# THE DEPARTMENT OF COMPUTING EXAMINATION TIMETABLE PROBLEM

IAIN PHILLIPS

Abstract. We describe the problem of devising an assignment of examinations to rooms and times, drawing on the experience of the Department of Computing, Imperial College. A particular point of interest is that the cost function is inverse square, since it seeks to penalise small gaps for candidates between examinations.

## 1. Introduction

We may contrast the examination timetable problem (ETP) with the teaching timetable problem (TTP). In the latter case, the timetables for students and for teaching rooms are typically fairly full, and there may be only a few or no solutions. Therefore minimising costs is not likely to be an issue. In the ETP we typically have many solutions, and we can either just take any solution, or, as is the practice in the Department of Computing at Imperial College, seek to minimise the cost for students, lecturers and invigilators.

The constraints may be summarised as follows:

- The capacity of an examination room may not be exceeded.
- No student can sit two examinations simultaneously (or with too short a gap).
- If two examinations share material then they must be scheduled simultaneously.
- If two examinations share a room then they must not overlap in time (or be separated by too short a gap), or else they must start and finish together.
- The number of calculators (or other resources) in use at any time must not exceed a given bound.
- If a student cannot sit examinations at a certain time then this time must be avoided for them.

The costs may be summarised as follows:

- The amount of invigilation
- The number of times a lecturer has an examination at an unfavourable time (e.g. when they wish to be away at a conference)
- A measure of the size of gaps for students between successive examinations. This is the most interesting cost, as we should penalise small gaps, so that it is natural to measure this by summing the inverse of the gaps, or to make the penalty higher, the inverse square, as is our current practice.

The simple algorithm (call it SA) we have written, and which is used to solve the ETP at present, works roughly as follows: First, rank the examinations in some order of scheduling difficulty. The heuristic used here is that an examination is harder

*Date*: December 1999. Version of May 9, 2000.

to schedule if it has been pre-allocated (e.g. to fit in with another department) or if it has more students. Then we schedule each examination in turn by rank. We look at each time and room and first see whether it is allowed by the constraints. If so then we calculate the cost. There are further heuristics to favour scheduling examinations at certain times - e.g. to place examinations on a Monday or Friday (so that there will be a large gap for the weekend after or before), and to place the longer examinations in the morning in preference to the afternoon (this is so that two sittings can be scheduled in the afternoon), and to use the larger rooms first. Finally we select the time and room with the least cost and move on to the next examination.

Of course we might reach an impasse. SA has no back-tracking, but we can simply inspect the results to see which examination failed, and pre-allocate this or otherwise increase its rank. Then we run the algorithm again. Experience shows that SA tends to succeed nearly always, but that one can do some 'hand-tuning' to achieve a lower cost.

SA is a partial solution, but both the ETP and the TTP (in all but its simplest versions) are known to be NP-complete. SA fits into the general pattern of 'successive augmentation' [Sch95, Sch99], where we proceed by successively selecting the most urgent (most tightly constrained) examination and allocating it in the most favourable fashion. A much more sophisticated algorithm which finds a solution and then seeks to improve it by moving examinations has been described in [CLC94] [not read]. Genetic algorithms for examination timetabling have been considered in [CFM93].

This note is intended to report on the precise constraints of the problem within the Department of Computing. There may or may not be some novel features, particularly concerning the measurement of invigilation cost and the cost of the spacing for students.

In the next section we formulate ETP exactly.

