

THE VALUE OF FLEXIBILITY IN NURSE SCHEDULING

by

Hasim Turhan

A thesis submitted in partial fulfillment
of the requirements for the degree

of

Master of Science

in

Industrial and Management Engineering

MONTANA STATE UNIVERSITY
Bozeman, Montana

August, 2011

©COPYRIGHT

by

Hasim Turhan

2011

All Rights Reserved

APPROVAL

of a thesis submitted by

Hasim Turhan

This thesis has been read by each member of the thesis committee and has been found to be satisfactory regarding content, English usage, format, citation, bibliographic style, and consistency and is ready for submission to The Graduate School.

Dr. Edward L. Mooney

Approved for the Department of Mechanical and Industrial Engineering

Dr. Christopher H. M. Jenkins

Approved for The Graduate School

Dr. Carl A. Fox

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfillment of the requirements for a master's degree at Montana State University, I agree that the Library shall make it available to borrowers under rules of the Library.

If I have indicated my intention to copyright this thesis by including a copyright notice page, copying is allowable only for scholarly purposes, consistent with "fair use" as prescribed in the U.S. Copyright Law. Requests for permission for extended quotation from or reproduction of this thesis in whole or in parts may be granted only by the copyright holder.

Hasim Turhan

August, 2011

TABLE OF CONTENTS

1. INTRODUCTION	1
2. APPROACH	9
Mathematical Model	9
Test Problems.....	13
3. EXPERIMENTAL RESULTS.....	19
4. SUMMARY AND CONCLUSIONS	30
REFERENCES CITED.....	34
APPENDICES	37
APPENDIX A: Revised Mathematical Model for CPLEX	37
APPENDIX B: OPL Code	41
APPENDIX C: C++ Code to Solve Problems with CPLEX	46
APPENDIX D: Nurse Scheduling Problem Generator.....	49
APPENDIX E: Problem Generator Source Code	66

LIST OF TABLES

Table	Page
1. Problem Classes for Experiment.....	14
2. Results for Uniform Demand.....	21
3. Results for Binomial (0.3) Demand.....	23
4. Results for Binomial (0.5) Demand.....	25
5. Results for Binomial (0.5) Demand.....	27
6. The Effect of the Number of Nurses.....	28
7. The Effect of FTE Mix	29
8. The Effect of Load Factor.....	29
9. Summary of Uniform Demand	30
10. Summary of Binomial (0.3) Demand.....	31
11. Summary of Binomial (0.5) Demand.....	32
12. Summary of Binomial (0.7) Demand.....	32
13. FTE versus Feasible Day Patterns	54
14. List of Day Patterns	55
15. List of Shifts.....	63

LIST OF FIGURES

Figure	Page
1. Contextual Nurse Scheduling Framework	3
2. Problem Generation Flow Diagram	53

ABSTRACT

Nurse scheduling is a complex class of workforce scheduling problems that involves constructing work schedules, or, assigning nurses to shifts to cover demand. Most of the studies of this problem assume fixed shifts and that demand is evenly distributed over the day. These assumptions can reduce utilization of nursing resources. In this thesis, we present a less constrained formulation of the nurse scheduling problem that minimizes staffing and preference costs while allowing intraday demand variability and staggered shifts. We developed a problem generator to create random instances and identified the most significant problem characteristics by preliminary runs. Varying and fixed shift policies were compared based on optimal solutions obtained with CPLEX. We demonstrate that nurse utilization can be improved if shift start times are allowed to vary for some types of intraday demand variability. Additionally, nurse pools with varied shift lengths were shown to improve the utilization.

INTRODUCTION

With expenditures reaching \$2.5 trillion in 2009 (17.6% of the gross domestic product) the health care industry is one of the biggest industries in the United States [1]. The expenditures of the industry have been rising for several years and are expected to reach 19.5% of gross domestic product by 2017 [2]. Despite job losses in nearly all of the major industries due to the recession, health care has created 613,000 jobs since the recession began [3]. This rapid growth has brought about a qualified nursing and technician shortage. The Bureau of Labor Statistics estimated that more than 581,500 new registered nurse positions will be created through 2018 [4]. According to another study by Health Affairs, the US nursing shortage is expected to grow to 260,000 registered nurses by 2025 [5].

A recent study found that lower patient to nurse ratios leads to lower mortality rates [6]. This study emphasizes the importance of nurses in the health care system. However, insufficient staffing is raising the stress level, impacting job satisfaction, and driving many nurses to leave the profession [3].

Given the importance of nurses, the current shortage, and future projections on the shortage of nurses, the issue becomes the management of scarce resources. Finding ways to efficiently schedule nurses will contribute to the well-being of nurses, increase job satisfaction and potentially lead to lower mortality rates. These efforts will also allow the health care decision makers to efficiently utilize their scarce resources.

Workforce scheduling, or rostering, is “the process of constructing work timetables for its staff so that an organization can satisfy the demand for its goods and

services” [7]. That process includes the arrangement of work schedules and staff assignment to the shifts to cover the demand using the variable resources [8]. In service industries, these problems are common and because of the highly constrained structure of these problems, they are very difficult to solve.

In particular, the nurse scheduling problems comprise a complex and a frequently encountered class of workforce scheduling problems. Problems are typically complicated by coverage (demand) and time-related constraints, work regulations, multiple shift types and skill categories as well as additional hard and soft constraints and the requirement to produce flexible, preferred schedules.

Automating nurse scheduling offers many potential benefits. The most significant benefit is the considerable time saving for the hospital managers and planners [9]. In general, this process is performed by a central planning department or by a senior nurse in each department. Automated approaches significantly reduce this process while producing high quality schedules for nurses.

Nurse scheduling involves several phases from macro level planning to daily adjustments. Consequently, there are several decisions made from short to long term planning which impact the decision making process in the other phases. Figure 1 below provides a context for the nurse scheduling framework.

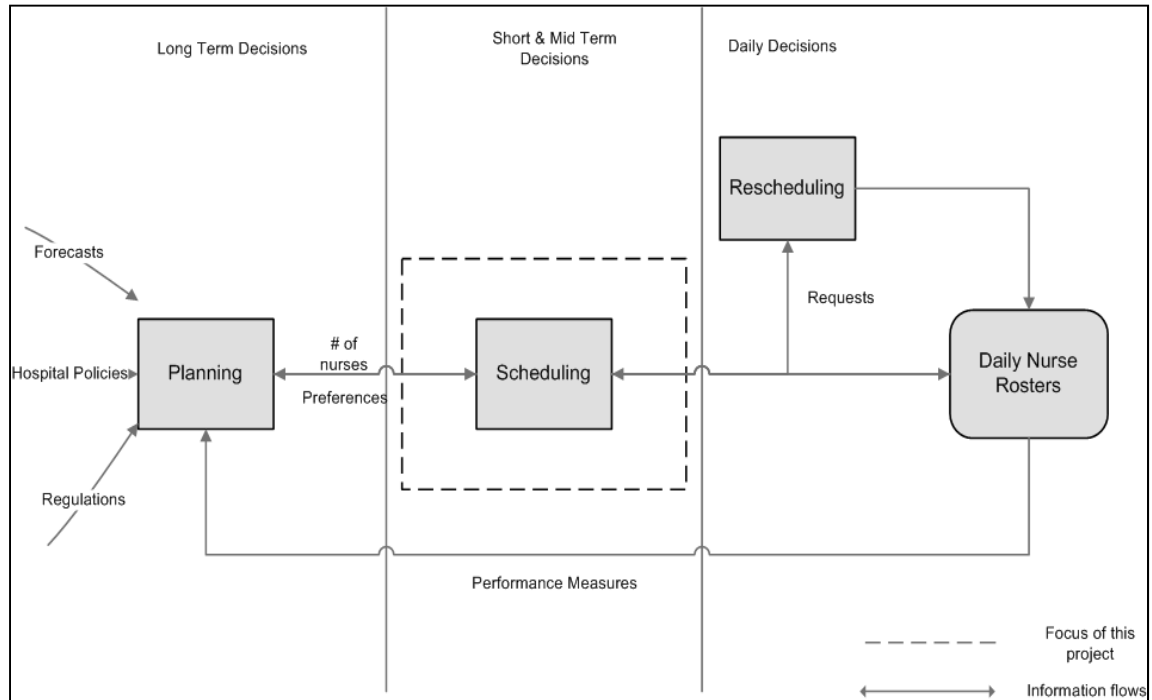


Figure 1. Contextual Nurse Scheduling Framework

In the planning phase, the decisions include how many nurses to dedicate to each ward, the number of nurses to train for management roles, and the number of nurses to recruit from outside sources to meet forecasted service. Multiple factors that affect those decisions include hospital policies, regulations, forecasts, nurse-to-patient ratios, skills of the personnel, etc. Considering the current shortage of nurses and future projections, this problem highly impacts daily nurse scheduling. Nursing resources are scarce and this scarcity is expected to grow in the future [3]. Consequently, planning models have gained importance in the current literature.

Lavieri and Puterman [22] analyzed the nursing human resources planning problem in British Columbia, Canada. The authors proposed a linear programming hierarchical planning model to determine the optimal number of nurses to train, promote

to management and recruit over a 20 year planning period. This article bridged the gap between the future estimates and impact of changes in the input mix by incorporating several factors such as FTE levels, promotion rules, learning, and parental leave in their model.

Once the longer term decisions are made, the next step is to allocate the workforce to the tasks. This widely studied problem is called nurse scheduling, or rostering. This phase involves assignment of individual nurses to shifts and days in a planning period that may range from one to six weeks. This stage can be roughly classified in three major sections: staffing, cyclical scheduling, and preference scheduling. The problems that belong to each class take into account some of the factors that may affect the decisions while keeping the model computationally tractable. Staffing involves determining the number of nurses to meet the predicted demand. There are several factors that affect staffing decisions including organizational structure and characteristics, personnel recruitment, skills of the personnel, and working preferences [10]. Another decision is to define work agreements for part time workers and to decide whether substitution of skill categories is allowed [9].

