

Nurse Scheduling System Using ILOG SOLVER

Sachio Kusumoto
NEC Computer Systems, Ltd.
6-1, Omorihoncho 1-chome, Ota-ku Tokyo 143, Japan
TEL:81-3-5762-1151, FAX:81-3-5762-1149
email: kusumoto@osd.necs.nec.co.jp

Introduction

Many hospital wards in Japan schedule nurses based on three rotations per day. Each nurse rotation comprises an eight hour work shift. In addition, since the summer of last year, some wards are introducing two rotations per day. Furthermore, taking into account the skill of the individual nurses, and each of the nurses personal preferences for shifts when generating the master schedule is becoming an increasingly difficult task for the management staff and head nurses. In order to address this problem, we have developed a Nurse Scheduling System named "ProAINUS" using ILOG SOLVER 3.0 libraries.

ProAINUS is a PC based system and runs on Windows and Windows NT. It was easily adopted by hospitals looking for a low cost solution.

System Overview

ProAINUS consists of two main component modules: the first, is the main application component, and the second, is the scheduling engine component (see **Figure 1**).

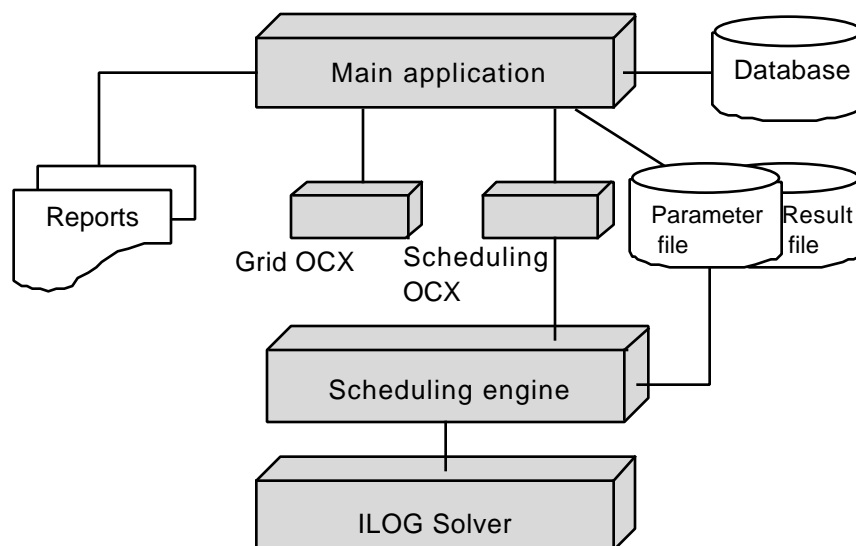


Figure 1. System overview of ProAINUS

The details of each of the components is as follows.

Main Application - the main application component is constructed using a relational database management system: Microsoft Access 2.0 for Windows. In order to generate a optimized schedule, database functions are needed to manage data such as personal data of each of the nurses, hospital records, data pertaining to the wards, statistical data of the shift patterns, and history of work done.

Database - database is defined as the Access tables. The main tables and the relations between the tables are shown in **Figure 2**.

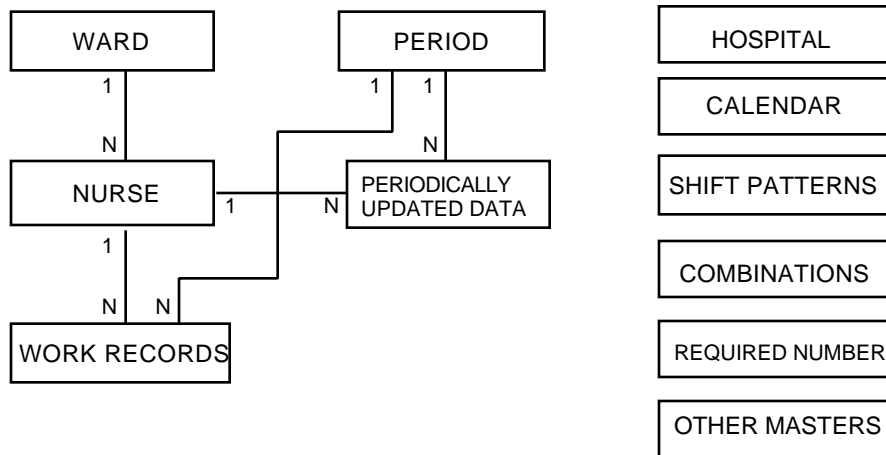


Figure 2. Main tables and the relations between the tables

Reports - reports mainly consists of two outputs: the schedule report and the work record report. In addition, statistical results is printed and used by the management to monitor the workloads.

