Nurse Rostering Optimization

20-Apr-2019 | Version 1.0

—

Team Name: ai.Orz

Dai Yirui, Dong Meirong, Gu Lijian, Guo Feng, Wong Yoke Keong, Zhang Le

Prepared for NUS-ISS Master of Technology in Intelligent Systems ISY5001 Reasoning System Group Project

Table of Contents

1.0 Executive Summary

An important aspect of hospital operations in Singapore is the effective scheduling of nurses in various wards, which is vital for the hospitals' patients to receive timely care from the nurses with the necessary skill sets. The objective is to balance operational costs and safe patient care needs while maximizing the degree to which nurses' request are met.

However, the scheduling of the nurses' roster is very time-consuming and challenging due to the need to balance the intensive demands of healthcare and the well-being of nurses. In addition, due to the dynamic nature of the workload and unforeseen circumstances, rescheduling is also very common.

Our project focuses on taking the chore out of the painstaking process of scheduling with the objective of reducing the time taken to achieve the best possible schedule in a highly repeatable manner.

2.0 Business Value

Nurse rostering, especially via manual approaches, is a time-consuming and painstaking task to nursing managements due of its complexity. The laborious task can be broken down into several steps.

First is the prediction of service demands. It requires Nursing Management to analyze a long list of reports like ward occupancy pattern and nursing productivity calculation to determine the number of staff needed to deliver safe patient care at different time.

Next is the assignment of staff to shifts and locations based on a set of operational constraints such as staffs' specialties, work agreements, grades and preferences. Meeting these constraints while achieving a satisfying roster can be a tedious and painful process as it requires a lot of communication and coordination.

Final process involves the review of the planned roster to ensure operational costs is minimized while staff request is maximized.

Although there are clear guidelines given at the hospital level to regulate nurse rostering, the process can still be problematic, especially in today's increasingly complex healthcare landscape, with budgets frequently over-run, and staffing levels too often fail to match demands.

In view of these challenges and the market's potential, quite a number of companies design and implement commercial software packages to seize market opportunities. However, many hospitals are still reluctant or even resistant to replace their manual planning process with an automated system, because of high licence and maintenance costs.

In this project, our aim is to apply the knowledge acquired from the Reasoning Systems course to build automated nurse rostering prototypes to facilitate more efficient roster planning. Additionally, we also put some effort into evaluating the benefits and limitations of such systems to help business users to make more informed decisions and to make our project more meaningful.

3.0 Knowledge Specification

In this section, we specify how we collect and represent requirements.

3.1 Knowledge Acquisition

Our team used these approaches to gather requirements for system designs:

- Interviewed domain experts to better understand the planning process;
- Analysed relevant academic researches and industrial reports to better study best practices;

3.2 Knowledge Representation

Project requirements are defined in table formats for easier viewing and understanding.

Scopes Specification

Item	Description
Shift Type and Length	Day, Late, Night or others (user definable)
Roster Days	The roster is constructed based on 24/7 working condition.
Roster Length	4 weeks roster will be planned each time.
Skill Mix	Head nurse/Nurse Manager, Normal nurse and others
Shift Requirements	Each staff will be given a rest day and a day off.
Employment Status	Full Time or Part-time

Hard Constraints

Hard constraints are a set of conditions that must be satisfied.

Item	Description
Working Hours	The working hours shall be regulated in accordance with the present practice on the basis of a maximum of 42 hours (excluding meal breaks) per week.
Leave Request	Staff on planned leave (annual leave, birthday leave, public holiday leave, family care leave, exam leave, maternity leave, long hospitalization leave) must not be rostered with shifts.
Shift Requirements	One staff can only be rostered with one shift per day. The end time of previous shift shall be 10 hours away from the start time of next shift minimally. (This is implemented to ensure night shift

	staff will not be assigned to early or day shift the next day.)
Day Off and Rest Day	Every staff shall be entitled to 1 rest day and 1 day off per week.
Skill Set Requirement	Some shifts require staff with special skills.

Soft Constraints

Soft constraints are a set of conditions that should be satisfied whenever possible.

Item	Description
Skill Utilization	Staff with specialized skill set should not be planned to cover ordinary shift unless there is no other resource available.
Contract Requirements	The schedule should be planned to fulfill the number of consecutive working days falls in a range as stated in staff contract. The planned working days of each staff should not be less than the minimum or more than maximum days stated in staff contract. Same rule applies to minimum or maximum number of rest days.
Day/Shift Off Request	The schedule shall avoid assigning staff to shift on their preferred day/shift off unless there is no other resource available.

