IME639A : ANALYTICS IN TRANSPORT AND TELECOM

University Course Timetabling Problem

Instructor: Dr. Faiz Hamid

GROUP 7:

Acanksha Jain (190045)

Chinmay Joshi (200299)

Gulshan Kumar Choudhary (190348)

Sonali Gupta (190849)

Introduction:

The University Course Timetabling Problem is the subset of the more general Scheduling problems that deals with assigning events to specific discrete time slots. It is an NP-Hard problem with a large roster of continuously growing applications.

The goal of the university course timetabling problem (UCTTP) is to find a method to allocate events to fixed, predefined time slots and rooms, such that all the constraints within the problem are satisfied.

Events include students, teachers and courses. Also, time slots include two components: daily and weekly time slots which vary from one institution to another.

Problem Definition:

In UCTTP number of classes (C), teachers (T), venues (V), periods per day (P) and days in a week (D) are given. Classes and teachers must be assigned such that there are no clashes.

In this problem, the number of times a class has to be held by a particular teacher in the given venue is given in the form of a matrix. Number of working days and number of periods are specified using which total number of timeslots can be calculated. The problem has been designed to be totally constrained i.e each class, teacher, and venue is required for each period. The objective for each of these problems is to allot class, teacher and venue for each of the time periods and ensure zero clashes among them.

Dataset Description:

The information about the number of teachers (T), classes (C), periods per day (P), working days in week (W) and venues (V) are given in the dataset. This problem has been designed to be totally constrained. That is, each class, teacher , and venue is required for each period. The optimal objective function for this problem is zero clashes by allotting class, venue and teacher to all the time periods. We can calculate the total total number of timeslots by using the data of number of periods and number of working days. A matrix is used to represent how many times a certain teacher has to teach a class at a certain venue. The first V rows of this matrix indicate the number of times each class-teacher combination is to meet each other in venue 1 across the P periods.

The next V rows indicate the number of times each class-teacher combination is to meet each other in venue 2 across the P periods, and so on.

Data set 1:

NUMBER OF TEACHERS: 5

NUMBER OF PERIODS PER DAY: 6

NUMBER OF WORKING DAYS PER WEEK: 5

NUMBER OF SLOTS PER WEEK = 6 x 5 = 30

NUMBER OF CLASSES: 5

NUMBER OF ROOM AVAILABLE: 5

NUMBER OF REQUIREMENTS: 150

Matrix grouping rows according to venue:

Teacher	1	2	3	4	5	
	1	0	0	1	2	
	2	0	1	3	1	
	0	0	1	2	1	
	1	2	3	1	0	
	0	2	1	4	1	Venue 1
	4	0	3	1	3	
	0	1	0	1	3	
	0	1	2	1	1	
	1	0	3	0	0	
	2	1	0	2	0	Venue 2
	2	2	0	0	2	
	3	2	1	2	1	
	1	0	3	1	1	
	3	0	0	0	3	
	1	0	0	1	1	Venue 3
	1	2	0	0	1	
	0	2	3	1	0	
	2	0	2	0	4	
	1	3	0	2	1	
	1	1	0	2	1	Venue 4

1	1	1	1	1	
0	3	0	0	0	
1	2	3	1	0	
1	2	1	1	1	
1	3	2	2	1	Venue 5

Here the first 5 rows of the matrix indicate the number of times each class-teacher combination is to meet each other in venue 1 across the P periods. For example, the 3rd row contains a “1” in the last column. This means that class 3 must meet teacher 5 in room 1 one time.

Problem Formulation:

No. of Classes = C

No. of Teachers = T

No. of Venues = R

No. of periods per day = P

No. of days per week = D

No. of slots per week = K = P * D

Decision Variables :

$xijkl$ = 1, if the ith class meets jth teacher in the kth slot and in the lth room.

= 0 otherwise

Objective Function: The objective function for the problem is to minimize the number of clashes. Since the problem is to find a feasible solution we may use other objective functions as well. One such function can be to minimize the summation of all decision variables. This ensures that our IP solver is not biased towards any decision variable. But the best objective function would be just a constant value as it won’t take computational time to calculate the objective function value. If CPLEX is able to solve the IP, we get a feasible solution.

