Surveys

Recent Developments in Practical Examination Timetabling

Michael W. Carter
Dept. of Industrial Engineering
University of Toronto
4 Taddlecreek Road
Toronto, Canada M5S 1A4

E-mail: carter@ie.UToronto.ca

Gilbert Laporte
GERAD
École des Hautes Études
Commerciales de Montréal
5255 avenue Decelles
Montreal, Canada H3T 1V6
E-mail: gilbert@crt.UMontreal.ca

Abstract

In 1986, Carter published a survey of papers on practical examination timetabling, in the intervening years, there have been a number of new applications, and several innovative techniques have been attempted. In this paper, we will classify the algorithms, discuss their reported results and try to draw some conclusions on the state of the art. We have not attempted to perform any experimental comparisons on the different methods.

1 Introduction

Examination timetabling is a difficult combinatorial problem that must normally be faced by educational institutions up to four times each year. There have been hundreds of research papers on the subject and probably thousands of computer programs written (mostly by amateurs at each of these schools) to "solve" their own particular variation on the theme. With all of this effort being expended on the timetabling, it is somewhat surprising that there have been relatively few published papers describing actual successful implementations. The purpose of this paper is to present an overview of such applications. While we have made every possible effort to produce an exhaustive review, we are conscious that some references may have been left out. We would therefore appreciate hearing from anyone who has been overlooked.

Initially, we had intended to include only those papers that had been implemented in a practical application and published; but that would have been a very short survey. We decided to extend our review to include some working papers (which are available from the authors) and papers describing algorithms that had been tested on real timetabling data. In the latter case, it is difficult to determine whether or not the algorithm has subsequently been implemented, but it is somewhat reassuring to see realistic results.

In 1986, Carter, published a fairly comprehensive survey of the literature on practical applications of examination timetabling. Our intention here is to update that often-cited survey with some of the more recent developments in the area. Therefore, we have not repeated the discussion of papers included in the previous review. A few older papers have been included here that were omitted from the original survey.

2 The Examination Timetabling Problem

The basic problem is to assign examinations to a limited number of available time periods in such a way that there are no *conflicts* or *clashes*. i.e., no student is required to write two examinations at the same time. In some cases, this constraint is relaxed, and the objective is to *minimize* the number of student conflicts. The following represents some of the more common side constraints that are observed in practice:

1) Certain groups of exams may be required to take place at the same time (e.g., some common examination questions);

2) Precedence constraints between exams restrict the ordering;

3) Certain exams are required to be in consecutive periods;

4) Certain exams are required to be on the same day;

5) Rules of the form "No student can write x examinations in any y consecutive periods";

6) Certain periods are excluded for an examination (e.g., an exam may be restricted to "evening periods only");

7) Certain (groups of) students are restricted to a subset of the available periods (note that this is a simple extension of excluding exams from certain periods);

8) There is a limit on the total number of students who can write examinations at the same time (total seats);

9) There is a limit on the number of examinations that can occur at the same time (e.g., number of rooms or invigilators).

There may also be a variety of constraints with respect to resources and examiners associated with each examination:

1) Limited number/size of rooms available;
   - Multiple exams in one location
   - Exams may be split over several geographically similar locations
   - Exams may have specific room/building preferences
   - Exams may be preassigned or excluded from certain rooms

2) Special resource requirements (large desks, TV, computers, etc.);

3) Schedule examiners (no examiner should have two exams on the same day; or examiners with two exams should have them in nearby locations);

4) Assign invigilators to exam rooms (minimize number of invigilators depending on the actual number of students in the room);

5) For special students, need to specify exam spacing, length, location, and special resources.

As described above, the primary objective is to find a conflict free timetable (or minimize the number of conflicts). However, there are a variety of secondary objectives:

1) Minimize the number of occurrences of x examinations in any y consecutive periods for all students;

2) Spread examinations out as evenly as possible (for each student);

3) Try to schedule all examinations as early as possible. Note that criterion 3) is directly contrary to objectives 1) and 2);

4) Retain the optimized timetable from the previous year as much as possible. This last criterion is not very appropriate unless there is very little change in student demand patterns from one year to the next. (Some schools want the opposite: do not put the same exam late every year);

5) Maximize flexibility: maximize the number of exams that can be moved to another time period without disrupting the rest of the schedule.

3 Algorithmic Approaches

In Tables 1 and 2, we have summarized a number of papers published in the past ten years. Not surprisingly, there are no exact algorithms being used. The papers can be divided roughly into four types of approaches which we have labelled Cluster, Sequential, Generalized Search and Constraint Based. We also briefly discuss the possibility of a global optimisation solution.

3.1 Cluster Methods

In these methods, the set of examinations are first combined into blocks corresponding to a set of compatible examinations, e.g., this group can feasibly be assigned to some period with no (or only few) conflicts. In a second phase, the clusters are sequenced into specific periods to minimize some objective or satisfy some constraints. The first published application paper describing an application of this approach was by White & Chan [1979] (discussed in Carter [1986]).