## 2. The problem formulated

Let $\mathbb{N}$ denote the natural numbers, $\mathbb{B}$ the Booleans $\{0, 1\}$ (with 1 as true and 0 as false) and $\mathbb{R}$ the real numbers. We are given the following finite sets, functions and relations:

- the set of examinations $E$
- the set of candidates (students) $S$
- the set of rooms $R$
- the set of examiners (lecturers) $L$
- the examination period, an interval $P \subseteq \mathbb{N}$
- the set of possible start times $T \subseteq P$
- the set of possible durations $D \subseteq \mathbb{N}$
- the duration function $dur : E \to D$
- the capacity function $cap : R \to \mathbb{N}$
- the unavailability relation $unavail : R \times T \to \mathbb{B}$
- the avoid relation $avoid : L \times T \to \mathbb{B}$
- the forbid relation $forbid : S \times T \to \mathbb{B}$
- the taking relation $taking : S \times E \to \mathbb{B}$
- the room pre-allocation function $prerm : E \to \wp(R)$
- the start time pre-allocation function $prest : E \to \wp(T)$

- a bound $K$ on the maximum number of examinations taken by a student, with $J = \{1, \ldots, K\}$
- the lecturing relation $lect : L \times E \to \mathbb{B}$
- the co-scheduling equivalence relation $cosched : E \times E \to \mathbb{B}$
- the possible number of invigilators for the rooms $I = \{0, \ldots, n\}$, some $n \in \mathbb{N}$
- the invigilation threshold function $invig : R \times I \to \mathbb{N}$
- the examinations requiring calculators $calc : E \to \mathbb{B}$
- the number of calculators $C \in \mathbb{N}$
- the minimum time gap between examinations in one room $RG \in \mathbb{N}$
- the minimum time gap between examinations for one student $SG \in \mathbb{N}$
- the maximum time gap between start times of co-scheduled examinations $EG \in \mathbb{N}$

We shall let $d, e, i, j, l, r, s, t$ range over $D, E, I, J, L, R, S, T$ respectively. Also we shall abbreviate $\forall d \in D$ by $\forall d$, $\sum_{d \in D}$ by $\sum_d$ and $\sum_d \sum_e$ by $\sum_{d,e}$ etc.

Let us further define the size (number of candidates) function $size : E \to \mathbb{N}$:

$$size(e) = \sum_s taking(s, e)$$

Most of the above should be self-explanatory. We treat time as discrete, taking values in $\mathbb{N}$. The invigilation threshold function measures how many candidates a certain number of invigilators can invigilate in a given room. It is strictly increasing in its second argument until a plateau is reached. The examination timetable problem (ETP) is to define three functions, for start times and rooms of examinations, and the number of invigilators assigned to particular sittings:

$$rm : E \to R$$

$$st : E \to T$$

$$inv : R \times T \to I$$

subject to various constraints to be defined and so as to minimise various costs to be defined.

Note that we assume that each examination takes place in a single room. We deal with examinations which are too big for a single room in two different ways: We combine two rooms to form a composite room, or, if this is not convenient, we split the examination into two smaller, co-scheduled examinations.

Also note that we allow sittings in different rooms to overlap in time, unlike in other approaches where there are non-overlapping periods.

We mention some facts and figures concerning Department of Computing examinations: There are some 100 examinations, with durations 90, 120 or 180 minutes. There are 4 rooms with capacities 130, 90, 80 and 25. The maximum number of invigilators is 3. There are 200 calculators. There are some 750 students, taking a maximum of 11 examinations. The possible starting times are 10:00, 14:00, 14:30, 16:00, and the examination period lasts for 3 weeks, with examinations Monday to Friday, interrupted by one Monday bank holiday (in the first, second or third week). $RG$ and $EG$ are 30 minutes and $SG$ is 60 minutes.

### 2.1. **Constraints.**

*Pre-allocation.* For all $e$:

$$rm(e) \in prerm(e)$$

$$st(e) \in prest(e)$$

*Capacity.* The capacity of an examination room may not be exceeded (it is enough to consider the start times):

$$\forall r, t \sum_{\substack{e \\ rm(e) = r \\ st(e) = t}} size(e) \leq cap(r)$$

*Calculators.* Similarly the number of calculators (or other portable resources for candidates) in use at any time must not exceed a given bound (again it is enough to consider the start times):

$$\forall t \sum_{\substack{e \\ st(e) \leq t < st(e) + dur(e)}} calc(e).size(e) \leq C$$

