

Staff Scheduling with ILOG Solver

Donovan R. Hare
Department of Mathematics & Statistics
Okanagan University College
3333 College Way
Kelowna, BC
Canada V1V 1V7

Error! Bookmark not defined.

Abstract

This paper describes the design and development of a staff scheduling software product called the Expert Rotation Generator (ERG). ERG's scheduling engine uses ILOG's Solver library to create staff rotations that satisfy a variety of staffing constraints in a reasonable amount of time. The user can interact with ERG during the generation process by modifying the constraints and by selecting a partial rotation from several that satisfy the current constraints. The way in which the search is performed can also be controlled.

Introduction

This paper describes the overall design and development of a component of a staff scheduling system called the Expert Rotation Generator (ERG). The component uses ILOG's Solver library to express the scheduling constraints and to generate rotations that satisfy the constraints.

ERG is the newest component of a commercial staff scheduling system that is installed in health care facilities around the world. The staff scheduling system (without ERG) is a powerful tool that runs on high-end PCs and allows a user to create, change, and report upon rotation-based employee schedules. When employees are not able to work their scheduled shifts, the user can book off the employees from those shifts and optionally schedule relief. The system allows a user to define scheduling-related policies that constrain their scheduling practices and the system will alert the user if a policy is violated. The system also automatically generates employee time cards and can write them to a variety of payroll system file formats. All in all, the system has fourteen distinct environments to perform scheduling related tasks: the configuration editor, the employee editor, the unit editor, the rotation editor, the assignments editor, book off, book on, schedule relief, shift swap, the exceptions editor, the availability calendar, the workbook, the time card editor, and the reporter.

While the staff scheduling system (without ERG) meets the very significant need to automate staffing transactions in large organizations, it does not create schedules. The user must construct a rotation first and then enter the rotation into the system. Once entered, staff can be inserted or deleted in the rotation, shifts can be swapped, and reports produced.

Constructing a good rotation by hand, however, can be a very difficult task. From an organization's perspective, a good rotation can reduce labor costs. On the other hand, the quality of the rotation can have a major impact on the quality of work life for employees. The scheduler must satisfy these interests as well as a host of other constraints. The process of creating a good rotation is costly and time-consuming. In many hospitals, the task is given to the head nurses who do not have training, time nor interest in rotation generation. ERG facilitates this process for them.

From a theoretical perspective, the problem of generating staff rotations (sometimes referred to as the staff scheduling problem) is a difficult combinatorial optimization problem for most staffing contexts (including hospital staffing). It is known that to find a polynomial-time algorithm to solve the general staff scheduling problem is equivalent to solving the most important problem in theoretical computer science (whether $P=NP$). Restricting the context to scheduling staff in hospitals does not make the problem any less difficult. This remains true even when restricted to cyclic nurse scheduling (see for example Bartholdi 1981). The attempts so far described in the literature have applied general classical approaches

(e.g. integer linear programming: Miller, Pierskalla and Rath 1976, Bartholdi 1981, Rosenbloom and Goertzen 1987, Kostreva and Jennings 1991; network flows: Kahn 1991; graph algorithms: Ball and Roberts 1985, Balakrishnan and Wong 1990, Hare 1990) to specific contexts and sub-problems (e.g. finding cyclic schedules that minimize the number of casual staff) with low levels of success. Other ad hoc methods have been used as well with similar limited results (Burns and Koop 1987, Emmons and Burns 1991).

In recent years, constraint logic programming has made some important contributions to the solution of the staff scheduling problem. Several applications using ILOG Solver have tackled the problem in different staffing contexts. In particular, ILOG Solver has been used for several nurse scheduling applications (Darmoni et al. 1996, Lázaro and Aristondo 1995, Kusumoto 1996, Weil et al. 1995).

ERG

The goal of the design of the ERG is to produce a rotation generator that:

- generate rotations that satisfy the given staffing constraints
- generate the rotations in a reasonable amount of time
- has an interactive component that allows a user to specify the part-time/full-time mix of the rotation and to change the constraints between generations
- is able to create linear and block rotations
- is flexible enough to be extendible for a variety of staffing contexts

The rotations ERG produces must satisfy the user-specified staffing constraints. Many of these constraints come from an employer/employee contract and must be satisfied for the acceptance of the rotation. Since the user can change the constraints between generations, the design of ERG requires that all the constraints specified will be satisfied for each rotation generated by ERG.

