Journal of the Operational Research Society (1996) 47, 373–383 0160-5682/96 $12.00

Examination Timetabling: Algorithmic Strategies and Applications

MICHAEL W. CARTER[1], GILBERT LAPORTE[2] and SAU YAN LEE[1]

[1] University of Toronto and [2] École des Hautes Études Commerciales de Montréal, Canada

Examination timetabling is an important operational problem in many schools, colleges and universities. The authors have developed a computerized examination timetabling system called EXAMINE. In this article several algorithmic strategies for this problem are investigated and compared. Computational results are reported on randomly generated problems and on some real problems.

Key words: examination timetabling, graph colouring

INTRODUCTION

Over the years, several algorithms have been suggested for the problem of scheduling examinations in schools and universities. The basic problem is to assign examinations to a limited number of time periods in such a way that there are no conflicts, i.e., no student can write more than one examination at a time, while ensuring that a number of side constraints are satisfied. The following side constraints represent some of the more common requirements.

(1) Some exams are preassigned to specific time slots or a group of slots (e.g., evenings).
(2) Limitations on available seating capacity or specific room availabilities.
(3) Certain exams must be consecutive or must take place in a specified sequence.
(4) There may be a maximum number of exams in one period or a maximum number of students per period.
(5) There may be restrictions of the form 'no student can write x exams in any y consecutive periods.' For example, 'no student can write 3 exams in any 4 consecutive periods.' These restrictions are sometimes treated as constraints, or penalized in the objective function. In some cases, the number of exams per day for the same student is limited or some situations are penalized.
(6) Certain exams have 'preferred' periods. For example, most schools try not to schedule large exams in the last few days.
(7) In real problems, the number of available periods is given as a fixed constraint.
(8) There may also be constraints on exams with the same instructor, e.g., two exams with the same instructor should be scheduled in nearby rooms.

In several institutions that allow a wide variety of course combinations, it is becoming increasingly difficult to keep the length of the examination schedule within acceptable limits. For example, in 1990–91, Carleton University in Ottawa was forced to curtail the length of the fall semester from 13 to 12 weeks in order to allow enough time for examinations to take place[1]. In such contexts any reduction in the length of the schedule, even by only one or two days, is significant. The authors have developed EXAMINE, a computerized examination scheduling system, based partly on the work of Laporte and Desroches[2], but incorporating several new features. A detailed description of how the system was implemented at the University of Toronto and at Carleton University can be found in Carter, Laporte and Chinneck[1]. We also summarise in the present article implementation results for a number of other institutions.

EXAMINE and a number of other examination timetabling algorithms incorporate costs to assess the quality of potential schedules[2,3]. These can be used to favour schedules in which examinations are well spaced out or in which the number of undesirable assignments is low. For

Correspondence: G. Laporte, GERAD, École des Hautes Études Commerciales de Montréal, 5255 avenue Decelles, Montreal, Canada H3T 1V6

example, EXAMINE uses a proximity cost w_s whenever a student has to write two examinations scheduled s periods apart: these weights are $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$ and $w_5 = 1$. In loosely constrained problems, determining a feasible schedule is not really a concern and more effort can be spent on producing a good schedule. Costs are useful in that context. As all real timetabling problems have different side constraints associated with them, for all practical purposes each school has a unique problem. Consequently, the definition of 'The Examination Timetabling Problem' becomes rather vague. For example, at the University of Toronto, the biggest problem was fitting exams into a limited number of rooms, whereas at Carleton the major problem was the number of exams that needed to be held in evening time slots. This makes it very difficult to compare algorithms, or to compare problems. In this article, we study the two issues of trying to find a conflict-free solution while spreading student exams as evenly as possible over the schedule.

It is well known that finding a conflict-free schedule without side constraints is a graph colouring problem (see, e.g., Carter[4]). Most examination scheduling algorithms, including EXAMINE, use a list processing scheme to solve the underlying graph colouring problem: examinations are sorted in order of decreasing difficulty according to some criterion, and are sequentially assigned to periods without creating conflicts (Barham and Westwood[5], Mehta[6], Johnson[3]). The length of the schedule produced by this approach can sometimes be excessive, and a number of authors have implemented backtracking procedures to alleviate this problem (Laporte and Desroches[2], Hertz[7]). One conclusion of the Laporte and Desroches[2] study is that the initial ordering of examinations has virtually no impact on the final outcome. This seems to contradict findings of other researchers who have shown that the initial ordering strategy can be critical in graph colouring (see Dunstan[8], Matula, Marble and Isaacson[9]). Two explanations offered by Carter[4] are that the backtracking feature of the Laporte and Desroches algorithm is sufficient to correct any colouring errors, and that there were probably sufficient periods available in the real problems solved by the authors so that it was not difficult to find a conflict-free solution. Further tests carried out by Carter indicate that on some data sets (University of Toronto, École des Hautes Études Commerciales) the initial ordering is not critical, but on other problems (University of Otago, University of Waterloo, Carleton University) it has a significant impact and can make a difference of a few days in the final schedule. In our opinion, this question deserves further study, especially in view of the fact that obtaining the shortest possible examination schedule is often essential.

