Department of Mathematics and Computer Science
University of Southern Denmark, Odense

October 1, 2009
Marco Chiarandini

DM811 - Heuristics for Combinatorial Optimization

Exam Project, Fall 2009

Note 1 The project is carried out individually and it is not allowed to collaborate. It consists of: algorithm design, implementation, experimentation and written report.

The evaluation of the project is based on the report. However, a program that implements the best algorithm described in the report must also be submitted. The program will serve to verify the correctness of the results presented. The report may be written in either English or Danish.

Note 2 Corrections or updates to the project description will be published on the course web page and will be announced by email to the addresses available in the Blackboard system. In any case, it remains students' responsibility to check for updates on the web page.

Note 3 *Submission*. An archive containing the electronic version of the written report and the source code of the program must be handed in through the Blackboard system **before 15:00 of Monday, 26 October 2009**. This is the procedure:

- choose the course DM811 in Blackboard,
- choose "Exam Project Hand in" in the menu on the left,
- fill the form and conclude with submit,
- print the receipt (there will be a receipt also per email).

See Appendix D for details on how to organize the electronic archive. Reports and codes handed in after the deadline will generally not be accepted. System failures, illness, etc. will not automatically give extra time.

---

1 Problem Description

Course timetabling at universities and other educational institutions involves scheduling a set of events (such as lectures, exercise or laboratory sessions, etc.) into given rooms and periods. The schedule is subject to resource and practicality constraints derived from the availability of rooms and teachers, student attendance to the events and precedence relations between events. Most commonly, periods extends over one week and the schedule is repeated weekly over the teaching term.

In this project we look at a specific formulation of the course timetabling problem, that we call Feasible Course Timetabling (FCTT). It consists in scheduling a set of events into periods and rooms. Students are enrolled into courses that may consist of lectures or exercise sessions and that make the set of events. Periods consist of a number of daily classes distributed over the working days of the week. Rooms have a capacity and specific "features", for example a projector or dry-erase boards. Each event has a set of required room

features, is taught by a teacher and has to be attended by a set of students. Both teachers and students may teach and attend, respectively, more than one event in the week. Teachers may also have a set of periods where they are "unavailable". Additionally, precedences may be specified between events in the form "event A must be scheduled in the week in an earlier period than lecture B".

A *timetable* or schedule is the assignement of an event to a period and a room. A complete timetable is a timetable in which all lectures are assigned to a period and a room. A *feasible timetable* is a complete timetable that satifies all the following constraints:

Events No more than one lecture is assigned in one room in each period.

Student conflicts No student attends more than one event in one period, i.e., events with students in common are assigned different periods.

Room suitability No lecture is assigned to a room that is not suitable, i.e., not meeting the capacity and the feature requirements of the event.

Availability No event is assigned to a period in which its teacher is unavailable.

Precedences No event violates the pre-defined event precedences in the week.

The FCTT asks for a given instance to find a feasible timetable or to determine that none exists.

More formally:

Definition 1. Feasible Course Timetabling (FCTT)
Input: *A set of r students* $S = \{s_1, s_2, \ldots, s_r\}$*, a set of n events* $E = \{e_1, e_2, \ldots, e_n\}$*, a set of p periods* $T = \{1, 2, \ldots, p\}$ *and a set of m rooms* $R = \{r_1, r_2, \ldots, r_m\}$*. For each student* s_x *a set of events to attend* $N_x \subseteq E$*, for each event* e_i *a set of unavailable periods* $U_i \subseteq T$ *and a set of suitable rooms* $Z_i \subseteq R$*. A precedence graph, i.e., a directed acyclic graph* $D = (V, A)$ *in which each vertex* $i \in V$ *represents an event* $e_i \in E$ *and each arc* $(i, j) \in A$ *a precedence cosntraint stating that the course* e_i *must precede course* e_j*.*
Task: *Find a timetable, i.e., a function* $\phi : E \mapsto T \times R$*, that satisfies all the following conditions:*

