Operations Research

Publication details, including instructions for authors and subscription information:
http://pubsonline.informs.org

OR Practice—A Survey of Practical Applications of Examination Timetabling Algorithms

Michael W. Carter,

A SURVEY OF PRACTICAL APPLICATIONS OF EXAMINATION TIMETABLING ALGORITHMS

MICHAEL W. CARTER
University of Toronto, Toronto, Ontario, Canada

(Received June 1984; revisions received December 1984, June 1985; accepted July 1985)

In examination timetabling, a set of examinations must be assigned to a fixed number of periods so that no student is required to take more than one examination at a time. Each institution will normally impose a list of secondary considerations that define a "good" timetable in its environment. This paper contains a survey of actual applications of timetabling at several universities and a tutorial guide for practitioners on selecting and/or designing an algorithm for their own institutions.

Most educational institutions must schedule a set of examinations at the end of each session or year. In its simplest form, the problem can be defined as assigning a set of examinations to a fixed number of time periods so that no student is required to take more than one examination at any time. A solution of this form is called a "conflict-free" assignment. Over the last 20 years, timetabling has become increasingly difficult in many North American schools, where the trend has been toward a flexible system of electives and programs tailored to individual needs and preferences. At the University of Waterloo, Ontario, for example, some examinations conflict with as many as 350 others out of the total of 552, and the average number of conflicts per course is 50. With this level of integration, the old manual timetabling systems have become unworkable, and most of the larger problems have been computerized to some extent. However, even in its simplest form, in which we try to find *any* conflict-free timetable, the problem is intractable. The solution techniques usually employed are "computer-assisted" manual approaches or relatively simple one-pass heuristics.

It is difficult to draw a clear distinction between the examination timetabling problem and the course timetabling problem. Course timetabling often involves situations in which students have requested a set of courses, and the objective is to *minimize* the total number of conflicts. In course timetabling, periods often overlap and have varying lengths. However, if a practical course timetabling problem requires a conflict-free schedule in uniform time periods, an examination timetabling algorithm would be appropriate. In the examination timetabling problems considered in this paper, courses as well as examinations must be scheduled conflict-free, and examination periods must be nonoverlapping and of uniform size. Although this paper restricts attention to examination timetabling, the author is currently preparing a survey of course timetabling.

In recent years, a number of heuristic algorithms have been developed and used on practical examination timetabling problems. Section 3 of this paper contains a brief chronological survey of successful applications. The reader will observe, however, that each of these routines was developed for some specific problem at a particular school. None of these "packages" have been used by more than one or two sites, and none of their developers compare their results against alternative approaches. In fact, most authors were unaware of the existence of other published material.

As a consequence of this disorganized state of the art, the typical examination scheduler has little or no basis for selecting an algorithm for his or her own institution. This paper attempts to organize and connect the state of the art. The last section presents a few recommendations and guidelines to help schedulers match their particular requirements to a specific (or some composite) algorithm.

Before proceeding with a description of actual applications, we briefly outline the theoretical basis un-

Subject classification 143 examination schedulings, 581 college examination scheduling.

0030-364X/86/3402-0193 $01.25

derlying each approach. As explained in Section 1, the simple timetabling problem is equivalent to the vertex coloring problem in graph theory. The latter problem has been studied extensively, and a wide variety of heuristics is available in the literature. This equivalence is particularly relevant for two reasons. First, most of the timetabling algorithms are based on one of the vertex coloring heuristics; and second, unlike the situation with timetabling literature, several authors have presented computational results comparing the performance of the various vertex coloring algorithms. These results provide a partial vehicle for connecting and comparing timetabling applications.

Section 1 develops the equivalence between the simple timetabling problem and the vertex coloring problem. Section 2 contains a non-technical description of several relevant vertex coloring heuristics and a summary of their computational effectiveness. We have given each algorithm a title suggestive of its operation. Each of the timetabling methods is later described in terms of variations of one of these heuristics.

1. The Vertex Coloring Problem

The problem of finding a conflict-free timetable is structurally similar to the vertex coloring problem studied extensively in the literature on graph theory. For a given examination timetabling problem, a graph is constructed as follows.

(i) Each course is represented by a vertex;
(ii) an edge connects two vertices if the corresponding courses have at least one student in common and, hence, cannot be scheduled in the same time period.

The graph coloring problem is usually posed as a question. Can the vertices of a graph be colored using a set of p colors so that no two vertices connected by an edge are both assigned the same color? The analogy with examination timetabling is completed by associating the p available exam periods with the p "colors."