Grid OCX - in order to display and allow user input of schedule and previous work records, we have developed a grid control as an OCX (OLE custom control) supported in Access 2.0. In our case, 900 cells are required to display the schedule for 30 nurses by 30 days. Using one text box as one cell is not feasible (too many windows); hence, a special OCX was developed.

The results such as the total workload of the individual nurses, daily satisfied requirements, the totals of columns and rows are displayed beside the schedule for ease of monitoring the output. The screen output using the grid OCX is shown in **Figure 3**.

Scheduling OCX - our initial plan was to create a scheduling engine as an OCX. However, with the version of SOLVER at the time, OCX (i.e., DLL) was not supported. Therefore, we have developed a OCX to call the scheduling engine and implemented the schedule engine as a separate executable. This OCX not only invoke the scheduling engine, but it converts Windows messages generated from the engine to Access events.

Scheduling Engine - details in regards to the schedule generating engine will be included later in this document.

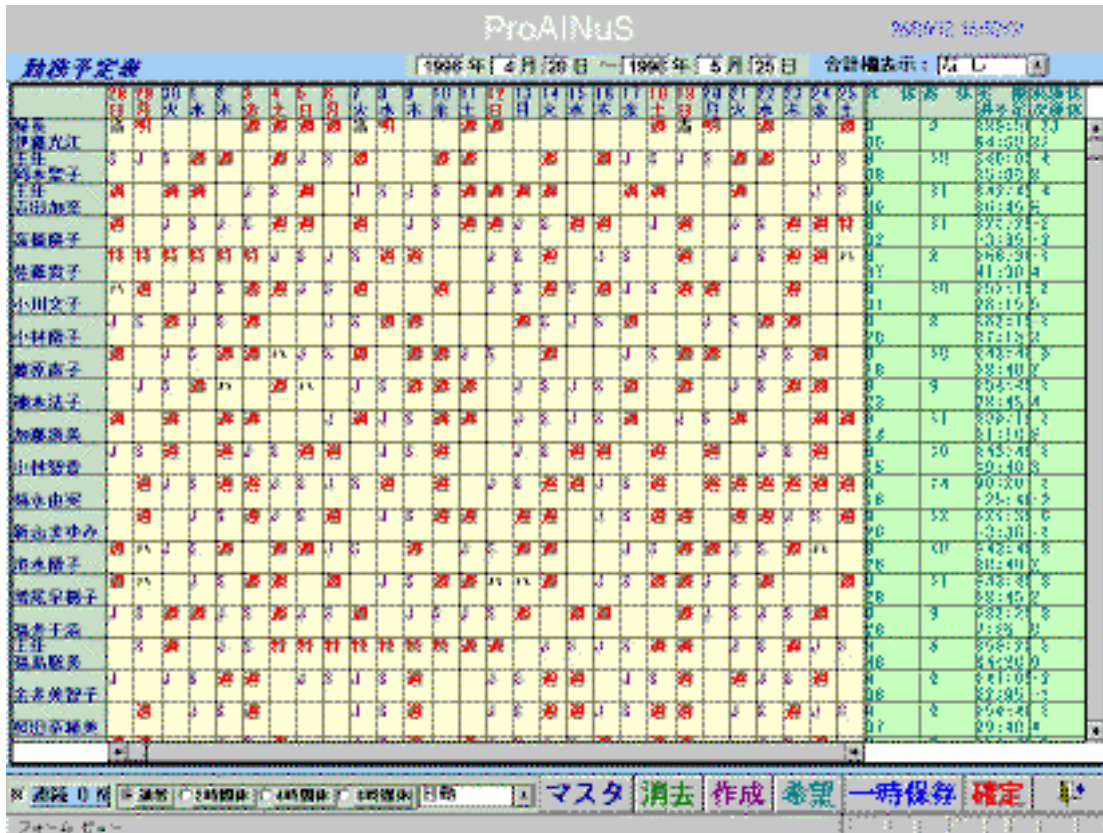


Figure 3. A schedule output form using grid OCX

Parameter File and Result File - these files are exchanged between the main application component and the scheduling engine component. Both are ASCII files and the data passed to the scheduling engine are as follows:

- Individual Data
 - Qualification.
 - Number of days-off allowed.
 - Team the individual belongs to.
 - Accumulated days-off.
 - Restricted dates.
 - Restricted number of shifts
- Compatibility Data
 - Desired combination.
 - Undesired combination.
- Shift Pattern Data
 - Restricted shift pattern.
 - Preferred shift pattern.
- Requirement Data
 - Team requirement.
 - Shift requirement.
 - Daily requirement.
- Calendar Data
 - Holidays, etc.
- Pre-assigned Data
 - Scheduled meetings.
 - Requested days-off, etc.
- Other Data
 - Type of rotation, etc.

