Nurse Rostering at the Hospital Authority of Hong Kong

Andy Hon Wai Chun

City University of Hong Kong
Department of Electronic Engineering
Tat Chee Avenue, Kowloon
Hong Kong
eehwchun@cityu.edu.hk

Steve Ho Chuen Chan, Garbbie Pui Shan Lam, Francis Ming Fai Tsang, Jean Wong and Dennis Wai Ming Yeung

Advanced Object Technologies Limited
Unit 602A, HK Industrial Technology Center
72 Tat Chee Avenue, Kowloon
Hong Kong
{steve, garbbie, francis, jean, dennis}@aotl.com

Abstract

This paper describes the Rostering Engine (RE) that we have developed for the Hospital Authority (HA), Hong Kong as part of their Staff Rostering System (SRS) using AI constraint-programming techniques. The Hospital Authority manages over 40 public hospitals in Hong Kong. With close to 1500 wards total, the amount of resources needed to produce weekly staff rosters for each ward is tremendous and extremely time consuming. Previously, most staff rosters were generated manually. Without computer records, it was difficult for HA to produce workforce statistics or to improve resource efficiency. In early 1997, the Hospital Authority embarked on a strategic Staff Rostering System project to provide automation support in managing their large workforce of over 48,000 full-time hospital staff. The Staff Rostering System performs ward-level rostering based on ward-specific constraints, staff requests, and work patterns. Version 1 of the system was completed early 1998 and Version 2 was released early 1999. The system is gradually being deployed in different public hospitals across Hong Kong.

Task Description

The Hong Kong Hospital Authority (http://www.ha.org.hk) was established in 1990 as an independent body to manage all public hospitals in Hong Kong. It is accountable to the Hong Kong Government through the Secretary for Health and Welfare. It provides medical treatment and rehabilitation services to patients through hospitals, specialist clinics and outreaching services. At the end of 1997, the Authority managed over 26,400 hospital beds; representing roughly 4.06 public hospital beds per 1,000 population. To fulfil its roles, the Authority employs over 48,000 full-time staff. In 1997, the Authority managed over 44 public hospitals/institutions and 49 specialist outpatient-centers.

The wards within each HA hospital schedules and manages its own nurses and clinical supporting staffs. Some staff members work across wards and need to be scheduled at the departmental level. This is obviously a very time-consuming task but needs to be performed regularly. In 1997, the Hospital Authority started to work with the City University of Hong Kong to design and develop the core rostering engine of their Staff Rostering System.

Ward managers use the Staff Rostering System (SRS) to schedule, reschedule and manage different types of staff such as nurses, student nurses and clinical supporting staff. Since the SRS will be used by a wide variety of different wards and hospitals which have their own specific needs and requirements, in additional to the standard Hospital Authority rules and constraints, the SRS was designed to be flexible enough to capture different types of operational needs. The way rostering is performed may also be different from ward to ward. For instance, rostering may be performed in stages for different ranks and shifts – a night roster may be prepared before the day duties are planned, or nursing officer's night shifts might be rostered first. For course, different groups of staff and different shifts will have different sets of rules and constraints. Rostering may also be performed in different intervals. For example, night shift rostering might be performed once every 4 weeks while other shifts may be rostered on a weekly or bi-weekly manner.

SRS will be used to roster many different types of staff and by many different types of wards and hospitals. Designing a sufficiently comprehensive set of rules and constraints and a rostering algorithm that is sufficiently flexible to handle all the different types of rostering needs were the main challenges in this project.

System Goals

Although many of the constraints used in SRS are unique to the Hospital Authority, the main goals and objectives of the SRS are similar to those of many other rostering systems. For example, the SRS should ensure that there is

an adequate number and mixture of skilled staff present to maintain committed level of service quality. At the same time, each staff member should be assigned an appropriate number of working hours in accordance with their terms of appointment, i.e., should not be over-worked or under-utilized.

For course the roster should also be as fair as possible to all staff members. For example, each staff should be equally given the same number of days off on weekends and public holidays or the same number of night duties. Understanding that each person has their individual needs, staff requests and preferences should also be considered during rostering and accepted as long as they do not impact the overall roster.

Another key objective is to make the roster as "friendly" as possible. For example, it should maximize the interval between performing two night duties. It should prevent or avoid shift patterns that are not desirable. For example, having to work two night duties on consecutive Sundays or having to work an afternoon shift just before a night shift.

The fundamental requirement that any rostering system must follow is of course to ensure that all Government labor regulations are followed and that all appointment terms are met. The system should also produce rosters within a reasonably short period of time.

