Proctor Assignment at Carleton University

Rania M. Awad — *Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, Ontario, Canada K1Y 3S1*

John W. Chinneck — *Systems and Computer Engineering, Carleton University*

Carleton University needs proctors during examination sessions to oversee the students as they write their final examinations. The process of assigning proctors to examinations is quite complex because of the sheer number of examinations and various complex constraints on allowable assignments. We developed a computer-based system to replace an existing manual system for assigning proctors, freeing up valuable staff time in the overloaded scheduling office. Our system combines some problem-specific heuristics, a genetic-algorithm framework, and a simple user interface based on readily available software tools.

About 60,000 person-examinations take place over 12 to 15 days at the end of the fall and winter terms at Carleton University, and a smaller number at the end of the summer term. During each of the larger fall and winter examination periods, the university hires about 120 seasonal employees as proctors to oversee the writing of the examinations.

The process of assigning proctors to examinations is quite complex. The proctors are divided into various classes (mainly by level of experience). Depending on their class, proctors are excluded from working in certain examination rooms and may or may not be eligible for certain roles (such as head proctor for a given building). There are also restrictions on unfavorable conditions, such as the assignment of many consecutive examinations to a particular proctor without a period off. Other factors include proctor preferences, such

EDUCATION SYSTEMS—OPERATIONS
PROGRAMMING—INTEGER—ALGORITHMS

as carpooling requirements (some proctors can be assigned only if others from their carpool are assigned at the same time) and refusal to work in certain periods (such as evenings).

People in the scheduling office at Carleton University were spending an inordinate amount of time assigning the proctors to examinations manually. To assist them, we developed a computer-based information system, Proctor, that carries out the initial assignment automatically, and permits its easy manual alteration. The system is PC based and designed for easy use by the nontechnical staff of the scheduling office. The main ingredients are an interface and a database that we created using Microsoft Access and Visual Basic, a set of heuristics for developing a population of assignment schedules, and a genetic algorithm for refining the original population to arrive at the initial assignment schedule. Proctor also provides various reports and statistics needed for the analysis of assignment schedules and for payroll purposes. It has been well received by the scheduling office and is now in regular use.

Related Work

The proctor-assignment problem at Carleton is characterized by predetermined and noncyclic shift requirements, various duties (for example, supervisor or regular) and shift characteristics (for example, room size and examination length), restricted employee availability, and heterogeneous employee types. In addition, it is important to try to satisfy numerous employee preferences. Since the problem is combinatorially explosive and the solutions are restricted by many constraints and preferences, we developed a hybrid genetic algorithm (GA) solution for the initial assignment problem.

The extensive literature on employee scheduling includes previous work on scheduling casual workers. There are numerous variations of the basic problem, generally distinguished by the types of constraints that the scheduler must handle. Examples include whether the shifts are cyclic or nonrepeating, whether the shifts are predetermined or generated as part of the solution, whether employee types (for example, skill levels or job classes) are homogeneous or heterogeneous, and whether general outside constraints must be satisfied (for example, employee preferences for shift types).

Levine [1996] describes a hybrid GA algorithm for scheduling airline crews. This problem is more structured than our proctor-assignment problem and does not deal with casual employees. Levine concludes that his hybrid GA is outperformed by special-purpose branch-and-bound and branch-and-cut algorithms on the well-structured airline-crew-scheduling problem. However, his work demonstrates the utility of hybrid GAs in solving scheduling problems and hints that the best use of hybrid GAs may be in less-structured problems.

Randhawa and Sitompul [1993] developed a heuristic-based decision support system that solves a nurse-scheduling problem similar to the assignment of proctors. The system first generates legal patterns for the nurse shifts: (1) days of the week to be worked and (2) sequences of shifts (day, evening, night). The two are then combined in the next stage to produce a schedule; penalty costs are assigned to undesirable schedules, and a best-first search is used to arrive at a solution. However, the actual assignment of nurses to this schedule is manual and is performed interactively. The schedule repeats for *n* weeks, and each nurse has the same schedule for the entire time, which is different from the proctor-assignment problem.

Love and Hoey [1990] formulated a fast-food-restaurant crew-scheduling problem as a two-stage network-flow problem, which they solved using a decomposition technique followed by the application of a custom network-flow algorithm and a postoptimality analysis. The problem is similar to proctor assignment in that employee preferences impose numerous constraints on the solution. The authors stress the importance of a friendly interface in gaining user acceptance of the system, mentioning in particular the appeal to users of the set of reports that the system generates.