All events assigned:	$\forall e \in E$	$\phi_T(e) \in T \wedge \phi_R(e) \in R$
Events:	$\forall e_i, e_j \in E$	$\phi_T(e_i) \neq \phi_T(e_j) \vee \phi_R(e_i) \neq \phi_R(e_j)$
Student conflicts:	$\forall e_i, e_j \in E : \exists s_x \in S : e_i, e_j \in N_x$	$\phi_T(e_i) \neq \phi_T(e_j)$
Room suitability:	$\forall e_i \in E$	$\phi_R(e_i) \in Z_i$
Availability:	$\forall e_i \in E$	$\phi_T(e_i) \notin U_i$
Precedences:	$\forall (i, j) \in A$	$\phi_T(e_i) < \phi_T(e_j)$

For instances of the problem where a solution may not exist it is useful to define the quality of a timetable, in such a way that timetables with the least number of violated constraints are preferred. In the context of this project the quality of a timetable will also be the criterium to compare diffrent solvers when a feasibile timetable is not found. More sepcifically, we define the cost of a timetable as the sum of the following values:

1. number of students in events that are not assigned to a period or are assigned to an unavailable period,
2. number of students in events that are not assigned to a room or are assigned to a room that is not suitable,
3. number of students in all events scheduled in the same room at the same period,

ID	n	r	m	p	$\sum N_x/r$	$\sum S_i/n$	$\sum Z_i/n$	$\sum(\lvert T \setminus U_i\rvert)/n$	$\lvert A\rvert$
E-1	400	500	10	45	21.02	26.27	4.08	[16 ;25.34 ;34]	40 (14)
E-2	400	500	10	45	21.03	26.29	3.95	[17 ;25.69 ;33]	36 (14)
E-9	400	500	10	45	21.43	26.79	2.91	[17 ;25.42 ;34]	41 (18)
E-10	400	500	10	45	20.98	26.23	3.20	[14 ;25.47 ;34]	40 (13)
E-11	200	1000	10	45	13.61	68.04	3.38	[17 ;25.32 ;35]	21 (17)
E-12	200	1000	10	45	13.61	68.03	3.35	[15 ;25.67 ;35]	20 (13)
U-1	160	38	6	30	10.184	2.4188	4.15	[15 ;27.925 ;30]	0 (0)
U-15	251	129	16	25	8.1783	4.2032	10.757	[7 ;19.61 ;25]	0 (0)
U-16	366	160	20	25	7.5812	3.3142	13.746	[5 ;20.383 ;25]	0 (0)
U-17	339	150	17	25	7.7	3.4071	11.906	[5 ;19.808 ;25]	0 (0)
U-18	138	99	9	36	6.2121	4.4565	6.2029	[10 ;23.239 ;29]	0 (0)
U-19	277	132	16	25	8.1364	3.8773	11.235	[6 ;19.101 ;25]	0 (0)

Table 1: Summary statistics on the test instances for the project. The columns report in the order, the identifier of the instance, the number of events, the number of students, the number of rooms, the number of periods, the average number of events per student, the average number of students per event, the average number of suitable rooms per event, the minimal, average and maximal number of periods available per event and the number of precedence cosntraints (between parenthesis the number of these constraints that occur between lectures that share students).

4. number of students attending two events that are scheduled in the same period,
5. number of students in two events for which a precedence constraint does not hold.

Formally, let $S_i, S_i \subseteq S$, be the set of students attending event $e_i \in E$. The cost c of a timetable ϕ that belongs to all conceivable timetables Φ is a function $c : \Phi \mapsto \mathbf{R}$ given by:

$$c(\phi) = \sum_{e_i \in E\,:\,\phi_T(e_i) \notin T \setminus U_i} |S_i| + \tag{1}$$

$$\sum_{e_i \in E\,:\,\phi_R(e_i) \notin Z_i} |S_i| + \tag{2}$$

$$\sum_{E' \subseteq E\,:\,\forall e_i, e_j \in E'\,:\,\phi_T(e_i)=\phi_T(e_j) \wedge \phi_R(e_i)=\phi_R(e_j)} \sum_{e_i \in E'} |S_i| + \tag{3}$$

$$\sum_{s_x \in S} |\{(e_i, e_j) \in E \times E \,:\, i > j \,\wedge\, e_i, e_j \in N_x \,\wedge\, \phi_T(e_i) = \phi_T(e_j)\}| + \tag{4}$$

$$\sum_{(i,j) \in A\,:\,\phi_T(e_i) \geq \phi_T(e_j)} (|S_i| + |S_j|) \tag{5}$$

Clearly, the quality of infeasible timetables increases as cost decreases and a feasibiel timetable has cost zero.

