Heuristics for the Exam Scheduling Problem

Fu Zhaohui and Andrew Lim
Department of Computer Science, National University of Singapore
3 Science Drive 2, Singapore 117543

Abstract

As part of the process of creating a campus-wide timetabling system for the National University of Singapore, the authors investigated examination-scheduling algorithms. The challenge in exam scheduling is to draw up the final examination timetable, taking into account a number of different constraints. The authors propose a different approach when the inter-examination gaps (termed paper spread) of each student should be maximized. The results compare favorably against the actual timetable produced by the current manual system.

1 Introduction

The National University of Singapore (NUS) is a state university in Singapore. Currently, NUS offers more than 2,000 modules to about 20,000 undergraduate and postgraduate students over two normal semesters.

2 The Examination Scheduling Problem

2.1 Constraint Types

The general timetable scheduling problem is known to be NP-complete [2]. For our work, we addressed the examination-timetabling problem, which involves the scheduling of a number of examinations (papers) in as few sessions (time slots) as possible, given the capacities of the venues and the examinations taken by each candidate. The following constraints, termed as critical constraints, must be satisfied for any valid timetable:

(1.1) Two examinations that are taken by any particular student cannot be scheduled in the same session.

(1.2) The combined number of candidates taking all the examinations at a particular venue for a particular session cannot exceed the capacity of that venue.

(1.3) Except for the case where the number of candidates taking an examination exceeds the capacity of all venues, all examinations should be scheduled into only one venue.

In addition to the critical constraints, there can be a number of other constraints that should be satisfied if possible, unless it results in the failure to satisfy any of the critical constraints. These are termed non-critical constraints. These possibilities include:

(2.1) No student should have examinations on consecutive sessions.

(2.2) Examinations with more candidates should be scheduled earlier in the examination period (to give the examiner more time for grading).

(2.3) The examinations should be scheduled such that the number of sessions between examinations for all students is maximized (to give the maximum amount of study time).

In this paper, we will concentrate on the most important non-critical constraint – (2.3), which is maximizing the paper spread. The problem is defined as follows.

DEFINITION: Let S be an examination schedule. Define $s(p)$ to be the number of sessions that paper p belongs to in S. Define $exam(c, i)$ to be the i^{th} exam taken by candidate c in schedule S. Define $num(c)$ to be the number of papers taken by candidate c. Then, *the paper spread of candidate c in S* is:

$$PS(S,c) = \frac{\sum_{i=1}^{i<num(c)} [s(exam(c,i+1)) - s(exam(c,i))]}{num(c) - 1}$$

Essentially, the paper spread for a student in a schedule is the average number of sessions between

1082-3409/00 $10.00 © 2000 IEEE

each successive paper. Thus, a student who has exams on days 1, 4 and 9 would have a paper spread of $[(4-1)+(9-4)]/2 = 4$ days.

2.2 Problem Domain

In order to test our eventual system, we obtained student registration data from the NUS Computer Center. The statistics are displayed in Table 1.

Year	Sem	#Stud	#Paper	#Tuples	#Const
97/98	I	20596	1586	96987	22745
	II	21350	1663	92812	22792
98/99	I	21607	1561	101197	23751
	II	22808	1594	97718	24600
99/00	I	23055	1468	98995	24472

Table 1. Statistics of the test data

Table 2 gives the list of examination venues that can be used for scheduling the examinations. All the examination halls are located within the NUS campus except the IMM Exhibition Hall, which is a commercial building that is not owned by NUS.

Alias	Venue	Capacity
IMM	IMM Exhibition Hall	1600
GYM	Gymnasium	312
MPH1	Multi-Purpose Sports Hall 1	750
MPH2	Multi-Purpose Sports Hall 2	850
CH	Competition Hall	396

Table 2. List of examination venues

3 The Graph Model

To facilitate our description of the scheduling algorithms, we will use a graph model to represent the interactions between different examinations. Let a timetabling problem be expressed as a graph

$G = \{V, E\}$, such that
$V = \{v | v \in$ set of all papers $\}$, and
$E = \{(v_1, v_2) |$ there exists a student taking v_1 and $v_2\}$.

Additionally, each edge e in G is weighted, such that the weight of $e = (v_1, v_2)$ and $w(e)$ =number of students taking both papers v_1 and v_2.

After constructing the graph, we have analyzed some special properties of our graph.