Application Description

The Staff Rostering System (SRS) is a computer system developed by HA's Information Technology Division to improve the delivery of health care services. It provides assistance to HA hospitals at the ward level in assigning shifts for nursing staff and supporting staff. Through process re-engineering, the system supports the streamlining of the rostering process and hence increases productivity. The functions of SRS include:

- Generating rosters using constraint programming
- Printing rosters for distribution to staff
- Storing roster records into a database
- Generating management reports

SRS adopts a two-tiered client-server architecture. Database operations such as retrieval, display, and modification of rosters, personnel information, and constraints are performed using Microsoft Visual Basic as the front-end to the back-end Microsoft SQL Server database. The Rostering Engine is a component of the front-end and is invoked when rosters need to be generated.

Typically, when a ward manager uses the Staff Rostering System, he/she would first update the system with any changes. These can be changes in ward information, such as special duty days, or staff information, such as promotions or resignations, etc. The ward manager would also adjust any constraints or parameters that might have changed. Normally, rules and constraints, once set are fairly stable. The ward manager then enters any staff requests, such as day off, leave encashment, etc., and any pre-assigned duties, such as training. Once all the pre-assignments, requests and changes have been made, the ward manager then invokes the Rostering Engine to produce a roster for a particular set of staff, shifts, and rostering period.

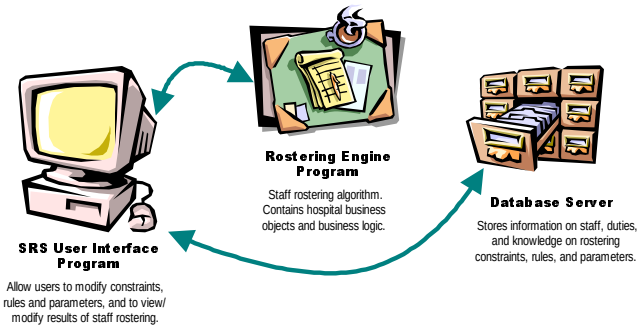

Figure 1. Overall system architecture.

The Rostering Engine creates business objects for the ward being rostered using the following four main types of information:

- **Static information about the ward**. This includes the department this ward belongs to, the shifts within a ward, the special duty days, and how rosters should be generated.
- **Staff information**. This includes information on rank, seniority, assigned wards, leave balances, and personal requests.
- **Constraints and parameters**. These are the constraints and parameters used during rostering. Constraints are described in further detail in Sections below.
- **Historical information**. This includes a set of roster-related statistics, such as number of accumulated leave, etc. and the previous roster history.

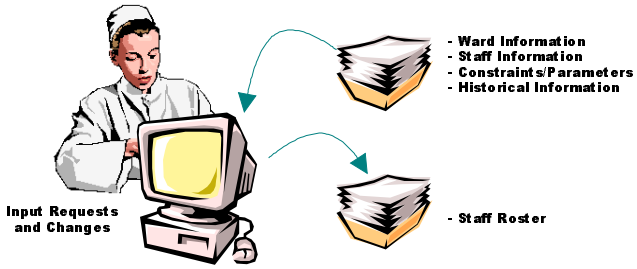

Figure 2. Typical operation of the Staff Rostering System.

Guided by the constraints of a ward, the past roster, and staff requests, the Rostering Engine produces the desired roster for the selected set of staff, shifts, and rostering period using AI constraint programming techniques (Chun 1999, Puget 1994).

Uses of AI Technology

The nurse rostering problem is modeled as a constraint satisfaction problem (CSP) (Cohen 1990, Kumar 1992, Steele 1980, Van Hentenryck 1989). The key components to any CSP algorithm are, of course, the variables, their domains, and the constraints that restrict how these variables may be assigned values from their domains. In our Rostering Engine, each variable represents the unknown shift that a nurse should work in, on a particular day. The domain consists of all the shifts defined for the group of staff that the roster is being generated for.

Our rostering constraints are classified as either hard or soft. Hard constraints must be followed strictly, such as enforcing labor regulations. Soft constraints represent preferences. Users are allowed to define different levels of preferences for each soft constraint. To further control how rostering constraints are used, two "scoping" mechanisms are provided – a "when-to-apply" scope and a "who-to-apply" scope. The "when-to-apply" scope defines when a constraint should be applicable, such as only during weekends or on Mondays as examples. The "who-to-apply" scope defines those staffs that are affected by a constraint, such as a particular individual, a group of staff, or staffs belonging to a particular rank.

