ILP - Class Notes

Wilmer Uruchi Ticona

November 2018

1 Knapsack Problem

Given a set of N items (of different types), each with a weight w_i and a value v_i $(1 \leq i \leq N)$, we want to determine how many items of each type to choose so that the value is maximized but we do not want to exceed a maximum weight W.

The objective is to maximize the number of items that we can carry: x_i $(1 \leq i \leq N)$ = number of items of type i, $\forall\ x_i \in \mathbb{N}$

Objective function:

Max $\sum_{i=1}^{N} v_i \cdot x_i$

Where we are trying to maximize the number of items of each type x_i that we can carry with value v_i (such that: $x_i \cdot v_i$ = total value for type i) while trying to keep sum w_i of each x_i less or equal to a max value weight W. To do this we have to add a constraint:

$\sum_{i=1}^{N} w_i \cdot x_i \leq W$

So, by applying this constraint, the sum of items x_i multiplied by their corresponding weights w_i should not be more than the max weight W. This constraint will be applied to the objective function to find an optimal solution.

1.1 The Boolean Approach

Define a decision variable $taken_i \in \mathbb{B}$ $(1 \leq i \leq N)$ that indicates if an items is included in the bag or not (1 = included). We also have w_i $(1 \leq i \leq N)$ that is the weight of a certain item, and v_i $(1 \leq i \leq N)$ that is the value that this items provides.

Objective function:

Max $= \sum_{i=1}^{N} taken_i \cdot v_i$

Constraint:

$\sum_{i=1}^{N} taken_i \cdot w_i \leq W$

The constraint makes sure that the total weight of the selected item (left hand of the equation), is less or equal to the total allows weight W.

2 Set Covering

We have a set of databases $d_1, ..., d_n$ where database d_i has size s_i. We receive m queries $q_i, ..., q_m$ and m subsets of databases $Q_1, ..., Q_m$ such that each query q_i can be answered by inspecting any of the databases in Q_i. Build a linear program that allows one to determine which is the smallest (in size of the its components) subset of databases that suffices to answer all queries.

Lets put it into an example: Suppose we receive a query q_1 that needs data from d_2, d_3, d_4, then, it would be optimal if the subset of databases Q_1 included all these databases (d_2, d_3, d_4) so q_1 only needed to access Q_1 to be completely answered. In another situation, there could be some subsets Q_j that include one or more of these databases d_2, d_3, d_4, but also include other databases that would increase the total size of the answer, so the minimization has to look for the answer with the less quantity of extra databases.

Lets start by defining a decision variable:

$c_i \in \mathbb{B}$ = database d_i is chosen to answer some query q_i

Objective function:

Min $\sum_{1 \leq i \leq n} s_i \cdot c_i \qquad \forall\ 1 \leq j \leq m$ (Queries)

The objective function tells us that for all queries q received, we minimize the sum of the sizes s of the chosen databases $c_i \rightarrow d_i$.

Then we have the constraint:

$\sum_{i=1:i\in Q_j}^{n} c_i \geq 1 \qquad \forall\ 1 \leq j \leq m$ (Subsets)

This constraint is making sure that at least one subset Q_j answers a query q_i using a database d_i. The relationship between the query and the database is given by the decision variable c_i.

3 Traveling Salesman Problem

A large number of holes are needed on printed Integrated Circuits boards for mounting chips and other hardware. Such holes are typically produced by automated drilling machines that move to drill holes between specified locations. Assume we are given a set of locations L where the holes need to be placed and that we know between which locations the drilling machine can move and the exact time of these moves (moving from A to B take the same as moving from B to A). Build a linear program that allows us to find the fastest way to produce all holes with the only requirement that the drilling machine should finish in the position of the first hole it has produced and should visit each location exactly once (except of course for the first hole).

A good start point is to visualize the holes as points with locations (x, y) that are all connected by edges that start at some point i and finish at the same point after connecting all the possible vertex by covering the less distance possible.

There are N holes (or cities).