Table 1: Problem Descriptions

Reference	Institution (Exams, Students, Periods)	Problem Description
Foxley & Lockyer(1968)	Nottingham University 1967: (651,15000,23)	Conflict-free; fixed spread between exams; user priorities.
Fisher & Shier (1983)	Clemson University (—,10875,18)	Minimize occurrence of consecutive exams (2 in 2)
Leong & Yeong (1987,1990)	National University of Singapore (800,16000,44)	Conflict-free, and minimize occurrence of (2 in 2)
Johnson (1990)	University of the South Pacific (200,2350,20)	Conflict-free; minimize occurrence of (2 in 2) with capacity constraints
Lotfi & Cerveny (1991)	SUNY at Buffalo (858,11331,15)	Minimize conflicts, 2 in 2, and 2 per day
Hertz (1991)	Swiss Federal Institute of Technology in Zürich (1851,4529,—)	Minimize 2 in 2, 2 in 4
Balakrishnan, Lucena & Wong (1992)	Purdue University (3569,—,30)	Minimize conflicts plus capacity constraints
Corne, Fang & Melish (1993)	University of Edinburgh, Dept. of Artificial Intelligence 1991: (38,60,28), 1992: (44,93,28)	Minimize conflicts and second and third order conflicts (2 in 2 and 2 in 3)
Nuijten, Kunnen, Aarts, Dignum (1994)	Eindoven University of Technology (275,7000,33)	Conflict-free; mandatory spread between exams within each program
Bozumault, Delon & Peridy (1994)	l'Université Catholique de l'Ouest, Angers, France (308, 4000,33)	Conflict-free, time windows, consecutives, fixed spacing (some exams), precedence.
Thompson & Dowsland (1994,1995)	Swansea University 1992: (600,3000,24)	Conflict-free; consecutives; minimize no. of students with consecutive exams.
Carter, Laporte & Chinneck (1994)	University of Toronto Faculty of Engineering (200,2400,20), and Carleton University (682,16925,34)	No x in y (any $x \leq 3$), or, Minimize x in y (any $x \leq 3$) plus capacity, and preference constraints
Carter, Laporte & Lee (1995)	Eleven institutions (81 to 2419, 611 to 30032, 20 to 51)	No x in y (any $x \leq 3$), or, Minimize x in y (any $x \leq 3$) plus capacity, and preference constraints
Leong (1995)	National University of Singapore (1000,16000,44)	Conflict-free.
Ergül (1995)	Middle East Technical University (1438,16000,39)	Minimize conflicts, 2 in 2, 2 in 3, 2 in 4; evenly spread required (must) course exams.

Table 2: Results on Practical Problems

Reference	Algorithm	Results
Foxley & Lockyer (1968)	Sequential Approach	Implemented and used since 1967
Fisher & Shier (1983)	Cluster Approach with TSP	Tested on real data but not shown to be implemented
Leong & Yeong (1987,1990)	Cluster Approach with QAP	Tested on real data; probably not implemented (see Leong (1995))
Johnson (1990)	Cluster Approach with Simulated Annealing	Implemented "very successful"
Lotfi & Cerveny (1991)	Cluster Approach with QAP & TSP	Implemented
Hertz (1991)	Tabu Search	Tested on real data but not shown to be implemented
Balakrishnan, Lucena & Wong (1992)	Cluster Approach with TSP	Tested on real data but not implemented
Corne, Fang & Melish (1993)	Genetic Algorithm	Implemented
Nuuten, Kunnen, Aarts, Dignum (1994)	Constraint Satisfaction	Tested on real data
Boizumault, Delon & Peridy (1994)	Constraint Logic Programming	Tested on real data
Thompson & Dowsland (1994,1995)	Simulated Annealing	Implemented since 1993
Carter, Laporte & Chinneck (1994)	Sequential with Backtracking	Implemented
Carter, Laporte & Lee (1995)	Sequential with Backtracking	Implemented in three, tested in eight institutions
Leong (1995)	Constraint Satisfaction ILOG	Implemented for 1994/95 year
Ergül (1995)	Genetic Algorithm	Implemented for Spring '95

There are a variety of approaches used to address the cluster phase. White & Chan create conflict free groups by selecting exams, one at a time, beginning with the largest ones first. Leong & Yeong [1990], do the same except that they select each exam in descending order of number of conflicts. This exam is added to a group with which it has no conflicts and has the largest number of conflicting exams in common. i.e., they try to minimize the number of new group conflicts that will be created.

A similar approach is used by Lotfi & Cerveny [1991]. They sort the exams by the degree times the number of student conflicts with all other exams. Exams are assigned to the first available conflict free block (with no instructor conflict and sufficient remaining capacity). If there are no conflict free blocks, the exam is assigned to the

block with the minimum number of conflicts (subject to instructor and capacity constraints). (They do not explain what they do if this fails.) They then apply simple interchange rules to improve the assignment (from Arani & Lotfi [1989]).

Johnson [1990] uses a linear combination of two criteria: the size of the exam (enrolment) and the number of conflicts with other courses. By varying the relative weights, a wide variety of different schedules will be produced. Johnson's algorithm constructs a sequence of solutions, discards the infeasible ones, and saves several of the best ones for the user to choose from. He also assigns rooms and verifies capacity constraints on each period as he constructs the groups.

