Problem 1 a)

To prove that the graduation requirements problem is an NP-complete problem we have to prove that it is both in NP and an NP-hard problem.

1. in NP

To show that Graduation Requirements (GR) is in NP we can say that we can clearly see that we can verify a witness of GR in poly time. The witness is simply a list of courses taken. Which we check if it is less than k and then check if all requirements are fulfilled.
Thus the existance of a poly time verifier tells us GR $\in$ NP.

2. NP-hard to show this we shall reduce from an NP-hard problem. That is, lets show 3-SDT $\leq_p$ GR.

Poly time mapping

Given an instance ϕ of 3-SDT create an instance of GR in the following manner.
The set of classes n contains all the literals in ϕ.
For each clause C_i make a requirement r_i st the literals in C_i are the classes for requirement r_i
k is the size of the literals in ϕ

Now it is clear that in this way we can map an instance ϕ of 3-SDT to an instance of GR. and it is poly time in terms of the variables and clauses in the instance ϕ as we create sets per each clause and a set with all variables.
To finish the reduction we have to prove that a yes instance of 3-SDT maps to a yes instance of GR and a yes instance of GR maps to a yes instance of 3-SDT

$\Rightarrow$ if ϕ is a yes instaree of 3-SAT β is a yes instance of 6R.

if ϕ is a yes instance of 3-SAT that means that at least one literal in every clause is evaluates to true. Thus if we pick classes taken in the 6R instance in the following manner if x_1 is T chose class x_1, if x_1 is False chose class $\overline{x}_1$ then every requirement has at least one class that was chosen. Moreover, because we couldn't have chosen more than $|h|$ classes the set of classes chosen was less than or equal to h and thus a yes instance of ~~3~~SAT maps to a yes instance of 6R

$\Leftarrow$ if β is a yes instance of 6R then ϕ is a yes instance of 3-SAT

if β is a yes instance of 6R then there is a way to pick classes st at least one class from every requirement was chosen while choosing at most k classes. Thus if we chose x_i we would set x_i to True and if we chose $\overline{x}_i$ we would set x_i to false. And because at least one class was chosen from each requirement it means that at least one literal in every clause evaluates to true so ϕ is satisfiable. therefore a yes instance of 6R is a yes instance of 3-SAT

thus 3-SAT $\leq_p$ 6R.

And this means that 6R is NP-hard because 3-SAT is NP-hard.

Lastly this means that 6R is NP complete.

1 b)

The problem statement is asking for an algorithm that would get an optimal solution for the 6t problem. That is, what is the minimal set of classes such that if a student takes those classes, the student could graduate. We can do this by using the poly time verifier for the decision problem. With this set of classes and this number k. Is it possible for a student to graduate by taking at most k of this classes?

Algorithm

1 First phase: discover size of the minimal set
2 starting with the original set of classes and k = |set of classes|
3 Check w/ the poly time verifier for the decision problem
4 If it returns true reduce k by 1 and go to line 3 else increase k by 1 and go to line 5.
5. Now we have the size of the minimal set = k now we need to find the set itself. Given the original set of classes enumerate all permutations of size k. From before we know that at least 1 works.
6. Verify all permutations until one returns yes, that is we have found the set of classes for which a student can graduate, return this set.

This algorithm is clearly exponential as we have to enumerate all permutations of size min k of classes. However it clearly gives the minimal set required to graduate because we discovered that for a set smaller than min k there is no set of at most that size that could fulfill the requirements. And this particular set of size min k does fulfil the requirements as checked by the verifier.