Lauer et al. [1994] describe an interactive decision support system for scheduling part-time computer-lab attendants. A linear program generates optimal daily shifts, which are manually assigned to crews via an interactive interface and combined to form weekly schedules. All shifts in a lab are of the same type so a solution schedule simply specifies whether or not an attendant is working. These authors also stress the importance of the user interface in gaining user acceptance.

Thompson [1996] describes a simulated annealing heuristic for scheduling employees with restricted availability and homogeneous skill levels.

The Proctor Assignment Problem

Examinations take place over a 12- to 15-day period at the end of the fall, winter, and summer terms. There are three examination time slots per day: morning (9:00 a.m. to noon), afternoon (2:00 p.m. to 5:00 p.m.), and evening (7:00 p.m. to 10:00 p.m.). Most exams are three hours in length, some are two hours, and professors occasionally prescribe other lengths, but all start at the same time in a given time slot. Exam length affects both the assignment of exams to rooms and the assignment of proctors. Exams are assigned to rooms throughout the university, including the gymnasium, which houses the larger exams. There are 10 to 55 exams scheduled in any particular time slot, employing 30 to 70 proctors.

The requirement for proctors is established as the fourth step of the examination-scheduling process. First the requirements for scheduled exams are established by consultation with the professors. Second, based on a knowledge of the number of students enrolled in the courses, the rules for scheduling examinations, and the available seating, schedulers assign exams to time slots [Carter, Laporte, and Chinneck 1994]. Third, the schedulers assign exams to rooms while respecting various constraints such as the requirement that large exams be kept in one room and that exams of differing lengths be placed in different rooms. Finally, once schedulers have assigned exams to rooms, they establish the requirements for proctors based on their experience as to the number of proctors needed to cover each room, the need for head proctors in various buildings, and the need for specially qualified proctors to handle special arrangement exams (SAE), for example, those for disabled students.

The scheduling office establishes a list of returning proctors prior to assigning proctors. The schedulers place returning proctors in one of four classes: (1) part-time rookie, (2) part-time experienced, (3) part-time veteran, or (4) full-time. The part-time subclassifications are based on experience, listed above from least to most experienced. Full-time proctors are willing to work from 9:00 a.m. to 10:00 p.m. and are available for work during every examination period.

The university hires new proctors only after the initial proctor assignment has been completed. The number and pattern of unassigned slots in the initial proctor assignment allows the schedulers to estimate the number of new proctors needed and to establish the periods during which they are required. An important goal of the initial proctor assignment is to minimize the number of unassigned slots so that fewer new proctors need to be hired. New proctors are difficult to find, and they must be trained.

Proctoring work is seasonal and does not pay well, so proctors can easily quit without advance notice if they are unhappy with their working conditions. This has an important negative impact on the functioning of the examination system. For this reason, various conditions are imposed on the proctor-assignment process to make the experience as pleasant as possible for the proctors and to encourage their retention. These conditions fall into three main classes: proctor preferences, constraints on unfavorable schedules, and constraints on number of shifts assigned.

Proctor Preferences

Proctor preferences are satisfied wherever possible:
—Proctors are scheduled to work only during periods in which they have indicated that they are available for work.
—Proctors are not assigned to rooms in which they have indicated that they will not work (for example, some proctors avoid rooms that require more standing than others or involve working in large teams).
—Members of carpools are always scheduled to work the same shifts.

Constraints on Unfavorable Schedules

Full-time proctors will accept any schedule; however, part-time proctors consider some schedules unfavorable:
—Split shifts, that is, a morning and an evening shift on the same day;
—Two-hour exams;
—An evening shift followed by a morning shift the next day.
The goal of the following rules is to distribute the unfavorable conditions as fairly as possible:
—Minimize the number of unfavorable proctor schedules.
—Distribute the unfavorable schedules evenly among proctors of the same class.
—Assign fewer unfavorable schedules to proctors of the more experienced classes.

Constraints on Number of Shifts Assigned

These rules govern the number of shifts assigned. The idea is to satisfy the wishes of the proctors in a fair and equitable manner:
—To assign part-time proctors a maximum of two shifts per day,

—To assign full-time proctors in every shift in which they are available,
—To assign more shifts to proctors of the more experienced classes,
—Within an experience class, to assign more shifts to the more flexible proctors, that is, those available for more shifts and willing to work in more rooms.

