Course assignment:
Examination Timetabling Problem

R. Tadei, D. Manerba

Dept. of Control and Computer Engineering

Politecnico of Torino, Italy

01TXCSM - Decision Making and Optimization (DMO) - 2019/2020

Timetabling

- A *timetabling problem* is a setting with four parameters:
  - a finite set of time-slots
  - a finite set of resources
  - a finite set of meetings
  - a finite set of constraints
- The aim is to assign time-slots and resources to the meetings so as to satisfy the constraints (hardly or as much as possible).
- Timetabling problems arise in every organizations of people, machines, activities and in the most various applicative domains:
  - transportation/logistics
  - machine/job-scheduling
  - personnel shifts
  - educational/academic institutions

Timetabling in educational context

- important and time-consuming tasks which occur periodically (i.e. annually, quarterly) in all academic institutions:
    - school timetabling
    - course timetabling
    - examination timetabling
- the complexity of such tasks may be exacerbated by several factors:
    - dimensions of the desired timetable
    - very limited resources (time horizon, number of rooms, capacity of rooms, ...)
    - general or institution-specific requirements
        - *hard* constraints: have to be satisfied in each timetable
        - *soft* constraints: not compulsory but preferable

Examination Timetabling Problem (ETP)

Let us consider a set E of exams, to be scheduled during an examination period at the end of the semester, and a set S of students. Each student is enrolled in a subset of exams, at least one. The examination period is divided in t_{max} ordered time-slots.

By law, *conflicting* exams (i.e., having enrolled students in common) cannot take place during the same time-slot. Moreover, to incentive universities at creating timetables more sustainable for the students, the *Ministry of Education* assigns a penalty for each couple of conflicting exams scheduled up to a *distance* of 5 time-slots. More precisely, given the number $n_{e,e'}$ of students enrolled in both conflicting exams e and e', scheduled at distance i of time-slots, the penalty is calculated as $2^{(5-i)} n_{e,e'}/|S|$.

The Examination Timetabling Problem (ETP) aims at assigning each exam to a specific time-slot ensuring that:

- each exam is scheduled once during the period,
- two conflicting exams are not scheduled in the same time-slot.

The objective is to minimize the total penalty resulting from the created timetable.

Assumptions:

- during each time-slot there is always a number of available rooms greater than the total number of exams;
- rooms have enough capacity with respect to the number of enrolled students.

A clarifying example

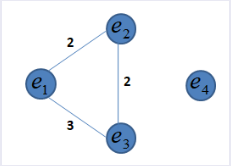
Input data

Exams: $\{e_1, \dots, e_4\}$
Students: $\{s_1, \dots, s_8\}$
Available time-slots: $\{t_1, \dots, t_6\}$

Enrollments:

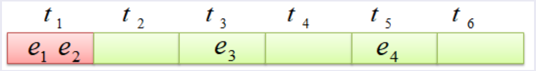
s_1: e_1 e_2 e_3
s_2: e_1 e_3
s_3: e_4
s_4: e_3
s_5: e_1 e_3
s_6: e_4
s_7: e_2 e_3
s_8: e_1 e_2

Conflict graph:



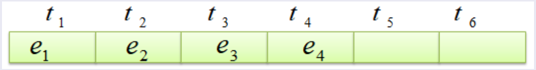
Sample solutions

An infeasible solution:



t_1	t_2	t_3	t_4	t_5	t_6
e_1 e_2		e_3		e_4	

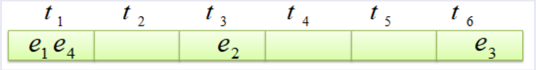
A feasible solution:



t_1	t_2	t_3	t_4	t_5	t_6
e_1	e_2	e_3	e_4		

with obj $= (2 * 2^{5-1} + 3 * 2^{5-2} + 2 * 2^{5-1})/8 =$
$(2 * 16 + 3 * 8 + 2 * 16)/8 = (32 + 24 + 32)/8 =$ **11**

A feasible (optimal) solution:



t_1	t_2	t_3	t_4	t_5	t_6
e_1 e_4		e_2			e_3

with obj $= (2 * 2^{5-2} + 3 * 2^{5-5} + 2 * 2^{5-3})/8 =$
$(2 * 8 + 3 * 1 + 2 * 4)/8 = (16 + 3 + 8)/8 =$ **3.375**