Provided data: $d_{ij} \in \mathbb{Z}\ (1 \leq i, j \leq N)$ = distance form hole i to hole j.

Decision variable 1: $x_{ij} \in \mathbb{B}\ (1 \leq i, j \leq N) = 1$ iff we travel from hole i to hole j.

Then, we have the objective function:

Min

$$\sum_{i,j=1}^{N} d_{ij} \cdot x_{ij}$$

In this objective function we are trying to minimize the sum of the distance covered by traveling from point i to point j, if that trip is required. By only applying this function, the linear programming will most likely return a value 0, since there are no other constraints that tell it that some trips are mandatory. So we add some constraints:

$$\sum_{j=1}^{N} x_{ij} = 1 \quad \forall\ 1 \leq i \leq N$$

$$\sum_{j=1}^{N} x_{ji} = 1 \quad \forall\ 1 \leq i \leq N$$

These constraints say that each vertex must have at least one edge in (an incoming trip), and one edge out (an outgoing trip), which make sense since the printing machine will travel to one point to mount the chip or hardware, and then go to the next point from the previous point, thus every vertex must fulfill this conditions, even the starting vertex. Nevertheless, as we are trying to find a solution that starts at one point, covers all the points, and finishes at the starting point; these conditions could lead the program to find solutions that include more than one path, that is, instead of having one path that covers all the points, we may end up with two paths, each one covering half the pints. We must forbid that.

$Num_i \in \mathbb{N} \qquad \forall\ 2 \leq i \leq N$

We will give each point a sequence number starting at vertex 2 (because the number 1 is reserved for the starting vertex), assuring that the solution will start and end at the same vertex; thus, forbidding a solution with many paths. But this variable lacks the logic that makes it work.

If $x_{ij} = 1 \rightarrow Num_i < Num_j \qquad \forall\ 2 \leq i, j \leq N$

This says that if there is a path from point i to j, then Num_i must be before Num_j in the sequence. However, this is not linear programming. We must find a way to represent this in form of a valid linear constraint:

$Num_i < Num_j$
$Num_i - Num_j < 0$
$Num_i - Num_j < 0 + M$
$Num_i - Num_j < 0 + (N - 1)$

Where this inequality will be always satisfied if M is sufficiently large enough. Now, to find an efficient value for M, suppose that Num_i takes the highest value possible, that is:

N; and Num_j takes the least value possible: 2. Then, for this inequality to hold, we must have something greater than $N-2$ on the right hand side, that is $N-1$. However, don't see it as a constraint, this is just some logic. Now, to put it as a constraint:

Constraint: $Num_i - Num_j < (N-1)(1-x_{ij}) \qquad \forall\ 2 \leq i, j \leq N$

If there is a path from $i \to j$, then $x_{ij} = 1$, so the right hand side will be 0. Consequently, Num_i must be less than Num_j, thus ensuring the sequence.

If there is no path from $i \to j$, then $x_{ij} = 0$, so the left hand side will be $N-1$. Consequently, Num_i could be less or greater than Num_j, we don't really care since there is no need for a sequence when there is no path between vertex $i \to j$.

4 Block 4 - Exercise 6

Show how these situations can be modeled with a set of linear constraints:

(a) At least one of the two constraints a_1x_1 +... $a_nx_n \leq$ b and a'_1x_1 +... $a'_nx_n \leq b'$ need to be true.

(b) At least K out of the N constraints $a^j_1x_1$ +... $a^j_nx_n \leq b^j$ with $j = 1, ...N$ need to be true.

(c) The linear expression $a_1x_1 + \cdots + a_nx_n$ can only take a finite number of values $d_1, ..., d_k$.

4.1 a)

For:

a_1x_1 +... $a_nx_n \leq b + M \cdot y$
a'_1x_1 +... $a'_nx_n \leq b' + M' \cdot (1-y)$

$y \in \mathbb{B}$

If $y = 1 \to$ 2nd constraint is to be satisfied.
If $y = 0 \to$ 2nd constraint is not satisfied.
Therefore, at least one constraint is true, but not both of them necessarily.