A closely related problem is to determine the minimum number of colors necessary to color the vertices of a graph, so that no two vertices connected by an edge have the same color. This unique minimum, denoted χ, is called the chromatic number of a graph. The problem of computing the chromatic number of a graph is known, in the terminology of computational complexity theory, to be NP-complete (Karp 1972). This theory, and practical experience, suggests that the CPU time required to compute χ necessarily grows exponentially with the number of vertices. Manvel (1981) has suggested that this property makes the problem intractable for graphs with over 100 vertices.

The implications for examination timetabling depend on the structure of the particular problem. If the number of periods p is much larger than χ, then the problem of assigning p conflict-free periods becomes relatively easy.

Grimmett and McDiarmid (1975) have shown that, at least for large random graphs, the simplest graph coloring heuristic will "almost always" use at most 2χ colors. And yet, as Manvel points out, for carefully constructed examples most heuristics require $k\chi$ colors for any value of k. Despite this result, it is likely that, for a graph coloring in which $p > 2\chi$, most heuristics will be sufficient to find conflict-free schedules. Carter (1983) presents some evidence to indicate that the graphs associated with timetabling problems are, in some sense, easier to solve than more general random graphs.

Practical examination timetabling problems differ from graph coloring problems when the following type of secondary constraints are added on the use of periods:

(i) a limit on the number of students and/or examinations in any one period;
(ii) room capacity constraints;
(iii) consecutive examination constraints (i.e., certain exams must occur in adjacent time periods);
(iv) nonconsecutive conflict constraints (e.g., no examinations in succession for any student);
(v) preassignments (i.e., certain examinations are preassigned to specific periods);
(vi) exclusions and time preferences (i.e., certain examinations are excluded from particular periods);
(vii) each student's examinations should be evenly spread over the examination period.

Our earlier discussion implies that, for a particular school or university, if p is much greater than 2χ, then there is likely to be considerable flexibility in accommodating secondary constraints. If p is close to χ, then finding a conflict-free schedule becomes the primary objective, and secondary constraints will typically be violated. Some schools try only to *minimize* the number of conflicts (Mehta 1981).

For this reason, it becomes difficult to compare the various practical applications. Each author presents data for his/her own school. There are no standard test problems or even standard measures of difficulty. Consequently, there is no opportunity to judge the robustness of a given algorithm on the general problem.

Each of the practical applications, with one exception (Laporte and Desroches 1984), uses a modified graph coloring heuristic. There is an extensive literature comparing these heuristics on random graphs. Therefore, we can gain some insight into timetabling by first describing these basic methods.

2. Graph Coloring Heuristics

Brelaz (1979) and Manvel give comparative results for graph coloring heuristics. Refer to Dunstan (1976), in particular, for an excellent review. We briefly describe those algorithms that have been applied to practical timetabling problems.

(i) "Largest degree first" (Welsh and Powell 1967). The vertices (courses) are ordered by degree (the number of edges adjacent to each vertex). Coloring proceeds by selecting courses from the top of the list and assigning the "lowest numbered" nonconflicting color. The rationale is simply that the vertices with the most edges will be the hardest to color (if we wait until their neighbors have been colored).

(ii) "Largest degree first: fill from top" (Peck and Williams 1966). As before, the vertices are sorted by degree. In this method, we scan the list, placing as many courses as possible in the first time slot (lowest color) and then go back to the top of the list and fill the second period, and so forth.

(iii) "Largest degree first recursive: fill from top" (Carter 1978, Dunstan). This method is similar to (ii) except that as we remove (color) each vertex from the list, we recalculate the degrees in the remaining subgraph and resort the list.

(iv) "Largest modified degree first" (Williams 1974, Dunstan). Williams proposed that the degree was not sufficient to determine how difficult a course was to schedule. He conjectured that a course was critical if a large number of its neighbors were critical. He used the following formula for computing the critical property of a vertex:

$$d_1(v_i) = \sum_{j \in A_i} d_0(v_j) \qquad i = i, \ldots, n,$$

where A_i is the set of vertices adjacent to vertex v_i and $d_0(v_j)$ is the degree of vertex v_j. *Hence* $d_1(v_i)$ is the sum of the degrees of the vertices adjacent to v_i. He normalizes the $d_1(v_i)$ values and then computes

$$d_2(v_i) = \sum_{j \in A_i} d_1(v_j) \qquad i = i, \ldots, n$$

and, recursively,

$$d_{k+1}(v_i) = \sum_{j \in A_i} d_k(v_j) \qquad i = i, \ldots, n.$$

After a while, these "modified degrees" values will stabilize. (One can show that they converge to the principle eigenvector of the vertex adjacency matrix.) Williams demonstrates that this approach is more expensive, but better, than the simple "largest first" heuristic in terms of the number of colors used.