The initial proctor assignment must also respect obvious physical constraints:
—Do not schedule a proctor to be in two places at once,
—Do not assign more proctors to a slot than are needed (for example, two proctors where one is required). Underassignment is permitted as part of the system for assessing the need for new proctors.

Existing Manual System for Proctor Assignment

Before Proctor, schedulers carried out the initial proctor assignment manually with no computer assistance and then entered it into a spreadsheet. The spreadsheet would typically have 36 to 45 columns for the examination time slots and 100 to 120 rows for the proctors. When printed, it was six pages wide and three pages long. The printed pages were glued together and taped to a desk for reference. The unmanageably large size of the spreadsheet made entering and updating data difficult and confusing. People extracted the information for reports by hand, copying it from the desk copy.

The proctor schedule is very dynamic, and it changes continually during the course of the examinations. Updates to the schedule used to be written on the desk copy, which was the only up-to-date record and which became very cluttered by the end of the examination period. One of the many problems associated with this manual system was in generating the payroll: at the end of the examination session, the desk copy was so cluttered that it was impossible to determine how many hours each proctor had worked. The scheduling officer was often forced to ask proctors how many hours they had worked and to pay them based on this figure.

The existing manual system had three major problems:
—It took too much time to create the initial proctor schedule.
—It was very difficult to keep track of changes in the schedule.
—It was difficult to produce accurate reports (such as payroll).
Our goal was to develop an information system that would alleviate these three problems.

Evaluating Proctor Assignments

Creating the initial proctor assignment automatically depends on having a measure of the quality of a proctor assignment. Any assignment that would satisfy all of the constraints simultaneously would earn the top quality ranking; however, such an assignment is unlikely because some of the constraints conflict. For this reason, we measure the quality of a particular proctor assignment by assigning penalty points for the failure to satisfy various constraints, converting the constraints into an objective function to be minimized. The proctor assignment having the fewest penalty points is the best; an assignment having zero penalty points satisfies all of the constraints.

Some constraints are inviolable and so are strictly enforced by the solution process rather than being treated via penalty points. The fact that a proctor cannot be scheduled in two places at once is enforced by the solution encoding scheme. "Proctors work only in locations where they have agreed to work" and "Proctors work only in time slots during which they are available" are both constraints that are enforced through the initial population rules.

More penalty points are assigned to violations of the more important constraints. For example, because the part-time proctors find split shifts to be more inconvenient than evening-followed-by-morning shifts, split shifts are more heavily penalized. The user determines the penalty associated with violating each constraint and can easily modify penalties through the information system as priorities change (appendix). The penalty scheme does not handle carpooling restrictions. Instead, the information system provides a report on carpooling errors, and they are corrected manually. Also, there is no penalty for deviating from the category average for the full-time proctors.

A Basic Genetic Algorithm for the Initial Assignment of Proctors

The number of possible proctor assignments is extremely large. In a worst case, there are p proctors, t examination time slots, and a possible assignments in every time slot. Where $p > a$, there will be $t \times p!/(p - a)!$ possible assignments. For example, where $p = 100$, $t = 45$, and $a = 30$, the number of possible assignments is 3.5×10^{59}. Because of the combinatorially explosive nature of the problem, enumeration and other deterministic methods fail. What are needed are techniques that sample the solution space.

Candidate solution methods for a problem of this type include pure random search, problem-specific heuristics, simulated annealing, tabu search, and genetic algorithms. We chose a genetic-algorithm (GA) framework for three primary reasons: (1) indications in the literature (for example, Levine [1996]) that GAs may be a good approach to such unstructured scheduling problems as our proctor-assignment problem, (2) the chromosome-string solution format required by a genetic algorithm is natural for this problem, so implementation is straightforward, and (3) we had had success with genetic algorithm solutions to other problems. The main drawback of GAs is that solution times can be long; however, we use the GA only to construct the initial proctor assignment, so a speedy solution is not important. Computation times on the order of hours are acceptable.

The first step in devising the GA is to find a meaningful way of encoding a solution (that is, a complete proctor assignment that is not necessarily feasible). We represent a solution by a matrix with examination time slots in the columns and proctors in the rows. Each cell in the matrix holds a code representing the building assignment for a proctor during a specific time slot (Table 1). Because there is only a single cell for each proctor-time-slot combination, proctors can be scheduled to work in only one place at a time. A post-processing step assigns the proctors to individual rooms.