This article describes a computational study of this question and it also analyzes the impact of cost and of backtracking. Our tests are carried out on unconstrained problems as we do not want our results and conclusions to be affected by too many secondary parameters. We let the number of exams and conflict matrix density vary. In some tests, the number of periods is given and the objective is to minimize average cost per student; in other tests, the objective is to minimize the number of periods. A first set of experiments are conducted on real-life problems from which all side constraints have been removed, and on randomly generated problems. Then, using the best strategy identified in these basic tests, we solve a number of real-life constrained problems. All experiments are conducted using the EXAMINE platform.

In the next sections we describe the various algorithmic rules used for the comparison and we report computational results. The conclusions follow.

ALGORITHMIC RULES

We consider the five following sorting criteria for the list processing scheme.

(1) *Largest degree* (LD): largest number of conflicting examinations. The rationale behind this rule is that examinations conflicting with many others are hard to schedule and should be assigned first.

(2) *Saturation degree* (SD): number of periods in conflict. This rule was proposed by Brelaz[10]. The examination that is selected next is the one with the fewest number of feasible available periods remaining. In a sense, the most difficult exam to schedule is the one that has the least flexibility in terms of choice of period. Ties are broken using the largest degree.

(3) *Largest weighted degree* (LWD): number of students in conflict. This rule can be thought of as a combination of rules 1 and 4; we select the exam with the largest degree, where each edge is

weighted by the number of students in conflict. We might expect that this rule would give priority to core courses where large numbers of students have identical conflicts.

(4) *Largest enrollment* (LE): largest number of students registered for an examination. It is often the case that examinations with a large enrollment are difficult to schedule as they create more conflicts.

(5) *Random ordering* (RO): this rule is mainly considered for benchmark comparisons. If the above strategies really have a significant impact on solution quality, then they should be better than just randomly selecting the next exam.

Five more rules were defined by first determining a large clique, i.e., a set of mutually conflicting examinations, and scheduling these first. The reason for this is that the length of the schedule is at least as large as the size of any clique and these core examinations should be dealt with while no assignment has yet been made. Moreover, in a sense, the largest clique represents the most difficult group of examinations to schedule since they are all in conflict. For the determination of the clique, we use the Carter and Gendreau[11] algorithm. For the remaining examinations, the above rules apply.

When the algorithm is executed without costs or side constraints, the examination timetabling problem reduces to a pure graph colouring problem. This is how some of the test problems were solved. We also tested a version of the algorithm that uses the same proximity costs w_s as in Laporte and Desroches[2], but no side constraints.

In addition, two versions of the list processing scheme can be defined according to whether backtracking is used or not. Backtracking simply means that when an examination to be scheduled is in conflict with every period, some assignments already made will be undone in order to schedule that examination. Any examination removed from the schedule through this process is bumped back into the waiting list. With this process, rules must be defined to avoid cycling.

We have implemented the following backtracking process. Suppose examination i cannot be scheduled in any period. Consider all periods p and all examinations j already in period p and in conflict with examination i. For each such j, identify the period to which j could be moved at least cost. Let m_p be the cost incurred by moving in this fashion all movable examinations to a different period. Each examination j that cannot be moved (no alternate conflict-free periods) has to be 'bumped' back on to the waiting list, i.e., j will later come back to be rescheduled. The cost b_p of bumping a set of examinations from period p is equal to the number of examinations that must be bumped back on to the waiting list. No examination j can be bumped by the same examination i more than once. If i has already bumped some exam j in period p, we set $b_p = \infty$, and say that i cannot be assigned to period p. When it is not necessary to bump any exams back to the waiting list, we set $b_p = 0$. At the end of the search, there are three outcomes possible:

(1) $\min_{p \in P_i} b_p = \infty$: No feasible schedule can be found. Examination i is removed from the waiting list and reported as unschedulable at the end of the algorithm.

(2) $\min_{p \in P_i} b_p = 0$: There exists at least one period p to which examination i can be assigned without bumping any examinations on to the waiting list. Select among these periods the period p^* with the minimum disruption cost, m_{p^*}. Examination i is then moved to period p^*, and the necessary conflicting exams are moved to other periods as previously identified.

(3) $0 < \min_{p \in P_i} b_p < \infty$: Let period p^* be the period yielding the minimum. Break any ties using the minimum value of m_p.

A total of forty different strategies were therefore tested: five basic rules, using cliques or not, using costs or not, using backtracking or not.

TEST PROBLEMS

One of the major drawbacks of most articles in the timetabling literature is that testing is limited to randomly generated problems and perhaps to one practical example. In this study, in addition to running randomly generated problems, we carried out a comparison of thirteen unconstrained real-life examples. Electronic mail access to these problems can be obtained by following the instructions given in the appendix.