(v) "Largest modified 1-degree first" (Dunstan). Dunstan presented results showing that Williams' first formula, $d_1(v_i)$, was sufficient to beat the simple largest first methods. Moreover, further recursion for $d_k(v_i)$ did not appear to result in significant improvements.

(vi) "Smallest degree last recursive" (Matula, Marble and Isaacson 1972). The rationale is similar to "largest first" methods in that vertices of *lowest* degree are *easy* to color. They are removed from the graph and placed at the *end* of the list. The degrees of the remaining vertices are recalculated. When the list is complete, we then color vertices from the top as before.

(vii) "Smallest degree last recursive with interchange" (Matula, Marble and Isaacson). The interchange rule applies equally to any of the largest first heuristics and proceeds as follows:

(a) The vertex, denoted by c_i, on the top of the list of unassigned courses is assigned to the lowest numbered conflict-free color that has already been used.

(b) If vertex c_i conflicts with all of the current colors, find a color k_j for which there is only one conflicting course c_j. If possible, recolor vertex c_j. Otherwise look for a bichromatic interchange. Specifically, for each color r, locate the set of vertices C_r with color r that conflicts with vertex c_j. If the set C_r does not conflict with vertex c_i or any of the other vertices in color k_j, then interchange vertex c_j with the set C_r, which allows vertex c_i to be assigned color k_j. (Refer to Matula, Marble and Isaacson for further details.) If no such interchange can be found, a new color is created for the course and the algorithm continues with the next course.

Several experiments on random graphs have shown that this heuristic tends to use fewer colors than the other methods and is only slightly more expensive computationally (Dunstan, Matula, Marble and Isaacson, and Mehta).

(viii) "Largest saturation degree first recursive" (Brelaz, Manvel). The uncolored vertices are sorted, first by the number of colors on adjacent colored vertices (called the saturation degree or color degree) and second, by the degree in the uncolored subgraph. Observe that the vertex with the largest saturation degree is also the vertex with the least number of possible colors to choose from. The rationale is simply that the vertex of highest degree with the least number of remaining colors is the most difficult to color and should be colored next. Both Manvel and Brelaz present computational results on large random graphs demonstrating that the approach requires fewer colors then any of the previous heuristic algorithms.

3. Timetabling Applications

This section presents a chronological survey of timetabling applications. Each algorithm is described in terms of its underlying vertex coloring algorithm, the modifications required to handle secondary constraints, and an overview of the implementation results. Many of the papers include suggestions and special considerations applicable in a wide variety of situations. The practitioner will probably find that the best approach is to develop a composite algorithm utilizing a cross section of ideas appropriate to his or her specific needs.

One of the earliest published examples of examination timetabling is presented by Broder (1964). His stated objective is to *minimize* the number of student conflicts. His algorithm is basically a "largest degree first" method; in case of ties, each course in the list is randomly assigned to one of the time periods that creates the fewest number of conflicts. Broder suggests a Monte Carlo simulation approach. The algorithm is run several times with a different random selection to break ties. The best run is selected. He observes that the same algorithm is easily adapted to minimize the number of times a student has: (1) two final examinations in the same day, (2) three final examinations in succession, (3) three final examinations in two days, and (4) four final examinations in three days.

Although his method is relatively simplistic, Broder's use of random selection to break ties is appropriate for most timetabling algorithms. Solutions that appear equivalent mathematically will often differ in many important ways that are not stated explicitly. Decision makers prefer to have a selection of possible schedules from which to choose.

Also in 1964, Cole published an algorithm that had been implemented successfully at Leicester University in England. Cole introduced constraints that allow

(i) certain sets of exams that must be consecutive;
(ii) precedence ordering for some exams;
(iii) space constraints on room sizes;
(iv) certain examinations to be scheduled only in the morning.

For each period, examinations that are required to satisfy any "consecutive" constraints are scheduled first (i.e., if one of a set of consecutive examinations is already scheduled in the previous period). The algorithm then uses the "largest degree first: fill from top" rule except that examinations are bypassed if they violate any of the constraints.

Computational results are presented based on actual data at Leicester University in June 1963. Cole's paper includes a sample of the data for the first year general program: 34 subjects requiring 57 examination periods (i.e., some examinations require two periods). Each subject conflicts with an average of 11 other courses. An implication of these results is that first year general courses do not conflict with any other courses at the university, making the problem separable into a number of small subproblems.

Probably the most important idea in Cole's paper is his use of a bit matrix to represent course conflicts. Element (i, j) is "1" if courses i and j have *any* conflicts and "0" otherwise. He does not store the *number* of student conflicts. Since the conflict matrix is often rather large for practical problems, this technique can be invaluable.