It is important to a user of ERG to have a quick response time to most of its operations. ERG provides a “what if” environment that allows the user to tweak the constraints of the rotation and re-generate. If ERG takes too long to generate a rotation or too long to decide that no rotation satisfies the given constraints, the process will be interrupted and the environment will lose its usefulness. The results of the testing of ERG (see the section entitled “ILOG Solver and ERG”) further qualify what is meant by a quick response time.

Another goal of the design of ERG is that it should allow for some interaction with the user during the rotation generation process. This feature would help the user tailor the rotations for their particular context. For example, a user could relax or strengthen a constraint if ERG was not producing an acceptable result or was taking too long to generate. The user should be able to make these modifications at any point during the generation process. This requires ERG to allow for interruptions and changes without having to restart the entire process.

ERG generates linear and block rotations. A linear rotation is a schedule where the work assignments for each employee repeat once the schedule ends. The day after the last day of the schedule is considered to be the first day of the next schedule, and so on. Although a linear rotation has a fixed length, it is understood that it represents an expanded schedule that continues indefinitely. Hence any appropriate constraints must be satisfied in the expanded schedule. For example, maximum tour length constraints (see below) must constrain tours that start on a day at the end of the rotation and end on a day at the beginning of the rotation. A block rotation is a special type of linear rotation. In a block rotation, each employee is assigned a block (typically a week long) and then switches (after the week) to the next employee’s block. At the end of the rotation, each employee has worked each block and has made the same changes from block to block. Block rotations are desirable because of their inherent fairness: everyone eventually works the same schedule. Depending on an organization’s policies or practices it may be important to have the ability to create block rotations.

Finally, the rotation generator needs to be flexible enough in design so that it can be used for other staffing contexts. This means that the algorithm that runs the rotation generator must not rely too heavily on the scheduling context (i.e. its particular constraints) but instead be able to handle general staffing constraints.

Constraints

ERG allows the user to specify values for the following constraints:

Baseline: the baseline constraint specifies the number of staff for each shift on each day that is needed in order to provide sufficient coverage. A schedule satisfies this constraint if for each shift and day the number of staff working the shift on the day does not exceed the specified number of the constraint. A rotation is considered to be complete if no baseline is remaining.

Maximum Tour Length: a tour is a consecutive group of “on” shifts or a consecutive group of off shifts (i.e. off days). The maximum tour constraints specify the maximum length of a tour. The maximum tour constraint can be expressed on three different levels: the individual-shift level, the length-group level, and the any-tour level. The individual-shift level expression of the constraint specifies for each shift the maximum number of consecutive days that shift can be worked. The length-group level expression of the constraint specifies for each shift length group (e.g. all 8 hour shifts, or whatever length groups the user is interested in constraining) the maximum number of consecutive days an employee can work shifts from the length group. The any-tour level expression of the constraint is the maximum number of consecutive days an employee can work.

Minimum Tour Length: a minimum tour constraint specifies the minimum length of a tour. This constraint specifies that if an employee changes to a particular shift, then they need to work a minimum number of the shift consecutively.

Minimum and Maximum Number of Hours: the minimum and maximum number of hours constraint is specified for each employee. Each employee has a number, called an FTE (full-time equivalence), that represents the percentage of full-time the employee works. The minimum and maximum number of hours constraint is calculated from the specified number of hours in a full-time week, the FTE of the employee, and the minimum length of the possible shifts the employee can work. The constraint requires the sum of hours of the shifts of each employee in the rotation to be at least the calculated minimum and at most the calculated maximum.

Shift Change Restrictions: the shift change restriction constraint specifies for each ordered pair of shifts (X,Y) whether or not Y can follow X in the schedule.

Allowable Shifts: The allowable shifts constraints specify for each employee which shifts the employee can work.

Weekends Off: an employee has a weekend off in a schedule if they do not work on a shift that starts or ends during a weekend. The start of weekend is a user-defined time on Friday or Saturday. The end of weekend is a user-defined time on Sunday or Monday. The weekends off constraint specifies the minimum number of weekends off an employee must have in the rotation.

Split Weekends Disallowed: this constraint specifies that split weekends are not allowed: either an employee works Saturday and Sunday, or works neither Saturday nor Sunday.

Period: a period is a repeating consecutive sequence of days defined by the user. A period constraint restricts an employee to work between a specified minimum and maximum number of hours in the period.

