EXAM SCHEDULING USING LOCAL SEARCH

CSE 318

Md Hasibul Husain Hisham

mdhasibulhussainh@gmail.com

PROBLEM SUMMARY

You are to design an exam timetable of students using local search, given all the ids of the courses taken by each student. In the end, you should get a day-wise plan of exam scheduling of the courses.

You cannot schedule two exams for the same student on the same day. Keep in mind that if you set two exams of the same student close to one another, you will incur a penalty amount. Penalty strategies are discussed later.

Your primary goal is to minimize the number of timeslots needed for the problem. Your secondary goal is to minimize the penalty amount of your scheduling.

(Here in the description, one day of an exam slot is equivalent to a timeslot)

CONSTRAINTS

- Hard: No two exams of one student can overlap (happening in same day) each other. Every exam of a student must be scheduled on different day.
- Soft: Exams should be placed in such a way that a student can get the maximum day difference between any two exams.

N.B.

Our problem can be considered as a graph/ map coloring problem. Where, nodes are the courses, and edges are placed between any two conflicting courses.

DATASET

Toronto Dataset

- Description: http://www.cs.nott.ac.uk/~pszrq/data.htm
- Dataset Download: https://www.cs.nott.ac.uk/~pszrq/files/Toronto.zip

Benchmark Datasets you will work with:

- car-f-92
- car-s-91
- kfu-s-93
- tre-s-92
- yor-f-83

DATASET

Example Dataset: *yor-f-83*

File: yor-f-83.crs

A portion of the *yor-f-83.crs* file (course file)

0001 23 (COURSE 0001 HAS 23 STUDENTS)

0002 19 (COURSE 0002 HAS 19 STUDENTS)

0003 67

…

…

…

0180 22

0181 22

DATASET

Example Dataset: *yor-f-83*

File: yor-f-83.stu

A portion of the *yor-f-83.stu* file (student file)

0031 0059 0091 0110 0133 0153 0166 0174 (COURSES OF STUDENT 0001)

0007 0024 0031 0060 0079 0110 0130 0152 (COURSES OF STUDENT 0002)

…

…

…

0040 0053 0074 0101 0103 0118 0179 (COURSES OF STUDENT 0939)

0073 0074 0088 (COURSES OF STUDENT 0940)

0043 0098 0100 0102 0151 0158 (COURSES OF STUDENT 0941)

PENALTY STRATEGY

- Linear Strategy:
  - Let, n = gap between any two exams
  - if (n <= 5) penalty = 2*(5-n)
  - else penalty = 0
- Exponential Strategy:
  - Let, n = gap between any two exams
  - if (n <= 5) penalty = 2^(5-n)
  - else penalty = 0

N.B.

You will show the final average penalty by summing all the penalties of all students followed by dividing by the number of students

HEURISTICS

Constructive Heuristics

Used for building a feasible solution (non-conflicting exam scheduling)

- Largest degree: The node with the largest number of edges (conflicting examinations) is scheduled first. Tie-break randomly.
- Saturation degree: The well-known Brelaz heuristic (used in DSatur algorithm) provides dynamic variable (or node) ordering. (Refer to the wiki link for the algo)
- Largest enrollment: The largest number of students registered for the examinations is scheduled first.
- Random ordering: One randomly picked node (course) will be colored (scheduled). Nevertheless, you are free to devise any creative heuristic here instead of randomly picking up any node.

HEURISTICS

Perturbative Heuristics

- Kempe-chain Interchange
  - A Kempe chain is defined as a connected subgraph that contains v, and that only comprises vertices colored with colors i and j.
  - Kempe (v, c(v), j) or Kempe (v, i, j)
  - Take a particular Kempe chain and swap the colors of all vertices

  (you should run kempe-chain minimum 1000 times as long as it is reducing penalty)
- Pair swap Operator
  - A pair-swap is the simultaneous application of two Kempe-chain interchanges applied to Kempe (v, c(v),c(u)) and Kempe (u, c(u),c(v)).
  - Let the Kempe chains KEMPE (u, i, j) and KEMPE (v, j, i) both contain just one vertex each (therefore implying that u and v are nonadjacent.) A pair swap involves swapping the colors of u and v. Used for building a feasible solution (non-conflicting exam scheduling).

  (you should run pair-swap minimum 1000 times as long as it is reducing penalty)

REPORT

- You are required to create a report
  - use exponential penalty strategy
  - apply any of the four constructive heuristics
  - apply kempe-chain heuristic first to reduce penalty
  - then apply pair-swap heuristic and see if penalty is reduced further

Rerun the above method for each of the four constructive heuristic (4 schemes)

Benchmark data	Known Best Solution		Scheme - 1 Largest-Degree + Kempe + Pairswap			
			Timeslots	Penalty		
	Timeslots	Penalty		After Largest-Degree	After Kempe	After Pairswap
CARF92	32	3.74	32	10.62	7.12	6.22
CARS91	35	4.42				
KFU93	20	12.96				
TRE92	23	7.75				
YOR83	21	34.84				

For linear penalty strategy, you are required to pick any one constructive heuristic of your choice and run both the perturbative heuristics in the mentioned order (minimum 1 scheme)

(In short, you will come up with total 5 schemes (minimum). Above is one example snippet of the desired report. Note that your result may differ based on the implementation.)

CONCLUSION

Special thanks to

- Abu Wasif sir
- Nazmul Islam Ananto

Deadline:

Monday, 23 January 2023

N.B.

Feel free to reach out at mdhasibulhussainh@gmail.com for any query