2 Project Requirements

The aim of the project is to study efficient heuristic algorithms for solving 12 test instances of FCTT. Summary statistics relative to these instances are reported in Table 1. All the instances admit a solution, i.e., a feasible timetable. They can be downloaded from the course web site.

All the following points must be addressed to pass the exam:

1. Implement the procedure RandomTimetable of Figure 1 to generate random timetables.

```
1 function RandomTimetable(π)
2 input an instance π of FCTT;
3 output a timetable;
4 for e_i ∈ E do
5     φ_T(e_i) ← choose uniformly at random a period from T \ U_i;
6     φ_R(e_i) ← choose uniformly at random a room from Z_i;
7 return (φ);
```

Figure 1: Procedure to generate a random timetable.

2. Design and implement one or more construction heuristics that perform better than RandomTimetable.

3. Design and implement one or more local search algorithms.

4. Design and implement an effective algorithm enhancing the heuristics at the previous two points with the use of stochastic local search methods and metaheuristics.

5. For all the methods above carry out an experimental analysis and draw sound conclusions.

6. For the best algorithm devised in point 4, determine experimentally whether the initial solution should be a good solution provided by a construction heuristic devised in point 2 or whether the use of RandomTimetable would lead to equally good (or possibly better) final results.

In the experimental assessment all algorithms should be given a maximum time per run of **1 minute** on each single instance.[1]

3 Remarks

Remark 1 For each point above a description must be provided in the report of the work undertaken. In particular for the best algorithms arising from the experimental analysis enough details must be provided in order to guarantee the reproducibility of the algorithm from the report only (i.e., without having to look at the source code).

Remark 2 The results of the experiments must be reported either in graphical form or in form of tables. Moreover, for the best solver resulting from the point 4, a table must be provided with the best results for each specific instance of Table 1.

Remark 3 The total length of the report should not be less than 8 pages and not be more than 14 pages, appendix included (lengths apply to font size of 11pt and 3cm margins). Although these bounds are not strict, their violation is highly discouraged. In the description of the algorithms, it is allowed (and encouraged) to use short algorithmic sketches in form of pseudo-code but not to include program codes.

[1] Times refer to machines in IMADA terminal room.

Remark 4 This is a list of factors that will be taken into account in the evaluation:

- quality of the final results;
- level of detail of the study;
- complexity and originality of the approaches chosen;
- organization of experiments which guarantee reproducibility of conclusions;
- clarity of the report;
- effective use of graphics in the presentation of experimental results.

Appendix A Instance Format

Input data for an istance of FCTT are given in a text file. Elements from the sets of events, rooms, students and timeslots are identified by an integer number running from zero to the number of elements in the set minus one. Lines may have variable length and all intra-line separators are spaces. The first character of a line is a letter that identifies the content of the line. The legal lines are:

- `c` comment: typically the name of the instance

  ```
  c U-19.dim
  ```

- `p` problem dimensions. Given in the form `p [n] [m] [r] [q] [d]` where `n` corresponds to the number of events n, `m` to the number of rooms m, `r` to the number of students r, `q` to the number of periods per day and `d` to the number of days to schedule. Consequently, the identifiers for events, rooms, students and periods are $E = \{0, \ldots, n-1\}$, $R = \{0, \ldots, m-1\}$, $S = \{0, \ldots, s-1\}$, $T = \{0, \ldots, \texttt{qd} - 1\}$, respectively.

  For example:

  ```
  p 400 10 500 9 5
  ```

  Note, `q` and `d` can be omitted in which case it is assumed $|T| = 45$

- `e` events attended by a student. Given in the form `e [student] [e1] [e2] [e3] ...` where `student` is the student identifier and `e1 e2 ...` are the events that the student should attend.

  ```
  e 0 13 43 58 85 106 138 139 162 168 230 237 ...
  e 1 2 13 15 66 68 85 106 138 162 178 199 ...
  ...
  ```