4.2 b)

For:

$$a_1^1 x_1 + a_2^1 x_2 + ... + a_n^1 x_n \leq b_1 + B_1 \cdot y_1$$
$$a_1^2 x_1 + a_2^2 x_2 + ... + a_n^2 x_n \leq b_2 + B_1 \cdot y_2$$
$$\vdots$$
$$a_1^m x_1 + a_2^m x_2 + ... + a_n^m x_n \leq b_m + B_m \cdot y_m$$

$$y_1 + y_2 + ... + y_m \leq m - K \qquad y_i \in \mathbb{B} \qquad \forall 1 \leq y \leq m$$

A constraint is active when $y_i = 0$. Therefore, in the case that 3 out of 10 constraints must be active, the sum of all y_i should include at least 3 y_i values equal to 0, so the sum must be less or equal to 7. We can encode 7 as: $m - K = 10 - 3$.

4.3 c)

$$a_1 x_1 + a_2 x_2 + \cdots + a_n x_n = y_1 d_1 + y_2 d_2 + \cdots + y_k d_k$$

Variables y_k condition values d_k, so if we want that only one value d_k is active, we can condition it by stating:

$$y_1 + y_2 + \cdots + y_k = 1$$

5 Integer Linear Programming vs Metaheuristics

A consulting company receives lots (assume it is an infinite number) of combinatorial optimization problems from two different costumers. So far, it is solving them using naive algorithms but it knows that both of them can be solved much better using linear programming and metaheuristics. Hence, it has contacted experts on these technologies. Hiring LP experts costs 45000/month, whereas MH experts 76000/month.

In order to decide which group to hire, it has sent them typical problems from companies A and B. This has allowed to estimate how many problems can be processed in one hour by each group. Moreover, after evaluating the quality of the solutions an estimation of which would be the profit to solve one problem with a particular group has been produced. Data are in the following table:

	A-Problems		B-Problems	
	Problems/h	Profit	Problems/h	Profit
LP Experts	52	12	38	17
MH Experts	42	16	23	10

In order to satisfy both companies, it wants to process at most twice the number of problems for one company than for another. Considering that the LP group can only offer 620 hours per month and the MH only 710, write an integer linear program to find out which group to hire, and how many problems of each type should be solved to maximize the benefit.

6 Research Department

Several research groups of the same department, all of which collaborate with private companies, have access to an external central computer. Each research group always sends jobs with the same processing time and knows which amount of money it would obtain from private companies as a result of executing one such job. We also know the number of jobs each group needs to execute during one month (720h of computer time), as shown in the table:

Group	Profit €	Processing Time (hours)	Number of jobs
1	100	1.0	400
2	340	5	75
3	350	3.7	30
4	190	2.4	15
5	550	4.5	40
6	100	0.7	20
7	1150	9.5	75

There are additional constraints regarding two research groups:

- As a consequence of previous misuse of the computer, a fixed amount of 2000€ has to be paid by the department if the 7th group uses the computer.
- The 2nd research group would like to process 50 additional jobs with processing time 0.5, but only if all the 75 with processing time 5.0 have been processed. These extra 50 jobs also give 340€ per job.

Knowing that the department gets 10% of the profit of all research groups, formulate a linear program to help the department head to decide how many jobs of each research group will be executed every month so as to maximize profit.

$x_i \in \mathbb{N}$ = number of jobs from group i executed. $(1 \leq i \leq 7)$
$\widetilde{x}_2 \in \mathbb{N}$ = number of extra jobs from group 2.
$b_7 \in \mathbb{B}$ = true (1) iff 7^{th} group executes some job.