Staff Skill Mix Ratio

	Day Shift		Evening Shift		Night Shift	
	Head Nurse	Normal Nurse	Head Nurse	Normal Nurse	Head Nurse	Normal Nurse
Skill Ratio	15%	85%	15%	85%	0%	100%

Staff Preference

- Preferred Shift
- Preferred Day Off
- Non-Preferred Shift
- Non-Preferred Day Off

Contract Requirement

- Minimum and maximum assignments
- Maximum consecutive working days
- Minimum consecutive working days
- Minimum consecutive rest days
- Maximum consecutive rest days

Problem Modeling

Nurse rostering is a two-dimensional timetabling problem that deals with the assignment of nursing staff to shifts across a scheduling period subject to certain constraints.

Shift Planning	Day 1	Day 2	Day 3	...	Day N
Nurse 1	Day	Day	Day off	Rest Day	Night
Nurse 2	Evening	Day	Day	Day	Day
Nurse 3	Night	Night	Night	Day off	Rest Day
Nurse 4	Day	Day	Day	Night	Night
Nurse 5	Evening	Evening	Evening	Evening	Day off
Nurse 6	Night	Night	Rest Day	Night	Night
...	...	...	...	...	...
Nurse N	Day	Evening	Evening	Day off	Evening

4.0 Solution

4.1 Solution Research

The team has done intensive research for the design and architecture for the Nurse Rostering Optimization System. Particularly, the team evaluated:

1) Java Swing Based desktop application for Nurse Rostering, an example in Optaplanner 7.x;
2) Rotabuilder, based on Apache ISIS and Optaplanner 6.4.
3) Optaplanner web version, Optaplanner integrated with comprehensive UI.

Choice of Optaplanner

Optaplanner is chosen because it is an open source software with source codes readily available for learning and prototyping. In addition, it provides the following benefits:

1) Built-in rule engine to accommodate the ever-changing business rules;
2) Score-based solver (Drools) to assign staff to shifts;
3) Supports Multi-threaded Incremental Solving, enabling problem-solving in drastically less time;
4) Support reproducible runs i.e. getting the same results after every run.

Choice of Spring Boot Framework

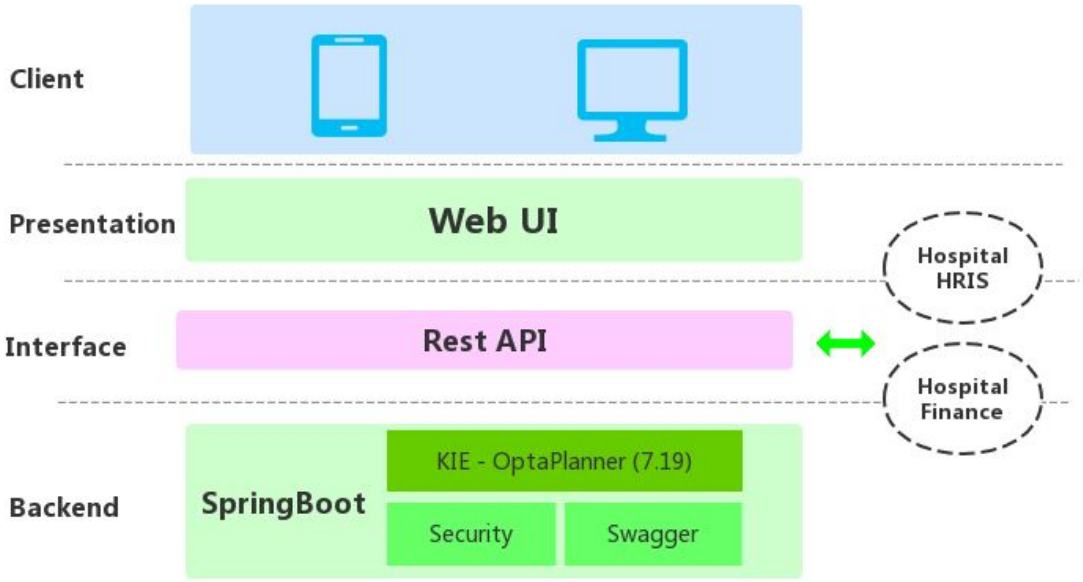
While these examples provide great education value on how Optaplanner works, they fall short in terms of enterprise-grade readiness, where factors like inter-system operability, security and scalability are highly valued. For example, in the financial services industry - a highly regulated industry like healthcare - large banks such as the Development Bank of Singapore (DBS) have

been utilizing cloud native solutions and micro-service oriented architecture like Spring Boot to rapidly introduce features to the market and integrate with existing banking systems.