Maenhout and Vanhoucke [13] investigated the solution approaches for a mid-term period nurse scheduling problem which involves construction of duty timetables for a set of nurses with different grades. The staffing problem is modified to handle specific constraints and objectives. This paper presents an exact branch and price algorithm for solving this problem which is known to be NP-hard. A multiple objective modeling approach is used and the authors discuss different branching and pruning strategies. The

contribution of the article is twofold. First, a branch and price procedure is presented that is able to solve real problem instances exactly. Second, unlike the other exact procedures, this algorithm simultaneously solves multiple conflicting objectives which occur in the heterogeneous workforce. However, staffing models leave out one important characteristic of the nurse scheduling problem which is assigning individual nurses to schedules. The outcome of this method is the number of nurses assigned to each shift on each day.

Another approach to nurse scheduling that addresses the missing component of staffing is cyclical scheduling. In this type of scheduling, a worker works for a cycle of n weeks. The day is split into distinct shifts and the demand pattern is cyclical. This approach has some advantages. First, nurses know their schedule a long time in advance. Second, the work is divided evenly and the same patterns are used repeatedly. Finally, it avoids unhealthy work rotations [9].

Bard and Purnomo [12] investigated the cyclic nurse scheduling. The problem is formulated as an integer program and Lagrangian relaxation-based decomposition techniques are used for solution. Two models were evaluated. In the first model the relaxation was based on preference constraints while the second one was based on demand constraints. Some outstanding results were produced in the end of that study. The results showed that problem instances with up to 100 nurses and 20 rotational profiles could be solved to near optimality in less than 20 minutes. Considering the size and complexity of the problem, those results are outstanding. However, the incorporation of

grades, the start and end time flexibility, and variable shift times may improve the solution in a more realistic way.

A special case of cyclical scheduling is tour scheduling. The goal of tour scheduling is to simultaneously determine daily shift start times and days-worked patterns that minimize total labor hours or cost over a one week planning cycle. The demand and other coverage requirements have to be met for each planning period [11]. The most important feature of this modeling approach is the incorporation of intraday demand variability to the scheduling process.

However, not all nurse schedules are cyclic. Beyond the many advantages, this method does not address personal preferences and it is not flexible in adjusting changes in personnel demand. This reduces the applicability of the method.

One of the most frequently used methods is preference scheduling which incorporates flexibility from the employee point of view. Preference scheduling considers workers' preferences for shift patterns. So, in this approach the objective function should measure the quality of the schedule in terms of the employee preferences. This method has a certain advantage especially in addressing personal requests. Adjustments can easily be made through blocking some previously feasible patterns and rescheduling.

Several authors attempted to solve this problem [19]. Parr and Thompson [16] and Burke et al. [17] and used metaheuristic methods [20] to solve preference scheduling models while Li et al [18] offered a heuristic approach. Parr and Thompson investigated the effectiveness of three metaheuristic techniques based on local search in producing suitable schedules. The main contribution of this paper was to evaluate two infrequently

used local search strategies (SAWing and noising) to solve the nurse scheduling problem. All of those authors proposed fast algorithms to solve the problem but they sacrificed some important problem characteristics especially intraday demand variability.

There are always unforeseen events occurring because of the dynamic nature of a hospital environment. Therefore, the planned schedules are subject to frequent changes which are handled through real-time adjustment and rescheduling efforts. In the literature, this is considered as a separate problem.

There are not many articles focusing on rescheduling because the model is rather small and can be solved efficiently through mathematical programming solvers. Furthermore, in many cases there is not enough time to run a program to handle situations occurring a short time before the shift starts.

Bard and Purnomo [23] addressed this problem and developed an integer programming model which produces a revised schedule that makes the most efficient use of available resources. They solved problem instances up to 120 nurses in a negligible amount of time.

The work that has been done in the literature includes several case studies, mathematical programming formulations, and algorithmic approaches for the solution of the nurse scheduling problem. However, most of these studies assume that nursing demand is evenly distributed over the day and nurses work for only fixed shifts, such as early, day, and night shifts. The fixed shift assumption disregards intraday demand variability, possibly resulting in lower utilization of nurses and increased risk of understaffing. This problem was identified at St. Peter's Hospital, Helena, Montana [24].

The authors presented an integer programming model to align nursing resources with the demand and physician hours. An important element missing in the current research seems to be a general framework which works well under different demand scenarios.

The scheduling phase indicated by the dashed lines in Figure 1 is the focus of this thesis. As discussed above, the nurse scheduling problem has been discussed extensively in the literature. However, the complex nature of the problem led researchers to address only parts of the problem characteristics. Ideas from tour scheduling and preference scheduling approaches were brought together to extend the current types of formulations into a more general one.

This thesis presents a new model of nurse scheduling that allows shift start times to float. We develop a problem generator to create random instances. We then demonstrate the value of this approach by comparing nurse utilization and preference satisfaction with the usual fixed and varying shift policies for various intraday demand patterns. Additionally, we point out the benefits of a flexible workforce in terms of shift lengths and start times.

The remainder of this thesis is organized as follows. We begin by describing the problem, assumptions, the integer programming model, and the approach. We conclude the thesis with a presentation of the experimental results and conclusions.

APPROACH

We began this study by developing an enhanced nurse scheduling model by incorporating intraday demand variability into the traditional sequence-dependent day pattern preference scheduling approach. In this approach, nurses work for a sequence of day and or night shifts and they have preferences associated with their feasible patterns. The goal is to maximize the total nurse preferences while minimizing the total number of nurses scheduled (maximizing nurse utilization). Once this mathematical model was formulated, we then developed a problem generator to create synthetic instances for use in evaluating scheduling policies. The experimental approach was to test the problem instances under two policies, fixed and varying shift start times, and to identify important policy principles and guidelines. We identified important problem and policy factors for the experiments with a series of preliminary runs using different problem instances from those used for the final experiments. Finally, we compared policies and effects based on optimal solutions obtained with CPLEX, integer programming solver. The remainder of this section presents our modeling assumptions, the mathematical model, problem generator, and the experimental approach.

Mathematical Model

Several assumptions were made in developing the model for this study. First, we assume that daily nurse demand is given and it may be spread unevenly within the day. Also, the nurses' start times for a shift may vary and each nurse may have a different start time. For instance, a nurse can start a shift at 7.30 a.m. while another may start at 8.30,

which means shifts may overlap. Related to that, the nurses have different full-time equivalent levels (FTE) which means a nurse can work different shift lengths over the planning period as long as they don't exceed the maximum work-hour requirements stated in the nurse's contract, or as regulated by hospital policies, union agreements, etc. Additionally, we do not consider the different grades that a nurse can have. In practice, some nurses may be qualified to do certain tasks while some are not. In this study, nurses are interchangeable which means all of them can perform all the tasks. Further, we assume the planning term is one week. In other words the number of days, n_d , is equal to seven in this study. In summary, the characteristics of the problem considered in this thesis are:

- Multiple FTE levels of nurses
- Flexible start times
- 24-hour working days
- Known demand

The preferences of the nurses should be considered because of the importance of well-being and motivation of the personnel. The hospital policies, contractual requirements of the personnel, and union agreements have to be met while generating shift schedules for each nurse.

A set of day patterns, P , allows us to implicitly enforce many difficult constraints. A day pattern is defined by a set of *days* (day or night parts) of the week when a nurse will be working. For example, (1111100 0000000) would be a feasible pattern for a full time nurse. The first seven entries correspond to the day shifts while the second seven

entries correspond to the night shifts in one week. In this example, this nurse works day shifts for five days (no night shifts) and takes two full days off. Hospital policies, regulations, and contracts restrict these types of patterns to a certain number. In this study, we consider $n_p = 411$ such patterns. The list of all possible day patterns can be found in Table 14 in Appendix D.

Each nurse should have a maximum of one day pattern assigned drawn from a list of feasible patterns. Each nurse typically can have 50 to 100 feasible patterns. We assume that preference costs, c_{ep} , are available for each nurse and each feasible day pattern. More preferred patterns have lower costs and are drawn from a Uniform (0.0, 2.0) distribution.

On each day a nurse works, a shift, s , has to be assigned. The complete set of shift patterns, S , is given in Table 15 in Appendix D. In contrast to the day patterns, the number possible shifts, $|S_e|$, that a nurse can work is smaller. Since a day is broken in to day and night halves, shifts have to be formed within 12 one-hour time periods. A nurse can work any contiguous block of these periods to fulfill his/her FTE level. For example, (111111110000) forms a shift for a nurse that works for 8 hours. The working hours are contiguous and formed in a 12-hour part of the day. Demand, or hourly nursing requirements, r_{dt} , is given for each one-hour period, t , of each day, d . This requirement simply states the required number of nurses at that period.