Patterns: a pattern is a consecutive sequence of on shifts. The patterns constraint restricts the rotation to contain off shifts or consecutive sequence of shifts that match the patterns of the constraint. The patterns constraint need not be activated.

Rules: a rule is a pattern that cannot occur in the rotation. The rules constraint restricts the rotation to *not* contain any consecutive sequence of shifts that match the rules of the constraint. This constraint allows the user to express constraints that are specific to their institution and that cannot already be expressed.

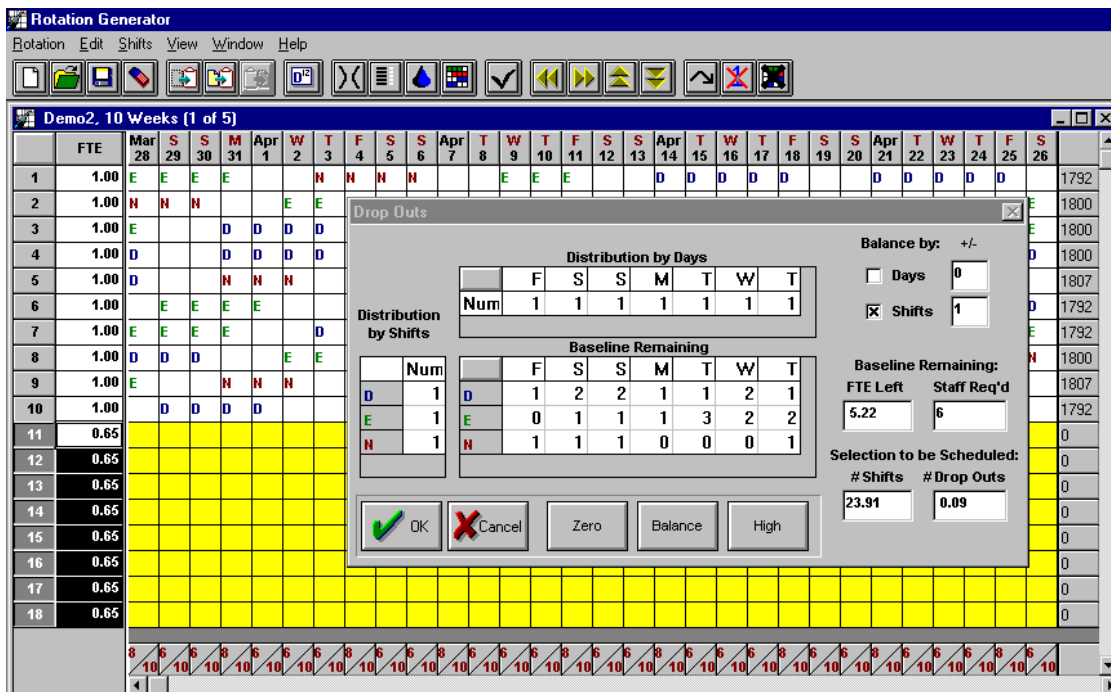
Preferences: a preference expresses an employee’s desire to work a particular shift on a particular day. A preference has one of three values: workable, unworkable and undecided (the default). The preferences constraint restricts the assignment of the shifts to be consistent with the specified preferences.

Dropouts: a dropout in a rotation is a shift that is required by the baseline constraint for a particular day that is not scheduled. The dropouts constraint specifies where the possible dropouts should occur in the rotation. If a maximum number of dropouts for a day or shift is specified, then the rotation generated will have at most that number of dropouts for the day or shift. The dropouts of a rotation can also be constrained by balancing them throughout the days or shifts with a specified flexibility parameter.

Using ERG

To generate a complete rotation, the user first enters the type of shifts (length, start time, etc.) and the baseline of the rotation. In the dropout control of ERG, the number of full-time equivalent staff is displayed. The user decides the part-time/full-time mix of the rotation using this number. The user chooses a group of staff and assigns them appropriate FTEs (an FTE of less than 1.00 is considered to be part-time). With this group, a set of rotations is generated that satisfy the constraints. The user then chooses a rotation from this set and repeats the process with the remaining baseline until all the shifts have been assigned.