$y \in \mathbb{B}$ = conditional auxiliary.

$$Max\ (100x_1 + 340(x_2 + \widetilde{x}_2) + 350x_3 + 190x_4 + 550x_5 + 100x_6 + 1150x_7)(0.1) - 2000b_7$$

Constraint: $1x_1 + 5x_2 + 0.5\widetilde{x}_2 + 3.7x_3 + 2.4x_4 + 4.5x_5 + 0.7x_6 + 9.5x_7 \leq 720$
Constraints: $x_1 \leq 400$, $x_2 \leq 75$, $\widetilde{x}_2 \leq 50$, $x_3 \leq 30$, $x_4 \leq 15$, $x_5 \leq 40$, $x_6 \leq 20$, $x_7 \leq 75$

Implication:

$$b_7 = 0 \Longleftrightarrow x_7 = 0$$

$$b_7 \leq 0 \Longleftrightarrow x_7$$

So we have two implications:

$$b_7 \leq 0 \Longrightarrow x_7 \leq 0$$

$$x_7 \leq 0 \Longrightarrow b_7 \leq 0$$

Meaning that if the group 7 does not execute any job, then b_7 is false (0). To encode these implications into constraints:

$$x_7 \leq 75b_7$$

$$b_7 \leq x_7$$

We have the case of the second group:

$$\widetilde{x}_2 > 0 \Longrightarrow x_2 = 75$$

It says that if group 2 has extra jobs, then they have already done their 75 original jobs. To encode this, we start with:

$$\widetilde{x}_2 \geq 1 \Longrightarrow x_2 \geq 75$$

By negating both inequalities, we arrive at a conditional constraint:

$$\widetilde{x}_2 < 1 \quad OR \quad -x_2 \leq -75$$

Only one of these constraints must be satisfied, so we apply a conditional:

$$\widetilde{x}_2 < M \cdot y$$

$$-x_2 \leq -75 + M' \cdot (1 - y)$$

An optimal value for M is 50, since that is the maximum number of extra jobs, and an optimal value for M' is 75, since that is the maximum number of original jobs from group 2. Then, we have:

$$\widetilde{x}_2 < 50 \cdot y$$
$$-x_2 \leq -75 + 75(1 - y)$$

This conditional constraint can also be expressed as:

$$\widetilde{x}_2 < 50 \cdot y$$
$$x_2 \geq 75 \cdot y$$

If $y = 1$, all the 75 original jobs have been completed, and no more than 50 extra jobs have been completed.

If $y = 0$, 0 or more original jobs could have been completed, but 0 extra jobs have been completed.

7 Tardor 2012

A large company that produces huge amounts of data does not have the necessary resources to store monthly backups and decides to hire external backup providers. The company has O different offices and we know, for each office o, which is the amount of data d_o, in PetaBytes (10^6 GB), that it needs to store.

After studying several possibilities, a backup provider is chosen that has C different backup centers. Using a backup center c has a fixed cost of $fixed_c$ thousands of euros and, for each PB it stores, an additional $storage_c$ thousands of euros will be paid. Each backup center c can store at most $capacity_c$ PBs of information.

Note that each office can use different backup centers, and that each backup center can store data for different offices. We assume that all d_o's, $fixed_c$'s, $capacity_c$'s and $storage_c$'s are integer values.

The goal is to find out which backup centers will be used and how many PBs of each office every backup center will store so that the total cost is minimized. An additional requirement we impose is that all offices can only store integer values of PBs in each backup center.

(a) Write down an integer linear program that solves the previous problem. Hint: use variables $x_{c,o}$ to represent the number of PBs that office o stores in backup center c, and variables $used_c$ to represent whether backup center c is used to store some PB.

(b) Create an OPL project to solve this problem. Concrete data will be found in the file exam.dat. Use the file initialModel.mod as the initial model and complete it to solve

the problem. Save this file as modelB.mod and send it through "El Racó". Which is the optimal value of the problem?

(c) Assume that the backup provider has one additional requirement: if the i-th backup center does not store any PB ($with 1 \leq i \leq C$), neither can the j-th backup center for all $i < j \leq C$. How would you extend your model to take into account this extra requirement? Write down the mathematical formulation and extend the OPL model. Save this file as modelC.mod and send it through "El Racó". Which is now the optimal value?