The following notation is used in the model.

n_e = the number of nurses

n_d = the number of days

n_p = the number of day patterns

n_s = the number of shifts

n_t = the number of periods

E = the set of nurse indices $\{e: e = 1, 2, \dots, n_e\}$

P = the set of day pattern indices $\{p: p = 1, 2, \dots, n_p\}$

D = the set of day indices $\{d: d = 1, 2, \dots, n_d\}$

S = the set of shift indices $\{s: s = 1, 2, \dots, n_s\}$

T = the set of period indices $\{t: t = 1, 2, \dots, n_t\}$

c_{ep} = the preference cost of assigning nurse e to a day pattern p

r_{dt} = the demand for nurses on period t of day d

S_t = $\{s: \text{period } t \text{ is in shift } s\}$

P_d = $\{p: \text{day } d \text{ is in pattern } p\}$

P_e = $\{p: \text{pattern } p \text{ is feasible for nurse } e\}$

S_e = $\{s: \text{shift } s \text{ is feasible for nurse } e\}$

Now we wish to choose x_{eds} and y_{ep} where:

$$x_{eds} = \begin{cases} 1, & \text{if nurse } e \text{ is assigned to shift } s \text{ on day } d \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ep} = \begin{cases} 1, & \text{if nurse } e \text{ is assigned to day pattern } p \\ 0, & \text{otherwise} \end{cases}$$

to

$$\text{Min } z = \sum_e \sum_p c_{ep} y_{ep} + \sum_e \sum_p y_{ep}$$

Subject To

$$\sum_{e \in E} \sum_{s \in S_e} x_{eds} \geq r_{dt} \quad \forall d, t \quad (1)$$

$$\sum_{s \in S_e} x_{eds} \leq 1 \quad \forall e, d \quad (2)$$

$$\sum_{p \in P_e} y_{ep} \leq 1 \quad \forall e \quad (3)$$

$$\sum_{s \in S_e} x_{eds} = \sum_{p \in P_e \cap P_d} y_{ep} \quad \forall e, d \quad (4)$$

$$x_{eds}, y_{ep} \in \{0,1\}$$

The objective function has two parts; the total preference-based cost of assigning nurses to a day-pattern and the number of nurses scheduled. Constraint (1) is the coverage constraint. The demand must be covered in each period of each day during the week. Constraint (2) limits the shift assignment to a maximum of one for each day for each nurse. When a nurse has to work on a particular day, he/she can only work one shift. Constraint (3) guarantees that at most one day pattern is assigned to each nurse. Constraint (4) guarantees consistency between the day patterns and the shift assignments. If a nurse is assigned a shift on a particular day, the day must be in the selected day pattern and nurse must be assigned days in the selected pattern. x_{eds} variable is not directly used in the objective function but constraint (4) guarantees that this variable is evaluated in the objective function indirectly through y_{ep} variable.

Test Problems

Problem characteristics varied for this study include the number of nurses, load factor, demand distribution, and FTE mix of the nurses. The parameters presented in Table 1 were used by the problem generator to output random instances with specified

characteristics. Other generator parameters, and hence problem characteristics, were fixed based on preliminary experiments. The generator is discussed further later in this section and Appendix D. Among the parameters not included as experimental factors were the range of preferences over day patterns and nurses' day pattern sets according to FTE levels. The problem instances used in the preliminary runs were not used in the final experiment.

Table 1. Problem Classes for Experiment

Factor	Description	Values
# of Nurses	Number of nurses	20, 50, 100
Load factor	The ratio of required nursing hours to available nursing hours	0.5, 0.7
Demand Distribution	Type of demand distribution	Uniform, Binomial(24, 0.3), Binomial(24, 0.5), Binomial(24, 0.7)
FTE Mix	Mixture of the nurses in terms of FTE levels	Triangular, Uniform

We generated three new problem instances of each 48 problem classes shown in Table 1. These instances were solved under two policies, varying and fixed shift start times.

The number of nurses, n_s , establishes, to a large extent, the size of the problem. This factor was set to three levels: 20, 50, and 100 nurses. 20-nurse problems are considered to be small. We restricted problems to be no more than 100 nurses because any number above would be unrealistic.

The load factor is the ratio of required nursing hours to available nursing hours. The load factor has two levels, 0.5 and 0.7. Problem instances with more than 0.7 load

factor are less likely to be feasible. In this case, the issue may be more of a capacity problem than a scheduling problem. The instances below the lower level of 0.5 can be solved very easily because of the excess resources. The numbers in between 0.5 and 0.7 do not provide any additional insight than the load factors we have tested.

The demand distribution can take many forms during the 24-hour day. That's why we only selected a representative set of these forms: discrete uniform and three variants of binomial. In the discrete uniform case, there is no intraday demand variability. In the variants of binomial distribution, demand is skewed on the left or right or centered in the middle. It is important to note that the distribution formulas are used to distribute a previously sampled total 24-hour demand over the 12 periods for each half-day of the week. Discrete uniform and variants of binomial distributions we used to decide what percentage of the daily demand will be allocated to each period. This percentage was equal for each period in the uniform case. However, the percentages were calculated from the binomial function and demand was allocated accordingly to each period. Details of how demand is distributed over periods can be found in Appendix D.

The FTE mix specifies the percentage of the nurses in each of the six FTE considered in this study: 1.0, 0.8, 0.6, 0.5, 0.4, and 0.2. For example, an FTE 0.6 nurse might work either eight hours for three days or six hours for four days in a week. The feasible patterns will be different accordingly. Further, these patterns may include day shifts, night shifts or both. For instance, a nurse might work only day shifts, only night shifts or a combination of day and night shifts without violating policies and contracts. The generator input must allocate a percentage to each of the six options and those values

should sum up to 1. We restricted FTE mix to just two combinations. In practice, FTE 1.0 nurses are more likely to be present. That's why we chose a steep discrete triangular form to represent that instance. We chose a discrete uniform mix to test the effect of a more flexible workforce.

The problem generator performs three steps: (1) nurse pool specification, (2) demand generation, and (3) adjustment, to produce random problem instances. After reading the input parameters described above, the generator outputs the random instances with desired formatting for solution methods.

The nurse pool specification step involves assignment of FTE, preference, and feasible patterns to each nurse. FTE assignment is accomplished by sampling from a discrete uniform or triangular distribution specified by the user. It was assumed that the nurses only have preferences, c_{ep} , over the day patterns and they are indifferent to the shift assignment within the day. Because preliminary runs demonstrated that the range of the preferences over patterns does not affect the solution day pattern preferences were sampled only from the Uniform (0,2) distribution. Day pattern assignments (i.e. feasible pattern sets), P_e , differ according to FTE levels assigned to nurses. We randomly assigned one of the possible pattern sets to a nurse because preliminary runs indicated that the pattern assignment does not have a significant effect on the solution. The table showing the possible feasible day patterns according to FTE levels is presented in Table13 of Appendix D.

The demand generation step mainly focuses on the distribution of demand over the periods to satisfy the load factor requirement specified by the user. To accomplish

that goal, available and required nurse hours were calculated. Available nurse hours is the sum of FTE times 40 hours (weekly full time load) while required nurse hours is available nurse hours times load factor. The nurses were utilized over the scheduling period with the specified load factor. Distribution over the days and periods was the next step of the problem generation. One important assumption made in this step was that the daily demand in a week is uniform and follows the same distribution within the day. Therefore, the required nurse hours for a day were sampled from a discrete uniform distribution. Then using a discrete uniform or binomial function, these daily required nurse hours were spread over 24 periods.

The adjustment step was necessary to check feasibility and consistency metrics for the generated problem instance. One of the feasibility metrics is that the number of nurses in a period of the day cannot exceed the number of available nurses. Another feasibility metric is the sum of the maximum demand in day and night periods. This sum should not exceed the number of nurses available. If any one of these feasibility metrics is not satisfied, demand in these periods is redistributed to other periods in the adjustment step. On the other hand, a consistency metric is the load factor. If the load factor is above or below the specified number because of rounding in the calculation steps, again the adjustment step will increase or decrease the demand in some periods to bring it to the specified level.

For the details of the problem generation process refer to the Appendix D. The source code can be found in Appendix E. The problem generator does not guarantee a feasible solution. However, we intentionally let this happen because some of the data sets

became feasible once we allowed the start times to vary. The results to justify this decision are presented in the results section.

The following section presents the results for the representative data sets. The results were categorized according to the characteristics above and compared for both fixed and varying shift start times.

EXPERIMENTAL RESULTS

In this section, we demonstrate the value of flexibility in nurse scheduling under intraday demand variability. Solutions are presented for a representative set of instances of random nurse scheduling problems. The code was developed using VC++ 2008 Express Edition and run under the release configuration. All runs were made on a CPU with dual 3.20 GHz Pentium 4 processors and 3 gigabytes of system memory. Computation times are reported in CPU seconds.

We solved a total of 144 problems defined by important characteristics under two different policies. We now summarize results comparing utilization and preferences under policies of fixed start time and varying start time. The preference objective values are given per nurse which gives us a normalized value for comparison between problems of different sizes. Results are presented for instances grouped by the demand distribution because the impact of varying start times changes depending on intraday demand distribution. Additionally, the effect of the increasing number of nurses, the effect of a flexible workforce, and the effect of load factor are presented in separate tables.

Tables 2 -5 present results for each of the four demand distribution levels. Columns 1-3 in each table correspond to the other three factors discussed in the previous section. Number of nurses has three levels: 20, 50, and 100 while load factor has levels of 0.5 and 0.7. FTE mix was set to two levels: Tr corresponds to discrete triangular distribution and U corresponds to discrete uniform mix. Three problem instances, or replications, of each combination were solved under two policies, fixed start time and varying start time, and normalized results are presented in utilization and average

preference per nurse columns. Solution time for each problem is presented in CPU seconds for each policy.

Table 2 presents the effects of the shift policy when intraday demand varies uniformly. As it can be observed in Table 2, the utilization and the preference objectives for both policies are equal for most the instances solved to optimality. This means there is no gain from varying shifts when the demand is uniform. In fact, the solution time is slightly longer in most of the instances if we let the shift start times float. This is simply because of the increasing number of nodes that needs to be evaluated by the solver. These results are consistent with our expectations because it is easier to match the demand with supply when the demand is evenly distributed over the periods.