An example is given in Figure 1. Here the user has already scheduled 10 full-time staff (all have FTEs of 1.00) and chosen the first of five generated rotations. The dropout control calculates the remaining baseline to be scheduled, the number of FTEs required for this baseline, and the minimum number of staff required. The user has chosen to schedule the 8 remaining part-time staff, numbered 11 through 18, all having FTEs of 0.65, and the dropout control calculates that there is 0.09 (= 0) shifts remaining once the rotation is generated.



Using this interactive process, ERG creates a “tree” of partial rotations and the user decides which branch is most preferable to pursue. It is possible that the chosen partial rotation with a given set of constraints does not lead to a complete rotation. In this case, the user would either relax some of the constraints and re-generate from the chosen partial rotation, or choose a different partial rotation to generate from.

The user can control the way the search is performed. The basic search involves choosing an employee/day pair and choosing a shift to assign to the chosen employee on the chosen day. The user can control the order of the employee/day choice with the following methods:

employee by employee: this method searches for a rotation for each employee consecutively completing an employee’s assignments before searching for the next employee’s assignments. It searches for an employee’s assignments by finding a shift for each consecutive day.

days off search: this method first searches for where the days off in the rotation might go. Once the days off are fixed, it then uses an employee by employee search to assign shifts to the employees for the days that are unassigned.

weekends first: this method first assigns shifts to the Saturdays and Sundays of the employees. Once the weekend shifts are fixed, it then uses an employee by employee search to assign shifts to the employees for the days that are unassigned.

random choice: this method searches for a rotation by randomly choosing an employee and a day and assigning a shift.

The shifts are represented by integers where 0 is reserved for the off shift. The user can also control the choice of the shift that is assigned using the following methods:

minimum shift: choose the minimum available shift.

maximum shift: choose the maximum available shift.

random shift: randomly choose a shift from those available.

successor shift: choose the smallest numbered available shift that is larger than the last assigned shift. If the last shift was the maximum valued shift, then choose the minimum available shift.

ILOG Solver and ERG

ERG's search engine is implemented using the ILOG Solver library and tests were run on a Pentium Pro 166 using Windows NT 4.0. Shifts are represented by integers and a rotation by a two dimensional array of type `IlcIntVar`. Many of Solver's basic goals were used as well as other significant heuristics (such as the ones used for the days off search). Several hospitals provided test cases for ERG. Most of these cases were block rotations of n weeks long by n employees with at most 4 shifts, where n is an integer between 9 and 15. When a solution existed and the right strategy used, the time to generate the first rotation for any of the cases was under 10 seconds. On the other hand, it took under 6 minutes to search the entire space of a 15 staff by 15 week block rotation with 4 shifts to determine that no rotation satisfying the constraints existed. It has been found that when there are few days off in a schedule (as in the case with full-time staff and 8 hour shifts), then the best method to use is the days off search along with the maximum shift choice. If there are many days off in the schedule, then the best method to use is the employee by employee strategy with the maximum shift choice.

Further research is necessary for the generation of linear rotations. They represent a class of problems that are significantly more difficult to generate in a reasonable amount of time than block rotations (although theoretically they are just as difficult). In a block rotation, every employee works the same rotation eventually. For this reason a block rotation with as many employees as desired can be modeled as a linear rotation with only one employee and some extra constraints. In a linear rotation, the FTE of each employee is allowed to be different. This prohibits a linear rotation from being modeled as a block rotation.

To illustrate the magnitude of difference between the search spaces and search times of linear and block rotations, consider the following example. Suppose the baseline requires 4 day shifts, 4 night shifts and 2 night shifts every day of the week. If a 14 week block rotation with 14 employees of FTE 1.0 has a search space of size x , then a 14 week linear rotation with 14 employees all with FTE 1.0 has a search space of size x^{14} . A crude estimate for the search space of block rotation case is $47 \times 14 \times 10^{59}$ and so for the linear rotation case is 10^{826} . If the block rotation case search takes 5 seconds to find a rotation, then the linear rotation case search without additional heuristics could take over 193 years.

Conclusion

ILOG Solver has proven to be a useful tool for developing the search engine of ERG. ERG generates rotations that satisfy a variety of staffing constraints in a reasonable amount of time. The constraints were implemented relatively painlessly with Solver. ERG is interactive by design offering the user a "what if" environment. Solver tools allow time limits to be set on any part of a search so that the user can interact with the search. The flexibility of the search mechanisms in Solver allowed the implementation to include heuristics tailored to its context such as the days off search. The generation of linear rotations by ERG requires more research to reduce the search times. Except for this case, all of ERG's design goals were implemented successfully using ILOG Solver.