- `u` event unavailabilities. Given in the form `u [event] [t1] [t2] [t3] ...` where `event` is the event identifier and `t1 t2 ...` are periods where the event cannot be scheduled because of teacher unavalabilities.

  ```
  u 0 0 5 6 7 10 12 19 20 21 23 24 25 27 28 29 31 34 36 37 39 40 41 42 43
  u 1 2 3 9 10 12 13 16 18 19 20 22 24 25 27 28 30 33 37 40 42
  ...
  ```

- `r` rooms suitabile for an event. Given in the form `r [event] [r1] [r2] [r3] ...`, where `event` is the event identifier and `r1 r2 ...` are the rooms where it can be scheduled.

  ```
  r 0 0 1 2 3 4 5 6 7 8 9
  r 1 0 1 2 3 4 5 6 7 8 9
  r 7 5 6 8 9
  r 8 5 6 8 9
  ...
  ```

- `a` precedences. Given in the form `a [ei] [ej]`, where `ei` and `ej` are event identifiers, meaning that `ei` has to be scheduled earlier than event `ej`.

```
a 24 107
a 31 130
a 35 276
a 41 319
...
```

Appendix B Solution Format

In order to check the validity of the results reported, the program submitted must output when finishing the best solution found during its execution in a file with extension `.sln`. The file must be in text format and contain a line for each event, in the order from 0 to n. In each line, separeted by a space there must be the *period number* and the *room number* to which the event has been assigned. The period number is an integer between 0 and $p-1$ representing the period allocated to the event. The room number is an integer from 0 to $m-1$ and corresponds to the room assigned to the event. A minus one in place of the period number or the room number indicates that the event has not been assigned a period or a room, respectively.

Example:

```
3 7
44 6
-1 -1
34 2
-1 -1
...
```

The example means that:

- the first event has been placed in period 3 and room 7
- the second event has been placed in the period 44 and room 6
- the third event has not been assigned to the timetable
- the forth event has been placed in period 34 and room 2
- the fifth event has not been assigned to the timetable

Appendix C Solution Validator

A program to check the validity of a given solution is made available at the course web page. The program is a binary exectuable that runs on the Linux machines of the IMADA terminal room.

To verify a solution type from the command line something like the following:

```
fctt-checker Inst-dim/fctt-1.dim Solutions/ctt_test_2-1_46688.txt.sln
```

Appendix D Handing in Electronically

The electronic archive to hand in must be organized as follows. It expands in a main directory named with the first 6 digits of your CPR number (e.g., 030907). The directory has the following content:

```
CPRN/README
CPRN/report/
CPRN/src/
```

The direcotry `reprot` contains a pdf or postscript version of your report. *Do not put your name in the author field of the report, instead put your CPR number*. The file README provides instructions for compilation of the program. The directory `src` contains the sources which may be in C, C++, Java or other languages. If needed a Makefile can be included either in the root directory or in `src`. After compilation the executable must be placed in `src`. For java programs, a jar package can also be submitted.

Programs must work on IMADA's computers under Linux environment and with the compilers and other applications present on IMADA's computers. Students are free to develop their program at home, but it is their own responsibility to transfer the program to IMADA's system and make the necessary adjustments such that it works at IMADA.[2]

The executable must be called `fctt`. It must execute from command line by typing in the directory `CPRN/src/`:

```
fctt -i INSTANCE -t TIME -s SEED -o OUTPUT
```

where the flags indicate:

- `-i INSTANCE` the input instance;
- `-t TIME` the time limit in seconds;
- `-s SEED` the random seed;
- `-o OUTPUT` the file name where the solution is written.

For example:

```
fctt -i Inst-dim/fctt-1.dim -o fctt-1.sln -t 180 -s 1 > fctt-1.log
```

will run the program on the instance `fctt-1.dim` opportunely retrieved from the given path for 180 seconds with random seed 1 and write the solution in the file `fctt-1.sln`.

It is advisable to have a log of algorithm activities during the run. This can be achieved by printing further information on the standard error or in a file. A suggested format is to output a line whenever a new best solution is found containing at least the following pieces of information:

```
best 853 time 10.000000 iter 1000
```

All process times are the sum of user and system CPU time spent during the execution of a program as returned by the linux C library routine `getrusage`. Process times include the time to read the instance.

[2]Past issue: the java compiler path is `/usr/local/bin/javac`; in C, any routine that uses subroutines from the math.c library should be compiled with the `-lm` flag – eg, `cc floor.c -lm`.