Table 2. Results for Uniform Demand

Experimental Factors			Rep.	Fixed Start Time Policy			Variable Start Time Policy		
# of nurses	Load Factor	FTE Mix		Util.	Pref.	Solution. Time	Util.	Pref.	Solution Time
20	0.5	Tr	1	0.75	0.03	2.78	0.75	0.03	3.91
			2	0.75	0.02	18.45	0.75	0.02	18.86
			3	0.75	0.02	22.20	0.75	0.02	16.48
		U	1	0.82	0.01	17.31	0.82	0.01	26.03
			2	0.88	0.02	1.77	0.88	0.02	3.39
			3	0.81	0.01	20.70	0.81	0.01	23.73
	0.7	Tr	1	0.75	0.03	2.73	0.75	0.03	3.81
			2	0.72	0.04	61.83	0.72	0.04	62.95
			3	0.75	0.02	21.81	0.75	0.02	16.41
		U	1	0.82	0.01	17.13	0.82	0.01	26.39
			2	0.88	0.04	2.98	0.88	0.04	4.30
			3	0.81	0.01	20.88	0.81	0.01	23.77
50	0.5	Tr	1	0.75	0.03	7.11	0.75	0.03	4.05
			2	0.75	0.03	69.75	0.75	0.03	65.45
			3	0.75	0.03	7.08	0.75	0.03	4.05
		U	1	0.88	0.01	306.75	0.88	0.01	303.75
			2	0.88	0.01	8.84	0.88	0.01	5.39
			3	0.92	0.01	306.73	0.92	0.01	303.77
	0.7	Tr	1	0.75	0.04	307.16	0.75	0.04	303.98
			2	0.75	0.04	307.02	0.75	0.03	303.97
			3	0.76	0.03	307.00	0.76	0.03	304.00
		U	1	0.90	0.02	306.84	0.88	0.02	303.80
			2	0.93	0.01	13.92	0.93	0.01	7.44
			3	0.90	0.02	306.94	0.90	0.02	303.86
100	0.5	Tr	1	0.75	0.03	8.03	0.75	0.03	14.34
			2	0.75	0.02	15.75	0.75	0.02	22.52
			3	0.75	0.02	8.31	0.75	0.02	14.42
		U	1	0.89	0.01	607.41	0.88	0.01	613.30
			2	0.92	0.01	607.58	0.94	0.01	613.50
			3	0.87	0.02	499.41	0.88	0.02	20.64
	0.7	Tr	1	0.75	0.03	607.84	0.75	0.03	614.20
			2	0.75	0.03	607.83	0.75	0.03	614.22
			3	0.75	0.03	607.86	0.75	0.03	614.14
		U	1	0.88	0.02	17.77	0.88	0.02	18.61
			2	0.96	0.02	607.67	0.96	0.02	613.89
			3	0.88	0.02	144.92	0.88	0.02	187.00

Similar results are shown in the following tables for different binomial demand specifications. In this group, most of the problem instances can be solved to optimality in both fixed and varying shifts case. However, there are some instances which were infeasible with the fixed shifts case that became feasible once we let the start times vary.

Table 3. Results for Binomial (0.3) Demand

Experimental Factors			Rep.	Fixed Start Time Policy			Variable Start Time Policy		
# of nurses	Load Factor	FTE Mix		Util.	Pref.	Solution. Time	Util.	Pref.	Solution Time
20	0.5	Tr	1	0.56	0.06	2.50	0.61	0.07	9.48
			2	0.56	0.04	11.02	0.63	0.04	4.55
			3	0.56	0.04	2.08	0.62	0.05	7.42
		U	1	0.64	0.05	18.22	0.68	0.05	4.55
			2	0.59	0.07	4.59	0.70	0.03	2.97
			3	0.65	0.07	2.31	0.78	0.05	3.56
	0.7	Tr	1	infeasible		infeasible			
			2	0.71	0.07	4.33	0.71	0.07	4.45
			3	infeasible		infeasible			
		U	1	0.67	0.11	8.41	0.69	0.06	62.94
			2	infeasible		0.69	0.07	3.33	
			3	infeasible		infeasible			
50	0.5	Tr	1	0.61	0.05	24.20	0.68	0.04	177.44
			2	0.60	0.05	9.13	0.72	0.08	11.98
			3	0.62	0.04	303.97	0.68	0.04	7.73
		U	1	0.66	0.05	303.88	0.79	0.04	9.27
			2	0.66	0.03	25.56	0.72	0.08	8.38
			3	0.66	0.04	48.92	0.84	0.02	18.25
	0.7	Tr	1	0.72	0.08	304.03	0.71	0.06	10.63
			2	0.71	0.06	8.44	0.68	0.04	7.66
			3	0.73	0.06	5.03	0.73	0.06	8.81
		U	1	infeasible			0.81	0.03	8.83
			2	0.71	0.06	28.13	0.84	0.03	14.84
			3	infeasible			0.81	0.04	11.28
100	0.5	Tr	1	0.61	0.04	607.77	0.67	0.04	70.19
			2	0.59	0.05	10.03	0.64	0.04	18.95
			3	0.61	0.04	607.78	0.67	0.03	21.36
		U	1	0.68	0.04	51.81	0.77	0.03	15.45
			2	0.68	0.04	551.17	0.85	0.03	34.05
			3	0.66	0.04	607.58	0.74	0.03	613.61
	0.7	Tr	1	0.73	0.07	9.30	0.73	0.07	15.97
			2	0.71	0.06	607.97	0.71	0.06	614.38
			3	0.72	0.06	607.95	0.72	0.06	614.30
		U	1	infeasible			0.70	0.06	16.39
			2	infeasible			0.82	0.04	613.88
			3	infeasible			0.72	0.05	26.25

As shown in Table 3 above, the utilization is improved significantly for all of the instances while the preference objective is improved for most of the instances when shift start times vary. This is because the demand is matched with the supply accordingly. There are a few problem instances which became feasible when the shift start times float.

When the demand is centered in the middle of the day as the results shown in Table 4, there is no significant benefit of staggering shifts. The reason why we don't benefit from staggering shifts is simply because we let the shifts to be formed in 12 hour periods. That's why only few of the shifts could be used in the assignments.

Table 4. Results for Binomial (0.5) Demand

Experimental Factors			Rep.	Fixed Start Time Policy			Varying Start Time Policy		
# of nurses	Load Factor	FTE Mix		Util.	Pref.	Solution Time	Util.	Pref.	Solution Time
20	0.5	Tr	1	0.55	0.04	8.17	0.55	0.04	9.19
			2	0.57	0.03	61.75	0.57	0.03	63.09
			3	0.55	0.03	61.63	0.55	0.03	63.00
		U	1	infeasible			infeasible		
			2	0.53	0.09	1.89	0.53	0.09	3.36
			3	infeasible			infeasible		
	0.7	Tr	1	0.70	0.04	1.98	0.70	0.04	3.27
			2	infeasible			infeasible		
			3	0.70	0.04	2.13	0.70	0.04	3.38
		U	1	infeasible			infeasible		
			2	infeasible			infeasible		
			3	infeasible			infeasible		
50	0.5	Tr	1	0.54	0.04	303.95	0.56	0.03	307.02
			2	0.54	0.04	5.53	0.55	0.04	307.11
			3	0.54	0.03	303.94	0.56	0.03	307.03
		U	1	infeasible			infeasible		
			2	0.54	0.03	303.95	0.54	0.03	307.09
			3	0.54	0.03	5.78	0.55	0.03	9.47
	0.7	Tr	1	infeasible			infeasible		
			2	infeasible			infeasible		
			3	infeasible			infeasible		
		U	1	infeasible			infeasible		
			2	infeasible			infeasible		
			3	infeasible			infeasible		
100	0.5	Tr	1	0.54	0.03	607.88	0.54	0.03	614.25
			2	0.54	0.03	22.81	0.55	0.03	17.25
			3	0.54	0.03	20.77	0.55	0.03	20.47
		U	1	infeasible			infeasible		
			2	0.49	0.03	9.80	0.49	0.03	16.36
			3	0.51	0.04	155.69	0.51	0.04	162.66
	0.7	Tr	1	infeasible			infeasible		
			2	infeasible			infeasible		
			3	infeasible			infeasible		
		U	1	infeasible			infeasible		
			2	infeasible			infeasible		
			3	infeasible			infeasible		

Another observation is that there are more infeasible instances in Binomial(0.5) demand group because the demand is concentrated around the day and night change period. They couldn't be driven to feasibility by varying the start times either because of the half-day structure stated above.

The results presented in Table 5 are consistent with our expectations. Both the utilization and the preference objectives were improved for most of the instances by varying shift start times. Furthermore, all of the infeasible problems became feasible as well.

Table 5. Results for Binomial (0.7) Demand

Experimental Factors				Fixed Start Time Policy			Varying Start Time Policy		
# of nurses	Load Factor	FTE Mix	Rep.	Util.	Pref.	Solution Time	Util.	Pref.	Solution Time
20	0.5	Tr	1	0.67	0.04	2.67	0.67	0.04	5.64
			2	0.63	0.05	1.77	0.69	0.06	3.06
			3	0.67	0.04	2.80	0.67	0.04	4.27
		U	1	0.68	0.03	5.58	0.82	0.03	3.56
			2	0.70	0.04	6.14	0.80	0.04	3.89
			3	0.72	0.05	2.84	0.89	0.02	3.36
	0.7	Tr	1	infeasible			0.71	0.07	10.84
			2	infeasible			0.71	0.06	3.75
			3	infeasible			0.72	0.06	5.70
		U	1	0.71	0.07	4.50	0.82	0.05	3.27
			2	0.72	0.11	1.64	0.80	0.07	3.28
			3	0.71	0.09	2.73	0.77	0.06	3.33
50	0.5	Tr	1	0.61	0.04	11.45	0.68	0.04	22.53
			2	0.63	0.05	303.91	0.68	0.05	8.97
			3	0.61	0.05	179.08	0.68	0.04	171.88
		U	1	0.69	0.04	6.00	0.86	0.03	19.39
			2	0.70	0.03	22.83	0.84	0.02	16.84
			3	0.69	0.03	4.84	0.86	0.02	307.06
	0.7	Tr	1	infeasible			0.74	0.06	75.22
			2	infeasible			0.74	0.08	8.00
			3	infeasible			0.74	0.06	161.77
		U	1	infeasible			0.76	0.05	7.89
			2	0.71	0.07	17.89	0.84	0.03	307.03
			3	0.69	0.07	6.50	0.84	0.03	307.00
100	0.5	Tr	1	0.63	0.06	364.50	0.68	0.04	56.69
			2	0.63	0.05	20.95	0.68	0.04	613.97
			3	0.62	0.06	9.58	0.69	0.04	15.91
		U	1	0.69	0.04	607.52	0.84	0.03	126.00
			2	0.70	0.03	132.33	0.90	0.02	613.58
			3	0.70	0.04	39.19	0.80	0.03	20.92
	0.7	Tr	1	infeasible			0.76	0.06	16.80
			2	infeasible			0.76	0.06	29.17
			3	infeasible			0.76	0.06	20.48
		U	1	0.71	0.07	127.34	0.77	0.05	613.92
			2	0.70	0.08	11.36	0.89	0.03	16.39
			3	infeasible			0.78	0.05	15.53

The average results for utilization and preference objectives are presented in Table 6 to show the effect of the number of nurses, or problem sizes. Each average includes 48 problem instances. Coefficient of variation is also calculated for each average and presented in separate columns for each objective. The effect of the number of nurses is mainly on the solution time. The solution time increases with the increasing number of nurses as shown in Table 6. However, this effect is not very important since most of the instances can be solved in less than five minutes with an optimality gap of less than 1%.