Table 3 shows that the graph is dominated by one huge connected component (CC) whose size is above 1000. The remaining connected components are relatively small. The graph is heavily connected since we can only find 1 or 2 articulation points which may disconnect the graph. Thus, it is not possible for us to divide the big graph into smaller sub-graphs.

Year, Sem	#CC	#AP	Top 3 CC		
97/98, I	51	1	1337	14	14
97/98, II	56	1	1324	22	21
98/99, I	50	0	1270	47	14
98/99, II	52	2	1306	41	22
99/00, I	58	1	1192	25	24

Table 3. Properties of the Graph

4 Satisfying Critical Constraints

We first examine our approach for creating a valid timetable when we take only the critical constraints into account. For critical constraint (1.1), we implemented a graph-coloring algorithm. For constraint (1.3), we implemented a multiple knapsack packing algorithm.

4.1 Multiple Knapsack

The problem becomes complicated when we take into account the capacity of our suite of venues. Since constraint (1.3) disallows the scheduling of an examination into more than one room, we need a packing algorithm to allocate the papers into the various venues within a session.

The packing problem we faced is in fact a Multiple 0-1 Knapsack Problem, which is known to be NP-hard [4].

We choose the Best Fit as a heuristic algorithm after several experiments.

4.2 Iterative Greedy

We implement the scheduling program using the Iterative Greedy heuristic [2] to find a good coloring solution. The details of the program are given as follows.

1. Let G be the problem graph (vertices are ordered by some *vertex-ordering*). Let c = number of papers. Let $n = 0$.

2. Perform a coloring on G using a greedy heuristic. Let c' be the number of colors used in this solution. If $c' < c$ then $c = c'$ $n = 0$.

3. Group the vertices by the coloring found in step 2. Reorder the vertices in G according to their groups, such that all vertices in the same group are consecutively ordered. The order of the groups is determined by some group-ordering.

4. Increment n. If $n < LIMIT$, go to Step 2 and restart.

Note step 3 reorders the vertices while keeping those vertices in the same group together in the sequence, using some ordering measure. Reordering vertices in this way and then re-coloring can never produce a coloring that requires more colors. The value of *LIMIT* determines the number of non-improving iterations that are made before the algorithm halts. For our algorithms, we used a *LIMIT* value of 15.

The main factor in the Iterative Greedy algorithm is the group-ordering that is used for reordering the groups in Step 3. We have investigated several methods regarding the populaton and the number of constraint, etc.

98/99, I	Vertex-ordering					
	Random		Des.#Stu		Des.#Cons	
	×	√	×	√	×	√
Simple	39	32	37	31	29	29
Random	39	34	37	33	29	29
Largest Pop	39	39	37	37	29	29
Smallest Pop	39	32	37	32	29	29
Most Const	39	31	37	32	29	29
Least Const	39	39	37	33	29	29
Most Papers	39	39	37	34	29	29
Least Papers	39	31	37	31	29	29

Table 4. Comparison of group-ordering methods (no. of sessions) using static greedy heuristics (× No Iterative Greedy; √ With Iterative Greedy)

5 Maximizing Paper Spread

We now take into account constraint (2.3) of maximizing paper spread for our timetable.

5.1 Schedule by Detecting Heavily Connected Components

From Chapter 3, it turns out that the graph is heavily connected with a dominant connected component. There are virtually no articulation points inside the graph. Some of the most heavily connected components complicate our problem by forming some large clique. It is easy to show that a size n clique will require n colors to color it. These exams affect the average inter-paper gap significantly since they have a large number of students. Thus, we will detect these *Big* exams and schedule them first.

Our strategy involves 5 steps:

1. Perform the greedy coloring algorithm given in the previous section. Set the number of sessions in the examination period N to be the number of colors used by this algorithm.

2. Find the most heavily connected components in the problem graph.

3. Schedule the papers in these components first, maximizing their spread over the examination period.

4. Schedule the remaining papers. If this is not possible, increment N and go to Step 2.

5. Further optimize the resultant schedule.

5.1.1 Schedule Optimization with Tabu Search

After the initial scheduling using the big exams list, we perform another procedure to further optimize the average spread of the schedule. It is not easy to achieve the optimal average spread. In fact, it is a NP-hard problem. We make use of a local search heuristic, Tabu search, for this purpose.