(d) Assume now that the backup provider adds yet another constraint: each backup center can store data for at most two different offices. How would you extend your model to take into account this extra requirement? Write down the mathematical formulation and extend the OPL model. Save this file as **modelD.mod** and send it through "El Racó". Which is now the optimal value? Hint: it may be necessary to add additional variables.

7.1 (a)

What we have:

C = set of backup centers
c = each center
$capacity_c$ = each center capacity
O = set of offices
o = each office
d_o = data office o needs to store
$fixed_c$ = cost of using a center (only once)
$storage_c$ = cost of each PB (from d_o)

Decision variables:

$x_{c,o} \in \mathbb{N}$ = number of PBs office o sends to center c
$used_c \in \mathbb{B}$ = true iff center c stores some PBs

Objective function:

$$Min \sum_{c \in C} fixed_c \cdot used_c + \sum_{c \in C} (\sum_{o \in O} x_{c,o}) \cdot storage_c$$

We are minimizing the fixed cost of using a certain amount of backup centers, plus the cost per PBs send to each center.

Constraints:

$$\sum_{o \in O} x_{c,o} \leq capacity_c \cdot used_c \quad \forall c \in C$$

For all centers c, the sum of all the PBs send by the offices o to that center, should not exceed the capacity of that center if that center is being used.

$$\sum_{c \in C} x_{c,o} = d_o \quad \forall o \in O$$

For all offices o, the sum of all the data send to the servers c should be equal to the total data the office needs to store d_o.

7.2 (c)

We have the following condition:

$$not(used_c) \Longrightarrow not(used_{c+1}$$

That's the same as:

$$used_c \leq 0 \Longrightarrow used_{c+1} \leq 0$$

One way to encode this into a constraint is:

$$used_{c+1} \leq used_c \quad \forall c \in C$$

We are stating that the next center should always have a used value less or equal than the previous, for example, if $used_c$ is 1 meaning that the center is in use, then only in that situation the value of $used_{c+1}$ can assume value 1 meaning that the center is in use. To implement this code into CPLEX, we have to be careful to set the upper limit of our centers iterator to the number of center minus 1, otherwise we will have an exception.

7.3 (d)

Add a new decision variable:

$sends_{c,o} \in \mathbb{B}$ = true iff office o sends data to center c.

Constraint:

$$\sum_{o \in O} sends_{c,o} \leq 2 \quad \forall c \in C$$

No more than 2 offices can use 1 center. However, until this point $sends_{c,o}$ can be 1 or 0, we have to condition the value of this variable. We have that:

$$x_{c,o} \Longrightarrow sends_{c,o} \geq 1$$

If some office o stores data in a center c, this implies that $sends_{c,o}$ should be greater or equal to 1. Since it is a boolean, it will be 1. As a constraint:

$$M \cdot sends_{c,o} \geq x_{c,o} \quad \forall o \in O, \forall c \in C$$

Where M could be the maximum number of PBs an office needs to store. This value can be calculated in pre processing.

8 Tardor 2013

A set of users U need to be connected to the Internet. For that purpose, we have available a set of access points A where we could install several routers (at most one per access point). We want to find out in which access points we should install a router and to which access point each user should be connected to, in order to minimize cost and satisfy other constraints that are defined in the following.

(a) Let us first assume that we can only install one type of router. The cost of installing such a router in an access point has two components: a fixed cost f, and an additional cost c for each user that is connected to it. Such a router has a given capacity k and we know, for each user $u \in U$, the amount cr_u of capacity units it consumes from the router it is connected to.

Write down an integer linear program to find out in which access points we should install a router and which access point each user should be connected to in order to minimize cost and not to exceed the capacity of each router.

Additionally, create an OPL project to solve this problem. Problem data will be found in the file examA.dat. Use the file initialModel.mod as the initial model and complete it. Save this file as modelA.mod and send it through "El Racó".