Table 6. The Effect of The Number of Nurses

# of nurses	Fixed Start Time					Varying Start Time				
	Utilization		Preference		Avg. Soln. Time	Utilization		Preference		Avg. Soln. Time
	Avg.	C.V.	Avg.	C.V.		Avg.	C.V.	Avg.	C.V.	
20	0.69	0.14	0.04	0.59	12.35	0.72	0.13	0.04	0.57	13.65
50	0.70	0.16	0.04	0.45	136.92	0.75	0.14	0.04	0.50	125.85
100	0.70	0.17	0.04	0.42	297.98	0.75	0.15	0.04	0.40	234.78

The effects of a more flexible workforce are summarized in Table 7. The results include overall averages for each objective and FTE mix as well as coefficient of variation. Solution times were not affected by FTE mix. 72 problem instances are averaged in each entry of the table. With a workforce where we have all kinds of FTEs, on average we can satisfy the demand with less number of nurses, increasing utilization. On average preference per nurse is higher but the high variation makes it hard to generalize.

Table 7. The Effect of FTE Mix

	Utilization		Preference	
	Avg.	C.V.	Avg.	C.V.
Uniform	0.75	0.12	0.05	0.49
Triangular	0.67	0.14	0.04	0.56

The effect of load factor is presented in Table 8. Load factor has a consistent effect on utilization. With a higher load factor, nurses can be utilized better as expected. More nursing resources will be used if the demand is higher.

Table 8. The Effect of Load Factor

Load Factor	Fixed Start Time					Varying Start Time				
	Utilization		Preference		Avg. Soln. Time	Utilization		Preference		Avg. Soln. Time
	Avg.	C.V.	Avg.	C.V.		Avg.	C.V.	Avg.	C.V.	
0.50	0.67	0.16	0.04	0.41	137.15	0.72	0.17	0.03	0.47	107.97
0.70	0.76	0.10	0.05	0.57	167.49	0.77	0.09	0.04	0.49	150.50

On the other hand, the preference objective is higher on average with a higher load factor. Since the system is highly loaded, nurses are less likely to be assigned to their preferred patterns. Since the variation is very high on preference objective, it is hard to conclude that on average preferences will get worse if the loading is increased.

SUMMARY AND CONCLUSIONS

This paper presented a new model for nurse scheduling which allows staggering shifts to match variable nursing demand with the available nursing hours. This model combines tour scheduling and preference scheduling ideas to minimize the number of nurses assigned while maximizing nurses day pattern preferences. We presented a synthetic nurse scheduling problem generator and documented results for several problem classes.

Comparisons of the results for the fixed and varying shifts demonstrated the value of varying shift start times for some intraday demand distributions. Each of the results presented in Tables 9-12 are the averages and coefficient of variation of utilization and preference objectives for 24 problems. The number of infeasible solutions in 24 instances is presented in a separate column. The percent difference in the objective and solution time under different policies is also displayed in the table.

Table 9. Summary of Uniform Demand

# of nurses	Policy	Utilization		Preference		Avg. Solution Time	# infeasible
		Avg.	C.V.	Avg.	C.V.		
20	Fixed	0.79	0.07	0.02	0.49	17.55	0
	Varying	0.79	0.07	0.02	0.49	19.17	0
	% difference	0.00		0.00		9.24	
50	Fixed	0.83	0.10	0.02	0.45	187.93	0
	Varying	0.82	0.09	0.02	0.44	184.46	0
	% difference	0.00		0.00		-1.85	
100	Fixed	0.83	0.10	0.02	0.34	361.70	0
	Varying	0.83	0.10	0.02	0.34	330.07	0
	% difference	0.00		0.00		-8.75	

Table 9 shows the summary results for discrete uniform demand. As expected, varying start times does not provide any benefits when intraday demand is uniform as presented in Table 9. When demand is centered in the middle of the 24-hour day (symmetric), the problems are more likely to be infeasible as presented in Table 10. Similarly, there are no significant benefits of varying shift start times either. Because of the demand structure, only a few of the start times can be used in each 12-hour day part in the solution which produces very similar results to the fixed policy.

Table 10. Summary of Binomial (0.5) demand

# of nurses	Policy	Utilization		Preference		Avg. Solution Time	# infeasible
		Avg.	C.V.	Avg.	C.V.		
20	Fixed	0.60	0.13	0.04	0.48	22.92	6
	Varying	0.60	0.13	0.04	0.48	24.21	6
	% difference	0.00		0.00		5.62	
50	Fixed	0.54	0.00	0.03	0.20	184.63	7
	Varying	0.55	0.01	0.03	0.16	247.54	7
	% difference	4.80		0.00		45.58	
100	Fixed	0.53	0.05	0.03	0.03	163.39	7
	Varying	0.53	0.05	0.03	0.03	166.20	7
	% difference	0.36		0.00		1.72	

However as shown in Table 11 and 12, varying shift start times increases the utilization of the nurses while improving the preference objective in most of the instances when intraday demand is asymmetric. Moreover, some of the infeasible data sets under fixed start times policy became feasible when the start times were varied.

Table 11. Summary of Binomial (0.3) Demand

# of nurses	Policy	Utilization		Preference		Avg. Solution Time	# infeasible
		Avg.	C.V.	Avg.	C.V.		
20	Fixed	0.62	0.10	0.06	0.37	6.68	4
	Varying	0.68	0.08	0.04	0.67	11.47	3
	% difference	10.17		-34.97		71.70	
50	Fixed	0.67	0.07	0.05	0.26	106.13	2
	Varying	0.75	0.08	0.05	0.40	24.59	0
	% difference	13.06		-1.68		-360.16	
100	Fixed	0.66	0.08	0.05	0.24	406.82	3
	Varying	0.73	0.08	0.04	0.32	222.90	0
	% difference	9.72		-5.30		-45.21	

Table 12. Summary of Binomial (0.7) demand

# of nurses	Policy	Utilization		Preference		Avg. Solution Time	# infeasible
		Avg.	C.V.	Avg.	C.V.		
20	Fixed	0.69	0.04	0.06	0.44	3.41	3
	Varying	0.76	0.09	0.05	0.30	4.50	0
	% difference	9.69		-12.57		31.93	
50	Fixed	0.66	0.06	0.05	0.32	69.06	4
	Varying	0.77	0.09	0.04	0.45	117.80	0
	% difference	15.26		-10.56		70.57	
100	Fixed	0.67	0.06	0.05	0.29	164.10	4
	Varying	0.78	0.09	0.04	0.30	179.95	0
	% difference	15.52		-15.37		9.66	

In summary, it appears that a flexible work force in terms of FTE and start time variability may often lead to better utilization. The intraday demand distribution will affect the value of this strategy. When the intraday demand distribution is symmetric, there is no significant benefit of varying shift start time policy. However, better utilization can be achieved when the intraday demand distribution is asymmetric.

Future work for this study will include the incorporation of grades in the formulation and test the hypothesis under this constraint. This will produce more realistic results because in practice not all the nurses are qualified to do all kinds of jobs. Another consideration will the planning period. Providing solutions for longer planning periods increases the applicability because hospital policies often require the announcement of schedules in advance for 4-6 weeks.

REFERENCES CITED

- [1] Centers for Medicare and Medicaid Services, Office of the Actuary, National Health Statistics Group, [National Health Care Expenditures Data](#), January 2010.
- [2] Keehan, S., Sisko, A., Truffer, C., Smith, S., Cowan, C., Poisal, J., Clemens, M.K, the National Health Expenditure Accounts Projections Team, ["Health Spending Projections Through 2017: The Baby-Boom Generation Is Coming To Medicare"](#), *Health Affairs* Web Exclusive, February, 2008.
- [3] Nursing Shortage Fact Sheet, American Association of College of Nursing, May, 2010
- [4] http://www.bls.gov/emp/ep_table_104.htm , December 2009
- [5] <http://content.healthaffairs.org> , August 2009
- [6] www.nursing.upenn.edu/chopr/Documents/Aiken.2010.CaliforniaStaffingRatios.pdf
- [7] Ernst, A.T., Jiang, H., Krishnamoorthy, M., Sier, D., “Staff Scheduling and Rostering: A Review of Applications, Methods, and Models”, *European Journal of Operational Research*, Vol: 153, pp. 3-27, 2003
- [8] Pinedo, M.L.,”Planning and Scheduling in Manufacturing and Services”, 2nd Edition, Springer, 2009
- [9] Burke, E.K., P. De Causmaecker, G. Vanden Berghe, H. Van Landeghem, “The state of the art of nurse rostering”, *Journal of Scheduling*, Vol:7, pp. 441-499, 2004
- [10] Gray, J.J., McIntire, D., Doller, H.J.,”Preferences for specific work schedules: Foundation for an expert-system scheduling program”, *Computers in Nursing*, Vol: 11, pp. 115-121, 1993
- [11] Isken, M. “An implicit tour scheduling model with applications in healthcare”, *Annals of Operations Research*, Vol: 128, pp. 91-109, 2004
- [12] Bard, J.F., Purnomo, H.W., “Cyclic preference scheduling of nurses using a Lagrangian-based heuristic”, *Annals of Operations Research*, Vol: 155, pp. 279-288, 2007
- [13] Maenhout, B., Vanhoucke, M., “Branching strategies in a branch-and-price approach for a multiple objective nurse scheduling problem”, *Journal of Scheduling*, Vol: 13, pp. 77-93, 2010
- [14] Schaerf, A., Meisels, A., “Solving Employee Timetabling Problems by Generalized Local Search”, *AI*IA*, pp. 380-389, 1999