In 1968, Wood devised an examination scheduling algorithm that was implemented at the University of Manchester, England, for more than 1,000 courses, 6,000 students and 30 examination periods. The average number of course conflicts for each examination was 15, producing a conflict matrix density of 1.5%. The interaction between the various faculties was very low. The low density and separability allowed Wood considerable flexibility in establishing his optimization criteria.

Wood's primary concern was that examinations had to be scheduled into a set of designated rooms on campus with limited capacities. For this reason, he sorted the courses according to the size of room required. Within each size group he used the "largest degree first" rule. For each course in the list, he searched for a feasible period with:

(i) no adjacent conflict, or, if none;
(ii) no conflict on the same day (2 examinations per day) or, if none;
(iii) the minimum number of students with another examination on the same day.

In the event of a tie, he computed the total number of unscheduled courses that conflicted with the current course and could feasibly be scheduled into each of the given periods. The period with the minimum interaction was selected. This "look ahead" feature tries to avoid later scheduling problems. He then selected the room with the "closest fit" (i.e., the least acceptable number of places).

Wood claims that, when the algorithm failed, inspection of the conflict pattern related to the unscheduled courses "clearly reveal[s] the subjects which cause the difficulty." These subjects are preassigned manually and the algorithm is repeated. Using some manual intervention, Wood was able to schedule all examinations in 24 periods, even though 30 periods were available.

Wood's algorithm illustrates a unique approach to a common situation. In his problem, the room capacity constraint is more restrictive than the feasibility constraint, which is reflected in his sort sequence. This idea can be generalized to other applications using the following rule: "Courses should always be scheduled beginning with the most difficult course." The scheduler should define "most difficult" in relation to the feasibility of satisfying each constraint. All algorithms of the "largest degree" type make an implicit assumption that finding a conflict-free schedule is the most difficult constraint.

The "look-ahead" feature of Wood's algorithm reflects an important difference between timetabling and coloring. In practical problems, the unscheduled courses may be restricted to particular time periods such as "morning only" or "during the first week." When constraints of this type are present, the "look-ahead" feature in an algorithm is useful for tie-breaking.

In 1978, Barham and Westwood presented an algorithm for timetabling courses under an elective system for the Manchester School of Business. The problem involved a small group of 36 students registering for 22 optional courses to be taken in 40 sessions, with each course requiring between 3 and 6 sessions. Each student selected 5 or 6 courses.

Barham and Westwood's approach is relevant to examination timetabling because they decided to construct a conflict-free schedule based on each student's first four selections. Students then would adjust their fifth and sixth electives, if necessary, after the fact.

Their proposed algorithm was simply a "largest degree first: fill from top" method. The results are interesting as an illustration of the use of a pure coloring algorithm on a difficult problem. For the purpose of comparison, the reader is referred to Tripathy (1980), who presents some results based on the same data using a Lagrangian relaxation algorithm. Lagrangian relaxation has been applied successfully to solving a number of large integer programming style problems. It appears, however, that further research is required on its application to timetabling.

Also in 1978, Carter developed an algorithm for final examination scheduling at the University of Waterloo. The system developed from this algorithm has now been in use for several years, and was recently implemented by the Waterloo County Board of Education for scheduling all area high school examinations. The algorithm basically uses the "largest degree first recursive: fill from top" rule. In the case of ties, a frequent occurrence in the recursive model, preference is given to large enrollment courses and then to certain special courses designated as "preferred" by the faculties. The major constraints were:

(i) several courses must be preassigned to fixed time periods;
(ii) no student should be required to sit for three or more consecutive examinations;
(iii) certain examinations are designated as evening or Saturday only.

Restriction (ii) was not implemented literally, due to the sheer volume of data associated with maintaining and verifying each student's record. Instead, a more restrictive rule disallowed scheduling any examination that had conflicts in the two previous periods.

The University of Waterloo has 17,000 students taking 600 examinations in 36 periods. There are 3 examination periods per day, 6 days per week, for 2 weeks (or 36 periods).

During the fall term in 1981, there were 552 examinations at Waterloo, with a conflict density of 5%. Over 100 courses conflicted with more than 90 other courses, and one course had conflicts with 312 others. There was at least one group of 22 mutually conflicting examinations. Since none of these examinations could be scheduled in any 3 consecutive periods, the schedule required a minimum of 32 examination periods. The situation was actually even more complicated, since many of the courses were constrained to the 16 evening or Saturday periods.

In light of this complexity, it is not surprising that Carter's algorithm has encountered problems scheduling all examinations in the 36 available periods. In some semesters, it has been necessary to relax the "3-in-a-row" constraint for the first few days. Even so, a few examinations are usually left over. These are inserted manually by shifting one or two "blocking" examinations. This manual procedure could be computerized using a constrained version of the "bichromatic interchange" routine (Matula, Marble and Isaacson).