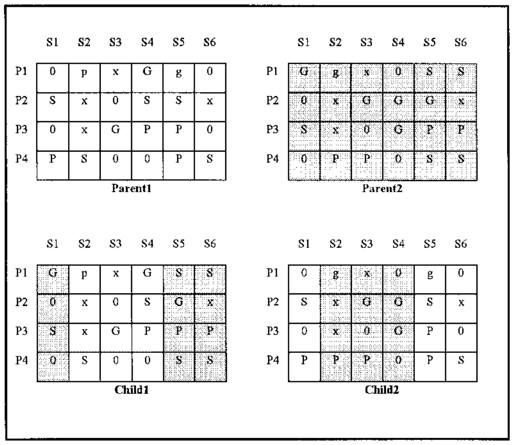
The Parent1 matrix in Figure 1 shows the assignment of four proctors over a period of six time slots (S1 through S6). Proctor P1 is assigned as follows: (S1) available

	Two-Hour Shift	Three-Hour Shift
Special	*	*
Gym	g	G
Southam Hall	s	S
Patterson Hall	p	P
Porter Hall	r	R
Steacie	t	T
Herzberg	h	H
Float	f	F
Other	o	O
Not available	X	X
Free (not scheduled)	0	0

Table 1: These codes represent proctor assignments.

but unassigned, (S2) assigned to Patterson Hall for two hours, (S3) not available to work, (S4) assigned to the gym for three hours, (S5) assigned to the gym for two hours, (S6) unassigned.

The second step is to randomly generate an initial population of solutions. We do



Parent1

	S1	S2	S3	S4	S5	S6
P1	0	p	x	G	g	0
P2	S	x	0	S	S	x
P3	0	x	G	P	P	0
P4	P	S	0	0	P	S

Parent2

	S1	S2	S3	S4	S5	S6
P1	G	g	x	0	S	S
P2	0	x	G	G	G	x
P3	S	x	0	G	P	P
P4	0	P	P	0	S	S

Child1

	S1	S2	S3	S4	S5	S6
P1	G	p	x	G	S	S
P2	0	x	0	S	G	x
P3	S	x	G	P	P	P
P4	0	S	0	0	S	S

Child2

	S1	S2	S3	S4	S5	S6
P1	0	g	x	0	g	0
P2	S	x	G	G	S	x
P3	0	x	0	G	P	0
P4	P	P	P	0	P	S

Figure 1: The crossover operation between Parent1 and Parent2 produces Child1 and Child2. Two columns are randomly selected (S2 and S4 in this example), and all the columns between and including the two selected columns are exchanged between the two parent assignments.

this using rules that reduce the number of infeasible proctor assignments:

(1) Assign proctors only to time slots in which they have agreed to work.

(2) Assign proctors only to rooms in which they have agreed to work.

(3) Preserve all preassigned slots.

To accomplish (1) we write-protect the matrix elements corresponding to periods in which a proctor is not available. We also write-protect slots in (3) to preassign nonclassifiable special proctor assignments (such as SAEs) that require a judgment call and cannot be automated. It is difficult to assign proctors to SAEs manually after the initial proctor assignment, but simple on a blank schedule.

The next step is to calculate the value of the fitness function for each solution in the population using the quality measure described previously. We use these values in a roulette-wheel selection [Goldberg 1989] to choose the schedules that will form the mating pool.

We then apply basic crossover and mutation operators to the schedules in the mating pool to create the next generation. We apply both operators in a manner that does not disturb the conditions established by the initial population rules.

The crossover operator preserves the following characteristics common to all schedules in the initial population: (1) all slots where no proctor is available for work are blank, (2) all assignments are for locations in which a proctor has agreed to work, and (3) for each proctor, preassigned slots in all schedules contain the same information. We maintain these characteristics during crossover by swapping cells that have identical locations in the two schedules, for example, we exchange a cell corresponding to proctor 7 in period 9 only with a cell corresponding to proctor 7 in period 9 in another schedule (Figure 1).

This crossover method is effective only when we are operating on assignments that fully satisfy room requirements for each period (along the columns), such as those produced by the initial population heuristics. Alterations within individual columns can only worsen the fitness-function value. This crossover procedure keeps columns intact while altering the proctor assignments along the rows, possibly improving the proctor conditions.