In the algorithm of Fisher & Shier [1983] conflict free groups are constructed by using the original *course* timetable. i.e., two exams belong to the same group if their corresponding lectures occurred on overlapping time periods. At Clemson University, course lecture patterns appear to be very regular; they describe a pattern consisting of all courses offered at 9:00 on any day. Apparently, these patterns do not overlap with any courses offered at 10:00. In many schools, this grouping would not work. Since lectures are 1, 2 or 3 hours long, combining courses that overlap would often produce a single group!

Once the examinations have been divided into compatible groups, there are several different methods used to sequence the groups. The simplest approach uses the objective of trying to minimize the number of students who have two exams in a row where the last examination on day t is considered *adjacent* to the first exam on day $t+1$. This problem can be modelled as a "Travelling Salesman Problem" (TSP) where each cluster is a city, and the "cost" to travel from cluster i to cluster j is equal to the number of students who must write an examination in *both* group i and in group j. Finding a *tour* that visits each city at minimum cost is equivalent to finding a *sequence* for the groups that minimizes the number of students with consecutive exams (see White & Chan [1979]). (A dummy city (group) with no students must be added to the tour so that our final sequence, when we remove the dummy city, is a linear ordering as opposed to a closed tour.)

Initially, Fisher & Shier sort the groups based on the number of students in each group, σ_i. They arrange the periods in the order: $\sigma_n\ \sigma_1\ \sigma_{n-2}\ \sigma_3\ \ldots\ \sigma_4\ \sigma_{n-3}\ \sigma_2\ \sigma_{n-1}$. If the students were uniformly randomly distributed, this sequence would minimize total 2 in 2 conflicts. They then use a simple pairwise interchange heuristic to solve the TSP.

A variation of this problem is studied by Balakrishnan, Lucena & Wong [1992]. Here, the last exam on day t is *not* considered adjacent to the first exam on day $t+1$. They model the problem as a "time dependent" TSP in the sense that adjacent groups over a day boundary have no cost. They use a network model with a Lagrangian penalty function and report very good results on large realistic problems; they produce upper and lower bounds that are always under 6%. They assume that the exams have already been divided into groups.

As pointed out by Arani, Karwan & Lotfi [1988], when there are exactly two periods per day, and no cost over a day boundary, then the problem can be solved as a minimum weighted matching problem, dividing the groups up into compatible pairs. In Johnson's application, there are two periods per day, over 10 days, and the objective at this stage was to minimize the number of same day conflicts. However, he chose to solve the problem using simulated annealing. He then solves the final step of the problem of sequencing these "days" based on the criterion that the day with the most "large" exams should be scheduled first; and all days are ordered using this rule.

The group sequencing problem can also be modelled as a Quadratic Assignment Problem (QAP) where the objective is to minimize the number of students who have two examinations in any y consecutive periods. The most common choice for y is 2 or 3. Leong & Yeong [1990] consider the objective:

$$\textit{minimize } \Sigma_i \Sigma_j \, q_{ij} \, c_{p(i)p(j)}$$

where

q_{ij} is the degree of interaction between lists i and j
c_{rs} is the cost of time spacing between sessions (periods) r and s, and
$p(i)$ is the session (period) that examination i has been assigned to.

For example, q_{ij} represents the number of students who have an exam in group i and in group j, and c_{rs} represents the "cost" associated with a student having exams in period r and period s. Leong & Yeong argue that, since the QAP approach uses costs associated with specific periods, it is possible to take advantage of natural breaks like weekends and holidays (where "consecutive exams" have no cost), while this is not easily done in the TSP. They used costs of 1000 for 2 in 2, 100 for 2 in 3, and 10 for 2 in 4. They construct an initial ordering randomly, and search for improving pairwise exchanges.

Lotfi & Cerveny [1991] include an additional stage in this type of approach; they first cluster the exams into "low conflict" groups. They then organize these groups into *days* using a simple QAP heuristic where they have three exam periods per day. In a third stage, they take the period each day with the least conflicts to the other two and put it in the middle, and then use a simple TSP heuristic to sequence the *days* based on the number of conflicts between the first and last period of each pair of days.

In general, the cluster approaches represent a form of decomposition of the problem. By combining exams into low conflict groups first, the size of sequencing problem is reduced which, theoretically, allows the use of more sophisticated optimization techniques on the period sequencing problem. This must necessarily detract from the potential quality of the final solution since the search space is drastically reduced. Moreover, in practical applications, people seem to be using fairly simple

heuristics to solve the decomposed problem. It is not clear to us that the approach is beneficial. A further disadvantage of these methods is that it becomes virtually impossible to implement rules such as "No student should have three exams in any four periods". Any reference back to individual exams or students has been lost. At the same time, Johnson [1990] reports success on a fairly small problem; and Lotfi & Cerveny [1991] report that their algorithm has been successfully implemented on a fairly large instance.