Real problems were obtained from eight Canadian institutions, from the London School of Economics, from King Fahd University of Petroleum and Minerals, Dhahran and from Purdue University, Indiana, by removing all side constraints. The main characteristics of these problems are summarized in Table 1. Each problem is identified by a code where the first three letters refer to the institution, F stands for the Fall semester and S for the Spring semester, and the two digits correspond to the year. Most entries of this table are self-explanatory except perhaps for the last column; the density represents the proportion of non-zero and non-diagonal entries of the matrix (c_{ij}), where c_{ij} is the number of students writing both examinations i and j.

TABLE 1. *Characteristics of real problems*

Code	Institution	Number of examinations	Number of students	Number of student examinations	Density d of the conflict matrix
CAR-F-92	Carleton University, Ottawa	543	18419	54062	0.14
CAR-S-91	Carleton University, Ottawa	682	16925	59022	0.13
EAR-F-83	Earl Haig Collegiate Institute, Toronto	181	941	6029	0.27
HEC-S-92	École des Hautes Études Commerciales, Montreal	81	2823	10634	0.20
KFU-S-93	King Fahd University of Petroleum and Minerals, Dharan	486	5349	25118	0.06
LSE-F-91	London School of Economics	381	2726	10919	0.06
PUR-S-93	Purdue University, Indiana	2419	30032	120690	0.03
RYE-F-92	Ryerson University, Toronto	487	11483	45052	0.07
STA-F-83	St. Andrew's Junior High School, Toronto	139	611	5539	0.14
TRE-S-92	Trent University, Peterborough, Ontario	261	4360	14901	0.18
UTA-S-92	Faculty of Arts and Sciences, University of Toronto	622	21267	58981	0.13
UTE-S-92	Faculty of Engineering, University of Toronto	184	2750	11796	0.08
YOR-F-83	York Mills Collegiate Institute, Toronto	190	1125	8108	0.29

Random problems were obtained by controlling the number n of examinations and the density d of the conflict matrix. These problems were generated by creating students one at a time, and then assigning them to r randomly selected courses, where r follows a discrete uniform distribution in the interval [2, 6]. This process ends when a specified density d is reached. Ten different instances were generated for each combination of $n = 200$, 400 and $d = 0.05$, 0.15, 0.25. Random problems are identified by the code $R - n - d$.

COMPUTATIONAL RESULTS FOR UNCONSTRAINED PROBLEMS

Extensive tests were conducted on the 13 real problems, and on 10 random problems of each type, using a SUN SPARC server 330 computer (16.7 MHz). We first analyzed the effect of using a clique initialization strategy on the number of periods required to construct a feasible schedule and on the number of times backtracking was required in the algorithm. One might expect the use of a clique as a starting point to be beneficial both for the number of periods required and for the number of backtracking steps. However, our tests suggest that on random problems, the use of a clique strategy for the initial solution has no effect on the final outcome. This is easily explained by the fact that random problems have no structure and usually have a small maximal clique. We conjecture that, since conflicts are uniformly distributed, every exam will be included in some clique close to the largest clique size. Therefore, identifying the largest clique does not really help. On the other hand, using a clique strategy is very useful on real problems, yielding fewer periods on most instances. This is particularly true in the case of the harder problems. We have thus decided to use this strategy in all tests.

We then examined the effect of backtracking on the solution. In a large number of tests on both real and random problems, we discovered that the use of backtracking reduces the overall length

of the schedule by an average of 50%. In every example, backtracking significantly reduces the required number of periods. This is particularly interesting in view of the fact that most existing examination timetabling algorithms (the older ones) do not use backtracking. Because of the clear superiority of the backtracking strategy for all test problems, this option will be selected in all subsequent computational results.

In the reported results, we have considered the impact of including 'proximity costs'. When costs are included, the problem is called 'Examination Timetabling'. When costs are not used, the problem is equivalent to 'Graph Colouring'. It was interesting to note that in our battery of tests, when we introduced costs without backtracking, the required number of periods increased significantly over the case of no costs without backtracking. In other words, when we did not use backtracking, the introduction of proximity costs made all problems much more difficult. This result was somewhat surprising to us; adding proximity costs does not affect the feasibility of any solution. However, when no proximity costs are used, all feasible periods are equivalent, and the algorithms will all tend to pack exams as early as possible. When proximity costs are used, the algorithms will try to space the exams out evenly. Clearly, the 'spreading' process has a detrimental impact on the solution quality. (The concept of packing exams as early as possible in the schedule really does make a difference to the solution.) On the positive side, we discovered that, when backtracking was employed, the differences between 'cost' and 'no cost' was almost eliminated. (We illustrate this in Tables 2–5.) When no costs are used, the number of periods tends to be slightly smaller across most of the problems. These results suggest that people who implement graph colouring algorithms on timetabling problems without backtracking are probably using many more than the minimum number of periods.