The mutation operator randomly alters the value of one element of the matrix, at a rate on the order of one alteration per 1,000 cells. The write-protected entries in the solution matrices are excluded from mutation. The cycle of reproduction into the mating pool, crossover, and mutation is repeated for many generations. The program retains the best solution generated so far (the incumbent solution) as the generations pass (Figure 2).

A final solution returned by the basic GA may provide fairly good proctor conditions but it may violate room requirements to an unacceptable degree. Unfortunately, manual correction of the solution to satisfy the room requirements can be as tedious as manually creating the initial proctor assignment from scratch. For this reason, we developed various heuristic and hybrid methods to improve the GA.

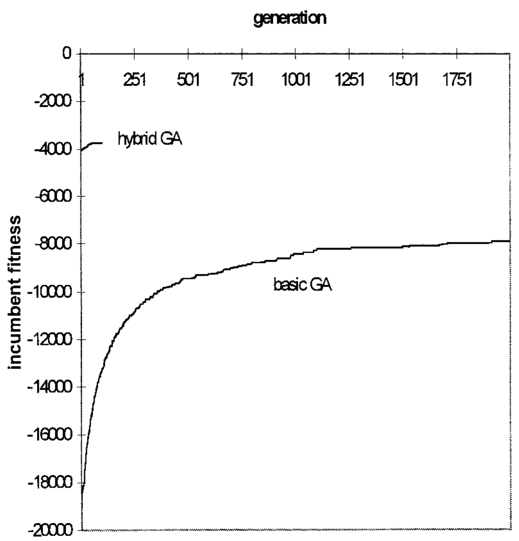


Figure 2: A typical plot of fitness of the incumbent solution versus the generation number. The basic genetic algorithm had a random initial population of 70 different proctor assignments and required approximately 12 hours of solution time on a 33 MHz 80486-based PC. The basic GA improved the incumbent solution fitness considerably over 2,500 generations, but its final incumbent solution is still much worse than the initial incumbent solution for the hybrid genetic algorithm.

Heuristic and Hybrid Methods for the Initial Proctor Assignment

Because the performance of the basic GA was disappointing, we developed a set of three heuristics for generating complete proctor assignments. We can use them in two ways: (1) to generate the initial proctor assignment directly without using the GA, or (2) to provide an improved initial population for the GA, which then operates to improve the solution. All three heuristics use the encoding scheme described above and satisfy the three rules for members of the initial population given previously.

Stand-Alone Heuristics

The goal of the heuristics is to generate good initial proctor assignments directly. All three heuristics try to satisfy room requirements first, while avoiding giving part-time proctors three slots in the same day and split shifts. We avoid both of these conditions simultaneously by masking either a morning slot or an evening slot for each part-time proctor who is available for work during all three time slots of a particular day. This masking is carried out randomly during the common Stage 1 of the heuristics.

The second stages of all three heuristics generate proctor assignments that meet these conditions:
—No excess assignments, although some slots may be underassigned as part of the system for assessing the need for new proctors,
—Proctors are scheduled only when available,
—Proctors are scheduled to work only in rooms where they agree to work, and
—Few split shifts.

A common algorithm is used for the second stages, as given in Algorithm 1; the heuristics differ only in the way in which they traverse the list of proctors.

Heuristic 1: We establish an initial randomly ordered list of proctors. We use the same list for each time slot, except that we choose a new starting point in the list randomly at the start of each time slot and treat the list as a circular queue.

Heuristic 2: We order proctors from most to least constrained based on the length of their lists of possible assignments (proctors having the shortest lists are the most constrained). This has the effect of reducing the number of unassigned slots so that Carleton needs to hire fewer new proctors. We traverse the proctor list from top to bottom each time.

Heuristic 3: This variation gives higher priority to more senior and more flexible proctors. We order the proctor list from most to least seniority. We then suborder proctors with the same seniority from longest to shortest list of available time slots. We traverse the proctor list from top

```
For each time slot:
  Order the list of buildings from largest number of proctors needed to smallest number of proctors
    needed.
  For each building needing proctors:
    $n$ = number of proctors needed in this building during this time slot.
    Number Assigned = 0
    For each proctor:
      If (proctor is available for work during the time slot) and
      (time slot is not masked for this proctor) and
      (proctor not already scheduled during this time slot) and
      (building is on the list of possible assignments for the proctor) then:
          Assign proctor to this building during this time slot.
          Increment Number Assigned.
          If Number Assigned = $n$, then exit the proctor loop.
      End if.
    Next proctor.
  Next building.
Next time slot.
```
Algorithm 1: This Stage 2 algorithm is common to all of the heuristics, which differ only in the ordering of the list of proctors.

to bottom each time.