3.2 Sequential Methods

In sequential methods, examinations are assigned to a specific period one at a time, chosen using some *sequencing strategy*. There exist a wide variety of approaches distinguished by the order for selecting the next exam, and the way the period is chosen. The period may be chosen as the earliest feasible period (in an attempt to minimize the total number of periods required), or it may be the best feasible period (where "best" is a measure of the objectives). These algorithms typically employ a two phase approach: a construction phase produces an initial timetable, and an improvement phase makes modifications and improvements. (Note that some of the papers described earlier used sequential methods to create an initial set of conflict-free blocks.)

Several earlier papers describe sequential methods, (cf. Barham & Westwood [1978], Broder [1964, 1966, 1968], Cole [1964], Desroches, Laporte & Rousseau [1978], Laporte & Desroches[1984], Mehta [1981,1982]) which are discussed in Carter [1986].

The following list illustrates some of the sequencing strategies that have been employed for the construction phase:

1) *Largest degree*: largest number of conflicting examinations. i.e., exams which conflict with many other exams should be scheduled early.

2) *Saturation degree*: number of *periods* in conflict. i.e., the exam with the least number of available periods should be selected next. This is a dynamic selection rule. Ties are broken using largest degree.

3) *Largest weighted degree*: total number of students in conflict with other exams.

4) *Largest enrollment*: select the exam with the largest number of students.

5) *Largest number of papers*: some exams may require multiple papers to be written in different, possibly consecutive, exam periods.

6) *User defined priority groups*: select high priority examinations first (e.g., exams for final year students may have high priority).
7) *Random ordering*: select examinations in random order (or select them in the order they appear in the input stream).

For almost 30 years, Nottingham University has been using an algorithm developed by Foxley & Lockyer [1968] based on Cole's [1964] method. The approach is a simple construction algorithm that fills the periods, one at a time from the beginning. There is no automatic improvement phase (although there is allowance for manual adjustments after the schedule has been generated). The structure of the procedure allows the authors to easily accommodate conflicts, consecutive exams, required minimum exam spacing and user defined exam priorities. They employ a flexible exam ordering and priority routine to allow the user to construct several different schedules and then choose the best.

Carter, Laporte & Chinneck [1994] include sequence strategies 1-4 and 7 as user selected options. Some strategies work better on some problems. In general, the "saturation degree" strategy proved the most robust, but occasionally, other strategies worked better. For a comparison of the strategies, refer to Carter, Laporte & Lee [1995]. They also include "random tie breaking" so that the algorithm can be run several times to generate a selection of timetables for the scheduler to choose from. A "limited backtracking" approach is used to try to ensure that the initial solution is feasible.

Typically, the improvement phase involves performing "k-opt" interchanges looking for improvements. For example, 1-opt checks if any single exam can move to a better period; 2-opt looks for interchanging or reassigning pairs of exams. Carter, Laporte & Chinneck use 1-opt and 2-opt to improve their final solution.

It is interesting to note that most of the algorithms use a *Conflict Matrix* as the internal method of representing the problem. Element i, j of the matrix represents the *number* of students who must write both exams i and j. This representation allows the implementation of rules like "minimize the number of students who have 2 exams in any y consecutive periods; they do not allow the evaluation of rules like "minimize the number of students who have 3 (or more) exams in y periods". In order to handle these types of measures, we need to retain the examination list for each student. Carter, Laporte & Chinneck do work directly with the list of exams by student, and allow rules of the form "no x in y" or "minimize x in y" for $x = 2, 3$ or 4, and any y.

3.3 Generalized Search Strategies

We use the term *Generalized Search* to describe the class of algorithms that begin with one or more initial solutions, and employs a search strategy that tries to avoid getting stuck in local solution areas, and investigates a more global region of the space. Examples of such approaches include simulated annealing, tabu search and genetic algorithms. These techniques all share the property that, if they are run long enough, they will almost certainly discover the "optimal" solution. They also share the characteristic that, for most problems, it is not practical to really run them "long enough", but they are still interesting as heuristic methods. In some respects, they are similar to sequential methods; they use a (usually simple) sorting criteria to construct one or more initial

timetables, and then apply a search procedure to investigate a variety of related solutions. In these algorithms, initial feasibility is not critical; they prefer to get started quickly, so backtracking is not used. The majority of the computation time will be spent in the improvement phase.

Simulated annealing has been applied to a wide range of problems. We already mentioned that Johnson [1990] used simulated annealing to solve one part of his "cluster" approach. The basic concept is that one tries to find a feasible solution fairly quickly, and then randomly, and iteratively, selects a "neighbourhood" solution. For example, one could define a neighbourhood as the set of all solutions with any single exam moved to a new period. If the new solution improves the objective function, it is accepted. If it is not an improvement, it is accepted with some probability. Initially, the probability is high so that most solutions are accepted. As the algorithm proceeds, the probability is slowly lowered (analogous to lowering the temperature in an annealing process) which eventually forces the solution into a local minimum. The temperature can then be increased again, and the procedure repeated until a new local minimum is discovered. The algorithm keeps track of the best solution(s) encountered. When some stopping criteria is met (total time or no improvement for a certain time), the best solution is reported.

