x[i,j] = 1;
c2 {i in I}: sum {j in J} A[j]*x[i,j] <= 7;
c3 {i in I}: sum {j in J} A[j]*x[i,j] >= 3;
c4 {i in L, j in H}: x[i,j] = 0;
```

Then, we wrote the corresponding values of each parameter from excel files mentioned before inside prefdata.dat (the values for each row in some parameters data are as from left to right, it has been divided to five sub-rows due to inability to transfer the AMPL data file properly to this report):

```
data;
param m:=30;
param n:=60;
param c:
param p:
param A:
```

Finally, we called the both files above in the file "prefdata.run":

```
reset;
```

```
model prefdata.mod;
data prefdata.dat;
expand z, c1, c2, c3, c4;
option solver cplex;
solve;
display x,z
```

Using the solution option "cplex" in the code before which indicates using branch-and-bound method and linear programming simplex iterations, we obtained the following results for the optimal solutions and the maximum objective value of the professor's happiness:

```
CPLEX 12.9.0.0: optimal integer solution; objective 4390
344 MIP simplex iterations
0 branch-and-bound nodes
```

	Course 1	Course 2
P1	MATH112-07	MATH352-01
P2	MATH206-01	MATH206-08
P3	MATH204-01	MATH213-01
P4	MATH206-05	MATH206-07
P5	MATH232-01	MATH313-01
P6	MATH112-05	MATH315-01
P7	MATH112-06	MATH261-01
P8	MATH317-01	MATH352-01
P9	MATH111-08	MATH261-02
P10	MATH112-08	MATH214-01
P11	MATH241-01	MATH243-01
P12	MATH101-01	MATH112-03
P13	MATH242-01	MATH319-01
P14	MATH112-04	MATH204-05
P15	MATH204-03	MATH204-07
P16	MATH206-04	MATH212-01
P17	MATH112-09	MATH231-02
P18	MATH111-04	MATH204-04
P19	MATH223-01	MATH416-01
P20	MATH111-03	MATH112-09
P21	MATH112-02	MATH234-01
P22	MATH111-02	MATH206-06
P23	MATH112-01	MATH206-04
P24	MATH111-01	MATH231-01
P25	MATH426-01	MATH461-02
P26	MATH204-02	MATH206-03
P27	MATH241-02	MATH461-01
P28	MATH111-05	MATH320-01
P29	MATH111-06	MATH413-01
P30	MATH111-07	MATH414-01

Table 1: AMPL optimal solutions of $x_{i,j}$

```
CPLEX 12.9.0.0: optimal integer solution; objective 4390
344 MIP simplex iterations
0 branch-and-bound nodes
```

The transpose of the solution matrix ($x_{i,j}$) for all i = 1,.....,30 and j = 1,.....,60 indicates how the assignment of a professor i in a course or course section j would affect the binary entries of each $x_{i,j}$. For example, $x_{30,8}$ = 1 implies that the 30th instructor should teach the course (or section) 8. The objective value z = 4390 represents the total happiness of load had by all instructors in our model.
It has been noted that the optimal values of $x_{i,j}$ are obtained as a result of more than 340 steps generated using LP relaxation simplex method, and since $x_{i,j}$ is restricted between 0 and 1, hence if there exists an optimal value for xi,j in the i-th and j-th entry, then the solution should have the value 1. Also, there are no branch-and-bound nodes since the optimal solutions and the maximum objective value have become an integer solution from the root node after doing simplex iterations.

2. Gurobi Solution

The Gurobi code that we used for our optimization model is presented in the appendix.

The objective function value we got was 4390 which is what we expect since this is what the AMPL optimal solution had. The solution that we get should be in the form of a 30 by 60 matrix which is the x_{ij} values for the optimal solution. The table below presents the all the non-zero values of x_{ij} with the course names which signify whether instructor i is assigned course j.

	Course 1	Course 2
P1	MATH112-7	MATH352
P2	MATH206-1	MATH206-3
P3	MATH204-1	MATH213
P4	MATH206-4	MATH206-5
P5	MATH111-6	MATH232
P6	MATH112-4	MATH315
P7	MATH112-6	MATH261-2
P8	MATH316	MATH320
P9	MATH111-8	MATH234
P10	MATH112-1	MATH214
P11	MATH241-2	MATH243
P12	MATH101	MATH112-3
P13	MATH242	MATH317
P14	MATH111-1	MATH204-5
P15	MATH112-9	MATH204-7
P16	MATH206-6	MATH206-8
P17	MATH112-5	MATH231-2
P18	MATH111-4	MATH204-4
P19	MATH223	MATH416
P20	MATH111-2	MATH204-6
P21	MATH112-2	MATH204-2
P22	MATH111-5	MATH261-1
P23	MATH206-7	MATH212

P24	MATH204-3	MATH231-1
P25	MATH426	MATH461-2
P26	MATH206-2	MATH313
P27	MATH241-1	MATH461-1
P28	MATH112-8	MATH319
P29	MATH111-3	MATH413
P30	MATH111-7	MATH414

This table shows the optimal way in which to assign course to instructors in order for the overall happiness of the department to be the maximum.

There are things to note with the solution to the instructor assignment problem. First, since sections of the same course are given the same weight/happiness, thus the optimal solution is not unique yet we still should get the same optimal value for the objective function. Furthermore, an infeasible solution may occur if the number of courses is double the number of instructors (violating constraint 3) and the number of courses is less than the number of instructors (violating constraint 2). An infeasible solution may arise as well if there are not enough professors to teach higher level courses (violating constraint 4).

Conclusion

The instructor assignment problem can be formulated into an integer linear programming model and solve using the Branch & Bound method as well as the Tabu-search algorithm. Gurobi and AMPL were used to optimize the assignment problem and finish the process of assigning instructors in record time. Our work can be added on to improve the optimization model so that all instructors can get their preferred courses. Further work that can be done would be incorporating a schedule into the instructor assignment problem which will be able to provide users with the schedules of professors in a department automatically taking into consideration professors' preference.

Appendix

Gurobi code:

```
# Importing needed libraries

from gurobipy import *
import pandas
import numpy as np

# Build the model
m = Model("OptimizationProject")
```

```
# Fitness/Happiness coeffecients c_ij

fitness = pandas.read_excel('Coef.xlsx')
fitness = np.array(fitness)
```

```
# Publication coeffecients p_ij

pub = pandas.read_excel('p.xlsx')
pub = np.array(pub)
```

```
# Set of all Professors & if lecturer or not

profdf = pandas.read_excel('profdata.xlsx')
profdf = np.array(profdf)
I = profdf[:,0]
Lect = profdf[:,1]
```

```
# Set of courses and if course is high/low.
# Credit parameter a_ij

coursedf = pandas.read_excel('CoursesData1.xlsx')
coursedf = np.array(coursedf)
J = coursedf[0,:]
Level = coursedf[1,:]
credit = coursedf[2,:]
```

```
# numP -> no. of professors, numC -> no. of courses
# H -> set of all high level courses, L -> set of all low level courses

profs = range(len(I))
courses = range(len(J))
numP = len(I)
numC = len(J)

H = np.where(Level == 1)[0]
L = np.where(Lect == 1)[0]
```

```
# Maximizing fitness/happiness
m.ModelSense = GRB.MAXIMIZE
```

```
# Adding decision variables x_ij

X = []

for i in range(numP):
    X.append([])
    for j in range(numC):
        X[i].append(m.addVar(vtype=GRB.BINARY,name="X%d,%d"% (i, j)))
```

```
# Adding constraints

c1 = []
c2 = []
c3 = []
c4 = []

# Each course can only be taught by one professor

for j in range(numC):
    c1.append(m.addConstr(quicksum(X[i][j] for i in range(numP)) == 1 ,'c1%d' % j))

# Each professor can teach between 3 to 7 credits each semester

for i in range(numP):
    c2.append(m.addConstr(quicksum(credit[j]*X[i][j] for j in range(numC)) <= 7 ,'c2%d' % i))

for i in range(numP):
    c3.append(m.addConstr(quicksum(credit[j]*X[i][j] for j in range(numC)) >= 3 ,'c3%d' % i))

# A lecturer cannot teach a high level course

for i in L:
    for j in H:
        c3.append(m.addConstr(X[i][j]  == 0 ,'c4%d,%d' % (i,j)))
```

```
# Objective function

m.setObjective(quicksum(quicksum([X[i][j]*(fitness[i][j]+pub[i][j]) for j in range(numC)]) for i in range(numP)))
```

```
# Update variables & constraints

m.update()
```

```
# Model Optimization

m.optimize()
```

```
Optimize a model with 198 rows, 1800 columns and 5478 nonzeros
Variable types: 0 continuous, 1800 integer (1800 binary)
Coefficient statistics:
  Matrix range     [1e+00, 4e+00]
  Objective range  [1e+01, 1e+02]
  Bounds range     [1e+00, 1e+00]
  RHS range        [1e+00, 7e+00]
Found heuristic solution: objective 1410.0000000
Presolve removed 78 rows and 78 columns
Presolve time: 0.01s
Presolved: 120 rows, 1722 columns, 5166 nonzeros
Variable types: 0 continuous, 1722 integer (1722 binary)