We compare in Tables 2–5 the five sorting criteria and illustrate the impact of proximity costs both on a set of randomly generated problems and on the 13 real problems. For many institutions, the most important criterion is to eliminate (or minimize) the number of student conflicts in a fixed number of periods. Therefore, one important measure of the effectiveness of a strategy is the ability to schedule all exams in as few periods as possible. In Tables 2 and 3, we compare each of the strategies based on how well they can schedule exams into the minimum number of periods. For each strategy and each class of problems, we report the minimum number of periods that were required to find a feasible solution along with the number of backtracks required and the computation time in seconds. For random problems, all reported values are averages over ten instances, while only one instance is solved for each of the real problems. The best results for each problem are shown in boldface characters.

TABLE 2. *Random problems: minimizing the number of periods*

		Examination timetabling (with costs)					Graph colouring (without costs)				
		LD[1]	SD	LWD	LE	RO	LD	SD	LWD	LE	RO
R-200-5	Periods	6.7	**6.3**	6.8	6.6	7.0	6.5	**6.1**	6.6	6.7	6.8
	Backtracks	14.2	6.1	5.5	17.3	18.8	19.2	8.2	11.3	6.2	32.0
	Seconds	3.6	2.7	2.8	3.9	140.1	2.6	1.8	2.0	1.6	155.6
R-200-15	Periods	11.4	**11.2**	11.8	11.4	11.9	11.4	**11.0**	11.2	11.3	11.4
	Backtracks	30.9	16.1	19.7	47.6	35.9	50.5	19.1	54.1	59.6	89.5
	Seconds	29.8	13.8	15.7	31.4	177.5	26.5	11.8	29.0	31.1	247.0
R-200-25	Periods	16.5	**16.2**	16.7	16.5	16.3	16.1	**15.6**	16.0	16.1	16.1
	Backtracks	51.1	18.9	37.2	49.2	110.5	51.0	139.4	63.0	76.6	99.9
	Seconds	78.0	30.9	57.9	85.1	367.8	69.0	169.0	83.5	100.8	340.7
R-400-5	Periods	9.7	**9.5**	9.7	9.8	10.1	9.5	**9.0**	9.5	9.3	10.0
	Backtracks	25.2	3.7	19.3	22.3	47.1	26.0	7.0	19.5	43.6	32.3
	Seconds	20.6	12.2	20.9	20.7	352.0	17.5	9.2	14.6	23.2	346.3
R-400-15	Periods	18.8	**18.2**	18.6	18.7	18.3	18.2	**17.9**	18.0	18.1	**17.9**
	Backtracks	55.4	47.4	54.3	52.2	175.1	52.7	109.9	101.0	86.6	165.2
	Seconds	209.4	511.3	205.1	209.1	1124.2	159.3	298.5	293.5	248.9	897.2
R-400-25	Periods	27.3	**26.3**	27.1	27.4	26.8	26.2	**26.1**	26.3	26.4	26.2
	Backtracks	93.0	284.6	66.5	89.9	246.1	264.7	248.1	239.1	160.0	354.2
	Seconds	1159.8	2230.2	692.4	1069.9	2615.9	2039.7	1649.2	1161.5	1218.8	3422.6

[1] LD: largest degree; SD: saturation degree; LWD: largest weighted degree; LE: largest enrollment; RO: random ordering.