(b) Assume that we have one additional requirement: routers can only be connected to users that are at distance at most d. To implement this constraint we know, for each user $u \in U$, its location (u_x, u_y) and for each access point $a \in A$, its location (a_x, a_y). Write down the mathematical formulation and extend the OPL model. Save this file as modelB.mod and send it through "El Racó". Use modelA.mod again as problem data.

(c) Finally, let us assume that we have a set of M different router models, each model $m \in M$ with its fixed cost f_m (there is no additional cost), capacity k_m and working distance d_m. The goal is now to decide which model should be installed in each access point so that the cost is minimized and all remaining constraints are satisfied. Write down the mathematical formulation and extend the OPL model, using now the data file examC.dat. Save the model as modelC.mod and send it through "El Racó".

8.1 a)

What we have:

U = set of users u that need to be connected to the internet, where $|u|$ is the number of users.
A = set of access points a where we can install routers.
k = capacity for each router.
c = cost for each connected user.
cr_u = capacity user u consumes from router r.

Find which access points should the routers and to which access point the users must connect in order to minimize the total cost of operation.

Decision variables:

a_u = access point a to which user u is connected.
$con_{u,a}$ = true iff user u is connected to access point a.
$inst_a$ = true iff router r is installed in access point a.
Objective function:

$$Min \sum_{a \in A} f \cdot inst_a + c \cdot |u|$$

We are minimizing the cost of the sum of all the installed routers plus the cost c for each connected user, assuming that all users must be connected.

Constraints:

$$\sum_{u \in U} cr_u \cdot con_{u,a} \leq k \cdot inst_a \quad \forall a \in A$$

For all the access points, the sum of the capacity consumed by all the users connected to this access point should be less or equal to the capacity of the router k that has been installed in this access point; otherwise, there sum should be equal to 0.

$$\sum_{a \in A} con_{u,a} = 1 \quad \forall u \in U$$

It basically says that each user u should be connected to one and only one access point a, implying that a router has been installed in that access point.

8.2 b)

Data:

d = maximum allowed distance from router to user.
(u_x, u_y) = user u location.
(a_x, a_y) = access point a location.

Then:

$$con_{u,a} = 0 \quad \forall u \in U : dist(u, a) > d, \forall a \in A$$

For all the access points, and for all the users such that the distance between the user and the access point is greater than the allowed distance d, the connection should not be allowed, i.e. $con_{u,a} = 0$. This is possible because both $dist(u, a)$ and d are input data, otherwise (it any of them were a decision variable), this can't be encoded in this way.

8.3 c)

Data:

M = set of m different router models.
f_m = fixed cost of installing a router model m.
k_m = capacity of router model m.
d_m = working distance of router model m.

Decision variable:

$inst_{a,m}$ = true iff router of model m is installed in access point a.

Objective function:

$$Min \sum_{a \in A} (\sum_{m \in M} f_m \cdot inst_{a,m})$$

Minimize the sum of the fixed cost of installing a router model m in access point a.
Constraints:

$$\sum_{m \in M} inst_{a,m} \leq 1 \quad \forall a \in A$$

There should not more than one router of model m installed in access point a.

$$\sum_{u \in U} cr_u \cdot con_{u,a} \leq \sum_{m \in M} k_m \cdot inst_{a,m} \quad \forall a \in A$$

For all access points, the sum of the capacity consumed cr_u of all the users connected ($con_{u,a}$) should be less or equal to the capacity k_m of the router model m installed ($inst_{a,m}$) in that access point.

We also have this condition:

$$inst_{a,m} = 1 \; AND \; con_{u,a} = 1 \Longrightarrow dist(u,a) \leq d_m \quad \forall u \in U, \forall a \in A, \forall m \in M$$

If there is a router model m installed in the access point a, and a connection between a user u and the access point a is possible, then the distance between the user u and the access point a should be less or equal to the working distance of the router model m. By the fact that $A \rightarrow B = \neg B \rightarrow \neg A$:

$$dist(u,a) > d_m \rightarrow inst_{a.m} = 0 \quad OR \quad con_{u,a} = 0$$

Constraint:

$$inst_{a,m} + con_{u,a} \leq 1 \quad \forall u \in U : dist(u,a) > d_m, \forall a \in A, \forall m \in M$$

When the distance between user u and access point a is greater than the working distance of a certain router model m, then only one of two conditions is possible, either the router model m is installed in access point a, or there is a connection between the user u and the access point a.