The following highlights the key constraints considered by our Rostering Engine:

- **Manpower Demand Constraint**

  This type of constraint defines manpower requirement for a specific rank, staff group, gender or their combination in a particular shift, as well as the alternative manpower demand patterns in case the original demand cannot be satisfied. For example:

  - *Exactly 4 registered nurses on Monday morning can be replaced by 3 registered nurses and 1 enrolled nurse or by 2 registered nurses and 1 student nurse.*
  - *At least 1 male registered nurse in night shift.*

- **Working Hours Constraint**

  This type of constraint defines the total number of working hours for a staff within the specified time period. For example:

  - *An enrolled nurse should work exactly 88 hours per fortnight.*
  - *A registered nurse should work at least 40 hours per week.*

- **Shift Distribution Constraint**

  This type of constraint defines the frequency that a staff may be assigned a particular shift during the specified time period. For example:

  - *At most 1 night shift on Sunday per fortnight.*
  - *At least 1 day off on Sunday every 4 weeks.*

- **Days Between Same Shift Constraint**

  This type of constraint defines the number of days that should elapse before a staff member takes another shift of the same type. For example:

  - *Minimum 4 days between night shifts.*
  - *Exactly 3 days between afternoon shifts.*
  - *A night shift on Thursday must be followed by a night shift on Sunday.*

- **Consecutive Day-Of-Week Constraint**

  This type of constraint defines the total number of consecutive occurrences of a particular shift assigned to a staff on a certain day. For example:

  - *Cannot assign night shifts on two consecutive Sundays.*
  - *Morning shifts must be assigned in two consecutive Saturdays each time.*
  - *Cannot assign day off on two consecutive public* holidays.

- **Shift Sequence Pattern Constraint**

  This type of constraint defines a pattern of shifts to be assigned on consecutive days.

  - *Afternoon shift is not preferred if before a night shift.*

- *Prefer a day off right after a night shift.*
- *Prefer a morning shift to be followed by a day off and then an afternoon shift.*
- *Should avoid assigning three morning shifts in a row.*
- *Should not assign three night shifts in a row.*

The Rostering Algorithm

The problem of nurse rostering or rostering in general have been subjects for decades of scheduling research (Abdennadher and Schlenker 1999, Martello and Toth 1886, Miller, Pierskalla, and Rath 1990, Randhawa 1983, Rosenbloom and Goertzen 1987). The combinatorial problems of shift assignment are well documented. It is only in recent years that researchers have begun to look into constraint programming as an alternative approach. Several researchers have documented successful applications of constraint programming to nurse and staff rostering (Dresse 1995, Kusumoto 1996, Lau and Lau 1997, Lazaro and Aristondo 1995).

The scheduling algorithm we have created for the Hospital Authority combines the power of constraint propagation with intelligent heuristics to avoid combinatorics associated with rostering. Constraint propagation reduces the search space early on while our heuristics guide the search through the remaining search space. The technique we use is a combination of look-ahead and intelligent scoring to determine which nurse to roster next and which shift would satisfy most of the soft constraints. Through half a year of pilot run, we have refined our heuristics so that the scheduling algorithm yields fairly good results within a short reasonable time.

Application Use and Payoff

Hospital Authority's SRS has been in daily use in wards of the Prince of Wales Hospital, Alice Ho Miu Ling Nethersole Hospital, and Shatin Hospital for over a year. Since then, Kwai Chung Hospital, Kwong Wah Hospital, Bradbury Hospice, and the United Christian Hospital have also gone live. Altogether, over 250 seats of SRS have been installed. Over three hundred people have been trained to use the system. Further deployment at other HA hospitals are being scheduled.

There are numerous benefits in using AI techniques for staff rostering. Some of the key application payoffs are outlined below:

- **Increased Productivity**

  Just the process of sitting down and drafting a weekly roster for a single ward is already quite time consuming, not to mention trying to make the roster more efficient and "friendly." Unfortunately, this mundane task of staff rostering must be performed by a highly experienced and skilled staff - the ward manager. In a large hospital, there may be twenty to thirty such ward managers each performing staff rostering for their respective wards. The availability of an automated system, such as the SRS, greatly improves productivity by relieving ward managers of the more mundane aspects of staff scheduling and lets them focus on the actual management of staffs and problem solving. SRS, by using intelligent AI techniques, also produces rosters that assigns staff more efficiently and hence improves the overall productivity of the hospital.