Generating the Schedule

Procedures

The general procedure for the operation of ProAINUS is as follows. First, prepare the master data for the hospital such as the hospital holidays, minimum required working days between days-off, minimum required night shifts, etc. Second, the data requested by the ward must be prepared. This data contains information regarding the type of rotation used, minimum required member in a team, preferred shift patterns and member combination, etc. Third, the individual data of the nurses must be prepared. This contains the nurse ID, name, qualifications, skills, date of employment, number of days-off allowed annually, minimum number of night shifts allowed, etc. Finally, after these data preparations, the personal requests such as night shift days and days-off, etc. are entered as pre-assigned schedule.

Constraints Satisfied by the Scheduling Engine

The scheduling engine satisfies the following constraints.

Absolute Constraints

- Daily required number of nurses are allocated
- Leader, assistant leader must be allocated per team.
- Individual nurse restrictions for the number of particular shift are respected.
- Undesirable combinations of nurses are avoided.
- Restricted shift patterns are respected.

Optimization Criteria

- Deviation of work load for each of the nurses is minimized.
- Satisfy as many preferred shift patterns as possible.
- Satisfy as many preferred combinations of members.
- Increase the assignment of days-off for the nurses with accumulated number of annual days-off as much as possible.
- Modify the assignment of days-off according to the previous schedule record of days-off assigned to that particular nurse.
- Distribute each type of shift evenly as much as possible for each of the nurses in a two-week period.

Structure of the Scheduling Engine

The structure of the engine is shown in **Figure 4**. The scheduling engine consists of three SOLVER goals. The first goal, SetConstraint() includes SetStatic() for posting static requirements, and SetDynamic() for posting requirements that are particular to the schedule. The constraints are listed in detail below.