To use these heuristics in a stand-alone manner, we generate a number of solution instances and choose the best solution from among those generated. The proctor assignments are evaluated as described previously. We would typically generate 350 proctor assignments in less than an hour on a 33 MHz 80486 PC. A typical proctor assignment generated this way has only 60 to 100 unassigned slots (out of a possible 1,550). The average number of split shifts and evening-followed-by-morning patterns per proctor is always less than one. These results are typically much better than those generated by the basic GA in 12 hours.

The Hybrid Method

We can easily create hybrid methods by using the three heuristics to generate an initial population of good solutions for the GA. The incumbent solution in the initial population of the hybrid method is usually much better than the final solution returned by the basic GA (Figure 2). While it is certainly possible to forego the GA entirely, it does provide some refinement of the initial incumbent solution, improving the fitness of the incumbent solution by about 10 percent within 200 generations. In fact, we usually needed only about 100 generations to produce an incumbent solution that satisfied the needs of the scheduling office.

The Proctor Information System

The Proctor information system is key to the successful implementation of the project. It performs several functions: (1) it provides users with an easy-to-use and nontechnical interface to the hybrid GA system so that they can obtain an initial proctor assignment, (2) it manages and displays the large amount of information needed to create and modify the initial assignment, and (3) it generates various reports the scheduling office needs.

Development Process: Prototyping

We used a prototyping approach during development to ensure that the scheduling office would accept the final product. We created various preliminary versions of the information system in consultation with the schedulers and then demonstrated or exercised them. This process prompted numerous refinements to our initial concepts and resulted in a system that closely matched the users needs. We used a beta version of the information system to assign proctors during fall term 1995. Based on this real-life test, we released a slightly modified final version and used it for final examinations in April 1996. It has been in regular use since then.

Features

The information system provides simple data-entry forms (Table 2) for the input of proctor information and constraints and for the preassignment of some proctors (for example, SAE proctors). It also allows the user to set the fitness-function penalty weights and to choose the initial population heuristic for the hybrid method. Once the initial proctor assignment has been generated, the user can view it and modify it easily. Table 2 lists the data-entry forms of the information system.

The system can generate 10 different reports describing the schedule:

—The individual proctor schedule lists the assignments for an individual proctor. These are sent to the proctors to notify them of their assignments.

Form	Purpose
Individual proctor information	Entering and modifying general proctor information, constraints and assignments.
Proctors needed	Viewing the exam list and entering the number of proctors needed for each exam.
Daily spreadsheet view	Displaying the proctor assignment for a single day at a time and updating the proctor assignments only.
Program configuration	Setting the fitness function penalties associated with violating each constraint and choosing the initial population heuristic.
Proctor rates	Entering the rate of pay for regular, SAE and head proctors.

Table 2: The information system includes various data-entry forms that so that the schedulers can enter data or modify existing information.

—The Payroll form is a weekly report showing the number and type of assignments a proctor worked, and the amount of money owed to each proctor.
—The proctor assignment by building by period form lists, for each building and each period, the proctors working. This is used by the head proctor in each building.
—The proctor assignment by building by day form lists, for each building and each day, all the proctors working.
—The proctor schedule by category form shows the schedule of all proctors, sorted by their proctor class, and contains various proctor category statistics and individual proctor statistics. This report is used by the scheduling officer as a working copy when adjusting the initial proctor schedule.
—The proctors needed form shows the number of proctors needed in each room for each period.
—The proctors scheduled form shows the number of proctors scheduled in each room for each period.
—The carpool errors form lists violations of the constraint that proctors who share the same carpool should work during the same periods.
—The proctors scheduled vs. proctors needed report lists all the rooms and periods where the number of proctors scheduled does not match the number of proctors required.
—The three slots same day report lists the part-time proctors who are assigned to three slots in the same day and shows the day on which that happens.

Some reports are useful during the final manual modifications of the initial proctor assignment (for example, reports on discrepancies in the schedule and statistics on proctor conditions), and others satisfy other management information needs (for example, the payroll report). All reports automatically reflect any manual modifications to the initial proctor assignment. Users generate the reports by clicking on-screen buttons.