Root relaxation: objective 5.235000e+03, 203 iterations, 0.00 seconds

    Nodes    |    Current Node    |     Objective Bounds      |     Work
 Expl Unexpl |  Obj  Depth IntInf | Incumbent    BestBd   Gap | It/Node Time

     0     0 5235.00000    0   38 1410.00000 5235.00000   271%     -    0s
H    0     0                    3950.0000000 5235.00000  32.5%     -    0s
H    0     0                    4000.0000000 5235.00000  30.9%     -    0s
     0     0 4468.33333    0   20 4000.00000 4468.33333  11.7%     -    0s
H    0     0                    4130.0000000 4468.33333  8.19%     -    0s
H    0     0                    4240.0000000 4468.33333  5.39%     -    0s
H    0     0                    4250.0000000 4468.33333  5.14%     -    0s
*    0     0               0    4390.0000000 4390.00000  0.00%     -    0s

Cutting planes:
  Cover: 24
  MIR: 2

Explored 1 nodes (438 simplex iterations) in 0.11 seconds
Thread count was 4 (of 4 available processors)

Solution count 7: 4390 4250 4240 ... 1410

Optimal solution found (tolerance 1.00e-04)
Best objective 4.390000000000e+03, best bound 4.390000000000e+03, gap 0.0000%
```

```
# Prints all non-zero X_ij entries

m.printAttr('X')
```

```python
import xlwt

workbook = xlwt.Workbook()

sheet = workbook.add_sheet("Sheet Name")

# Specifying style
style = xlwt.easyxf('pattern: pattern solid, fore_colour light_blue;'
                              'font: colour white, bold True;')

# Specifying column
for i in range(numP):
    for j in range(numC):
        if(X[i][j].X == 1):
            sheet.write(i+1, j+1, X[i][j].X, style)
        else:
            sheet.write(i+1, j+1, X[i][j].X)

for i in range(numP):
    sheet.write(i+1, 0, 'P%d' % (i+1))

for j in range(numC):
    sheet.write(0, j+1, 'C%d' % (j+1))

workbook.save("sample.xls")
```

```python
Course_names = pandas.read_excel('Coursenames.xlsx')
Course_names = np.array(Course_names)
```

```python
import xlwt

workbook = xlwt.Workbook()

sheet = workbook.add_sheet("Sheet Name1")

# Specifying style
style = xlwt.easyxf('pattern: pattern solid, fore_colour light_blue;'
                              'font: colour white, bold True;')

# Specifying column

for i in range(numP):
    sheet.write(i, 0, 'P%d' % (i+1))

count = 0
for i in range(numP):
    count = 0
    for j in range(numC):
        if(X[i][j].X == 1):
            sheet.write(i, count + 1, Course_names[j,0])
            count = count + 1

workbook.save("sample.xls")
```

References;

[1] Bradley, Hax, and Magnanti. *Applied Mathematical Programming*, 1st ed.

Reading, MA: Addison Wesley, 1977. [E-book] Available: mit.edu

[2] Huang. Chung-Yang, Cheng. Kwang-Ting, "Branch-and-Bound Algorithm Design" in Electronic Design Automation, 2009. Available: Science Direct Online.

https://www.sciencedirect.com/topics/computer-science/branch-and-bound-algorithm-design

[3] Taylor. B, *Introduction to Management Science,* 10th ed.

Virginia Polytechnic Institute and State University. [E-book] Available: academia.edu

[4] An, Heungjo (2019). 05.Heuristics_updated [Powerpoint slides]. Retrieved from https://elearn.ku.ac.ae/course/view.php?id=16554&section=1

[5] Glover, F. (1989). Tabu Search-Part I. ORSA Journal on Computing, 1(3), 190-206. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.302.4060&rep=rep1&type=pdf

[6] Smaili, A., Atallah, N. & Zeineddine, F. (2005, January). OptimaLink: A MATLAB-Based Code for Teaching/Learning Precision-Point and Optimum Synthesis and Simulation of Mechanisms. International Journal of Engineering Education. 21(5), 874-884. Retrieved from https://www.researchgate.net/publication/228875898_OptimaLink_A_MATLAB-Based_Code_for_TeachingLearning_Precision-Point_and_Optimum_Synthesis_and_Simulation_of_Mechanisms

[7] Gunawan, A. & Ng, K. M. (2011, March). Solving the Teacher Assignment Problem by Two Metaheuristics. International Journal of Information and Management Sciences, 22(1), 73-86. Retrieved from https://www.researchgate.net/publication/228579574_Solving_the_Teacher_Assignment_Problem_by_Two_Metaheuristics