TABLE 3. *Real problems: minimizing the number of periods*

		Examination timetabling (with costs)					Graph colouring (without costs)				
		LD[1]	SD	LWD	LE	RO	LD	SD	LWD	LE	RO
CAR-F-92	Periods	31	**28**	30	31	32	31	**28**	30	31	32
	Backtracks	137	175	211	35	199	104	510	432	6	205
	Seconds	618.3	510.8	911.6	121.2	1143.4	392.5	227.2	1775.6	19.1	996.6
CAR-S-91	Periods	32	**28**	30	32	35	32	**28**	30	32	35
	Backtracks	60	23	255	54	114	1	311	210	107	59
	Seconds	239.8	75.1	1046.4	210.4	712.7	8.9	1080.6	807.9	412.1	265.2
EAR-F-83	Periods	**23**	24	**23**	**23**	24	**22**	**22**	**22**	**22**	**22**
	Backtracks	18	29	38	39	27	100	1	35	56	114
	Seconds	80.2	173.0	156.6	151.8	256.5	391.6	8.7	111.5	184.5	630.4
HEC-S-92	Periods	**17**	**17**	18	**17**	**17**	18	**17**	**17**	**17**	**17**
	Backtracks	44	28	28	249	90	5	0	102	155	39
	Seconds	31.8	17.1	22.8	219.5	83.1	3.2	0.5	66.8	105.0	30.8
KFU-S-93	Periods	20	**19**	**19**	**19**	**19**	**19**	**19**	**19**	20	**19**
	Backtracks	4	1	247	180	137	27	101	124	6	247
	Seconds	120.1	97.2	2396.5	1852.7	2217.0	522.1	1159.6	1156.9	73.6	3361.6
LSE-F-91	Periods	**17**	**17**	**17**	**17**	18	**17**	**17**	**17**	**17**	**17**
	Backtracks	7	14	124	126	45	95	10	25	164	79
	Seconds	78.7	164.6	763.9	786.2	674.5	551.9	78.0	158.8	943.3	848.1
PUR-S-93	Periods	43	**38**	40	40	42	36	**35**	38	38	n.a.
	Backtracks	35	141	24	26	145	111	1	19	20	n.a.
	Seconds	31922.2	73338.3	8782.3	9304.9	72028.5	53899.0	1608.8	10885.7	8703.6	>200000
RYE-F-93	Periods	23	**21**	22	22	22	**21**	**21**	22	22	22
	Backtracks	6	4	10	12	53	245	6	309	18	7
	Seconds	507.2	380.0	419.4	502.5	3467.4	4809.0	343.8	6619.0	516.7	662.8
STA-F-83	Periods	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**	**13**
	Backtracks	1	0	0	0	5	0	0	0	0	2
	Seconds	6.0	6.0	5.7	6.1	98.9	2.7	2.7	2.7	2.8	90.3
TRE-S-92	Periods	22	**21**	**21**	23	22	22	20	20	22	22
	Backtracks	45	9	116	9	65	51	2	63	2	132
	Seconds	398.8	108.4	1067.9	107.4	836.3	468.3	32.8	402.6	29.8	1342.5
UTA-S-93	Periods	**33**	35	35	34	35	33	**32**	33	33	**34**
	Backtracks	51	29	109	97	200	5	2	92	11	89
	Seconds	4294.1	2557.2	9223.0	7340.4	17323.1	549.8	272.3	6878.4	755.6	6539.2
UTE-S-92	Periods	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**	**10**
	Backtracks	3	0	5	7	8	0	0	2	0	4
	Seconds	6.3	4.7	8.5	9.1	9.9	1.6	1.6	2.7	1.6	4.3
YOR-F-83	Periods	21	**19**	20	20	20	20	20	20	**19**	20
	Backtracks	81	60	18	166	72	255	1	378	310	208
	Seconds	649.4	190.4	67.1	516.4	351.5	673.1	6.6	966.5	705.8	740.2

[1] LD: largest degree; SD: saturation degree; LWD: largest weighted degree; LE: largest enrollment; RO: random ordering.

Tables 2 and 3 clearly indicate that the saturation degree strategy (SD) yields the minimum number of periods in most cases. For the real problems, there is often a tie or a close tie between several strategies. However, on closer consideration of our results, it is apparent that some real problems stand out as being much more difficult than others. In particular, three problems (CAR-F-92, CAR-S-91, PUR-S-93) appear much more difficult based on the difference between the number of periods corresponding to the best and the worst strategies in Table 3; in each case, this difference is at least three. For all but one of the ten variations tested on these three problems, SD was the best strategy. For the 'easy' problems, we see that, in several cases, the random ordering is as good as anything else. (We reiterate that this was only true when used in conjunction with backtracking. Without backtracking, the random strategies are uniformly worse than the others.) In addition, as mentioned above, Tables 2 and 3 illustrate that the 'examination timetabling' style problems (with costs) tend to require slightly more periods than the pure 'graph colouring' (no costs) formulations.

For our second set of test results, we were interested in comparing the efficiency (speed) of each strategy, and the relative quality of the answers in terms of the average cost per student. In Tables 2 and 3, the SD strategy typically used fewer periods; therefore, we would expect that it also may require more CPU time, and possibly pack students in a little tighter than, say, the random

TABLE 4. *Random problems: minimizing cost per student*

		Examination timetabling (with costs)					Graph colouring (without costs)				
		LD[1]	SD	LWD	LE	RO	LD	SD	LWD	LE	RO
R-200-5	Cost	36.4	**36.1**	36.9	37.1	39.6	0	0	0	0	0
	Backtracks	3.5	**0.6**	4.1	5.6	18.8	1.9	**0.2**	0.8	2.0	15.1
	Seconds	2.6	**2.3**	2.7	3.0	140.1	1.3	**1.1**	1.4	1.3	139.7
R-200-15	Cost	**19.2**	19.4	19.4	19.9	21.0	0	0	0	0	0
	Backtracks	16.1	**1.6**	8.1	17.2	27.8	7.6	**0.5**	6.4	7.9	31.9
	Seconds	23.0	**5.8**	9.5	15.7	166.7	6.5	**2.8**	5.6	6.2	171.1
R-200-25	Cost	**15.1**	**15.1**	15.4	15.5	16.2	0	0	0	0	0
	Backtracks	15.9	**4.1**	19.0	25.4	36.8	40.2	**8.9**	47.4	70.9	79.3
	Seconds	27.1	**13.0**	32.3	44.7	213.6	54.7	**15.1**	63.6	94.1	295.4
R-400-5	Cost	34.6	**33.4**	34.6	34.8	38.2	0	0	0	0	0
	Backtracks	13.0	**1.0**	10.3	14.4	47.1	1.5	**0.0**	3.3	4.4	32.3
	Seconds	15.8	**11.6**	16.6	49.7	352.0	7.7	**5.5**	8.7	7.6	346.3
R-400-15	Cost	18.7	**18.4**	18.5	18.5	19.8	0	0	0	0	0
	Backtracks	36.3	**8.1**	31.6	36.1	80.0	38.2	**13.9**	44.9	60.3	89.9
	Seconds	157.6	**118.5**	119.5	164.2	748.6	121.2	**50.4**	142.2	180.7	629.2
R-400-25	Cost	**13.5**	13.6	**13.5**	**13.5**	14.0	0	0	0	0	0
	Backtracks	65.7	**7.5**	59.5	73.0	102.3	78.5	**38.1**	138.0	73.5	145.5
	Seconds	800.3	**91.6**	493.7	939.8	1280.2	608.7	**344.0**	1088.9	576.0	1605.0