User Reaction

Carleton has used Proctor during three examination sessions so far, and the user reaction has been very positive. The examinations scheduling officer reports that Proctor eliminates about one person-week of work during each session, time formerly spent generating the initial proctor schedule and cross-checking changes manually. She also reports that the user interface is easy to use, that the various reports are very helpful, and that the initial proctor assignments are extremely good, requiring little manual adjustment. In short, the client is well satisfied with the system, and the scheduling office has adopted it as an essential tool.

Conclusions

Proctor proved successful in solving the problems plaguing the existing manual system. It greatly diminished the workload of the scheduling office in managing the proctors. Two main elements contribute to this success: (1) an operations-research-based system for generating an initial proctor assignment, and (2) a simple and user-friendly information system that eases the administration of the proctors by providing needed reports and allowing easy modification of the proctor-assignment schedule.

A basic genetic algorithm combined with problem-specific heuristics for generating an initial population of solutions provided an effective and practical solution to the combinatorially explosive problem of finding an initial proctor assignment. While it is possible that a more advanced GA or other algorithm may give superior results, the trade-offs in complexity of implementation and maintenance are not likely to be worth the extra effort in development and maintenance. The simple method implemented more than meets the client's needs.

The process of assigning proctors to examinations is quite complex.

The prototyping approach we used in developing the package is an important reason for its success. We obtained effective feedback from the client, leading to an improved end product; it also promoted a feeling of project ownership in the client.

Lessons can be drawn from this project about the relationship between the hard mathematical tools of operations research and the softer issues addressed in designing information systems. In particular, the client reported a great improvement in productivity when we supplied a beta test version that lacked the capability to generate an initial proctor assignment. In other words, we realized a good portion of the benefits of the project simply by organizing and managing the huge amount of data via the information system. In contrast, an excellent algorithm for generating an initial proctor assignment with a poor and inconvenient system for inputting data and retrieving and displaying results would likely have had a very poor reception.

Many operations research projects are really nothing more than computer-based information systems with embedded complex algorithms. For project success, operations researchers need to pay as much attention to the information systems design issues as to the algorithmic issues [Chinneck 1992].

APPENDIX

We evaluate a given proctor schedule by assigning penalty points for constraint violations and for deviation from the category average for numbers of split shifts, numbers of shifts, and so forth. Refer to Table 3 for a summary of the symbols used below and for an example penalty score calculation.

Type of Penalty	Symbols	Example Values	Symbol for Penalty Weight	Example Penalty Weights (Default Values)	Example Penalty Contribution
Constraint Violations	Number of Constraint Violations	Example Number of Constraint Violations			
Under staffing	*under*	89	C_1	10	890
Three shifts same day (part-time proctors)	*pt3s*	4	C_2	9	36
Split shift: Veteran	$split_v$	0	C_3	6	0
Split shift: Experienced	$split_e$	8	C_4	5	40
Split shift: Rookie	$split_r$	0	C_5	3	0
Two hour: Veteran	two_v	52	C_6	5	260
Two hour: Experienced	two_e	200	C_7	4	800
Two hour: Rookie	two_r	28	C_8	2	56
Evening followed by a.m.: Veteran	$eveam_v$	13	C_9	5	65
Evening followed by a.m.: Experienced	$eveam_e$	72	C_{10}	4	288
Evening followed by a.m.: Rookie	$eveam_r$	1	C_{11}	3	3
Deviations from Category Average	Category Standard Deviation	Example Standard Deviations			
Split shift: Veteran	$split\delta_v$	0	C_{12}	5	0
Split shift: Experienced	$split\delta_e$	4	C_{12}	5	20
Split shift: Rookie	$split\delta_r$	0	C_{12}	5	0
Two hour: Veteran	$two\delta_v$	9	C_{13}	5	45
Two hour: Experienced	$two\delta_e$	18	C_{13}	5	90
Two hour: Rookie	$two\delta_r$	5	C_{13}	5	25
Evening followed by a.m.: Veteran	$eveam\delta_v$	5	C_{14}	5	25
Evening followed by a.m.: Experienced	$eveam\delta_e$	13	C_{14}	5	65
Evening followed by a.m.: Rookie	$eveam\delta_r$	1	C_{14}	5	5
Number of shifts: Veteran	$shifts\delta_v$	68	C_{15}	5	340
Number of shifts: Experienced	$shifts\delta_e$	127	C_{15}	5	635
Number of shifts: Rookie	$shifts\delta_r$	24	C_{15}	5	120
Example Total					3,808

Table 3: This is an example of a penalty score calculation for a particular proctor schedule produced by the hybrid genetic algorithm for 74 experienced, 17 veteran, and 9 rookie part-time proctors and 6 full-time proctors. Relevant symbols are also summarized.