Minimize $\sum i \sum j \sum k \sum l \; xijk$

Constraints:

1. Requirement Constraint $\sum k \; xijkl = M[C * l + i]\,[\,j\,]$ $\quad \forall \; i,j,\, l$ Where M = Matrix

2. Constraint for no clashes for classes $\sum j \sum l \; xijkl \leq 1 \quad \forall k,\, i$

3. Constraint for no clashes for teachers $\sum_i \sum_l x_{ijkl} \leq 1 \quad \forall k, j$

4. Constraint for no clashes for rooms $\sum_i \sum_j x_{ijkl} \leq 1 \quad \forall k, l$

Heuristic approach:

<u>Evolutionary and genetic algorithms</u>

Inspired by natural evolutionary mechanisms, evolutionary algorithms (EAs) are built on computing models. The three steps of an EA are selection, regeneration, and replacement. EAs operate on a population of potential solutions. To be parents for the upcoming generation, people with excellent levels of fitness are chosen during the selection phase. The parents who were chosen in the first phase are subjected to two crossover and mutation operations during the regeneration phase, and during the replacement phase, the original (starting) population's inhabitants are replaced by the newly produced inhabitants.

The following steps constitute the foundation of genetic algorithms, which are a subset of EAs: The first step is to create the initial population. The second is to evaluate the generated population using the evaluation function. The third step is to choose some individuals as parents to crossover based on the information obtained from the evaluation functions. The fourth step is to apply the crossover operator to produce children. The fifth step is to apply the mutation operator to children. The sixth step is to choose parents and children to form the new population for the following generation. The seventh step is to determine whether the termination condition is satisfied.

<u>Ant colony optimization algorithm:</u>

This approach was developed to establish a path between a formicary and food after studying the behavior of ants. Ants always move in a haphazard manner in search of food before leaving a scent trail. When other ants discover this path, they take it, follow it, and if they succeed in reaching the food, they return to their nest and lay a new track next to the earlier one. The random search is restricted to that food since the pheromone slowly evaporates where results along the path would make it less alluring to the next ant. This strategy aims to have artificial ants travel in order to determine the shortest path.

Socha et al.'s (2002) use of the Max-Min ant system to schedule university courses has resulted in the construction of an ideal path where each path may produce a helpful graph to assign courses to timeslots affected by the amount of pheromone within a range. The entire distribution of courses to timeslots is carried out by a large number of ants using a predetermined list. Ants use heuristic information and data from an indirect coordinator mechanism among agents and activities in an environment to choose the probabilities of time slots for allocating courses.

Memetic algorithm:

When people's views change, a meme is created. We might think of a meme as a unit of information that generates itself. A meme differs from a gene in that when it is shared among individuals, each person alters the meme if they think it is useful, whereas a gene is passed along unchanged. One of the best features of the memetic algorithm is that the range of potential solutions is constrained to local subspace optimization. According to Obit (2010), the memetic algorithm combines the genetic and hill climbing algorithms.

Jat and Shengxiang (2008) used the memetic algorithm to combine the local search approach with genetic algorithms to solve the UCTTP problem. A local search was conducted over the events, and another was conducted over the timeslots.

Results:

We formulated the problem and implemented it using C++ in Visual Studio 2022 and solved it using CPLEX. To store the decision variables, we created a 4 dimensional matrix. Below is a chart that shows how long it took the programme to find a feasible way to arrange teachers and classes.

NO. OF TEACHERS	NO. OF CLASSES	NO. OF ROOM AVAILABLE	TIME TAKEN (sec)
4	4	4	0.24
5	5	5	0.51
6	6	6	0.89
7	7	7	29.01

It is clear from this table that our algorithm will take hours to solve this problem when it has a large number (>90) of classes, teachers and rooms. It is not very efficient as it takes around half a minute to solve the problem having a size of 7 classes, 7 rooms and 7 teachers.

Conclusion:

The problem was formulated as Integer Programming. The exact method was used to solve the problem of how to schedule classes at a university as it was less complicated to code. For a problem with 7 classes, 7 teachers, and 7 rooms, it took 29.01 seconds to figure out how the classes and teachers could be spread out over a week. Since there were fewer groups and teachers in the dataset, we were able to solve it using the exact method. With bigger numbers, the IP method will take a lot more time. So, to solve real-life university schedule problems, we need a more complex algorithm.

References:

1. https://www.sciencedirect.com/science/article/pii/S0360835214003714
2. https://www.itc2019.org/home
3. http://people.brunel.ac.uk/~mastjjb/jeb/orlib/tableinfo.html