[1] LD: largest degree; SD: saturation degree; LWD: largest weighted degree; LE: largest enrollment; RO: random ordering.

strategy which spread the exams out further. Therefore, to compare solution quality, we fixed the number of periods for each test problem to be the minimum value for which each strategy was able to find a feasible solution. For example, in Table 3, the CAR-F-92 problem was solved in at most 32 periods by all strategies. By giving all strategies 32 periods, we can compare them on CPU time and cost per student on an equal basis. These results are presented in Tables 4 and 5.

The analysis of the random test problems in Table 4 reveals again the superiority of the SD strategy in terms of both solution quality and computing time. The largest degree (LD) strategy is a close second in terms of solution costs. The random ordering (RO) strategy is by far the worst for all criteria. These results translate to some extent to the real problem results in Table 5. While the salutation degree (SD) criterion is still the best for the number of backtracking steps and for computation time, the largest degree (LD) strategy produces a better solution cost most of the time, followed by the largest enrolment (LE). However, for the three 'difficult' problems identified above, SD is better than LD on all measures: solution quality, backtracking and CPU time. Note that solution costs are lower on high density problems as the number of required periods is larger for these problems, resulting in more speading of exams and in smaller interaction costs.

REAL-LIFE APPLICATIONS

It should be stressed that none of our test problems contained side constraints as we wanted all instances to be as comparable as possible. In practice, however, timetabling restrictions such as those described in the introduction mean that longer schedules should be expected and that some of the comparisons just made may no longer hold. In this section, we provide statistics on a number of real-life instances. Incorporating side constraints in our algorithm is straightforward: whenever an attempt is made to schedule an exam, feasibility with respect to the side constraints has to be checked in addition to potential conflicts.

We summarize in Table 6 the main results obtained on seven of the real-life problems described in Table 1. Most entries of this table are self-explanatory, except perhaps for the following. Two versions of the KFU-S-93 problem were defined, by using different sets of constraints. When KFU first contacted us, they were using an eight-day schedule. They wanted to know if we could find a conflict-free seven day schedule. We could not. In column 4, the value of C gives the number of exams that were assigned *a priori* to a specific period, while the value of D gives the number of exams that had to be scheduled within a restricted set of periods (in the case of CAR-S-91 and