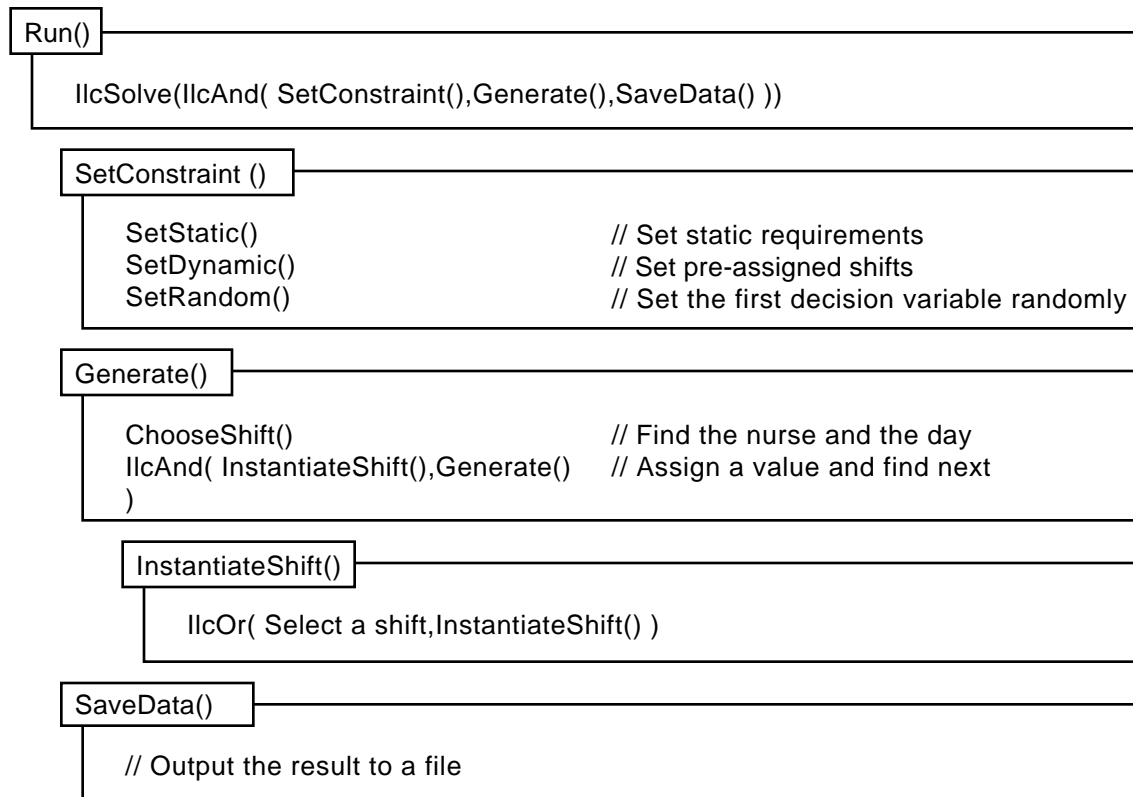


Figure 4. Structure of the schedule engine

SetStatic()

- Required number of nurses.
- Required number of leaders, and assistant leader members.
- Rules compliant with the type of rotation.
- Minimum and Maximum number of shifts to be performed per nurse per period.
- Sub-divide the period into two to restrict deviation of number of shifts assigned to a nurse.
- Restrict certain shift patterns.
- Restrict certain combination of members.
- Sum of each of the shifts for days (columns) must equal the sum of the corresponding shifts for the nurses (rows).

SetDynamic()

- Pre-assigned shifts.

The second goal, Generate() launches the generating engine which iterates the search space. First, ChooseShift() finds the constrained decision variable defined by the nurse with the least workload, and first unassigned day of that nurse. Then, InstantiateShift() goal assigns a value from the constrained variable. This last goal is linked with Generate() by IlcAnd() and the assigning of the values to variables is recursively called until all the variables in the search space are assigned.

In InstantiateShift(), if possible, values that satisfy the preferred patterns, desired combinations are assigned. If pattern with n length cannot be assigned to satisfy the variable chosen, n-1 length of the pattern is tried etc. If none of the pattern or part of the pattern can be assigned to the variable, the least assigned shift so far for the nurse is consecutively tried. This trial and error process is repeated until the possible yet most desired value is assigned to the variable using IlcOr().

The last goal, SaveData() outputs a file containing the scheduling result.

Performance Considerations

The performance of the prototype version of ProAINUS was not satisfactory as a practical application. However, with additional tuning of the search mechanism and insertion of redundant and real constraints, practical performance was achieved.

Additional Constraints

In the prototype, constraint equating the sum of each of the shifts for days (columns) and the sum of the corresponding shifts for the nurses (rows) was not implemented. This was later added to increase performance.

In addition, total number of shift carnality constraint for the period (e.g., 28 days) was divided into two for each of the nurses, which in effect, introduced new constraint consistent with the problem and further increased performance.

First Move

By experimenting with the engine, we were able to determine that the first variable, and consequently the first value choice in the search space greatly affected the efficiency of the search. Currently, we have used this observation to implement the use of random function to choose a variable within the first week period at the beginning of search, then again, randomly assign a value to that first variable chosen. Then the search continues in Generate() until the time limit is reached. When the time limit is reached, the search is re-initialized and a new search begins. The time out procedure is called in two steps. At the first time limit, the engine checks to see how much of the variables have been instantiated. In the second step, the search is stopped and a new search is initiated. If many of the variables have been instantiated, the second limit value is extended by the time proportional to the number of variables instantiated at the first check point. Furthermore, the user can see the search progress in real time using the monitor window, and stop the search at any time.

Conclusions

ProAINUS has been in operation in the pediatrics ward in Toho University Omori Hospital in Tokyo since February of 1996. Originally, two shift rotation was used to generate the schedule. Currently, other wards within the same hospital are using three shift rotation to generate the schedule. After extensive use of the software by these wards, the hospital concluded that the schedule generated by ProAINUS is comparable to those generated manually by a well experienced head nurse.

In the future, we are planning to allow the user to dynamically and seamlessly modify the heuristics by linking the input of the additional constraints with an automatic modification agent in the search mechanism. Further, by using SOLVER 3.1, we are planning to pack the schedule component into an OLE component and allow smoother integration with the main application.

Acknowledgments

We would like to extend our appreciation to ILOG Pte. Ltd. and Bull K.K. for their support. We also thank the staffs of Omori Hospital for their helpful suggestions and comments.