In 1978, Desroches, Laporte and Rousseau presented the algorithm HOREX, an appellation derived from the French "horaire" for timetable. They did not elaborate on the details of their method. Instead, they described the general stages.

Step 1. Find an initial solution consisting of p, the number of periods allowed, sets of nonconflicting examinations using a basic coloring algorithm. They refer to the method of Matula, Marble and Isaacson.

Step 2. Using a minimum matching algorithm, they combine these sets in pairs corresponding to a day (morning/afternoon) with the minimum number of "doubles" (students who must take two examinations in the same day). This problem is solved using a branch and bound code for integer linear programming as described in Land and Powell (1973).

Step 3. Simple moves of one examination at a time are made to further reduce the number of "doubles."

Step 4. Within each pair, examinations are moved to maximize the number of "morning" examinations. Again, the Land and Powell code is used to solve the resulting knapsack problem.

Step 5. The "days" are ordered using a traveling salesman problem heuristic to minimize the number of "successions" in which a student must take examinations on consecutive days. For the traveling salesman problem heuristic, each city corresponds to one day, and distance is defined as the number of successions between each pair of days. The authors employed an algorithm developed by Miliotis (1976).

Step 6. The minimum "tour" is now rotated through each possible starting day to determine the minimum number of successions, ignoring those over weekends and holidays.

The algorithm was implemented at L'École Polytechnique de Montréal; the typical problems had 160 examinations, 12 days (24 periods), a conflict density between 20% and 25%, and a constraint of 800 students per period with a total of 12,000 student examinations.

This approach represents a considerable departure from the more traditional methods. Although no comparative computational results are available, the following comments apply.

(i) The initial assignment requires fewer time periods than the traditional methods, since the procedure virtually ignores all secondary constraints.
(ii) Consequently, the initial solution is very poor from the perspective of the secondary constraints.
(iii) The algorithms required to improve the initial solution are relatively expensive.
(iv) The approach becomes impractical when more than a few courses are preassigned to fixed time periods.

Hence the traditional approaches are preferable, if a feasible solution can be found. If the number of available periods is small, then an approach similar to HOREX is clearly superior.

In 1979, White and Chan published an algorithm that had been in use at the University of Ottawa for several years. Their approach is similar to HOREX and proceeds in phases.

Step 1. Courses are assigned to periods using a variation of the "largest degree first: fill from top" procedure in which the degree is first multiplied by the course enrollment to give the "degree of conflict." One suspects that this modification will have a mixed effect. Large courses will tend to be scheduled early, a highly desirable situation for academic reasons. However, the ordering may require more time periods to schedule all courses.

Step 2. If any courses remain after the pre-determined number of examination periods has been exhausted, the method checks to see if any individual courses can be moved to create an opening. If not, the unscheduled courses are abandoned as unschedulable.

Step 3. As in HOREX, the periods are ordered using a traveling salesman heuristic (Colijn 1979, Chan and White 1978) to find an ordering of the periods that minimizes "second-order conflicts" between adjacent periods.

Step 4. Total second-order conflicts are reduced further by trying to move each course to an alternate, feasible period with fewer conflicting courses in the adjacent periods.

Step 5. In the final stage, the authors attempt to move every pair of courses simultaneously.

In a typical run, scheduling 390 courses in 25 periods for 16,000 students requires 40 minutes of CPU time on an IBM 360/65. The comments presented earlier for HOREX apply equally to this approach. White and Chan claim that preassignments can be accommodated. Presumably, doing so would have a significant impact on the traveling salesman phase.

In 1981, Mehta described an algorithm implemented at Cedar Crest College, Allentown, Pennsylvania. The college problem involved 750 students, 84 examinations and 12 time periods. His stated objective was to minimize the number of conflicts and to minimize the number of occurrences of 3 examinations in a row. The conflict matrix had a density of 29.4%. There was some preslotting (3 courses on Thursday evening and 1 course in the first period).

He used the "largest saturation degree first" algorithm of Brelaz to attempt to color the vertices of the graph using the minimum number of colors. In the given problem, he obtained a 13-color solution and 4 groups of 12 mutually conflicting examinations. He was unable to determine whether or not there was a solution using 12 colors.

He then used a "compression" routine that computed the number of conflicts that would result if all examinations in a period were assigned to the alternate period with the fewest conflicts. He eliminated the period with the lowest "compression cost" and redistributed its examinations.

Mehta's work is significant, since his example appears to be the most difficult of the published applications with respect to the "conflict-free" objective. His successful approach supports earlier results on random graphs (Brelaz, Manvel) that showed that saturation degree methods are the best coloring techniques.