Thompson & Dowsland [1995a, 1995b, 1996] used a two phase approach. In the first phase, they only consider the more important, "binding" criteria (conflict-free, limited number of students per period and preassignments). Assuming that they can find a feasible solution, they optimize secondary constraints in the second phase (prefer large exams early, minimize students writing consecutive exams), while the neighbourhood search is restricted to maintaining feasibility of the binding constraints. By restricting the search space in this way, they may disconnect it and thereby lose the "global search" property of simulated annealing. They attempt to counter this behaviour by borrowing some "diversification" ideas from tabu search and constructing several runs investigating a variety of local solutions. They also employ "Kempe chains" which consist of two subsets of exams in time periods i and j where the two sets of assignments can be feasibly interchanged. This increases the size of the neighbourhood. They construct solutions in under an hour on a DEC alpha computer (equivalent to about 18.5 hours on a 486 PC according to the authors) although this can be reduced by using a faster cooling schedule with a corresponding reduction in solution quality. The system has been used successfully at Swansea since 1993.

In simple terms, tabu search is similar to simulated annealing in the sense that neighbourhood moves are used to move out of a local optimum. However, instead of choosing randomly, the method maintains a list of "tabu moves". These represent solutions that have been visited before, and the algorithm is forced to look elsewhere. Subject to this constraint, the algorithm looks for the best improving move it can find without going backwards. The new move is accepted, even if it leads to a worse solution.

Tabu search has been applied to both course and examination timetabling problems by Hertz [1991]. He does not state exactly how he generated an initial solution; but one can assume that he used one of the simple ordering strategies. For exam timetabling, the neighbourhood is defined as the set of schedules obtained by moving a single exam to a new period. Since this set is quite large, he restricts it by considering a random subset; he chooses $\frac{1}{2}n$ potential moves where n is the number of exams. Each selection consists of randomly picking an exam which violates a constraint, and randomly choosing a new period. The algorithm selects the best of these as the new solution. The tabu list "of all previous solutions" is also too large to save directly. Therefore, a list of the last seven moves is saved as pairs $\{x, i\}$ where x is an exam and i represents the previous period; he does not allow any shift that would move course x back to period i. The number seven is a program parameter. This type of restriction is typical of tabu search algorithms.

Boufflet and Negre [1996] have applied Tabu search to the examination timetabling problem at the University of Technology of Compiègne in France. Their problem involved up to 130 exams in 20 periods.

Genetic algorithms derive their name from the underlying process; they attempt to imitate the genetic function of combining chromosomes to produce new "children". Initially, a large "population" of schedules is constructed (using a simple sequential method). Although there are a variety of coding schemes, the simplest way to represent an examination timetable is in the form of a vector (the chromosome) of length n (the number of exams) where the i-th entry (gene) of the vector states which period the exam is assigned to. Two different solutions (parents) can be combined to produce two new solutions (children) by randomly selecting a point to split the two vectors into a front and back part. The front of the "father" is merged with the back of the "mother" to produce one child, and vice versa to produce a second child. Both children are new feasible schedules in the sense that every exam is assigned to a unique period. Some algorithms will also include a feature to try to make the children feasible using simple moves. The hope is that perhaps good parts of the father will match with good parts of the mother to produce an improved child. The new solutions are evaluated in a "fitness" function which evaluates how good the schedule is with respect to the criteria. Corne, Fang & Mellish [1993] use a population of size 50. Each one is evaluated, and then 25 pairs are randomly selected (with a bias toward the fittest solutions). These 50 parents produce the next generation of 50 children, and they repeat the procedure for 300 generations. They repeat the whole process ten times and select the best schedule from all runs.

There are many different coding schemes, different strategies for producing children, a process called "mutation" where individual genes are randomly selected to be randomly moved, and "elitism" where the best "parents" are always selected. Corne, Fang and Mellish [1993] used a fairly simple strategy on the exam timetable for the A.I. Dept. at the University of Edinburgh. The evaluation function included conflicting exams (weight 30), more than two exams in a day (weight 10), two consecutive exams (weight3) and two exams just before and after lunch on the same day (weight 1). They

achieved far better solutions compared to those that had been produced manually in 1991/92. One student had consecutive exams (score 3), compared with the manual score of 135 from 33 consecutives, 16 before & after lunch, and two students with three exams on one day. Their approach was implemented in 1992/93. Corne and Ross [1996] looked at alternate initialization strategies using the same data sets.

In 1995, Middle East Technical University (METU) in Ankara, Turkey used a genetic algorithm approach called GUNES developed by Ergül [1996]. The *fitness* (objective) function included costs associated with conflicts and attempting to spread exams out for students. It also gave a higher weight to the spread between "must" (required) course exams. The author used a surrogate for "must" exams as all those exams with a *low* conflict density. Exams with a high conflict density are assumed to be electives. One assumes that elective courses can be taken by a wider variety of students than must courses. The schedule produced by the program was accepted in the Spring '95 exams after 46 exams were moved to new slots. This implies, perhaps, that the results are good, but the fitness function still requires work.