Problem 1 c)

let s_j be the score that a student would have to get in the course j
the score is 0 if the course is not taken

let the requirement be that $\sum_{s_i \in S_i} \geq h_i$ for requirement r_i

So setting this up as a linear programming problem is the following

variables: all scores s_i for all available classes.

constraints: $\sum_{s_j \in S_i} \geq k_i \ \forall \ r_i$ That is the sum of my scores for classes in a given requirement need to be at least h_i for requirement r_i and this has to hold for all requirements

$0 \leq s_j \leq 1$

objective $\min \sum_{s_j}$

A solution for this linnear program would be a solution for the problem statement because it would give the minimum sum of all the scores for classes a student has taken (has to take) while making sure that each independent score is between 0 and 1 and the sum of the scores for a particular requirement is at least the score required for the requirement. That is a solution for the problem statement.
Moreover if we were to solve the problem statement, that is come up with the sum of scores, this would correspond to the solution of the linear programming problem proposed above.
Lastly, because we can solve this linear problem in poly time this means we can get the answer to the problem statement in poly time

Algorithms Problem 2

To prove the lecture planning problem is NP-complete we have to prove it is both in NP and it is NP-hard.

a) Lecture Planning $\in$ NP.

It is easy to see that the witness for an instance of the lecture planning problem is a schedule of lecturers chosen for each week. then the verifier would check that the selection is valid (the lecturer chosen for week i was available in week i) and that every project has at least one lecturer that was chosen.

This would run in polytime with respect to the size of weeks and projects.

Therefore Lecture Planning $\in$ NP.

b) lecture Planning is NP-hard.

To show this lets reduce from 3-SAT. That is 3-SAT $\leq_p$ Lecture Planning.

To do this we have to create a mapping from 3-SAT instance α to Lecture Planning instance β in polytime. And show that if α was a yes instance of 3-SAT then β is a yesInstance of Lecture Planning, and vice versa.

Polytime mapping

Given ϕ a 3-CNF boolean formula with k clauses and r distinct variables do the following.

create set $n = \{x_i, \bar{x}_i$ $\forall$ variables $1, \ldots, r\}$. That is the set of speakers is created by creating two speakers per variable.

Create sets $L_1, \ldots L_r = \{x_i, \bar{x}_i\}$ in L_i. That is there are r weeks where in week i two lecturers are available: x_i and $\bar{x}_i$

create sets $P_1, \ldots P_k$ where set P_i has the lecturers represented by the variables in clause i that is if clause i is $(x_1 \vee x_2 \vee x_3)$

$P_i = \{x_1, x_2, x_3\}$.

Thus from the original ϕ we have created an instance of the lecture planning problem.

Moreover, you can see that this mapping was poly time in the number of variables r and the number of clauses k.

Now we need to show a yes instance of 3-SAT maps to a yes instance of lecture planning and vice versa

$\Rightarrow$ if α is a yes instance of 3-SAT then β is a yes instance of lecture planning.

if α is a yes instance of 3-SAT then that means there is a configuration of the variables of ϕ is True. This means that at least one variable per clause is true. Moreover it means that we set each variable x_i to either true or false. This means that if we say that x_i was set to true we selected x_i to speak at week i and if we set it to false we pick $\overline{x_i}$ to speak at week i. Moreover because each clause has at least one true literal in ϕ this means that every project has at least one lecturer that presented for that project. Thus we see that a yes instance of 3-SAT maps to a yes instance of Lecture Planning.

$\Leftarrow$ if β is a yes instance of Lecture Planning then α is a yes instance of 3-SAT.

if β is a yes instance of lecture planning then that means we can select speakers to speak every week st at least one speaker that was necessary for P_i was selected $\forall i$. Therefore if we think of selecting a speaker at week i as setting that variable in ϕ (if we select x_i we set $x_i = T$, if we select $\overline{x_i}$ we set $x_i = F$) we have set all variables in ϕ to either T or F. Moreover, because we have a yes instance of lecture planning at least one speaker for P_i was selected. That is at least one literal in clause i is T $\forall i$ thus all

clauses of ϕ are T and ϕ is T, thus the instance of 3-SAT is satisfiable, that is, a yes instance of 3-SAT.

Altogether this means that 3-SAT reduces to Lecture planning, and because 3-SAT is NP-hard, so is lecture planning.

THUS, because Lecture planning is in NP and is NP-hard then lecture planning is NP-complete. ▨

Problem 3 a

variables: the variables are the units of flow sent on every edge for every commodity. That is $f_1(e_1)$ is the units of flow of commodity 1 sent across edge 1

constraints: $0 \leq \sum_{i \in k} f_i(e_j) \leq c(e_j)$ that is the sum of flows of all commodities on edge j cannot be less than 0 or greater than the capacity of the edge.