*Student gap.* No student can sit two examinations simultaneously (or with too short a gap):

$$\forall s, e, e'[(taking(s,e) = taking(s,e') = 1) \Rightarrow$$

$$(st(e') + dur(e') + SG \leq st(e) \vee st(e') \geq st(e) + dur(e) + SG)]$$

*Cluster.* Another constraint prevents students from having a 'cluster' of examinations. Let us say that no student can sit more than $M$ examinations in a time interval of length $H$:

$$\forall s, t |\{e \in E : taking(s,e) = 1 \wedge t \leq st(e) \leq t + H\}| \leq M$$

It might for instance be desirable to disallow more than 1 examination in a day. In the Department of Computing this would be too strong a requirement. However we might require no more than 3 examinations in 2 days. One could impose a series of cluster constraints.

*Co-scheduling.* If two examinations share material then they must be scheduled within the maximum time gap allowed:

$$\forall e, e'[cosched(e,e') = 1 \Rightarrow |st(e') - st(e)| \leq EG]$$

*Co-sitting.* If two examinations share a room then they must not overlap in time (or be separated by too short a gap), or else they must start and finish together:

$$\forall e, e'[rm(e) = rm(e') \Rightarrow (st(e') + dur(e') + RG \leq st(e) \vee$$

$$st(e') \geq st(e) + dur(e) + RG \vee (st(e) = st(e') \wedge dur(e) = dur(e'))]$$

In view of this constraint each 'sitting' (one or more examinations in the same room at the same time) has a well-defined duration, and by abuse of notation we may define a function $dur : R \times T \to D$ (where if there is no sitting any value in $D$ can be chosen).

*Availability.* Rooms can only be used when available:

$$\forall r, e.unavail(rm(e), st(e)) = 0$$

*Forbidden times.* If a student cannot sit examinations at a certain time then this time must be avoided for them:

$$\forall s, e[(taking(s, e) = 1) \Rightarrow (forbid(s, st(e)) = 0)]$$

*Invigilators.* The number of invigilators in a room is sufficient for the number of candidates:

$$\forall r, t \quad \sum_{\substack{e \\ rm(e) = r \\ st(e) = t}} size(e) \leq invig(r, inv(r, t))$$

## 2.2. **Costs.**

*Invigilation.* The simplest cost is the amount of invigilation:

$$\sum_{r,t} dur(r, t).inv(r, t)$$

Actually the number of invigilators can be calculated trivially once the timetable is known, but we phrase the problem this way so that the invigilation cost function is simple.

*Lecturer.* The next cost is the number of times a lecturer has an examination at an unfavourable time (e.g. when they wish to be away at a conference). The more times a lecturer seeks to avoid, the lower is the weighting given to their preferences. This is achieved by letting:

$$weight(l) = \begin{cases} \frac{1}{\sum_t avoid(l,t)} & if \sum_t avoid(l, t) > 0 \\ 0 & otherwise \end{cases}$$

The lecturer cost is then:

$$\sum_{l,e} weight(l).lect(l,e).avoid(l,st(e))$$

*Student.* The most significant, and the most interesting, cost is a measure of the size of gaps for students between successive examinations. We should penalise small gaps, so that it is natural to measure this by summing the inverse of the gaps, or to make the penalty higher, the inverse square, as is our current practice. If student $s$ takes examinations with start and finish times $t_1, f_1, \ldots, t_n, f_n$ (in ascending order) then the gaps are $t_2 - f_1, \ldots, t_n - f_{n-1}$. Let $G \subseteq \mathbb{N}$ be the (finite) set of all possible gaps, ranged over by $g$. This can be computed easily from $T$ and $D$. Let $pen : G \to \mathbb{R}$ be the (decreasing) penalty function. In our implementation of SA we have taken $pen(g) = g^{-2}$. Then:

$$studentcost(s) = \sum_{k=1}^{n-1} pen(t_{k+1} - f_k)$$

Clearly we wish to minimise:

$$\sum_{s} studentcost(s)$$

Finally we must decide on the relative weighting of the three kinds of cost. This would seem to be a matter of judgment and experiment.