- [15] Bellanti, F., Carello, G., Della Croce, F., Tadei, R., “A greedy-based neighborhood search approach to a nurse rostering problem”, *European Journal of Operational Research*, Vol: 153, pp. 28-40, 2004
- [16] Parr, D., Thompson, J.M., “Solving the multi-objective nurse scheduling problem with a weighted cost function”, *Annals of Operations Research*, Vol: 155, pp. 279-288, 2007
- [17] Burke, E.K, Li, J., Qu, R.,” A hybrid model of integer programming and variable neighborhood search for highly-constrained nurse rostering problems”, *European Journal of Operational Research*, pp. 484-493, 2010
- [18] Li J., Aickelin U. and Burke E.K., "A Component-Based Heuristic Search Method with Evolutionary Eliminations for Hospital Personnel Scheduling", *INFORMS Journal on Computing*, Vol: 21, pp. 468-479, 2009
- [19] B. Cheang, H. Li, A. Lim, B. Rodrigues, "Nurse Rostering Problems - A Bibliographic Survey", *European Journal of Operational Research*, Vol: 151, pp. 447-460, 2003
- [20] Luke, S., 2009, *Essentials of Metaheuristics*, available at <http://cs.gmu.edu/~sean/book/metaheuristics/>
- [21] Maenhout, B., Vanhoucke, M., “Characterization and Generation of Nurse Scheduling Problem Instances”, Working Paper, 2005
- [22] Lavieri, M.S., Puterman, M.L., “Optimizing nursing human resource planning in British Columbia”, *Health Care Management Science*, Vol: 12, pp. 119-128, 2009
- [23] Bard, J.F., Purnomo, H.W., “Real-Time Scheduling for Nurses to Demand Fluctuations and Personnel Shortages”, *Proceedings of the 5th International Conference on the Practice and Theory of Automated Timetabling*, 2004
- [24] Kindrick, B.E., Lenssen, S.L., “St. Peter’s Hospital Same Day Services Nurse Scheduling Project”, Senior Design Project, Montana State University, 2009

APPENDICES

APPENDIX A

REVISED MATHEMATICAL MODEL FOR CPLEX

A slightly different model is implemented for the CPLEX solver to allow use of arrays to define the sets **S** and **P**. The OPL code and C++ code to solve problem instances with CPLEX presented respectively in Appendices C and D use the notation provided in this appendix.

The following notation is used in this model:

n_e = the number of nurses

n_d = the number of days

n_p = the number of day patterns

n_s = the number of shifts

n_t = the number of periods

NE = the set of nurse indices $\{1, 2, \dots, n_e\}$

NP = the set of day pattern indices $\{1, 2, \dots, n_p\}$

ND = the set of day indices $\{1, 2, \dots, n_d\}$

NS = the set of shift indices $\{1, 2, \dots, n_s\}$

NT = the set of period indices $\{1, 2, \dots, n_t\}$

c_{ep} = the cost of assigning nurse e to a day pattern p

r_{dt} = the demand of nurses on period t of day d

S = Binary 0-1 matrix of all possible shifts where

$$s_{st} = \begin{cases} 1, & \text{if period } t \text{ is in shift } s \\ 0, & \text{otherwise} \end{cases}$$

Note that set $S_t = \{s: \text{period } t \text{ is in shift } s\}$ corresponds to the element in column t of **S**, $S_{\mathbf{t}}$, with a value of 1.

Similarly, **P** matrix is introduced for all day patterns.

$\mathbf{P}$ = Binary 0-1 matrix of all possible day patterns where

$$p_{pd} = \begin{cases} 1, & \text{if day } d \text{ is in pattern } p \\ 0, & \text{otherwise} \end{cases}$$

Set $P_d = \{p: \text{day } d \text{ is in pattern } p\}$ is induced by elements of column d in $\mathbf{P}$, $\mathbf{P}_d$,

with $p_{pd}=1$.

P_e = the set of feasible day patterns for nurse e

S_e = the set of feasible shifts for nurse e

We wish to choose:

$$x_{eds} = \begin{cases} 1, & \text{if nurse } e \text{ is assigned to shift } s \text{ on day } d \\ 0, & \text{otherwise} \end{cases}$$

$$y_{ep} = \begin{cases} 1, & \text{if nurse } e \text{ is assigned to day pattern } p \\ 0, & \text{otherwise} \end{cases}$$

to

$$\text{Min } z = \sum_e \sum_p c_{ep} y_{ep} + \sum_e \sum_p y_{ep}$$

Subject to

$$\sum_{e \in E} \sum_{s \in S_e} x_{eds} * s_{st} \geq r_{dt} \quad \forall d, t \quad (1)$$

$$\sum_{s \in S_e} x_{eds} \leq 1 \quad \forall e, d \quad (2)$$

$$\sum_{p \in P_e} y_{ep} \leq 1 \quad \forall e \quad (3)$$

$$\sum_{s \in S_e} x_{eds} = \sum_{p \in P_e} y_{ep} * p_{pd} \quad \forall e, d \quad (4)$$

$$x_{eds}, y_{ep} \in \{0,1\}$$

APPENDIX B

OPL CODE

This appendix provides OPL Code for the model presented in Appendix B. This code includes the implementation of the mathematical model as well as post processing statements that allows further analysis.

```

/*****
* OPL 6.3 Model
* Author: hasim.turhan
* Creation Date: Jul 6, 2011 at 2:02:39 PM
*****/
int ne = ...; // number of nurses
int nd = ...; // number of days
int np = ...; // number of patterns
int ns = ...; // number of shifts
int nt = ...; // number of periods

range NE = 1..ne;
range ND = 1..nd;
range NP = 1..np;
range NS = 1..ns;
range NT = 1..nt;

int c[NE][NP] = ...; // cost of assigning a pattern
int r[ND][NT] = ...; // demand array

{int} Pe[NE] = ...; // set of feasible patterns for a nurse
{int} St[NE] = ...; // set of feasible shifts for a nurse

int P[NP][ND] = ...; // matrix of all day patterns
int S[NS][NT] = ...; // matrix of all shifts

dvar boolean X[NE][ND][NS]; // decision variable for shift assignment
dvar boolean Y[NE][NP]; // decision variable for pattern assignment

constraint MeetDemand;
constraint OneAssignedPatternforOneNurse;
constraint OneAssignedShiftforOneNurse;
constraint MatchPatternDay;

//=====objective
minimize
    sum (e in NE, p in NP) c[e][p]*Y[e][p] + sum (e in NE, p in Pe[e])
    Y[e][p];

```

```

//=====constraints
subject to{
  MeetDemand =
  forall ( d in ND, t in NT)
    sum (e in NE, s in St[e])
      X[e][d][s]*S[s][t] >= r[d][t];

  OneAssignedShiftforOneNurse =
  forall (e in NE, d in ND)
    sum (s in St[e])
      X[e][d][s] <= 1;

  OneAssignedPatternforOneNurse =
  forall (e in NE)
    sum (p in Pe[e])
      Y[e][p] <= 1;

  // change this to include day and night
  MatchPatternDay =
  forall(e in NE, d in ND){

    sum(p in Pe[e]) (Y[e][p]*P[p][d]) <= sum(s in St[e])X[e][d][s];
    sum(p in Pe[e]) (Y[e][p]*P[p][d]) >= sum(s in St[e])X[e][d][s];
  }

} // end of constraints

execute{// postprocessing
  var e; var p; var d; var s; var t;
  var num_assigned = 0;
  var sum_demand =0;
  var total_cover =0;
  var utilization;
  var preference=0;
  writeln("Nurses Assignments");
  for (e in NE){
    writeln("Nurse,", e);
    for(p in NP){
      if(Y[e][p] == 1){
        writeln("day-pattern,", p);
        for(d in ND){
          write(P[p][d],",");
          for(s in NS){
            if(X[e][d][s]==1){
              for(t in NT){
                write(S[s][t],",");
              }
              writeln();
            }
          }
        }
        if(P[p][d]==0){
          for(t in NT){
            write("0,");

```

```

    }
    writeln();
  }
}

for(e in NE)
  for(d in ND)
    for(s in St[e])
      if(X[e][d][s]==1)
        for(t in NT)
          total_cover += S[s][t]

for (e in NE)
  for(p in NP)
    if(Y[e][p]==1)
      num_assigned++;

for (d in ND)
  for(t in NT)
    sum_demand += r[d][t];
for (e in NE)
  for(d in ND)
    for (s in NS)
      //if(X[e][d][s]==1)
      for(t in NT)
        r[d][t]=r[d][t] - (X[e][d][s]*S[s][t]);
//writeln("Final coverage is :")
for (d in ND){
  write("day", d, " : ")
  for(t in NT){
    write(r[d][t],",");
  }
  writeln();
}

// writeln ("Total demand is ", sum_demand);
// writeln ("Total coverage is ", total_cover);
utilization = sum_demand/total_cover;
// writeln ("utilization: ", utilization);
for (e in NE)
  for (p in NP)
    if(Y[e][p]==1)
      preference += c[e][p]*Y[e][p];
//for(e in NE)
//  for(d in ND)
//    for( s in NS)
//      if(X[e][d][s]==1)
//        preference += b[e][d][s]*X[e][d][s];

//writeln("Total preference = ", preference);