In 1984, Laporte and Desroches described a different approach to the problem. They defined an "aversion cost," p_{il} as the aversion of examination i to period l. (They used $p_{il} = 0$ or 0.75.) They also defined a proximity cost, w_S, which is the penalty for scheduling two conflicting examinations s periods apart. They currently use $w_1 = 16$, $w_2 = 8$, $w_3 = 4$, $w_4 = 2$, $w_5 = 1$, and $w_k = 0$, $k > 5$. As such, they associate a penalty with scheduling exams fewer than 6 periods (2 days) apart. For any schedule the objective function is the sum of these two costs for every occurrence.

Their algorithm proceeds as follows.

Step 1. Take an examination i from the unscheduled set.

Step 2. Find all periods p to which i can feasibly be assigned subject to conflict and room capacity constraints. If none are found, go to Step 4.

Step 3. For each feasible period, compute the increase in the cost function (aversion and proximity costs). Assign examination i to the period with the lowest cost. Go to Step 1.

Step 4. Examination i conflicts with some other examination in every possible period.

- *4a.* Find the periods into which i can be scheduled by rescheduling all examinations that conflict with i. Calculate the net cost of moving the conflicting examinations. Schedule i in the period with the lowest "disruption" cost.
- *4b.* If there are no periods for which the conflicting examinations can be rescheduled without conflict, then count the number of examinations that cannot be rescheduled. Schedule i in the period with the least number of such conflicts. Reschedule as many conflicts as possible, and "bump" the others back onto the list of unscheduled examinations. In order to make the procedure finite, limit (currently 3) the number of times that any examination j can be "bumped" by the *same* examination i. If this limit is reached, examination i is dropped as unschedulable.

This last step can be viewed as a polynomially bounded backtracking mechanism that allows us to "correct earlier errors." It is likely, however, that if a pair of examinations is difficult to schedule, they will bump each other relatively early in the backtracking process.

After the algorithm has finished, Laporte and Desroches try a "simple move" routine to determine if the schedule can be improved by moving any examinations to an alternate period.

Computational results are given with actual data from L'École des Hautes Études Commerciales de Montréal. There were 168 examinations in 5 programs, with some overlapping courses. Each program had from 9 to 62 examinations with conflict densities from 32% to 44%.

This approach represents a dramatic departure from virtually all of the other algorithms. There is no underlying coloring heuristic. The authors take this approach one step further. They experiment with three different strategies for ordering the initial list: largest degree first, largest enrollment first, and random. They claim that all three methods yielded solutions with roughly the same cost.

There are two plausible explanations for this unexpected behavior. First, the limited backtracking feature of the algorithm is sufficient to correct any coloring "errors." Second, for this particular problem, conflicts are not the most serious problem. The latter is definitely a factor, since the highest density program (the M.B.A. program, which contains 49% of the total student population) has 28 examinations to be timetabled into 15 periods, or only 2 examinations per period. This evidence suggests that the method represents an excellent, flexible approach to satisfying secondary constraints when conflict feasibility is not the predominant issue.

In 1982, Romero described a computer assisted participative procedure for examination scheduling used by the Industrial Engineering Department at the Polytechnical University in Madrid, Spain. Essentially he describes a committee composed of administrators, faculty and students who schedule examinations one at a time using a microcomputer. The computer keeps track of classroom availability on each date, notifies the committee of conflicts, and produces summary reports. There is no attempt at optimization.

Romero's approach illustrates the extreme end of the spectrum of computational complexity. One can afford to put the emphasis on subtle political implications of a timetable only when the problem of satisfying all other constraints is essentially trivial.

In 1984, Tripathy described a second attempt to solve the timetabling problem as an integer linear programming problem using Lagrangian relaxation. His specific application involved finding a course schedule for each term of a one-year graduate program. There were about 400–450 subjects per term to be assigned to 30 periods. Since his formulation requires a conflict-free schedule, the approach is relevant to examination timetabling.

When the problem is expressed mathematically, the numbers of variables and constraints become unmanageably large for practical size problems. In an attempt to reduce the problem size, Tripathy identifies groups of students who are in the same stream, or specialization. In his problem, there were 25 streams. He defines 25 "super-students" who are assigned every subject taken by any student in the corresponding stream. Some of these "students" require more than the available number of periods, and they are further subdivided. In his example, he solved the problem using 33 "students."

The value of this operation is debatable. Tripathy actually *creates* conflicts between subjects in order to reduce the number of variables. He defines similar grouping operations for subjects and rooms. Instead of finding approximate solutions to the original problem using a heuristic procedure, he tries to find "good" answers to a similar problem. His approach may fail to find a feasible solution when one exists. Although the idea is interesting, it requires further computational and theoretical research.