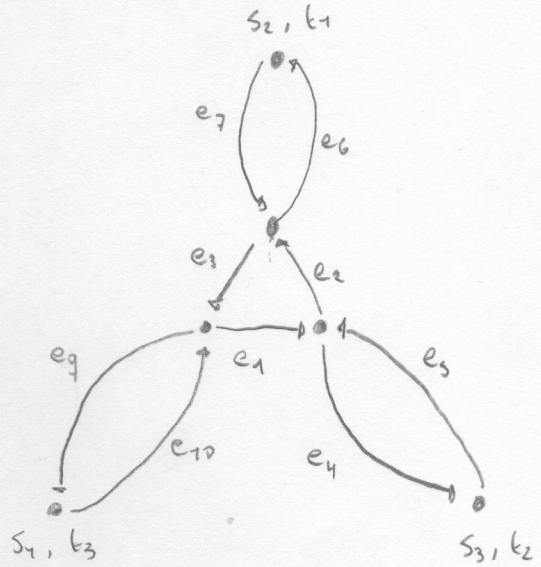
$f_{i\,in}(v_j) = f_{i\,out}(v_j)$ conservation of flow of a particular commodity

Objective $\max \sum_{i \in k} \left(\sum_{in} f_i(t_i) - \sum_{out} f_i(s_i) \right)$

that is maximize the amount of total flow across all commodities.

A solution to the linear programming problem above would give us the maximum sum of all individual commodity flows while making sure that the flows in an edge are positive and less than the capacity of an edge, and that flow conservation is preserved. That is, a solution to the linear problem would be a solution to the maximum multicommodity flow problem. On the other hand a solution to the maximum multicommodity flow problem would equate to the maximal sum of separate flows that are legal, in other words a solution to the linear problem proposed above.

for parts b and c i refer to some edges like so.



Problem 3

b) Show the above graph has maximum flow 3/2.

Notice that each commodity flow i from s_i to t_i has to take at minimum 2 edges on the internal triangle (edges 1, 2, 3). Notice further, then, that a particular edge is shared by at most 2 commodities.

Lastly Notice the edges in the outside (edges 4, 5, 6, 7, 8, 9, 10) do not constraint the flow of a particular commodity as they do not have to be shared by more than 1 commodity. Therefore we have that the maximum multicommodity flow is achieved by maximizing the flow at every edge in the triangle. That is:

$$f_1(e_1) + f_2(e_1) = c(e_1) = 1$$
$$f_3(e_2) + f_1(e_2) = c(e_2) = 1$$
$$f_3(e_3) + f_2(e_3) = c(e_3) = 1$$

And notice that

$$f_1(e_1) = f_1(e_2) = a$$
$$f_2(e_3) = f_2(e_1) = b$$
$$f_3(e_2) = f_3(e_3) = c$$

In order to conserve flow of a commodity.

Therefore we can rewrite the above system as

$$a + b = 1$$
$$b + c = 1$$
$$c + a = 1$$

and

$$a = 1 - b$$
$$c + 1 - b = 1$$
$$\text{so } c = b$$
$$\text{so } b + b = 1$$
$$2b = 1$$
$$b = \frac{1}{2}$$

so $a = 1/2$, $c = 1/2$

So the flow of every commodity is $1/2$

and the maximum multicommodity flow is $3/2$.

c) Any valid multicut in the above graph must be any in which both side of the cut must contain at least one vertex in the inner triangle but not all of them. This is because all source-sink paths touch all vertices in the triangle and if we include all the vertices in one side of the cut we could remove all edges across the cut without restricting flow of at least one commodity.

Thus any valid multicut in this graph has exactly two edges and because the capacity of every edge is 1 then any valid multicut has value 2.

And because <u>any</u> valid multicut has value 2 then it must be that the <u>minimum</u> multicut has value 2.