The team further researched and understood that RedHat Process Automation Manager (the superset of Optaplanner), from version 7.1 release, it supports the deployment of the process automation manager runtime as a “capability” within Spring Boot applications. Thus the team chose to adopt the micro-service oriented architecture to ensure easier and better integration in heavily regulated enterprise environments in hospital and healthcare industry in general.

4.2 System Architecture

The system architecture diagram shown below illustrates how the front-end web platform has been integrated with the back-end optimization system through REST API.

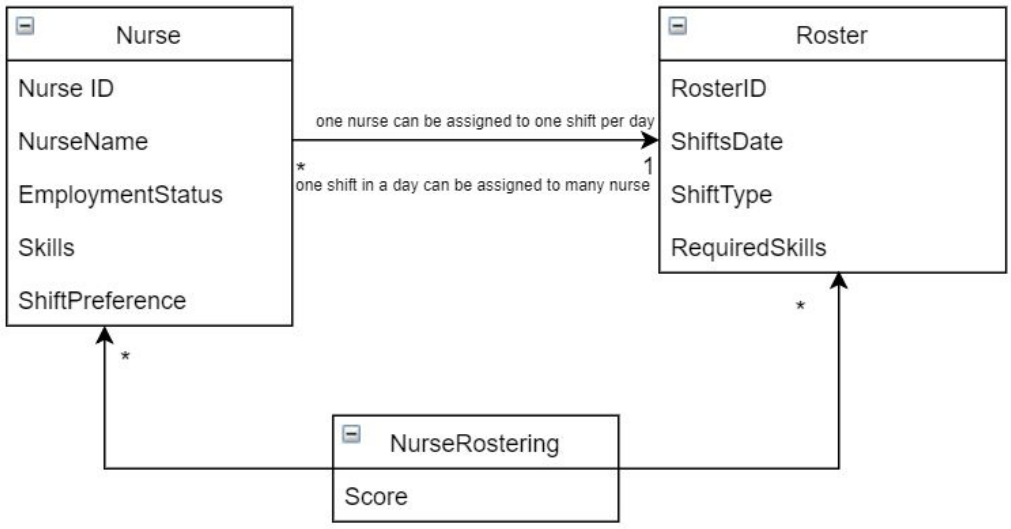


4.3 Solution Modeling

This sections shows how solutions is modelled.

4.3.1 Relationship Model

The Relationship Model is as below. It depicts how a nurse (an employee) is assigned to a shift in a roster based on a scoring system.



4.3.2 Rule and Score Configurations

The sections shows the rules and score configurations.

Rules Configurations

Rules are implemented with Drools DRL in nurseRosterScoreRules.drl (path: src/main/resources/config/) to achieve incremental score calculations and to ensure hard constraints (hard score) and soft constraints (soft score) defined section 3 of the report are not violated.

The rule engine is quite powerful as it enables the solution to have a huge scalability gain without the team having to write a complicated incremental score calculation algorithm in Java.

Score Configurations

The score is calculated based on the weight assigned to each rule in nurseRosterScoreRules.drl.

Weight is defined in data imports. Weight is also user-configurable to enable users to adjust it based on their resource availabilities. For example, when there is a shortage of staff, day off request might be adjusted to a lower a value.

In this project, the higher the score, the more optimal a solution is. Because the scoring system works on a deductive mode. It means when a rule is violated, the weight assigned to the rule will be deducted from the total score.For example: zero means no rule is violated. -1 means a rule with a weight of -1 is violated.

4.4 Tabu Search

For combinatorial problems such as rostering, deriving the best solution usually requires exceptionally large computational times and resources, which is not practical in the real business world. Therefore,heuristic approaches that can produce satisfactory results in reasonably short times are usually adopted for such problems.

We use Tabu Search, one of the most widely utilised heuristic approaches for combinatorial problems solving, in this project. Tabu Search works by maintaining a “tabu size” that holds recently used objects that cannot be reused for N number of moves to avoid getting stuck in local optima.

While hard constraints remained fulfilled, Tabu solutions move in the following way: calculate the best possible move which is not tabu, perform the move and add characteristics of the move to the tabu list.The tabu list stores objects such as the planning entity or planning value.

The objective is to ensure enough nurses on duty at all times while taking account of individual preferences and requests for days off.

Tabu Size in Optaplanner can be configured in nurseRosteringSolverConfig.xml (path: src/main/resources/config/) as below.

```
  <acceptor>
    <entityTabuSize>7</entityTabuSize>
  </acceptor>
  <forager>
    <acceptedCountLimit>800</acceptedCountLimit>
  </forager>
```

4.5 Project Scope and Assumptions

To effectively demonstrate business value and functionalities of the product, the following scope and assumptions have been defined.