One final technique that has been used to solve practical examination timetabling problems is based on the reasonable assumption that the corresponding *course* timetable is already conflict-free. Therefore, if each course has a separate examination, one can construct a conflict-free examination timetable by assigning all courses with the same starting time to the same examination period.

Although the computational benefits are certainly attractive, several drawbacks should be considered. First, there must be at least as many examination periods as there are course start times. This is often undesirable. In addition, the approach does not account for possible secondary constraints. Preassignments, space limitations and consecutive examination limitations can restrict the applicability of the method. For example, three examinations in a row is usually considered to be more serious than three one-hour courses in a row. Indeed, using this approach could have the reverse effect of imposing examination timetabling constraints on the course timetabling problem.

Finally, one suspects that if such an obvious conflict-free examination timetable exists, then considerable improvements could be achieved using an algorithm that directly addresses secondary constraints.

4. Some Recommendations on Selecting an Appropriate Approach

The preceding section described a number of algorithms for solving the timetabling problem. Unfortunately, it is not possible to select an overall "best" method. Evidence certainly suggests that those algorithms that rely on a coloring heuristic might be improved by using a "largest saturation degree" approach; but, aside from this result, the best algorithm is really a function of the structure of the problem at each individual school. The most appropriate algorithm will be a composite of the ideas developed by the various authors. This section offers some suggestions to assist practitioners in deciding which approach is most appropriate to a particular institution.

One of the most important distinctions between the various algorithms involves a classification on a scale between two extreme cases:

(i) those that concentrate exclusively on finding a conflict-free timetable, and
(ii) those that also try to minimize the number of violations of secondary constraints.

Although the latter clearly is preferred, one can afford the luxury of giving high priority to secondary constraints only if there are sufficient time periods available to make conflict-free timetabling relatively easy.

The first step is to estimate the minimum number of periods required for a typical problem. If some academic terms are harder than others to timetable, use actual data from the most difficult term or session. Use one of the coloring heuristics to approximate the minimum number of periods required for a conflict-free schedule, ignoring other constraints. The DSATUR algorithm of Brelaz is recommended for this stage. When this method is applied to practical problems, there are often only a few courses in the last two or three periods.

Suppose the minimum number determined from this step is p^* and the number of periods available is p. Clearly, if p is less than (or approximately equal to) p^*, an algorithm that concentrates almost exclusively on finding a conflict-free solution must be used. Mehta's approach at Cedar Crest College would be a reasonable method.

If p is sufficiently greater than p^*, it may be possible to consider secondary constraints. In some cases, the effect of secondary constraints on the number of periods required is predictable. For example, if there is a constraint that prohibits a student from taking more than one examination per day (out of two periods per day), then strict enforcement of this constraint will probably double the minimum number of required periods. In this case, if p is less than $2p^*$, then it is unlikely that a conflict-free schedule exists that will satisfy the constraints. If p is approximately equal to $2p^*$, one of the heuristics for finding a conflict-free schedule should be used with the modification that each examination must have no conflicts on either side.

In some situations, the effect of adding secondary constraints can be approximated more accurately by incorporating these constraints directly in the conflict graph and the vertex coloring model. For example, there may be only one room on campus with sufficient capacity to accommodate the largest examinations. Since the affected courses cannot be scheduled simultaneously, we can introduce a "pseudo-conflict" between each pair. The resulting vertex coloring solution will satisfy this space constraint and produce a more realistic value of p^*.

Preprocessing the conflict matrix for the purpose of estimating p^* can also be applied to preassigned courses. If two or more courses have been preassigned to the same period, they can be combined into one large course that will have the union of the conflicts of the originals. Every pair of courses preassigned to different periods can have a pseudo-conflict edge inserted between them, since these pairs must not be timetabled in the same period.

Another useful technique for the purpose of analyzing a problem is first to eliminate any examinations that are easy to schedule. The idea is based on the same principle as the "smallest last: recursive" coloring heuristic, and is best illustrated by a specific example. At the University of Waterloo, we had 552 examinations to schedule in 36 periods. We felt that any course that conflicted with 24 or fewer other courses could be considered easy to timetable. Given the secondary constraint of "no set of three conflicting examinations in succession," we could almost always timetable any such courses after the fact, no matter where we place its 24 conflicting courses. By recursively eliminating all courses with fewer than 25 conflicts, we discovered a core group of 121 difficult courses, and we based our analysis on this reduced set. In a truly easy timetabling problem, the reduced set will literally disappear.

This idea can be incorporated explicitly into most of the timetabling algorithms. Easily scheduled examinations are eliminated recursively and one of the standard methods can be used to timetable the reduced set. Upon completion, the eliminated courses are then added back and timetabled in reverse order from which they were removed.