9 Tardor 2017

A university wants us to design to weekly schedule of a certain degree, whose duration in years is given by $years$. The schedule should range from Monday to Friday and from 8am until 8pm (i.e. last lecture finishes at 8pm). For that purpose, we know the set C of existing courses and the set P of available professors. All lectures are one-hour long, and all lectures of a given course are given by the same professor. However, not every professor can teach any course.

We have available the following data:

- A Boolean matrix $cy[c][y]$ telling whether course c belongs to year y.
- An integer vector $hoursPerCourse[c]$ containing the number of one-hour lectures of course c per week.
- A Boolean vector $cpAvail[c][p]$ telling whether professor p can teach course c.

(a) Consider that each course can have at most one-hour lecture per day. Obviously, two different courses cannot be simultaneously taught by the same professor. Similarly, we do not want two courses of the same year to overlap in time, since otherwise students of that year could not attend both. Taking into account the previous constraints, write an Integer Linear Program to build a weekly schedule that minimizes the number of professors that teach, at least, one course. The schedule should specify which professor teaches each course, and in which day/hours each course is taught.

Hint: use the following Boolean variables:

- $cdh_{c,d,h} \quad \forall c \in C \quad 1 \leq d \leq 5 \quad 1 \leq h \leq 12$
- $cp_{c,p} \quad \forall c \in C \quad \forall p \in P$

stating whether a course is given in some day/hour and by some professor, respectively. Note that it is very likely that you need additional variables.

(b) Modify the previous model to satisfy one additional constraint: each year should have at least one free day, i.e., for each year there is at least one day where no lecture of that year is given.

(c) Modify the previous model in order to satisfy the following requirement: for each year, there are at most 3 professors teaching all courses of that year.

9.1 a)

Decision variable:

$teaches_p \in \mathbb{B}$ = true iff professor p teaches at least 1 course. $\quad \forall p \in P$

Objective function:

$$Min \sum_{p \in P} teaches_p$$

Minimize the quantity of professors that teach courses.
Constraints:

$$|C| \cdot teaches_p \geq \sum_{c \in C} cp_{c,p} \quad \forall p \in P$$

For all professors, the sum of the courses a professor can teach should less or equal to the maximum number of courses the professor can teach, which is equal to the number of courses $|C|$ multiplied by the boolean variable $teaches_p$ that tells us if the professors teaches at least one course.

$$\sum_{h=1}^{12} cdh_{c,d,h} \leq 1 \quad \forall c \in C, \forall 1 \leq d \leq 5$$

A course cannot have more than 1 hour per day.

$$\sum_{d=1}^{5} \sum_{h=1}^{12} cdh_{c,d,h} = hoursPerCourse[c] \quad \forall c \in C$$

For all courses, the sum of the hours of a course for the whole week (12 hours per day, 5 days a week), is equal to the value of $hoursPerCourse_c$ that specifies the number of hours per week per course.

Implication: $cp_{c,p} = 1 \implies cpAvail_{c,p} = 1$ (If our decision variable determines that a professor teaches a course, then that professor should be also allowed to teach that course, since $cpAvail_{c,p}$ is input data, it should be the ruling condition.)

$$cp_{c,p} = 0 \quad \forall c \in C, \forall p \in P : cpAvail[c][p] = 0$$

For all courses, and professors, if the input data ($cpAvail_{c,p} = 0$) indicates that a professor cannot teach a course, then the corresponding value of the decision variable $cp_{c,p}$ should be 0.

$$\sum_{p \in P} cp_{c,p} = 1 \quad \forall c \in C$$

For all courses, only one professor teaches a course.