```

```
        writeln("***Summary** ", num_assigned, ",", utilization, ",",  
preference);  
    }
```

APPENDIX C

C++ CODE TO SOLVE PROBLEM INSTANCES WITH CPLEX

This appendix presents the C++ code that is used to solve problem instances with CPLEX solver. This code utilizes the OPL code provided in Appendix B and reads problem instances from .dat files produced by the problem generator. By using the CPLEX solver libraries, the instances are solved and the results are stored in separate output files.

```
// ----- *- C++ *-
// File: nurseSolve.cpp
// -----
// Definition: Program that solves nurse scheduling problem instances
// with IBM ILOG CPLEX solver
// Revisions:
// created, 07/06/2011, Hasim TURHAN
////////////////////////////////////
/////////
#include <iopl/iloopl.h>
#include <ilcplex/ilocplex.h>

#include <sstream>

#ifdef ILO_WINDOWS
#define DIRSEP "\\\"
#else
#define DIRSEP "/"
#endif
#ifndef DATADIR
#define DATADIR "C:" DIRSEP "WINNT" DIRSEP "Profiles" DIRSEP
"hasim.turhan" DIRSEP "Desktop" DIRSEP "OPL" DIRSEP
"nurse_scheduling_07.22" DIRSEP
#endif
#ifndef DATADIR_2
#define DATADIR_2 "C:" DIRSEP "WINNT" DIRSEP "Profiles" DIRSEP
"hasim.turhan" DIRSEP "My" " " "Documents" DIRSEP "Nurse_Scheduling"
DIRSEP "dataGenOPL" DIRSEP "dataGenOPL.v4" DIRSEP
"converted_07.24_100s" DIRSEP
#endif

int main(int argc, char* argv[]) {
    IloEnv env;
    char* filename;
    filename = new char[1000];
    strcpy(filename, DATADIR);
    filename = strcat (filename, "nurse_scheduling_07.06.dat");
    if(argc > 1){
        strcpy(filename, DATADIR_2);
        strcat (filename, argv[1]);
    }
}
```

```

    }

    int status = 127;
    try {
        IloCplex cplex(env);

        cplex.setParam(cplex.TiLim, 600);
        IloOplErrorHandler handler(env,cout);
        IloOplModelSource modelSource(env, DATADIR
"nurse_scheduling_07.22.mod");
        IloOplSettings settings(env,handler);
        IloOplModelDefinition def(modelSource,settings);
        IloOplModel opl(def,cplex);
        IloOplDataSource dataSource(env, filename);
        opl.addDataSource(dataSource);
        opl.generate();

        if ( cplex.solve() ) {

            opl.postProcess();
            cout << "***Solution time," <<
opl.getCplex().getCplexTime()<< endl;
            opl.printSolution(cout);
            status = 0;
        } else {
            cout <<"**Summary," << "infeasible"<< endl;
            cout << "No solution!" << endl;
            status = 1;
        }
    } catch( IloException & e ) {
        cout << "### exception: ";
        e.print(cout);
        cout <<"**Summary," << "exception"<< endl;
        status = 2;
    } catch (...) {
        cout << "### UNEXPECTED ERROR ..." << endl;
        cout <<"**Summary," << "unexpected error!"<< endl;
        status = 3;
    }

    env.end();

    return status;

}

```

APPENDIX D

NURSE SCHEDULING PROBLEM GENERATOR

This appendix provides the details of the problem generation that are not addressed in the body of the thesis. These details include from the motivation to code a problem generator to the specific details of input/output requirements and problem generation steps.

Some authors have pointed out that the lack of benchmark problem sets restricts the development of future nurse scheduling algorithms [9], [21]. There have been recent efforts to overcome this shortcoming in the literature. First, Automated Scheduling Optimization and Planning (ASAP) research group at University of Nottingham created a webpage (<http://www.cs.nott.ac.uk/~tec/NRP/>) to present benchmark problem instances for staff rostering problems. Those instances include several instances of nurse scheduling problems. ASAP also maintains the best known solutions and pointers to the papers where those solutions were presented. Second, a random problem generator called NSPGen was developed recently by [21].

These problem sets and the problem generator were presented for some specific versions of the nurse scheduling problem. That's why, we couldn't directly use any of the data sets for the test cases. For example, NSPGen simply ignores case-specific preference structure such as sequence-dependent preferences (i.e day patterns) which is the fundamental approach that we use in our mathematical model.

The closest problem set that we could use as test problems was presented in the work by [18]. The authors provide the problems on their webpage. However, after careful analysis, we concluded that the problems were not feasible. The required nursing hours were more than available nursing hours but the authors solved these problems with an

unrealistic assumption. The assumption was simply counting even a fractional FTE nurse as full-time nurse and making the shift assignments accordingly. This assumption simply ignores the possibility of uncovered periods.

The restrictions of the current problem sets and unavailability of source code for the current problem generators directed us to develop a new problem generator to produce random instances. Our generator is specifically designed for the purpose of testing the procedure presented in this project. The principle behind generating the data sets can simply be explained in three stages: input, generation, and output. The generation stage is comprised of the nurse pool specification, demand generation, and adjustment steps. The output section simply focuses on the output data formatting.

Input

There are two main complexity indicators considered in this problem generator. Those are the number of nurses and the load factor. The size of the problem instance highly depends on the number of nurses to be scheduled. However, the load factor has a more important effect on the feasibility of the solution. Load factor can be described as the ratio of required nursing hours to available nursing hours. The closer this ratio to 1, the less likely producing a feasible solution to this problem instance.

A user has to specify the discrete distribution parameters to generate data sets with different FTE mixes. The program considers 6 different FTE levels: 1.0, 0.8, 0.6, 0.5, 0.4, 0.2. The input format should allocate a percentage to each of the 6 levels and those values should sum up to 1. If a certain FTE level will not be considered in the data set, it can simply be set to zero. On the other hand, if only a certain type of FTE will be

considered (e.g. all the nurses are full time), then it has to be set to one and all the others have to be set to zero.

Intraday demand distribution is handled in a different fashion. The user can choose between different demand distribution functions. The demand distribution function options are 1 = triangular, 2 = exponential, 3 = uniform, and 4 = binomial. First, the user has to specify the distribution, and then the required parameters. For example, for the binomial function user must enter 4, and then the parameter p which is the probability.

Problem generation process

Figure 1 shows the problem generation process. After reading the input file described in the previous section, the steps followed can be grouped as nurse pool specification, demand generation, and adjustment. The final output is discussed in a separated section.

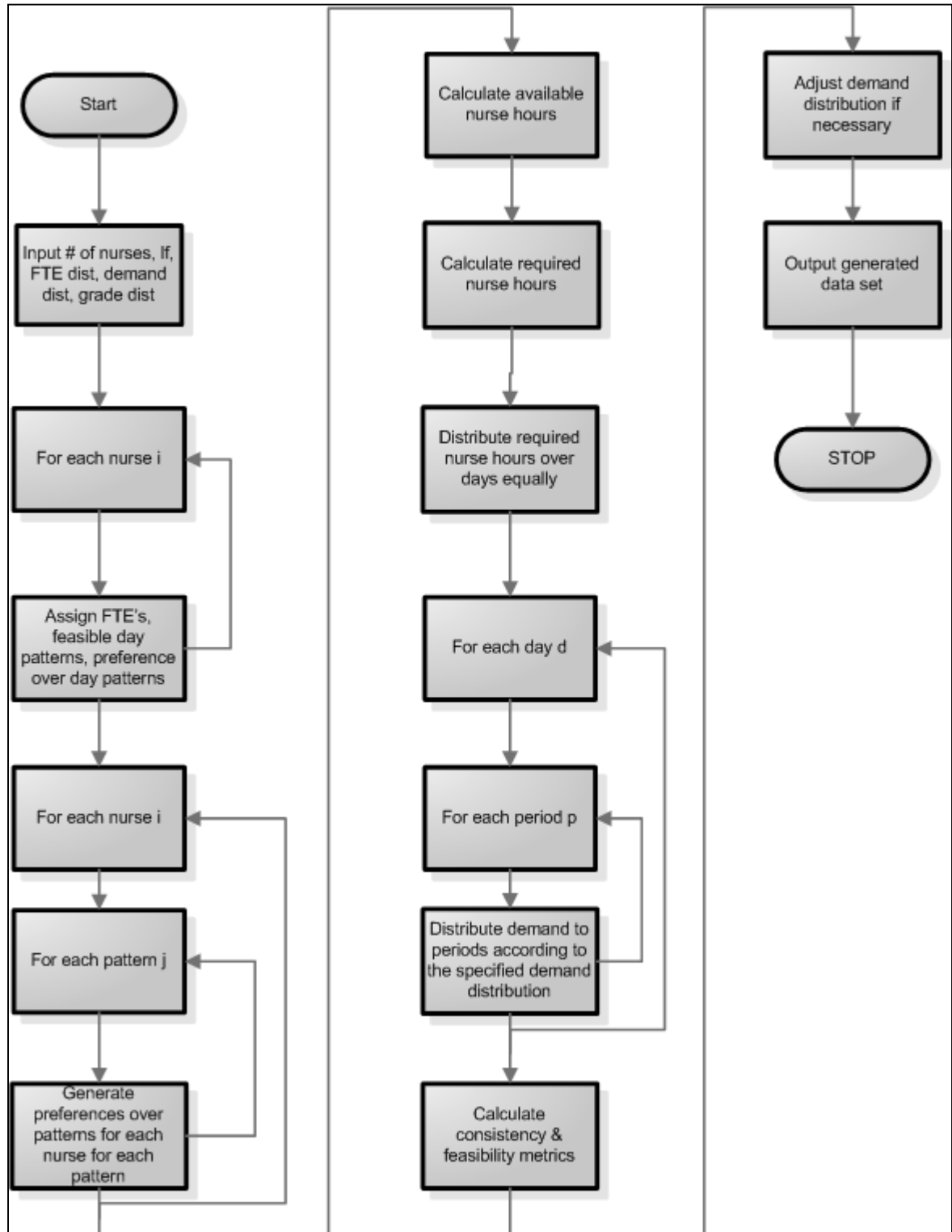


Figure 2. Problem Generation Flow Diagram

The nurse pool specification step involves assignment of FTE, preference, and feasible patterns to each nurse. FTE assignment is accomplished by sampling from a discrete distribution specified by the user. When an FTE is assigned to a nurse, feasible day patterns can be decided accordingly using Table 12.