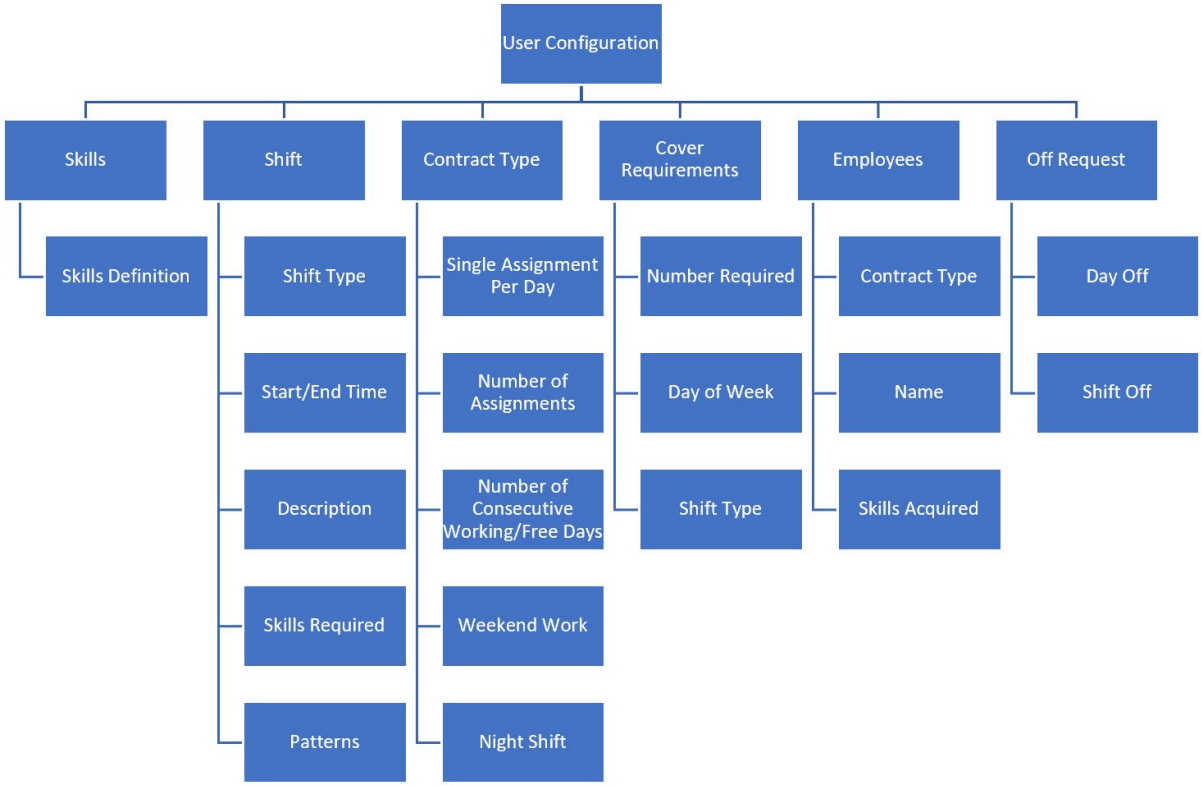
- Geographical context and scheduling constraints has been localized to Singapore.
- For this project, only the rostering for nurses have been considered.
- Planning duration has been defaulted to four calendar weeks. Users can amend the value.

4.6 System Features

Despite many limitations, the team has leveraged on the pros and cons of different technologies to come up with the most optimized design, which can substantially add value to our users, mainly the hospital or wards.

4.6.1 System Intelligence - Constraints Configuration Variables

Most of our constraints variables are user configurable so that users can adjust them accordingly to meet their business needs. These variables will then be used by the constraints solver engine. The below chart shows the major configuration entities that users can change.



4.6.2 Ease of Access

The system front-end is a web interface, which allows user to select data as input for back-end solver to come up with the best solution. In order to free user from tedious data input work, current system only allows user to select from a predefined list of files that include all required details for scheduling. User simply select a file and then wait for the computation to complete. The team has leveraged on the pros and cons of this approach. To allow more interactiveness, users are advised to use the other proposed solution by our team that is based on optaplanner web version.

4.6.3 Integration and Segregation

The front-end web interface integrates with back-end solver via RESTful API. Input data are stored in files that sit on server side, while user can select and pass filename as parameter through URL. The rostering result will return to front-end in JSON format, and will be rendered by HTML and Javascript. Each part of the system serves different purpose and is designed with different degree of complexity and weights.

4.6.4 Scalability

The system is developed as a micro-service that can be scaled to cater to high workloads in the future.