$$cp_{c,p} + cp_{c_2,p} + cdh_{c_1,d,h} + cdh_{c_2,d,h} \leq 3 \quad \forall\, 1 \leq d \leq 5, \forall\, 1 \leq h \leq 12, \forall\, p \in P, \forall\, c_1, c_2 \in C, \forall\, c_1 < c_2$$

This one is a little bit tough to explain. Certainly, two different courses c_1 and c_2 can be taught by the same professor and that's the first part of the constraint: $cp_{c_1,p} + cp_{c_2,p}$ where this sum is less or equal to 2. Then there is $cdh_{c_1,d,h} + cdh_{c_2,d,h}$, this sum should be less or equal to 1, because a professor cannot teach two courses that have the same day and hour, and that is the purpose of this constraint.

$$\sum_{c \in C} cdh_{c,d,h} \leq 1 \quad \forall\ 1 \leq d \leq 4, \forall\ 1 \leq h \leq 12, \forall\ 1 \leq y \leq years : cy[c][y] = 1$$

For all days, hours, and years, the sum of all the courses taught at the same time ($cdh_{c,d,h}$) should be less or equal to 1. Using $cy[c][y] = 1$ we get all the courses that belong to the same year.

9.2 b)

A new decision variable:

$free_{y,d}$ = true iff there is at least 1 free day per year.

Constraints:

$$\sum_{d=1}^{5} free_{y,d} \geq 1 \quad \forall 1 \leq y \leq years$$

Each year, must have at least one free day per week.

Then, there is the following implication:

$$cdh_{c,d,h} \implies free_{y,d}$$

If there is a course in some day and hour, then that day is not a free day. Encode that into a constraint:

$$\sum_{c \in C : cy[c][y]=1} \sum_{h=1}^{12} cdh_{c,d,h} \leq (1 - free_{y,d}) \cdot |C| \cdot 12 \quad \forall\ 1 \leq y \leq years, \forall\ 1 \leq d \leq 5$$

For all the years, and days, the sum of hours of all the courses that belong to a certain year should be less or equal to a big enough value (represented. by $|C| \cdot 12$). This is a conditional constraint that enforces that when $free_{y,d}$ is equal to 0 (false), the sum of hours for that day should be 0, thus, making that day a free day. Since in the previous constraint we enforced that there should at least one $free_{y,d} = 1$ per week, then this will result in at least one free day for each year. Also, previous constraints enforce that not every day will be a free day.

Another way to encode the same condition:

$$free_{y,d} \leq 1 - cdh_{c,d,h} \quad \forall\ 1 \leq y \leq years, \forall\ 1 \leq d \leq 5, \forall\ 1 \leq h \leq 12, \forall\ c \in C : cy[c][y] = 1$$

Whenever $cdh_{c,d,h} = 1$, then $free_{y,d}$ is 0. However, since we are enforcing that there should be at least 1 free day per week, then there must be at least one combination of y, d for which $free_{y,d}$ will be 1 and all the $cdh_{c,d,h}$ corresponding to that day, course, and year will be 0, making that a free day.

9.3 c)

For each year, there are at most 3 professors teaching all the courses of that year.

New decision variable:

$teachesYear_{p,y} \in \mathbb{B}$ = true iff professor p teaches at least 1 course in year y.

Constraint:

$$\sum_{p \in P} teachesYear_{p,y} \leq 3 \quad \forall\ 1 \leq y \leq years$$

For all years, at most 3 professor teach the courses of each year. However, this is not enough, since we are not restriction $teachesYear_{p,y}$, which we do in the following constraint:

$$teachesYear_{p,y} \geq cp_{c,p} \quad \forall\ p \in P, \forall\ 1 \leq y \leq years, \forall\ c \in C : cy[c][y] = 1$$

By using $teachesYear_{p,y}$ we are restricting $cp_{c,p}$.

Another way to encode the solution for part c without introducing a new decision variable is:

$$\sum_{p \in P} \sum_{c \in C : cy[c][y] = 1} cp_{c,p} \leq 3 \quad \forall\ 1 \leq y \leq years$$