- **Increased Morale**

  Because the rules, constraints and parameters used by our Rostering Engine for each ward are clearly specified and well publicized, suspicions of favoritism are eliminated. Having a set of open criteria, by itself, has already greatly improved staff morale in general. In addition, our Rostering Engine ensures that all staffs are treated fairly by using different types of historical statistics to generate the roster. The statistics indicate how "well" each staff has been treated so far, in terms of the number of more desirable or less desirable shifts assigned in the past. Based on the statistics, the Rostering Engine tries to evenly assign "good" shifts to balance out the statistics. Using fairness measures is another way of improving staff morale. Furthermore, the Rostering Engine also tries to satisfy as many staff preferences and requests as possible without violating the fairness criteria, which provides a better sense of belonging. Due to complexity of all the computations involved, the time needed to produce rosters of this quality manually will be overly time consuming.

- **Facilitate Quality Management**

  Previously, many wards used only manual approaches to staff scheduling with only paper records of staff work assignments and leave schedules. Trying to produce any management statistics or reports from the paper records was difficult and prone to errors. SRS, with its connected databases, allows the Hospital Authority to quickly and accurate produce any type of management reports at a click of a button. This allows senior management instant access to statistics on workforce productivity and utilization for planning and review.

- **Improved Quality of Service**

  The most important objective of any workforce scheduling system is, of course, to improve the quality of service provided by the organization. This quality of service can be ensured by scheduling an adequate

number of staff with a well-balanced set of skills and experiences to handle any potential work that might be needed in the ward. The most important criterion considered by the Rostering Engine is to ensure that all workload requirements are satisfied first.

Application Development and Deployment

The SRS project began in early 1997 and was considered as one of the key strategic IT projects of the Hospital Authority.

The project began with an extensive user requirement study. Several larger hospitals within Hong Kong were selected for requirement analysis. The objective was to be able to obtain a board set of requirements that balanced the needs of a wide variety of wards across Hong Kong.

Based on the results of the requirement study, a set of generalized rules and constraints was extracted that encapsulated a majority of the rostering knowledge used by different wards in Hong Kong. Although the Hospital Authority has a fixed set of rules and guideline governing the scheduling of staff, each ward operates slightly differently and has additional constraints and parameters that must also be considered.

Software and database design of the Staff Rostering System began mid-1997. A set of Booch Diagrams (Booch 1994) was used to document the design of the Rostering Engine. Actual software development began soon after that and lasted roughly nine months. Development was performed in parallel; the Hospital Authority designed and implemented the backend database and the front-end Visual Basic graphic user interfaces, while we focused on the C++ Rostering Engine. Total project team size was roughly fifteen people.

The Rostering Engine was developed using the MS Visual Studio development environment. C++ class libraries from RogueWave (http://www.roguewave.com) were used to create the foundation classes and hospital business objects. Class libraries from ILOG (http://www.ilog.com) were used to implement the CSP algorithm and to provide constraint-programming capabilities.

The Hospital Authority began to field test the Staff Rostering System early 1998 in a few selected sites including Prince of Wales Hospital and Alice Ho Miu Ling Nethersole Hospital. During this period, comments and feedback from end users were used to refine the user interfaces, reporting facilities and the backend rostering component. The Rostering Engine was extended, in terms of its scheduling algorithm and heuristics, to better match the needs for individual test wards but without loosing generality. After half a year of field-testing and refinement, Version 1 of the system was officially deployed in mid-1998.

By end of 1998, SRS was deployed in all the general wards of Alice Ho Miu Ling Nethersole Hospital and all the wards of the Shatin Hospital. The Prince of Wales Hospital also began to roll out SRS in all its wards starting early 1999. Also within 1999, over a hundred seats of SRS were installed at Kwai Chung Hospital and Kwong Wah Hospital. In early 2000, SRS began roll out at Bradbury Hospice and the United Christian Hospital. The Staff Rostering System is now in daily use at seven of the larger hospitals in Hong Kong, with a total application installation base of roughly 250 wards total. The implementation of SRS at other public hospitals are still underway.

The Staff Rostering System is an ongoing HA strategic project. New enhancements and software features were added in 1999 and also planed for 2000. One potential enhancement is departmental-level rostering that further enhances staff utilization by sharing staff across wards depending on workload requirements of the wards and the skill sets of the available staff. There are numerous other areas, within the hospital environment, that can also benefit from AI constraint programming technology. For instance, scheduling and managing operation theatres, radiology rooms and operators, hospital beds, ambulance dispatching, etc. The Hospital Authority is now looking into some of these potential areas as well.