5.0 Limitations and Future Enhancements

5.1 Limitations

Given the limited time and resources, the team has decided to prioritize the project execution in the following order

1) **Understanding of Singapore specific nurse rostering challenges and requirements**
   We took time to understand the nurse rostering challenges from our Healthcare domain experts and defined the hard and soft constraints.

2) **Definition and design of a micro-services oriented system architecture**
   To ensure future enhancements can be added, we evaluated several Optaplanner sample implementations and decided on the Spring Boot framework that is based on the industry standard Micro-services oriented system architecture.

3) **Translation business and technical requirements into Optaplanner implementation**
   The team deep dived into sample Optaplanner implementations to understand its various components and implemented the aforementioned goals in Optaplanner.

4) **User Interface**
   User Interface is more of a prototype currently with a more developed version planned as future components. This section will be further addressed in later sections of the report.

5.2 Future Enhancements

As our system design is based on the micro-services oriented architecture that provides flexibility and scalability, the team is confident that future enhancements in the areas below can potentially be added

1) **Multi-tenancy to support other hospital departments**
   At the current state, the system supports on nurses and hospital wards. The system can be expanded to support other hospital departments such as clinics and & central kitchens, where similar requirements exist.

2) **Near real-time support**
   In the future, near real-time planning can be implemented, where planning can be triggered automatically and roster updated with notifications to affected employees.

3) **Finance System Integration for Cost Optimization**
   Integration with Finance systems can be done so that the optimization model can consider additional finance-related factors with the aim of achieving more operational savings.

4) **Hybrid Reasoning System with ML models for skills use & demand forecasting**
   Machine Learning techniques can be employed to predict demands for number of nurses and their skills. This can then lead to a self-learning hybrid reasoning system where the optimization model can cater to changing demand and skill use trends.

5) **Enhanced User Interface (UI)**
   The team has worked on the enhancing the web UI and additional functions. A demo version can be view and downloaded from the URL below.

   Enhanced UI Prototype Github:
   https://github.com/gu-lijian/IRS-RS-2019-03-09-IS1PT-GRP-ai.Orz-NurseRosteringWeb.git

6.0 Recommendation to Potential Users

After assessing the various products in market, here are some of our observations.

Benefits

- Automated rostering systems significantly lower planning costs and effort by reducing the amount of time required for planners to comply with rostering rules.
- Automated rostering systems reduce the need for communication as users can input their preferences into the system for rostering.
- Automated rostering systems planned shifts based on predefined rules which can help to increase rostering transparency and minimise any resentment over unfaired scheduling.

Drawbacks

- Automated rostering systems cannot eliminate manual inputs completely. For example, users' inputs are still needed for rules definitions and adjustments.
- Automated rostering systems do not guarantee best solutions. They are designed to produce optimal solutions in a given time frame only. Before releasing the solutions to staff, nurse managers must review the outcomes first.
- Automated rostering systems do not automatically predict the amount of nursing resources required unless integrating with a patient acuity predication system. A patient acuity prediction system forecasts nursing resources by analysing ward occupancy pattern and hours of care required by each patient type.

7.0 Conclusion

In this report, the team has demonstrated the business case of why effective nurse rostering is vital for hospital operations and how reasoning system implementations like optimization with Optaplanner helps to achieve business benefits and operational savings.

Furthermore, this report elaborates the benefits and limitations of nurse rostering systems to help audiences to better understand the business values of such systems.

8.0 Bibliography

OptaPlanner (2019, April 19). Employee Rostering Use Case: https://www.optaplanner.org/learn/useCases/employeeRostering.html

Red Hat, OptaPlanner (2019, April 19). Optaplanner User Guide: https://docs.optaplanner.org/7.20.0.Final/optaplanner-docs/html_single/index.html

Optaplanner (2019, April 19). A giant leap forward with multithreaded incremental solving https://www.optaplanner.org/blog/2018/07/03/AGiantLeapForwardWithMultithreadedIncrementalSolving.html

Red Hat Developer (2019, April 19). Spring Boot-enabled business process automation with Red Hat Process Automation Manager: https://developers.redhat.com/blog/2018/11/01/spring-boot-enabled-business-process-automation-with-red-hat-process-automation-manager/

KIE Group Github (2019, April 19). Optaplanner Examples: https://github.com/kiegroup/optaplanner/tree/master/optaplanner-examples

Bilgin Ibryam Github (2019, April 19). Rotabuilder: https://github.com/bibryam/rotabuilder

DBS Newsroom (2019, April 19). DBS partners with Pivotal to accelerate innovation for customers: https://www.dbs.com/newsroom/DBS_partners_with_Pivotal_to_accelerate_innovation_for_customers