Burke et al [1996] have used a memetic algorithm for examination timetabling and tested it on a number of real problems including the University of Nottingham. A memetic algorithm is similar to a genetic algorithm except that a local improvement heuristic is applied to each child so that every member of the population represents a local optimal solution. This will dramatically reduce the population size; however, it also requires considerable additional time for each iteration. The authors presented evidence of the value of their approach; but they did not compare directly with a genetic approach.

All of these techniques are relatively recent innovations in the area of heuristic search. If one only considers problem size, all of these techniques have been applied successfully on some rather large example problems. This looks like a very promising area for future research. Of course, all of these search algorithms require considerable computer time and/or horse power. It is difficult to draw more accurate conclusions without comparisons with other approaches on identical problems to see if the additional effort produces better quality solutions.

3.4 Constraint Based Approaches

There are a number a papers from the AI (Artificial Intelligence) research community that use general systems for constraint representation. They are described as Constraint Satisfaction Problems or Constraint Logic Programming. Most of the constraint programming literature models a problem as a set of variables with a finite domain, to which values must be assigned so as to satisfy a number of constraints. However, the approaches can also be presented based on the concept that a wide variety of optimization problems can be described as a set of "activities" (exams) which require "resources" (rooms and/or periods) and there are "constraints" which apply to groups (pairs) of activities, and constraints that restrict the use and/or timing on resources. Examples of constraints include two exams that cannot be assigned to the same period;

two exams that must be consecutive, rooms that have seating limits, etc. i.e., most of the exam scheduling rules can be translated into constraints. Typically, these systems employ fairly simple rules for assigning exams to periods. They will not allow infeasible assignments, and when they get stuck, they use backtracking until they find a feasible solution that satisfies all of the constraints. They do not explicitly minimize or maximize an objective function. The beauty of these systems is that they are not designed for exam scheduling. They are general solvers that simply interpret constraints. The user enters a list of constraints to describe their particular application. (In practice, the search strategy or heuristics that are used to guide the search are often specific to the application.)

A general constraint satisfaction technique has been applied to the problem of examination timetabling at the Eindoven University of Technology, Netherlands by Nuijten, Kunnen, Aarts and Dignum [1994]. (They tested their approach on real data, but there is no indication that it has been implemented.) The problem is fairly small with 275 examinations, 7,000 students and 33 exam periods over three weeks. They have constraints on the number of students per period, "time windows" on when the exams can take place, conflict constraints, and mandatory spread constraints (between exams in exam groups corresponding to programs). They had two variations on their algorithm, and both required about 3,000 constraints. The random selection algorithm consists of assigning as many exams as possible to the first period using random selection, and then continuing with the second period. If no feasible solution is found, they try backtracking. If backtracking is not successful after a while, they try a complete restart in the expectation that random selection will provide a different path. They ran several different tests varying the capacity of the exam room to test their approach. On average, the successful runs required between two and twelve minutes of CPU time on a SUN SPARC station ELC.

Constraint logic programming (CLP) uses techniques inherited from Constraint Satisfaction Problems (CSP). In simple terms, the difference is that CSP methods ask the user to specify their constraints, while CLP is an actual programming language, so that the user can do much more customization in the solution design. Presumably, the added flexibility also increases the development time required. Boizumault, Delon & Péridy [1994] describe using Constraint Logic Programming for l'Université Catholique de l'Ouest in Angers, France. In addition to the usual rules (conflicting exams, preassignments, consecutive exams, precedence constraints), they include room assignment where multiple exams may be assigned to each of seven rooms. i.e., a bin packing problem. They tested there model on the 308 examinations which produced a set of 2600 constraints. Of these, the most frequent were 1930 exam conflict constraints. Their search procedure used a "Best Fit Decreasing" (bin packing) strategy. They solved the real problem in under one minute of CPU time (although they did not mention the processor).

Although there are no formal papers presenting the problem or the results, there is an article that appeared in Computerworld (Leong [1995]) describing the use of a commercial package called the ILOG SOLVER and ILOG SCHEDULE. ILOG is a

French company that has become very successful in the development of constraint based object oriented tools. The article gives an overview of an implementation of examination timetabling at the National University of Singapore where one of the systems analysts, Goh Guay Im, in the University Computer Centre, developed and implemented a model in about one month. Unlike the previous two examples, this problem was fairly large with over 1,000 examinations. However, there is an indication in the article that the problem may have been decomposed, and each faculty solved separately. There was no indication of CPU solve times. The article was primarily an interview with Ms. Goh. The system has been used for two exam schedules in 1994-95. The ILOG system allows the user to model one or more objective functions. As the branch and bound search progresses, the program will locate feasible solutions. The system can then add a constraint that only allows solutions with a lower objective function value. In the case of multiple objectives, the user can interact with the system to determine which objectives should be restricted.

All of the papers that we considered for constraint based methods have been used to solve fairly small problems (around 300 examinations). We are not sure what size problem the ILOG solver has tackled, but it is likely around the same size. In constraint methods, the number of constraints grows quite quickly as problem size increases (at least as the square), so it would appear that they may have trouble with large examples in the range of 1,000 exams. Obviously, this limitation will be reduced over time.

