Question: How the problem instance was generated?

The problem instance was generated using a problem instance generator code that I wrote in python. In this code, I took the following parameter:

number of courses (> 100)

number of students (> 200)

minimum number of courses that each student can take (>=2)

maximum number of courses that each student can take.

For each student , a random number was selected between the minimum and the maximum number of courses the student can take. Let's say that the minimum number of courses = 2 and the maximum number of courses the student can take is 5. Then a student s_i will have between 2 to 5 courses.

For the next part of the discussion, we will assume that our

minimum number of courses = 2 and

maximum number of courses the student can take is 5

To ensure that there are no courses with 0 students, we employ the following strategy, assuming a student s_i can have x_i number of courses ($2 <= x_i <= 5$):

Initialize choice list C of courses with all available courses (from a master course list C.

For i = 1 to number of students:

(selecting courses for s_i)

If C has at least x_i number of courses then

x_i number of courses are removed randomly from Choice list C and put into the list of exams taken by student s_i . Updated choice list becomes $C = C – C_x$. Next student picks his courses from this updated choice list.

i <- i + 1 (move to the next student)

else

all courses n are removed from C and put into the list of exams taken by student s_i . Say this list of all courses is C_n .

Choice list is now empty. Re-initialize choice list C of courses with all available courses.

Remove all elements in C that are in C_n . The student can pick (x_i – n) number of courses from C` = C - C_n . This ensures that in this list there are no courses that this student s_i has previously taken.

(x_i – n) courses are removed randomly from C` and put into the list of exams taken by student $s_i$. C` updates to C`` after removing (x_i – n) courses.

Elements in C_n are added to C`` and this list is updated to C``` . This ensures that the next student will have all available courses to choose from from the master course list C, except for the ones that student s_i has chosen.

i <- i + 1 (move to the next student)

This strategy ensures that there are no courses with 0 students, and ensures that each student will have between 2 and 5 courses.

Determining Room Capacity:

Once we have generated courselist for each student, we determine the number and identity of students in each course, and from here determine the maximum number of students in a course, and minimum number of student in a course.

To ensure that the exam room can accommodate all the students of the course with the maximum number of students, we generate the exam room size with the following equation:

Exam room size = 2 * (maximum number of students in a course + minimum number of students in a course).

This ensures that the exam room size is big enough to accommodate the exam for the course with the largest student enrollment, and can house multiple course exam simultaneously if we want to.

Determining Maximum Number of Slots:

Maximum number of slots can not exceed the number of exams (number of courses). So for generating problem instance with a confirmed valid solution, we used the following formula:

Maximum number of slots = number of courses.

Determining the problem should be solvable:

The generated .crs and .stu files were tested against assignment 2 code, and a feasible solution was generated. The generated files are instance1.crs, instance1.stu and instance1.sol. The solution was generated against first objective function (most packed schedule).

generateProblem.py (Problem instance generator)

```
import random
from copy import deepcopy

numberOfCourses = 150
numberOfStudents = 250
minNumberOfCourses = 2
maxNumberOfCourses = 5

courseName = range(1,numberOfCourses+1)
course = {}
for i in courseName:
    student = list()
    course[i] = student
#    print i

for i in courseName:
    print i,course[i]

studentList = []
tempCourseList = deepcopy(courseName)
for student in range(0,numberOfStudents):
    randomNumberOfCoursesForStudent =
random.choice(range(minNumberOfCourses,maxNumberOfCourses+1))
    #print "Student:",student,"#ofCourses:",randomNumberOfCoursesForStudent
    listOfCourses = set()
#    tempCourseList = deepcopy(courseName)
    #print "len:",len(tempCourseList)
    #print tempCourseList
    if len(tempCourseList) > randomNumberOfCoursesForStudent:
        for i in range(0,randomNumberOfCoursesForStudent):
            tempElement = tempCourseList.pop(random.randrange(len(tempCourseList)))
            listOfCourses.add(tempElement)
            #print "element:",tempElement,listOfCourses
    else:
        count = 0
        tempSet = set()
        for i in range(0,len(tempCourseList)):
            tempElement = tempCourseList.pop(random.randrange(len(tempCourseList)))
            tempSet.add(tempElement)
            listOfCourses.add(tempElement)
            #print "element:",tempElement,listOfCourses
            count += 1

        print tempSet
        tempCourseList = deepcopy(courseName)
        print "copied master list"
        print tempCourseList
        print "removing from list"
        #removing previously added courses for current student
        for eachElement in tempSet:
            print eachElement
```

```
            tempCourseList.remove(eachElement)
        print tempCourseList

        #choosing courses from courses not previously added
        for i in range(count,randomNumberOfCoursesForStudent):
            tempElement = tempCourseList.pop(random.randrange(len(tempCourseList)))
            listOfCourses.add(tempElement)
            #print "element:",tempElement,listOfCourses

        #all courses for the current student has been added. adding previously
removed item from tempList
        #as this list is initialized for a new student now
        for eachElement in tempSet:
            print eachElement
            tempCourseList.append(eachElement)
        print "This list is for the next student"
        print tempCourseList


    studentList.append(listOfCourses)

print "printing student list"

studentFile = open("instance1.stu","wb+")
for student in range(0,numberOfStudents):
    print "Student",student,studentList[student]
    for courseNumber in studentList[student]:
        #print courseNumber
        #print course[courseNumber]
        course[courseNumber].append(student)
        #print course[courseNumber]
        studentFile.write(str(courseNumber)+"    ")
    if(student!=numberOfStudents-1):
        studentFile.write("\n")

studentFile.close()


print "Printing CourseList"
#determining number of students and max number of students in a course
maxNumberOfStudents = 0;
for i in courseName:
    print i,len(course[i]),course[i]
    if len(course[i]) >  maxNumberOfStudents:
        maxNumberOfStudents = len(course[i])

print "max number of students in a course",maxNumberOfStudents

#determine minimum number of students in a course
minNumberOfStudents = maxNumberOfStudents
for i in courseName:
    #print i,len(course[i]),course[i]
    if len(course[i]) <  maxNumberOfStudents:
        minNumberOfStudents = len(course[i])
```

```
print "min number of students in a course",minNumberOfStudents

#determining min number of seat in a room
minNumberOfSeatInARoom = 2*(maxNumberOfStudents+minNumberOfStudents)
print "min number of seat in a room :", minNumberOfSeatInARoom

#determining max number of timeslot
maxTimeSlot = numberOfCourses
print "max timeslot:",maxTimeSlot

#writing course file
courseFile = open("instance1.crs","wb+")
courseFile.write(str(minNumberOfSeatInARoom)+"    "+str(maxTimeSlot)+"\n");
for i in courseName:
    print i,len(course[i]),course[i]
    courseFile.write(str(i)+"    "+str(len(course[i])))
    if(i != courseName[len(courseName)-1]):
        courseFile.write("\n")
courseFile.close()
```