Benchmark instances: properties

$\longrightarrow$ **8 public instances** will be shared in order to let you test and assess the results of your algorithms with respect to the benchmarks
$\longrightarrow$ **? private instances** will be used to evaluate how the same algorithms are robust when solving instances unknown during the development

	Visibility	$\lvert E\rvert$	$\lvert S\rvert$	enr	t_{max}	$density$	$benchmark$
instance01	Public	139	611	5751	13	0.14	**157.033**
instance02	Public	181	941	6034	21	0.29	**34.709**
instance03	Public	190	1125	8109	24	0.27	**32.627**
instance04	Public	261	4360	14901	23	0.18	**7.717**
instance05	Public	461	5349	25113	20	0.06	**12.901**
instance06	Public	622	21266	58979	35	0.13	**3.045**
instance07	Public	81	2823	10632	18	0.42	**10.050**
instance08	Public	184	2750	11793	10	0.08	**24.769**
instance09 ... instance??	Private	-	-	-	-	-	-

- enr: total enrolments
- $density$: ratio between the number of pairs of conflicting exams and the total number of exams pairs (arc density of the conflicting graph)
- $benchmark$: obj value (total penalty) of the best solution available in the literature, **not necessarily optimal**

Benchmark instances: format

Each instance is defined by 3 plain text files named *instanceXX*:

- *instanceXX.exm*: defines the total number of students enrolled per exam.
  Format: 1 line per exam. Each line has the format

  ```
  INT1 INT2
  ```

  where `INT1` is the exam ID and `INT2` is the number of enrolled students in `INT1`.
- *instanceXX.slo*: defines the length of the examination period.
  Format: a single value in the format

  ```
  INT
  ```

  where `INT` is the number of available time-slots t_{max}. Hence, time-slots will have numeric IDs $\{1, 2, \ldots, t_{max}\}$.
- *instanceXX.stu*: defines the exams in which each student is enrolled.
  Format: 1 line for each enrollment. Each line has the format

  ```
  sINT1 INT2
  ```

  where `INT1` is the student ID and `INT2` is the ID of the exam in which student `INT1` is enrolled.

Assignment: tasks and deadlines organization

Required tasks:

- provide an Linear Programming formulation for the ETP (deadline 1)
- propose a solution approach for the ETP by exploiting one or more heuristic and meta-heuristic algorithms presented during the course
- develop/implement such a solution algorithm through a know programming language (C/C++ or Java)
- solve the benchmark instances through the implemented algorithm
- deliver the project code, the results, and a report of the work (deadline 2)
- present the solution method adopted and the results obtained (to be defined within Jan 13–16, 2020, during the last lessons of the course)

Deadlines:

- deadline 1 (problem formulation): 24/11/2019, 11:59 p.m.
- deadline 2 (project code, results, and report): 12/01/2020, 11:59 p.m.

Assignment: groups and evaluation

Groups:

- 6-7 students per group (one *leader* must be chosen for corresponding)
- group composition: up to you, must be decided and submitted before Oct 20, 2019 by filling up the spreadsheet at `https://tinyurl.com/DMO20192020`

Evaluation:

- one single grade [0,10] per group (no individual grades), based on:
  - correctness/completeness of your formulation
  - soundness/rationality/efficiency of your solution algorithm
  - goodness of your solutions (closeness to the benchmark) on public and private instances (3, 2, and 1 additional points to the first, second, and third best group, respectively)
  - quality/clarity of the presentation.

Support:

- during the course, 2 lessons (1h:30m each) will be dedicated to assistance in assignment development
- the *Forum* section of the course's site: a main thread with FAQs and, obviously, other Q/A that may interest all the groups

Assignment: given materials

- **instances.zip**: set of public instances (format already explained) plus a *test* instance (with solutions) corresponding to the example shown before
- **ETPchecker.exe**: a feasibility checker and obj function calculator. It is a black-box tool able to read an ETP instance and a relative specified solution. If the provided solution is feasible, it returns the value of the objective value (total penalty) of that solution, otherwise it returns an error relative to some infeasible characteristic of the solution

  How to use it from a command line (example):

  ```
  $ ETPchecker.exe instancename -check feasiblesolution
  ```

  PS: only 'logical' feasibility is checked! no guarantees if a non-correctly formatted solution is provided
- **benchmarks.xlsx**: an Excel file containing the properties and the benchmark values of the public instances. Given the solution values generated by your algorithm, it automatically calculates the percentage gap with respect to the benchmarks

Assignment: deliveries specifications (1)

Model formulation:

- upload a pdf file (max 2 pages) named **ETPmodel_DMOgroupXX**, where 'XX' is the number (two digits!) of your group, in *'Elaborati'* section of the course's site
- the file must contain:
  - the definition of the notation used for the parameters and for the variables (notation already introduced here must be maintained)
  - the Linear Programming formulation
  - a brief explanation of the meaning of its parts (objective function and constraints)

Assignment: deliveries specifications (2)

Project code, results and report:

- upload a .zip archive file named **ETPproject_DMOgroupXX** in *'Elaborati'* section of the course's site
- the archive must contain: risultati ottenuti nei 180-300 sec o runnate all'infinito
  - 8 text files containing your best solutions of the 8 given public instances (specifications follow)
  - a .xlsx file named **benchmarks_DMOgroupXX**: it is the given *benchmarks.xlsx* file duly completed with your results
  - an executable file (both .exe and .jar are allowed) named **ETPsolver_DMOgroupXX** (specifications follow)
  - a folder containing the complete development of the software (source code, project files, ...)
  - a .pdf file named **ETPreport_DMOgroupXX** containing a report of the work. Max 3 pages, focused on algorithm and results

Assignment: solver and solutions specifications

The solver:

- a command line application that, inputed with an instance name and a time limit `tlim` (in seconds), solves the ETP problem on the specified instance within the time limit and generates a solution text file
- the software must respond to one of the following syntaxes:
  ```
  $ ETPsolver_DMOgroupXX.exe instancename -t tlim
  $ java -jar ETPsolver_DMOgroupXX.jar instancename -t tlim
  ```
- stand-alone, i.e. without the need of installing any additional software
- must write/overwrite a text file any time a new best solution is found!

A solution:

- a text file named **instancename_DMOgroupXX.sol** containing, for each exam, the assigned time-slot (the assignment must be feasible!)
- format: 1 line per exam. Each line has the format

  ```
  INT1 INT2
  ```

  where `INT1` is the exam ID and `INT2` is the ID of the assigned time-slot (IDs must correspond to those read in the instance files)

Assignment: deliveries specifications (3)

Final presentation:

- in English, to the whole class, using your laptop linked to the projector
- max 15 minutes per group (including set-up time and 2-3 minutes for possible questions)
- at least 4 elements of the group must present
- DO NOT present the problem, the instances, or whatever is common to all groups
- FOCUS on your solution ideas and algorithms, on their tuning, on results and performances
- include some conclusive considerations
- upload the presented slides as a .pdf file named **ETPslides_DMOgroupXX** in *'Elaborati'* section of the course's site after the presentation

F.A.Q.s

Q: Is it possible to check the correctness of our Linear Programming (LP) model?

A: There exist several tools dedicated to model and solve LP problems.

- **MPL**: an high level language for mathematical programming. Free for students. Available in LADISPE. `http://www.maximalsoftware.com/mpl/`
- **IBM ILOG Cplex**: a powerful solver. Has a MPL-like language (OPL) and APIs for all programming languages. `https://www.ibm.com/developerworks/community/blogs/jfp/entry/CPLEX_Is_Free_For_Students`
- **Xpress**: a solver with its own modelling language. Available in LADISPE. `http://www.fico.com/en/latest-thinking/brochures/xpress-moselan-overview`
- **MS Excel Solver**: `https://www.solver.com/excel-solver-integer-programming`
- ...

By implementing your model through one of these tools, you can solve toy instances and check for the feasibility of their solutions with respect to the ETP requirements.

NB: The use of these tools is not strictly required for completing the assignment and, hence, it is not part of the course program.

F.A.Q.s

Q: Which machine will be used for the final test of the algorithms on the hidden instances?

A: The machine that will be used to test your algorithms has the following features:

- CPU: Intel(R) Core(TM) i7 - 860 @2.80 GHz
- 4 cores, 8 logical CPUs
- RAM: 16 GB
- OS: Windows 10 Pro, 64 bit

Q: How much time is supposed to be given to our algorithm for the tests on the hidden instances?

A: Since the benchmarks have been obtained through the use of exact solvers running for some hours, it is reasonable to provide a heuristic able to produce good solutions in few minutes. Hence, the hidden instances will be tested by imposing ≈ **180-300 seconds** of time limit. However, you should develop your algorithm assuming that the time limit is not known *a-priori*. Trivial suggestions:
a) develop algorithms that exploit all the available time
b) do not assign fixed times to subroutines