## 3. Preliminary Analysis

In this section we describe ETP in terms of the standard language of optimisation problems. The *integer programming problem* (IPP) [Bea96] requires us to find values for Boolean variables $x_i$ ($i \in I$ some index set $I$) which minimise the linear objective function $c^T.x$ subject to the constraints $A.x \leq b$.

*Variables and basic constraints.* Let us take Boolean-valued variables

$$x_{s,e,j,g,t,r,d,i} \quad (s \in S, e \in E, j \in J, g \in G, t \in T, r \in R, d \in D, i \in I)$$

The intended meaning is that $x_{s,e,j,g,t,r,d,i} = 1$ iff student $s$ takes $e$ as $s$'s $j$th examination with gap $g$ after $e$, starting at time $t$, with duration $d$, in room $r$ and with $i$ invigilators. Let us add the value $\infty$ to $G$ to represent the 'gap' after a student's last examination.

For each $s$, $e$ the values of the remaining parameters are unique:

$$\sum_{j,g,t,r,d,i} x_{s,e,j,g,t,r,d,i} = taking(s,e) \tag{3.1}$$

The next four constraints capture the fact that if $x_{s,e,j,g,t,r,d,i} = 1$ then $e$ is the $j$th examination of $s$ and the gap after $e$ for $s$ is $g$, with the last gap being $\infty$.

For each $s$, $j$ the values of the remaining parameters are unique:

$$\sum_{e,g,t,r,d,i} x_{s,e,j,g,t,r,d,i} \leq 1 \tag{3.2}$$

For each $s$ the values of $j$ form an initial segment of $J$, i.e. for each $s$, $e$, $j' < j$, $g$, $t$, $r$, $d$, $i$:

$$x_{s,e,j,g,t,r,d,i} \leq \sum_{e',g',t',r',d',i'} x_{s,e',j',g',t',r',d',i'} \tag{3.3}$$

For each $s$, $e$, $j$, $g < \infty$, $t$, $r$, $d$, $i$:

$$x_{s,e,j,g,t,r,d,i} \leq \sum_{e',g',r',d',i'} x_{s,e',j+1,g',t+d+g,r',d',i'} \tag{3.4}$$

For each $s$, $e$, $j$, $t$, $r$, $d$, $i$:

$$x_{s,e,j,\infty,t,r,d,i} + \sum_{e',g',t',r',d',i'} x_{s,e',j+1,g',t',r',d',i'} \leq 1 \tag{3.5}$$

Let us further have Boolean variables

$$y_{e,t,r,d,i} \quad (e \in E, t \in T, r \in R, d \in D, i \in I)$$

The intended meaning is that $y_{e,t,r,d,i} = 1$ iff examination $e$ starts at time $t$, in room $r$, with duration $d$ and with $i$ invigilators. For each $e$ the values of the remaining parameters are unique:

$$\sum_{t,r,d,i} y_{e,t,r,d,i} = 1 \tag{3.6}$$

Furthermore for each $s$, $e$, $j$, $g$, $t$, $r$, $d$, $i$:

$$x_{s,e,j,g,t,r,d,i} \leq y_{e,t,r,d,i} \tag{3.7}$$

Let us further have Boolean variables

$$z_{t,r,d,i} \quad (t \in T, r \in R, d \in D, i \in I)$$

The intended meaning is that $z_{t,r,d,i} = 1$ iff the sitting starting at time $t$ in room $r$ has duration $d$ and $i$ invigilators. For each $t \in T$, $r \in R$ the values of the remaining parameters are unique:

$$\sum_{d,i} z_{t,r,d,i} \leq 1 \tag{3.8}$$

Furthermore for each $e$, $t$, $r$, $d$, $i$:

$$y_{e,t,r,d,i} \leq z_{t,r,d,i} \tag{3.9}$$

### 3.1. **Constraints.**

*Pre-allocation.* For all $e$:

$$\sum_{e,t \in prest(e), r \in prerm(e), d, i} y_{e,t,r,d,i} = 1 \tag{3.10}$$

*Capacity.* For all $r$, $t$:

$$\sum_{s,e,j,g,d,i} x_{s,e,j,g,t,r,d,i} \leq cap(r) \tag{3.11}$$

*Calculators.* For all $t$:

$$\sum_{s,e,j,g,t-dur(e)<t' \leq t,r,d,i} x_{s,e,j,g,t',r,d,i} \leq C \tag{3.12}$$

*Student gap.*

$$\sum_{s,e,j,g<SG,t,r,d,i} x_{s,e,j,g,t,r,d,i} = 0 \tag{3.13}$$

*Cluster.* For all $s$, $t$:

$$\sum_{e,j,g,t \leq t' \leq t+H,r,d,i} x_{s,e,j,g,t',r,d,i} \leq M \tag{3.14}$$

*Co-scheduling.* For all $e$, $e'$:

$$cosched(e,e').y_{e,t,r,d,i} \leq \sum_{t-EG \leq t' \leq t+EG,r',d',i'} y_{e',t',r',d',i'} \tag{3.15}$$

*Co-sitting.* For all $r$, $t$, $d$, $i$:

$$z_{t,r,d,i} + \sum_{t<t'<d+RG,d',i'} z_{t',r,d',i'} \leq 1 \tag{3.16}$$

*Availability.*

$$\sum_{t,r,d,i} unavail(r,t).z_{t,r,d,i} = 0 \tag{3.17}$$

*Forbidden times.*

$$\sum_{s,e,j,g,t,r,d,i} forbid(s,t).x_{s,e,j,g,t,r,d,i} = 0 \tag{3.18}$$

*Invigilator.* For all $r$, $t$:

$$\sum_{s,e,j,g,d,i} x_{s,e,j,g,t,r,d,i} \leq \sum_{d,i} invig(r,i).z_{t,r,d,i} \tag{3.19}$$

We see that all the constraints are linear.

3.2. **Costs.** Minimise the following objective functions:

*Invigilation.*

$$\sum_{r,t,d,i} i.d.z_{r,t,d,i}$$

*Lecturer.*

$$\sum_{l,e,t,r,d,i} weight(l).lect(l,e).avoid(l,t).y_{e,t,r,d,i}$$

*Student.*

$$\sum_{s,e,j,g,t,r,d,i} pen(g).x_{s,e,j,g,t,r,d,i}$$

(We set $pen(\infty) = 0$).

These are all linear. As in Section 2.2, we have to decide what relative weight to give to the three objective functions.

## 4. Further Work

This could include:

- Classification of the problem within the class of IPPs
- Exact solution of the IPP of Section 3
- Comparison of the exact solution with the solution given by SA (see Introduction)
- Solution by a genetic algorithm
- Formulation and solution as a constraint logic programming problem

## References

[Bea96] J.E. Beasley, editor. *Advances in linear and integer programming.* Oxford University Press, 1996.

[CFM93] D. Corne, H. L. Fang, and C. Mellish. Solving the modular exam scheduling problem with genetic algorithms. In *Proceedings of the Sixth International Conference on Industrial and Engineering Aplications of Artificial Intelligence and Expert Systems*, pages 370–373, 1993.

[CLC94] M. W. Carter, G. Laporte, and J. W. Chinneck. A general examination scheduling system. *Interfaces*, 24:109–120, 1994.

[Sch95] Andrea Schaerf. A survey of automated timetabling. Technical Report CS-R9567, CWI, 1995.

[Sch99] Andrea Schaerf. A survey of automated timetabling. *Artificial Intelligence Review*, 13(2):87–127, 1999.

Department of Computing, Imperial College, 180 Queen's Gate, London SW7 2BZ, England. iccp@doc.ic.ac.uk