With the help of a few tricks like these, the scheduler can determine just how serious the conflict situation is at his or her own institution. Similar experiments should be performed with respect to the major secondary constraints such as room sizes, spread constraints, or preferences. When the most critical factors have been determined, an algorithm can be designed that is tailored specifically to the particular problem.

Finally, the scheduler should be prepared to experiment with several variations of the algorithm. It is rarely possible to solve a problem on the first attempt.

Acknowledgment

This research was supported by a grant to the author from the National Sciences and Engineering Research Council of Canada.

References

BARHAM, A. M., AND J. B. WESTWOOD. 1978. A Simple Heuristic to Facilitate Course Timetabling. *J. Opnl. Res. Soc.* **29,** 1055–1060.

BRELAZ, D. 1979. New Methods to Color the Vertices of a Graph. *Comm. A.C.M.* **22,** 251–256.

BRODER, S. 1964. Final Examination Scheduling. *Comm. A.C.M.* **7,** 494–498.

CARTER, M. W. 1978. Examination Scheduling: Documentation. Internal Report, Data Processing Department, University of Waterloo.

CARTER, M. W. 1983. A Decomposition Algorithm for Practical Timetabling Problems. Working Paper No. 83-06, Department of Industrial Engineering, University of Toronto.

CHAN, P. W., AND G. M. WHITE. 1978. An Algorithm for Examination Scheduling: Theory and Practice. In *Proceedings of the C.I.P.S., Session 1978,* pp. 244–249. Canadian Computer Conference, Edmonton.

COLE, A. J. 1964. The Preparation of Examination Timetables Using a Small-Store Computer. *Comput. J.* **7,** 117–121.

COLIJN, A. W. 1979. Reduction of Second Order Conflicts in Examination Timetables, as cited in White and Chan (1979).

DESROCHES, S., G. LAPORTE AND J. M. ROUSSEAU. 1978. HOREX: A Computer Program for the Construction of Examination Schedules. *INFOR* **16,** 294–298.

DUNSTAN, F. D. J. 1976. Sequential Colorings of Graphs. In *Proceedings of the 5th British Combined Conference, Aberdeen, 1975, Cong. Num. XV, Utilitas Math.,* pp. 151–158.

GRIMMETT, G. R., AND C. J. H. MCDIARMID. 1975. On Colouring Random Graphs. *Math. Proc. Camb. Phil. Soc.* **77,** 313–324.

KARP, R. M. 1972. Reducibility among Combinatorial Problems. In *Complexity of Computer Computations,* pp. 85–104. Plenum Press, New York.

LAPORTE, G., AND S. DESROCHES. 1984. Examination Timetabling by Computer. *Comput. Opns. Res.* **11,** 351–360.

LAND, S. H., AND S. POWELL. 1973. *FORTRAN Codes for Mathematical Programming.* John Wiley & Sons, New York.

MANVEL, B. 1981. Colouring Large Graphs. In *Proceedings of the 1981 Southeast Conference on Graph Theory, Combinatorics and Computer Science.*

MATULA, D. W., G. MARBLE, AND I. D. ISAACSON. 1972. Graph Coloring Algorithms. In *Graph Theory and Computing,* R. C. Read (ed.). Academic Press, New York.

MEHTA, N. K. 1981. The Application of a Graph Coloring Method to an Examination Scheduling Problem. *Interfaces* **11,** 57–64.

MILIOTIS, P. 1976. Integer Programming Approaches to the Travelling Salesman Problem. *Math. Program.* **19,** 367–378.

PECK, J. E. L., AND M. R. WILLIAMS. 1966. Algorithm 286—Examination Scheduling. *Commun. A.C.M.* **9,** 433–434.

ROMERO, B. P. 1982. Examination Scheduling in a Large Engineering School: A Computer-Assisted Participative Approach. *Interfaces* **12,** 17–24.

TRIPATHY, A. 1980. A Lagrangian Relaxation Approach to Course Timetabling. *J. Opnl. Res. Soc.* **31,** 599–603.

TRIPATHY, A. 1984. School Timetabling—A Case in Large Binary Linear Integer Programming. *Mgmt. Sci.* **30,** 1473–1489.

WELSH, D. J. A., AND M. B. POWELL. 1967. An Upper Bound for the Chromatic Number of a Graph and Its Application to Timetabling Problems. *Comput. J.* **10,** 85–86.

WHITE, G. M., AND P. W. CHAN. 1979. Towards the Construction of Optimal Examination Schedules. *INFOR* **17,** 219–229.

WILLIAMS, M. R. 1974. Heuristic Procedures—(If They Work, Leave Them Alone). *Software Pract. Exp.* **4,** 237–240.

WOOD, D. C. 1968. A System for Computing University Examination Timetables. *Comput. J.* **11,** 41–47.