TABLE 5. *Real problems: minimizing cost per student*

		Examination timetabling (with costs)					Graph colouring (without costs)				
		LD[1]	SD	LWD	LE	RO	LD	SD	LWD	LE	RO
CAR-R-92	Cost	7.6	6.6	6.6	**6.2**	8.2	0	0	0	0	0
	Backtracks	134	**0**	170	12	199	46	**0**	37	1	205
	Seconds	637.9	**15.0**	886.0	47.0	1143.4	201.5	**5.4**	1625.2	7.7	996.6
CAR-S-91	Cost	7.9	**7.1**	7.4	7.6	11.9	0	0	0	0	0
	Backtracks	3	**0**	14	14	114	**0**	**0**	**0**	**0**	59
	Seconds	29.2	**20.7**	67.5	58.3	712.7	**6.6**	**6.6**	**6.6**	**6.6**	265.2
EAR-F-83	Cost	**36.4**	46.5	37.3	42.3	42.1	0	0	0	0	0
	Backtracks	**3**	29	18	57	27	**0**	**0**	5	4	77
	Seconds	24.7	**6.0**	76.1	242.4	256.5	24.7	**6.0**	21.2	18.6	492.5
HEC-S-92	Cost	**10.8**	12.7	15.8	15.9	20.0	0	0	0	0	0
	Backtracks	**8**	9	28	9	56	5	**0**	28	9	28
	Seconds	7.4	**6.5**	22.8	8.7	54.9	3.2	**0.5**	17.2	5.6	23.2
KFU-S-93	Cost	**14.0**	15.9	22.1	20.8	21.0	0	0	0	0	0
	Backtracks	**4**	31	8	15	35	**2**	12	9	6	43
	Seconds	**120.2**	559.4	147.3	208.9	977.6	**53.0**	171.4	95.8	73.6	909.7
LSE-F-91	Cost	12.0	12.9	13.1	**10.5**	13.3	0	0	0	0	0
	Backtracks	**2**	**2**	29	2	45	**0**	**0**	**0**	2	2
	Seconds	50.3	50.5	233.5	**48.0**	674.5	14.9	**15.0**	**15.0**	22.8	311.7
PUR-S-93	Cost	4.4	4.1	5.0	**3.9**	n.a.	0	0	0	0	0
	Backtracks	35	28	**17**	34	n.a.	**0**	**0**	**0**	**0**	n.a.
	Seconds	31922.1	22921.9	**10601.1**	21729.4	>200000	1175.1	1178.2	**1172.1**	1175.3	>200000
RYE-F-92	Cost	**7.3**	7.4	10.0	7.7	11.0	0	0	0	0	0
	Backtracks	6	**4**	8	8	50	97	**0**	309	18	7
	Seconds	507.2	403.0	**360.0**	391.9	3019.2	3386.0	**68.1**	6619.0	516.7	662.8
STA-F-83	Cost	162.9	165.7	**161.5**	**161.5**	184.1	0	0	0	0	0
	Backtracks	1	**0**	0	0	5	**0**	**0**	**0**	**0**	2
	Seconds	**6.0**	**6.0**	5.7	6.1	98.9	2.7	2.7	2.7	2.8	90.3
TRE-S-92	Cost	11.0	10.4	9.9	**9.6**	13.0	0	0	0	0	0
	Backtracks	13	**6**	7	9	27	51	**0**	2	2	132
	Seconds	150.6	89.3	**74.1**	107.4	468.4	468.3	**13.5**	29.7	29.8	1342.5
UTA-S-93	Cost	4.5	**3.5**	5.3	4.3	6.4	0	0	0	0	0
	Backtracks	15	**5**	64	28	83	0	**0**	1	2	103
	Seconds	1287.8	**664.3**	5929.7	2705.1	7135.6	**113.0**	113.2	197.8	231.0	8092.5
UTE-S-92	Cost	38.3	31.5	26.7	**25.8**	31.6	0	0	0	0	0
	Backtracks	3	**0**	5	7	8	**0**	**0**	2	0	4
	Seconds	6.3	**4.7**	8.5	9.1	9.9	**1.6**	**1.6**	2.8	**1.6**	43
YOR-F-83	Cost	49.9	44.8	**41.7**	45.1	46.5	0	0	0	0	0
	Backtracks	262	142	78	58	**37**	255	**1**	378	46	208
	Seconds	215.7	428.0	271.4	**174.5**	231.8	673.1	**66**	966.5	123.8	740.2

[1] LD: largest degree; SD: saturation degree; LWD: largest weighted degree; LE: largest enrollment; RO: random ordering.

TABLE 6. *Characteristics of seven real-life problems*

Code	Number of examinations	Number of periods	Assignment constraints (A, B, C, D)[1]	Sequencing constraints		
				No x in y (constraints)	No x in y (objective)	Number of students
CAR-S-91	682	51	(1550, —, —, 191)	3 in 3	2 in 2	1232
					2 in 3	3369
					2 in 4	3797
					2 in 5	2939
					2 in 6	3386
					3 in 4	39
					3 in 5	305
					4 in 5	4
					4 in 6	9
					5 in 7	1
CAR-F-92	543	36	(2000, —, 1, 126)	—	2 in 2	3164
					2 in 3	3394
					2 in 4	4295
					2 in 5	3681
					2 in 6	3536
					3 in 3	33
					3 in 4	467
					4 in 4	1
					4 in 5	6
					5 in 7	0
KFU-S-93a	486	24	(1955, 35, 8, 6)	3 in 3	3 in 4	265
					2 per day	3360
KFU-S-93b	486	21	(1955, 35, —, —)	—	3 in 3	68
					3 in 4	281
					2 per day	4104
PUR-S-93	2419	30	(5000, —, 156, —)	—	2 in 1	106
					2 in 2	18540
					3 in 3	2685
					3 in 4	5216
					3 in 5	6556
TRE-S-92	261	46	(655, —, 4, —)	3 in 3	2 in 2	0
					2 in 3	1589
					2 in 4	233
					2 in 5	1203
					3 in 4	0
UTA-S-93	622	37	(2800, 100, 50, —)	4 in 4	2 in 2	2754
					2 in 3	3600
					2 in 4	3059
					2 in 5	2418
					2 in 6	2764
					3 in 3	42
					3 in 4	296
					3 in 5	374
					4 in 5	11
					4 in 6	18

[1] A: Maximum number of students per period; B: Maximum number of exams per period; C: Number of exams preassigned to one period; D: Number of students preassigned to multiple periods.

