Dept. of Computer Science and Engineering,

CS69011: Computing Lab 1

Assignment 4 Linear Programming

Maximum Marks: 40

Problem 1: (10 marks)

You have n students and m courses. Each student wants to credit a set of courses in a particular semester. Let $A = [a_{sc}] s \in \{1, \dots, n\}, c \in \{1, \dots, m\}$ be the indicator variable for student preference, i.e. $a_{sc} = 1$ if student s wants to credit course c, 0 otherwise. Each course c can accommodate a maximum of M_c students. Each student can take a maximum of N courses. Given the input matrix A, output a course allocation such that the total number of courses taken by all students is maximized.

Is the course allocation unique?

Input format:
The input file will have the following format:
<Number of students> n, <Number of courses> m
<max. course per student> N
<List of max. students per course> $M_1, M_2, \dots, M_m$
<Number of student preferences> p
s_1, c_1
…
s_p, c_p

Output format:
List of courses allocated to each student.

Problem 2: (30 marks)

Now consider a more realistic scenario, where there are n students, m courses and o slots in the timetable. Assume that all timetable slots are identical, hence all courses can be offered in all slots. The task is to both assign courses to students, and courses to slots. The objective of assigning courses to slots is to minimize the number of slot clashes, while that of assigning courses to students is to both maximize the number of courses taken by students, while also minimizing the slot clashes.

As mentioned in part 1, the inputs are student course preferences: a_{sc}, limit on number of students per course: M_c, and maximum number of courses per student N. In order to solve the problem, we first define the following variables:
$x_{sc} \in [0,1]$: Indicator that student s is assigned course c.
$y_{ct} \in [0,1]$: Indicator that course c is assigned slot t.

For the problem of assigning slots to courses, we assume the course assignment to students, x_{sc}, is fixed. Then the total number of slot clashes can be calculated as: $\sum_{s,c,c',t} x_{sc} x_{sc'} y_{ct} y_{c't}$. However, this objective is quadratic in both x_{sc} and y_{ct}. Hence, we define the following joint variable:
$y_{cc't} \in [0,1]$: Indicator that courses c and c' are clashing through slot t.
Hence, we have the following formula for the number of slot clashes: $\sum_{s,c,c',t} x_{sc} x_{sc'} y_{cc't}$.

Also, $y_{cc't}$ can be 1 only if both y_{ct} and $y_{c't}$ are 1. Hence, we add the following constraints:
$y_{cc't} \geq y_{ct}$ and $y_{cc't} \geq y_{c't}$.
Moreover, every course should have a slot. Hence, we add the following constraints:
$\sum_t y_{ct} = 1 \; \forall c$
Hence the final formulation is:

LP1: $Minimize_{y_{ct}, y_{cc't}} \quad \sum_{s,c,c',t} x_{sc} x_{sc'} y_{cc't}$
Subject to:

$\sum_t y_{ct} = 1 \; \forall c$
$y_{cc't} \geq y_{ct} + y_{c't} - 1 \quad \forall \, c, c', t$
$y_{cc't} \in [0,1] \quad \forall \, c, c', t$
$y_{ct} \in [0,1] \quad \forall . c, t$

Similarly, with the definition that $x_{scc'} \in [0,1]$: Indicator that courses c and c' are clashing, and have been allotted to the student s, derive another linear program

LP2: $Minimize_{x_{sc}, x_{scc'}} \quad \lambda \sum_{s,c,c',t} x_{scc'} y_{ct} y_{c't} - \sum_{sc} a_{sc} x_{sc}$
Subject to:

$\sum_s x_{sc} \leq M_c \; \forall c$
$\sum_c x_{sc} \leq N. \; \forall s$
$x_{scc'} \geq x_{sc} + x_{sc'} - 1 \quad \forall \, s, c, c'$ such that for some t, $y_{ct} y_{c't} > 0$ (there is a clash between c and c')
$x_{scc'} \in [0,1] \quad \forall \, s, c, c'$
$x_{sc} \in [0,1] \quad \forall . c, t$

For the final problem of finding both allocations, you can run the following steps iteratively:

- Solve LP2 to find the optimal x_{sc} given the current y_{ct}. In the first step, you can assume that there are no slots, corresponding to $\lambda = 0$
- Solve LP1 to find the optimal y_{ct} given the current x_{sc}.

With every step you can increase the value of λ, so as to give more importance to minimizing slot clashes as opposed to assigning as many courses to students as possible.

Input format:
The input file will have the following format:
<Number of students> n, <Number of courses> m, <Number of slots> o
<max. course per student> N
<List of max. students per course> $M_1, M_2, \ldots, M_m$
<Number of student preferences> p
s_1, c_1
…
s_p, c_p

Output format:
List of courses allocated to each student.
List of courses allocated to each slot.

GLPK Guide:

Get yourself introduced to the GLPK API. If you want to install it in your personal machine (like laptop), you need to download the package from the standard repositories (http://ftp.gnu.org/gnu/glpk/) and compiling it from source. For Ubuntu, pre-built binaries can be installed using:

```
$ sudo apt install glpk-utils libglpk-dev glpk-doc
```

Read the user's manual from (https://cse.iitkgp.ac.in/~abhij/course/lab/CompLab-I/Autumn19/glpk.pdf). Include the following directive in your program.

```
#include <glpk.h>
```

Compile your code with the following flags.

```
gcc -Wall glpkdemo.c -lglpk -lm
```

GLPK supports both real-valued linear programming and mixed-integer optimization. The basic API calls are `glp_simplex` and `glp_intopt`. The integer optimizer needs an initial solution. You can start with the output of the simplex solver.