References

1. Balakrishnan, N. and R. T. Wong. "A network model for the rotating workforce scheduling problem." *Networks*, 20:25–42, 1990.
2. Ball, M. and A. Roberts. "A graph partitioning approach to airline crew scheduling." *Transportation Sci.*, 19:107–126, 1985.
3. Bartholdi III, J.J. "A guaranteed-accuracy round-off algorithm for cyclic scheduling and set covering." *Oper. Res.*, 29:501–510, 1981.
4. Bell, P.C., H. Genevieve and Y. Liang, "A visual interactive decision support system for workforce (nurse) scheduling," *INFOR* (1985), 24:134–144.
5. Burns, R.N. and M.W. Carter, "Work force size and single shift schedules with variable demands," *Management Science* (1985), 31(5):599–607.
6. Burns, R.N. and G. J. Koop. "A modular approach to optimal multiple-shift manpower scheduling." *Oper. Res.*, 35:100–110, 1987.
7. Darmoni, S. J., A. Fajner, N. Mahe, A. Leforestier, M. Vondracek, O. Stelian, M. Baldenweck, "Horoplan: computer-assisted nurse scheduling using constraint-based programming," located at <http://www.chu-rouen.fr/dsii/publi/plao.html>.
8. Emmons, H. and R. N. Burns. "Off-day scheduling with hierarchical worker categories." *Oper. Res.*, 39:484–495, 1991.
9. Handler, G.Y. and I. Zhang, "A dual algorithm for the constrained shortest path problem," *Networks* (1980), 10:293–310.
10. Hare, D.R. "Nurse scheduling algorithm: Algorithm description." Technical report, Glen Cooper and Associates, Oct. 1990.
11. Hung, R., "Single shift workforce scheduling under a compressed workweek," *Omega* (1991), 19(5):494–497.
12. Hung, R., "A three day workweek multiple shift scheduling model," *JORS* (1993), 44(2):141–146.
13. Khan, Z.A. "A note on a network model for nursing staff scheduling problems." *Inform. and Decision Tech.*, 17:63–69, 1991.
14. Koop, G.J., "Optimal multiple shift manpower scheduling: models and algorithms," Ph.D. Thesis, U. Waterloo (1983).
15. Koop, G.J., "Multiple shift workforce lower bounds," *Management Science* (1988), 34(10):1221–1230.
16. Kostreva, M.M. and K. S. B. Jennings. "Nurse scheduling on a microcomputer." *Comput. Oper. Res.*, 18:731–739, 1991.
17. Kusumoto, S., "Nurse scheduling system using ILOG Solver," in *ILOG Solver & ILOG Schedule Second International Users' Conference Proceedings* (1996), 6 pages.
18. Lázaro, J.M. and P. Aristondo, "Using Solver for nurse scheduling," in *ILOG Solver & ILOG Schedule First International Users' Conference Proceedings* (1995), 10 pages.
19. Morris, J.G. and M.J. Showalter, "Simple approaches to shift, days off and tour scheduling problems," *Management Science* (1983), 29(8):942–950.
20. Miller, H.E., W. P. Pierskalla, and G. J. Rath. "Nurse scheduling using mathematical programming." *Naval Res. Logist.*, 37:559–577, 1990.
21. Mills, R.G.J. and D.M. Panton, "Scheduling of casino security officers," *Omega* (1992), 20(2):183–191.
22. Randhawa, S.U. and D. Sitompul, "A heuristic based computerized nurse scheduling system," *Computers & Operations Research* (1983), 20(8):837–844.
23. Rosenbloom, E.S. and N. F. Goertzen, "Cyclic nurse scheduling." *European J. Oper. Res.*, 31:19–23, 1987.
24. Smith, L.D., "The application of an interactive algorithm to develop cyclical rotation schedules for nursing personnel," *INFOR* (1976), 14(1):53–70.
25. Tien, J.M. and A. Kamiyama, "On Manpower Scheduling Algorithms," *SIAM Review* (1982), 24(3):275–287.
26. Vohra, R.V., "A quick heuristic for some cyclic staffing problems with breaks," *JORS* (1988), 39(11):1057–1061.
27. Weil, G., K. Heus, P. François and M. Poujade, "Constraint programming for nurse scheduling," *IEEE Engineering in Medicine and Biology* (1995) July/August: 417–422.