3.5 Global Optimization Approaches

Of course, it would be marvellous if someone could develop a globally optimal algorithm for examination timetabling. Although there have been some research papers on this topic, there is only one paper that we are aware of where the author, Tripathy [1980, 1984], has attempted to solve the problem on real data. The approach is described in Carter [1986]. It is probably fair to say that the algorithm is not a feasible way to solve typical practical problems.

4 Room Assignment

Several of the algorithms (Leong & Yeong [1990], Carter, Laporte & Chinneck [1995], Lotfi & Cerveny [1991], Boizumault, Delon & Péridy [1995]) include a routine for assigning examinations to rooms after the exams have been assigned to periods. All of them are similar. Lotfi & Cerveny describe their approach as follows. For each examination period, sort the available rooms in descending order by capacity; sort the examinations in descending order by number of participants. The largest exam is assigned to the smallest room with sufficient capacity to hold the students. If no room is large enough, then the largest room is filled, and remaining students are assigned to a different room.

As mentioned earlier, Johnson assigns rooms as part of the timetable construction. In producing the initial clusters, he first checks if there is an available room for the exam in that cluster, and then assigns the exam to the smallest room which is large enough.

Carter [1995] has recently added a room assignment module to the EXAMINE program that allows a wide variety of additional criteria: some exams cannot be split in multiple rooms, some rooms can handle more than one exam, split exams or exams with the same instructor need to be geographically nearby, some exams require special room facilities, some exams are preassigned to specific (groups of) rooms.

5 Conclusions

One of the most striking observations from preparing this survey is that the algorithms that have been implemented in practice are uniformly relatively simple. For example, there has been a considerable interest in genetic algorithms and a wide variety of approaches. However, the ones that have been implemented are among the simplest of GA strategies. The same is true for the other successful approaches. Part of this simplicity is that most algorithms only address a subset of the constraints and objectives of the general timetabling problem as outlined at the beginning of this paper. Some authors claim that additional criteria could easily be incorporated. For example, an *evaluation* function for a GA can be very flexible. However, complex evaluation functions may severely restrict the speed required to produce large potential populations. There is a trade-off between speed and comprehensiveness.

Carter, Laporte and Chinneck incorporate most of the criteria. Their algorithm is also the only algorithm in the literature that has been implemented at more than one institution. It is currently in use at the University of Toronto (Engineering), Carleton University, London School of Economics, the University of Otago (New Zealand), and the University of Limerick (Ireland).

We should state that this survey is in a sense rather restricted in scope. There has been a lot of practical work on the related, but more complex problem of course timetabling with practical applications. In many cases, course timetabling algorithms can be applied directly to the examination timetabling problem with excellent results. However, a detailed survey of applications in course timetabling will have to wait for another paper. We strongly encourage future researchers make use of the test problems now available at the ftp site: ftp.cs.nott.ac.uk or the web site site http://tawny.cs.nott.ac.uk/ttg/ttp.html. By reporting test results on some of these benchmark problems, it will be possible to gain a better understanding of the relative merits of the various approaches.

References

Arani, T., Karwan, M., and Lotfi, V., "A Lagrangian Relaxation Approach to Solve the Second Phase of the Exam Scheduling Problem", **European Journal of Operations Research 34**, 1988, 372-383.

Arani, T., and Lotfi, V., "A Three Phased Approach to Final Exam Scheduling", **IIE Transactions 21**, 1989, 86-96.

Balakrishnan, N., Lucena, A., and Wong, R.T., "Scheduling Examinations to Reduce Second-order Conflicts", **Computers & Operations Research 19**, 1992, 353-361.

Balakrishnan, N., "Examination Scheduling: A Computerized Application", **Omega 19**, 1991, 37-41.

Barham, A.M. and Westwood, J.B., "A Simple Heuristic to Facilitate Course Timetabling", **Journal of the Operational Research Society 29**, 1978, 1055-1060.

Boizumault, P., Delon, Y., and Péridy, L., "Constraint Logic Programming for Examination Timetabling", **Journal of Logic Programming**, 1995, 1-17.

Boufflet, J.P., and Negre, S., "Three Methods Used to Solve an Examination Timetabling Problem", in The Practice and Theory of Automated Timetabling (E.K. Burke and P. Ross eds.), Springer-Verlag Lecture Notes in Computer Science, 1996.

Broder, S., "Final Examination Scheduling", **Communications of the A.C.M. 7**, 1964, 494-498.

Broder, S., "A Heuristic Algorithm for Timetable Construction", **Journal of Educational Data Processing 5**, 1968, 117-123.

Broder S., On the Problem of Time Table Construction, **Communications of the ACM, 9**, 1966, 257.

Burke, E.K., Newal, J.P., and Weare, R.F., "A Memetic Algorithm for University Exam Timetabling", in The Practice and Theory of Automated Timetabling (E.K. Burke and P. Ross eds.), Springer-Verlag Lecture Notes in Computer Science, 1996.