In the case of constraint violations, the system calculates a penalty score by multiplying the number of violations by a specified penalty weight (C_i). In the case of deviations from the category average, it first calculates the standard deviation for the category, and then multiplies this by a penalty weight.

Equation 1 shows the calculation of the total penalty score, including penalties for both constraint violations and deviations from the category average.

$$\begin{aligned}\textit{Total penalty} = & \textit{under} \times C_1 + \textit{pt3s} \times C_2 \\ & + \textit{split}_v \times C_3 + \textit{split}_e \times C_4 + \textit{split}_r \\ & \times C_5 + \textit{two}_v \times C_6 + \textit{two}_e \times C_7 \\ & + \textit{two}_r \times C_8 + \textit{eveam}_v \times C_9 + \textit{eveam}_e \\ & \times C_{10} + \textit{eveam}_r \times C_{11} + (\textit{split}\sigma_v \\ & + \textit{split}\sigma_e + \textit{split}\sigma_r) \times C_{12} + (\textit{two}\sigma_v \\ & + \textit{two}\sigma_e + \textit{two}\sigma_r) \times C_{13} + (\textit{eveam}\sigma_v \\ & + \textit{eveam}\sigma_e + \textit{eveam}\sigma_r) \times C_{14} + (\textit{shifts}\sigma_v \\ & + \textit{shifts}\sigma_e + \textit{shifts}\sigma_r) \times C_{15}.\end{aligned} \quad (1)$$

References

Carter, M. W.; Laporte, G.; and Chinneck, J. W. 1994, "A general examination scheduling system," *Interfaces,* Vol. 24, No. 3, pp. 109–120.

Chinneck, J. W. 1992, "Advent of the operations analyst," *OR/MS Today,* October.

Goldberg, D. E. 1989, *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, Massachusetts.

Lauer, J.; Jacobs, L. W.; Brusco, M. J.; and Bechtold, S. E. 1994, "An interactive, optimization-based decision support system for scheduling part-time, computer lab attendants," *Omega,* Vol. 22, No. 6, pp. 613–626.

Levine, D. 1996, "Application of a hybrid genetic algorithm to airline crew scheduling," *Computers and Operations Research,* Vol. 23, No. 6, pp. 547–558.

Love, R. R. and Hoey, J. M. 1990, "Management science improves fast-food operations," *Interfaces,* Vol. 20, No. 2, pp. 21–29.

Randhawa, S. U. and Sitompul, D. 1993, "A heuristic-based computerized nurse scheduling system," *Computers and Operations Research,* Vol. 20, No. 8, pp. 837–844.

Thompson, G. M. 1996, "A simulated annealing-heuristic for shift scheduling using non-continuously available employees," *Computers and Operations Research,* Vol. 23, No. 3, pp. 275–288.

Helen Zaluska, Assistant Director, Scheduling and Examination Services, Carleton University, Ottawa, Ontario, Canada K1Y 3S1, writes: "The proctor-assignment system designed by Rania Awad has been implemented in the Scheduling Office. I used a version of the program to assign proctors for the April 1996 examination period. In 30 minutes, it did a very good job of what would have taken 35 hours using our manual system.

"Proctors were assigned to appropriate room types and were not assigned to overloads unless they were designated as full-time. Predictably, I did have to make some manual adjustments to correct car-pool problems.

"The error reports produced by the program are very useful. They have allowed us to eliminate from the examination operation, which is far too dependent on manual systems, an estimated eight person-hours of cross-checking.

"Other reports, particularly the 'Individual Proctor Schedule' and 'Proctor Assignments by Building and Period,' are to my specifications and are already a part of Carleton's examinations 'culture' even though we've been using them only since December 1995.

"The system is easy to use. It mirrors the good parts of the manual system and improves on the others. Data-entry for proctor availability and proctors needed is uncomplicated and the rest of the job gets done by following the menus.

"I consider that my time on this project was well spent. It was a pleasure to work with Rania who had a good understanding of the problem, was enthusiastic and able in seeking the solution and produced a useful system."