CAR-F-92, these were all evening periods). Column 5 gives the number of hard constraints of the type 'no x exams in y periods'. Column 6 enumerates all restrictions of the type 'no x in y' that were incorporated into the objective function. The number of students corresponding to each case in the final timetable is given in column 7.

Most of the problems described in Table 6 took from 5–60 minutes on a PC486 machine. We do not have accurate times since some were run several years ago on a variety of machines and we did not record all CPU times.

Brief comments are in order for some of the problems. At Carleton University, there is a large proportion of evening courses for which the number of allowable periods is limited. When we experimented with different strategies for the real problem, we discovered that the SD strategy without cliques worked a lot better (it produced fewer periods) than the SD strategy with cliques. This was rather unexpected. Then we realized that in this example, the difficult exams to schedule were the evening ones. SD without clique was automatically assigning the evening examinations

first since they had the least number of available periods. If the clique exams are assigned first, the algorithm has trouble solving the evening problem.

At Purdue, the calendar limits the examination week to six days. Five two-hour exams can be scheduled each day. Direct conflicts (2 in 1) are unavoidable. We tried to minimize the number of such occurrences, then the number of occurrences of two consecutive exams in the same day for the same student, and then three exams in the same day. There are also severe room constraints. The University asked us to pack exams in the early days and use historical period placement data as there are complaints if an exam is on the last day several semesters in a row.

These examples and others that we solved indicate that EXAMINE can handle a wide variety of constraints and is a good tool to experiment with various sets of rules and conditions. A number of schools and universities have now adopted this program.

CONCLUSIONS

Examination timetabling is an important problem for schools, colleges and universities. In several institutions, students are given a wide degree of latitude in their course selections and it is becoming more difficult to control the length of the examination timetable. We have shown that the combined use of a backtracking strategy and of a saturation degree sorting rule almost systematically yields shorter schedules in less computing time. This is clearly the case for the random test problems and for the more difficult real problems. The importance of choosing a good algorithmic strategy cannot be overstressed. In the CAR-S-91 problem, for example, the schedule length varies from a low of 28 periods if saturation degree and backtracking are used, to a high of 66 if a random ordering strategy without backtracking is employed. Given the critical importance of keeping semester length to a minimum at Carleton, this university has decided to adopt EXAMINE[1] incorporating the SD and backtracking options. Overall, the algorithm we have developed seems relatively robust and flexible. When applied to real problems, it provides good quality solutions within very reasonable computing times.

APPENDIX

Test data (including side constraints) used in this paper can be obtained from the Internet at ie.utoronto.ca via anonymous ftp. Use 'anonymous' as the login name and your complete email address as the password. All data files reside in the directory/mwc/testprob with the file names of the form *.crs and *.stu. You should also check the README file in this directory for a more detailed description of the various data files and the formats. All files have been compressed to facilitate data transfer. An archived and compressed version of all the data files is combined in the file/mwc/data/all_data.tar.Z. New problems will be added to the directory over time. We encourage readers to send us their own test problems (in our format please) to carter@ie.utoronto.ca.

Acknowledgements—This work was partly supported by the Canadian Natural Sciences and Engineering Research Council under grants OGP0001359 and OGP0039682. This support is gratefully acknowledged. Thanks are also due to two referees for their valuable comments.

REFERENCES

1. M. W. Carter, G. Laporte and J. W. Chinneck (1994) A general examination scheduling system. *Interfaces* **24**(3), 109–120.
2. G. Laporte and S. Desroches (1984) Examination timetabling by computer. *Comps and Opns Res.* **11**, 351–360.
3. D. Johnson (1990) Timetabling university examinations. *J. Opl Res. Soc.* **41**, 39–47.
4. M. W. Carter (1986) A survey of practical applications of examination timetabling problems. *Opns Res.* **34**, 193–202.
5. A. M. Barham and J. B. Westwood (1978) A simple heuristic to facilitate course timetabling. *J. Opl Res. Soc.* **29**, 1055–1060.
6. N. K. Mehta (1981) The application of a graph colouring method to an examination scheduling problem. *Interfaces* **11**(5), 57–64.
7. A. Hertz (1991) Tabu search for large scale timetabling problems. *Eur. J. Opl Res.* **54**, 39–47.

8. F. D. J. DUNSTAN (1976) Sequential colourings of graphs. *Congressus Numerantium XV*, 151–158.
9. D. W. MATULA, G. MARBLE and J. D. ISAACSON (1972) Graph coloring algorithms. In *Graph Theory and Computing* (R. C. READ, Ed.) pp 109–122, Academic Press New York.
10. D. BRELAZ (1979) New methods to color the vertices of a graph. *Communications of the Association for Computing Machinery*, **22**, 251–256.
11. M. W. CARTER and M. GENDREAU (1992). A practical algorithm for finding the largest clique in a graph. Publication CRT-820, Centre de recherche sur les transports, Montreal.

Received April 1994; accepted July 1995 after two revisions