Our iterative step involves exchanging all the examinations that have been scheduled in 2 different time slots.

There are several options for defining the objective functions for Tabu Search. They are given as follows.

1. Directly use of the value of Average Inter-Paper Gap.

2. Suppose a student has c_2 2-consecutive exams, c_3 3-consecutive exams,..., c_x x-consecutive exams. Then the objective function for this student will be

$$\sum c_i * i^2$$

3. Make use of inequality $\sum_{i=1}^{n} a_i \leq \sqrt[n]{\prod_{i=1}^{n} a_i}$, where the equality is achieved only when $a_1 = a_2 = ... = a_n$.
DEFINITION Define e_i to be the i^{th} exam time slot of a student, the objective function is $\sqrt[n]{\prod_{i=1}^{n} e_i}$. This is because $\sum_{i=1}^{n} e_i$ = the length of the exam

period, which is fixed. The more spread out of the exams, the larger the value of $\sqrt[n]{\prod_{i=1}^{n} e_i}$ will be returned

In the experiment, we used mainly the 2 objective functions because we were primarily concerned about the length of the consecutive exams. And this is also one of the criterion in the evaluation of the result.

To find the best cut-off between the result and the number of iterations, consider Table 5.1.1

Semester, Year	No. of Nonimprovements			
	0	5	10	15
1997/1998, Sem I	37	30	29	29
1997/1998, Sem II	36	31	30	30
1998/1999, Sem I	39	32	31	31
1998/1999, Sem II	38	32	30	30
1999/2000, Sem I	44	36	35	35

Table 5. Comparisons of No. of Nonimprovement

Our experiments were conducted using a length-5 Tabu list, with 10 non-improving iterations as the terminating condition.

Note that Tabu search would not affect the number of sessions taken by the schedule, only the paper spread.

5.2 Test Results

Table 6 shows the results for the various algorithms on the data for 1998/1999 Semester I. Paper spread alone is misleading since a schedule with more sessions is likely to give a better spread. Hence the value of paper spread per session is also given.[5].

Method	Result			Time
	#Sess	PS	PS/#Sess	(sec)
Greedy	28	3.628	0.130	22
Greedy (Tabu)	28	4.815	0.172	3196
Big E	28	3.785	0.135	642
Big E(Tabu)	28	4.996	0.178	3362
Manual	47	5.712	0.121	+++

Table 6. Comparisons of algorithms

Although bare statistics seem to suggest that the manual system is inferior in all ways, this is misleading. This is because we did not take into consideration all the non-critical constraints that were considered by the human schedulers. Also, since the length of an exam period of the manual system is much longer than the automated system, the Average Inter-Paper Gap is also longer.

5.3 Transforming the Constraints

When we consider constraint (2.3), there are various ways to interpret it. The main idea is to restate the constraints in an unambiguous way. Or sometimes, constraints like (2.3) are explicitly requested by the university, as we will state in this section.

1. No student should have more than X consecutive exams.
2. All the students must have exams at least Y slots apart.
3. Assume Z exams per day. No student should have more than E exams in D days.

During the Iterative Greedy coloring process, the exam will be scheduled into a particular time slot if and only if it satisfies all the constraints.

6 Conclusion

Our results show that a sequence of greedy graph coloring, pre-scheduling heavily-connected components (big exams list), greedily scheduling the remaining papers without backtracking, and Tabu search optimization based on slot swapping produces good results. In any case, the automated solutions give shorter examination periods with better paper spreads than the actual timetables created using manual methods.

References

[1] Culberson. Iterated greedy graph coloring and the difficulty landscape. Technical report, University of Alberta Computing Science, 1992.

[2] J. Culberson and Luo. *Cliques, Coloring and Satisfiability*, volume 26 of *DIMACS Series*. Eds D.S. Johnson and M.A. Trick, 1996.

[3] A. Lim, W. C. Oon, J. C. Ang, and W. K. Ho. Development of a campus-wide university course timetabling application. In *PATAT*, pages 71-77, 2000.

[4] S. Martello and P. Toth. a program for the 0-1 multiple knapsack problem. *ACM Transactions on Mathematical Software*, 11:135–140, 6 1985.

[5] Z. Fu, and A. Lim. The university examination scheduling problem. In *Artificial Intelligence and Soft Computing Conference*, pages 350-354, 2000.