Maintenance

The rules and constraints used by the Rostering Engine were designed to be fully maintainable by the hospital end-user, which are mainly the ward managers. A set of simple-to-use MS Visual Basic screens and menus allow HA staff to quickly and conveniently display and update the knowledge base. All ward-specific knowledge, constraints, parameters and data are stored in the local MS SQL Server database. Changes in the knowledge base to reflect changes in operational needs can be routinely performed without any Rostering Engine source code modification. HA's IT Department provides front-line technical and end-user support while we provide additional assistance whenever needed. The City University of Hong Kong (http://www.cityu.edu.hk) and its subsidiary Advanced Object Technologies Limited (http://www.aotl.com) provides development and consulting services on enhancements to the Rostering Engine.

Conclusion

This paper provided a brief overview of the Rostering Engine, which is part of the Hong Kong Hospital Authority's Staff Rostering System. The Rostering

Engine is the scheduling program that generates a roster using constraint-programming techniques given a set of rostering parameters and constraints. The paper described the general architecture of the SRS and the key constraints considered by the RE. Given that there are over a thousand public hospital wards in Hong Kong, our Rostering Engine might eventually become one of the largest installation of any AI system in the Asia Pacific region.

Acknowledgements

The authors would like to thank the Hospital Authority, Hong Kong for providing us with an opportunity to participate in this crucial project. We would also like to thank HA for allowing us to include information on the Rostering Engine in this paper. In particular, we would like to thank Barbara Kwan and her team members Deirdre Chiu and Derek Tang for suggestions made to an earlier version of this paper.

Research performed was funded in part by a Hong Kong RGC Earmarked Grant and a Strategic Research Grant provided by the City University of Hong Kong.

References

Abdennadher, S. and Schlenker, H. 1999. Nurse Scheduling using Constraint Logic Programming. In *Proceedings of the Eleventh Conference on Innovative Applications of Artificial Intelligence*, 838-843. Menlo Park, Calif.: AAAI Press.

Chun, H.W. 1999. Constraint Programming in Java with JSolver. In *Proceedings of the First International Conference and Exhibition on the Practical Application of Constraint Technologies and Logic Programming*. London.

Cohen, J. 1990. Constraint Logic Programming. *Communications of the ACM* 33(7):52-68.

Booch, G. 1994. *Object-Oriented Analysis and Design with Applications*, 2nd ed. Benjamin/Cummings Publishing Company Inc.

Dresse, A. 1995. A Constraint Programming Library Dedicated to Timetabling. In *Proceedings of the First ILOG Solver and Scheduler Users Conference*. Paris: ILOG.

Kumar, V. 1992. Algorithms for Constraint Satisfaction Problems: A Survey. *AI Magazine* 13(1):32-44.

Kusumoto, S. 1996. Nurse Scheduling System Using ILOG Solver. In *Proceedings of the Second ILOG Solver and Scheduler Users Conference*. Paris: ILOG.

Lau, H.C. and Lau, S.C. 1997. Efficient Multi-Skill Crew Rostering via Constrained Sets. In *Proceedings of the Second ILOG Solver and Scheduler Users Conference*. Paris: ILOG.

Lazaro, J.M. and Aristondo, P. 1995. Using Solver for Nurse Scheduling. In *Proceedings of the First ILOG Solver and Scheduler Users Conference*. Paris: ILOG.

Martello, S. and Toth, P. 1986. A Heuristic Approach to the Bus Driver Scheduling Problem. *European Journal of Operations Research* 24 (1):106-117.

Miller, H.E., Pierskalla, W.P. and Rath, G.J. 1990. Nurse Scheduling Using Mathematical Programming. *Naval Research Logistics* 37:559-577.

Puget, J.-F. 1994. A C++ Implementation of CLP. In *ILOG Solver Collected Papers*. ILOG SA, France.

Randhawa, S.U. and Sitompul, D. 1983. A Heuristic Based Computerized Nurse Scheduling System. *Computers and Operations Research* 20(8):837-844.

E.S. Rosenbloom, E.S. and Goertzen, N.F. 1987. Cyclic Nurse Scheduling. *European Journal of Operations Research* 31:19-23.

Steele, G.L. Jr. 1980. The Definition and Implementation of a Computer Programming Language Based on Constraints, Ph.D. Thesis, MIT.

Van Hentenryck, P. 1989. *Constraint Satisfaction in Logic Programming*, MIT Press.