Carter, M.W., "A Survey of Practical Applications of Examination Timetabling Algorithms", **Operations Research 34**, 1986, 193-202.

Carter, M.W., Laporte, G., and Chinneck, J.W., "A General Examination Scheduling System", **Interfaces 24**, 1994, 109-120.

Carter, M.W., Laporte, G., and Lee, S.Y., "Examination Timetabling: Algorithmic Strategies and Applications", **Journal of the Operational Research Society 47**, No. 3, 1996, 373-383.

Carter, M.W., "Examination Room Assignment for EXAMINE", Working Paper, May 1995.

Cole, A.J., "The Preparation of Examination Timetables Using a Small-Store Computer", **The Computer Journal 7**, 1964, 117-121.

Corne, D., Fang, H.L., and Mellish, C., "Solving the Modular Exam Scheduling Problem with Genetic Algorithms", in **Proc. of the Sixth International Conf. on Industrial and Engineering Applications of Artificial Intelligence and Expert Systems**, P.W.H. Chung, G. Lovegrove & M. Ali (eds), 1993, 370-373.

Corne, D., and Ross, P., "Peckish Initialization Strategies For Evolutionary Timetabling" in The Practice and Theory of Automated Timetabling (E.K. Burke and P. Ross eds.), Springer-Verlag Lecture Notes in Computer Science, 1996.

Desroches, S., Laporte, G. and Rousseau, J., "HOREX: A Computer Program for the Construction of Examination Timetables", **INFOR 16**, 1978, 294-298.

Ergül, A., "GA-Based Examination Scheduling Experience at Middle East Technical University" in The Practice and Theory of Automated Timetabling (E.K. Burke and P. Ross eds.), Springer-Verlag Lecture Notes in Computer Science, 1996.

Fisher, J.G. and Shier, D.R., "A Heuristic Procedure for Large-Scale Examination Scheduling Problems", Tech. Report 417, Dept. of Mathematical Sciences, Clemson University, March, 1983.

Foxley, E., and Lockyer, K., "The Construction of Examination Timetables by Computer", **The Computer Journal 11**, 1968, pp. 264-268.

Hertz, A., "Tabu Search for Large Scale Timetabling Problems", **European Journal of Operations Research 54**, 1991, 39-47.

Johnson, D., "Timetabling University Examinations", **Journal of the Operational Research Society 41**, 1990, 39-47.

Laporte, G. and Desroches, S., "Examination Timetabling by Computer", **Computers and Operations Research 11**, 1984, 351-360.

Leong, T.Y., and Yeong, W.Y., "Examination Scheduling: A Quadratic Assignment Perspective", **Proc. Int'l Conf. on Optimization: Techniques and Applications**, Singapore, April 1987, 550-558.

Leong, T.Y., and Yeong, W.Y., "A Hierarchical Decision Support System for University Examination Scheduling", working paper, National University of Singapore, Sept., 1990.

Leong Y.L., "Right on Schedule: Exam Timetabling at the National University of Singapore Gets an Added Boost with New Tools", **ComputerWorld**, May 12-18 1995, 22-23.

Lotfi, V., and Cerveny, R., "A Final-exam-scheduling Package", **Journal of the Operational Research Society 42**, 1991, 205-216.

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

Mehta N.K., Computer-Based Examination Management System, **Journal of Educational Technology Systems, 11**, 1982, 185-198.

Nuijten, W.P.M., Kunnen, G.M., Aarts, E.H.L. and Dignum, F.P.M., "Examination Time Tabling: A Case Study for Constraint Satisfaction", **Proceedings of the ECAI '94 Workshop on Constraint Satisfaction Issues Raised by Practical Applications**, 11-19.

Thompson, J.M., and Dowsland, K.A., "Variants of Simulated Annealing for the Examination Timetabling Problem", working paper, European Business Management School, Swansea University, Swansea, UK 1995a, to appear in **Annals of O.R.**

Thompson, J.M., and Dowsland, K.A., "TISSUE Wipes Away Exam Time Tears: A Computerised System Helps Swansea University Improve Examination Timetabling", **O.R. Insight 8**, No. 4, 1995, 28-32.

Thompson, J.M., and Dowsland, K.A., "General Cooling Schedules for a Simulated Annealing Based Timetabling System" in The Practice and Theory of Automated Timetabling (E.K. Burke and P. Ross eds.), Springer-Verlag Lecture Notes in Computer Science, 1996.

Tripathy, A., "A Lagrangian Relaxation Approach to Course Timetabling", **Journal of the Operational Research Society 31**, 1980, 599-603.

Tripathy, A., "School Timetabling - A Case in Large Binary Integer Programming", **Management Science 30**, 1984, 1473-1489.

White, G.M. and Chan, P.W., "Towards the Construction of Optimal Examination Timetables", **INFOR 17**, 1979, 219-229.

White, G.M., and Haddad, M., "An Heuristic Method for Optimizing Examination Schedules Which Have Day and Night Courses", **Computers and Education 7**, 1983, pp. 235-238.

Wood, D.C., "A System for Computing University Examination Timetables", **Computer Journal 11**, 1968, 41-47.