Table 13. FTE versus Feasible Day Patterns

FTE	Day*Hours	Patterns		
		Day	Night	Mixed
1.0	5x8	1-21	228-247	N/A
0.8	4x8	57-91	22-56	332-411
0.6	3x8	127-161	92-126	257-331
	2x12	183-203	162-182	N/A
0.5	4x5	57-91	22-56	332-411
	5x4	1-21	228-247	N/A
	2x10	183-203	162-182	N/A
0.4	2x8	183-203	162-182	N/A
	4x4	57-91	22-56	332-411
0.2	1x8	211-217	204-210	N/A
	2x4	183-203	162-182	N/A

This table shows the feasible day patterns for each FTE. There may be multiple weekly work structures available for some FTEs. For instance, a 0.6 FTE nurse can work either 8x3 (8 hours for 3 days) or 12x2 (12 hours for 2 days). In that case, one of the working structures is assigned randomly to a nurse. A general preference assignment is performed on day, night, or mixed patterns so that nurses' preferences on each feasible pattern can be adjusted accordingly. For example, if a nurse prefers days over nights, the cost of assigning a nurse to a night pattern will always be higher than assigning to a night

pattern e.g. the costs will be sampled from mutually exclusive ranges. Finally, the nurses' preferences on the individual feasible patterns are assigned generating uniform random numbers from a specified range using the preference information assigned in the previous step. The complete list of patterns is presented in Table 13. Note that Table 13 includes all possible combinations including 6-day patterns. However, in this study we did not use 6-day patterns. The patterns assigned to nurses according to FTE levels were given in Table 12.

Table 14. List of Day Patterns

	Day Shifts							Night Shifts						
i	S	M	T	W	T	F	S	S	M	T	W	T	F	S
1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
2	1	1	1	1	0	1	0	0	0	0	0	0	0	0
3	1	1	1	1	0	0	1	0	0	0	0	0	0	0
4	1	1	1	0	1	1	0	0	0	0	0	0	0	0
5	1	1	1	0	1	0	1	0	0	0	0	0	0	0
6	1	1	1	0	0	1	1	0	0	0	0	0	0	0
7	1	1	0	1	1	1	0	0	0	0	0	0	0	0
8	1	1	0	1	1	0	1	0	0	0	0	0	0	0
9	1	1	0	1	0	1	1	0	0	0	0	0	0	0
10	1	1	0	0	1	1	1	0	0	0	0	0	0	0
11	1	0	1	1	1	1	0	0	0	0	0	0	0	0
12	1	0	1	1	1	0	1	0	0	0	0	0	0	0
13	1	0	1	1	0	1	1	0	0	0	0	0	0	0
14	1	0	1	0	1	1	1	0	0	0	0	0	0	0
15	1	0	0	1	1	1	1	0	0	0	0	0	0	0
16	0	1	1	1	1	1	0	0	0	0	0	0	0	0
17	0	1	1	1	1	0	1	0	0	0	0	0	0	0
18	0	1	1	1	0	1	1	0	0	0	0	0	0	0
19	0	1	1	0	1	1	1	0	0	0	0	0	0	0
20	0	1	0	1	1	1	1	0	0	0	0	0	0	0
21	0	0	1	1	1	1	1	0	0	0	0	0	0	0
22	0	0	0	0	0	0	0	1	1	1	1	0	0	0
23	0	0	0	0	0	0	0	1	1	1	0	1	0	0
24	0	0	0	0	0	0	0	1	1	1	0	0	1	0
25	0	0	0	0	0	0	0	1	1	1	0	0	0	1
26	0	0	0	0	0	0	0	1	1	0	1	1	0	0
27	0	0	0	0	0	0	0	1	1	0	1	0	1	0
28	0	0	0	0	0	0	0	1	1	0	1	0	0	1
29	0	0	0	0	0	0	0	1	1	0	0	1	1	0
30	0	0	0	0	0	0	0	1	1	0	0	1	0	1
31	0	0	0	0	0	0	0	1	1	0	0	0	1	1
32	0	0	0	0	0	0	0	1	0	1	1	1	0	0
33	0	0	0	0	0	0	0	1	0	1	1	0	1	0
34	0	0	0	0	0	0	0	1	0	1	1	0	0	1
35	0	0	0	0	0	0	0	1	0	1	0	1	1	0

36	0	0	0	0	0	0	0	1	0	1	0	1	0	1
37	0	0	0	0	0	0	0	1	0	1	0	0	1	1
38	0	0	0	0	0	0	0	1	0	0	1	1	1	0
39	0	0	0	0	0	0	0	1	0	0	1	1	0	1
40	0	0	0	0	0	0	0	1	0	0	1	0	1	1
41	0	0	0	0	0	0	0	1	0	0	0	1	1	1
42	0	0	0	0	0	0	0	0	1	1	1	1	0	0
43	0	0	0	0	0	0	0	0	1	1	1	0	1	0
44	0	0	0	0	0	0	0	0	1	1	1	0	0	1
45	0	0	0	0	0	0	0	0	1	1	0	1	1	0
46	0	0	0	0	0	0	0	0	1	1	0	1	0	1
47	0	0	0	0	0	0	0	0	1	1	0	0	1	1
48	0	0	0	0	0	0	0	0	1	0	1	1	1	0
49	0	0	0	0	0	0	0	0	1	0	1	1	0	1
50	0	0	0	0	0	0	0	0	1	0	1	0	1	1
51	0	0	0	0	0	0	0	0	1	0	0	1	1	1
52	0	0	0	0	0	0	0	0	0	1	1	1	1	0
53	0	0	0	0	0	0	0	0	0	1	1	1	0	1
54	0	0	0	0	0	0	0	0	0	1	1	0	1	1
55	0	0	0	0	0	0	0	0	0	1	0	1	1	1
56	0	0	0	0	0	0	0	0	0	0	1	1	1	1
57	1	1	1	1	0	0	0	0	0	0	0	0	0	0
58	1	1	1	0	1	0	0	0	0	0	0	0	0	0
59	1	1	1	0	0	1	0	0	0	0	0	0	0	0
60	1	1	1	0	0	0	1	0	0	0	0	0	0	0
61	1	1	0	1	1	0	0	0	0	0	0	0	0	0
62	1	1	0	1	0	1	0	0	0	0	0	0	0	0
63	1	1	0	1	0	0	1	0	0	0	0	0	0	0
64	1	1	0	0	1	1	0	0	0	0	0	0	0	0
65	1	1	0	0	1	0	1	0	0	0	0	0	0	0
66	1	1	0	0	0	1	1	0	0	0	0	0	0	0
67	1	0	1	1	1	0	0	0	0	0	0	0	0	0
68	1	0	1	1	0	1	0	0	0	0	0	0	0	0
69	1	0	1	1	0	0	1	0	0	0	0	0	0	0
70	1	0	1	0	1	1	0	0	0	0	0	0	0	0
71	1	0	1	0	1	0	1	0	0	0	0	0	0	0
72	1	0	1	0	0	1	1	0	0	0	0	0	0	0
73	1	0	0	1	1	1	0	0	0	0	0	0	0	0
74	1	0	0	1	1	0	1	0	0	0	0	0	0	0
75	1	0	0	1	0	1	1	0	0	0	0	0	0	0
76	1	0	0	0	1	1	1	0	0	0	0	0	0	0
77	0	1	1	1	1	0	0	0	0	0	0	0	0	0
78	0	1	1	1	0	1	0	0	0	0	0	0	0	0
79	0	1	1	1	0	0	1	0	0	0	0	0	0	0
80	0	1	1	0	1	1	0	0	0	0	0	0	0	0
81	0	1	1	0	1	0	1	0	0	0	0	0	0	0
82	0	1	1	0	0	1	1	0	0	0	0	0	0	0
83	0	1	0	1	1	1	0	0	0	0	0	0	0	0
84	0	1	0	1	1	0	1	0	0	0	0	0	0	0
85	0	1	0	1	0	1	1	0	0	0	0	0	0	0
86	0	1	0	0	1	1	1	0	0	0	0	0	0	0
87	0	0	1	1	1	1	0	0	0	0	0	0	0	0
88	0	0	1	1	1	0	1	0	0	0	0	0	0	0
89	0	0	1	1	0	1	1	0	0	0	0	0	0	0
90	0	0	1	0	1	1	1	0	0	0	0	0	0	0
91	0	0	0	1	1	1	1	0	0	0	0	0	0	0
92	0	0	0	0	0	0	0	0	0	0	0	1	1	1
93	0	0	0	0	0	0	0	0	0	0	1	0	1	1
94	0	0	0	0	0	0	0	0	0	0	1	1	0	1
95	0	0	0	0	0	0	0	0	0	0	1	1	1	0

96	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
97	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	1
98	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0
99	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	1
100	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	0
101	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	0
102	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	1
103	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	1
104	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0
105	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	1
106	0	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0
107	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0
108	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1
109	0	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
110	0	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
111	0	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
112	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	1
113	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	1
114	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
115	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	1
116	0	0	0	0	0	0	0	0	1	0	0	1	0	1	0	0
117	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	0
118	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	1
119	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0
120	0	0	0	0	0	0	0	0	1	0	1	0	1	0	0	0
121	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0
122	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	1
123	0	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0
124	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0
125	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0	0
126	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0	0
127	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0	0
128	0	0	0	1	0	1	1	1	0	0	0	0	0	0	0	0
129	0	0	0	1	1	0	1	1	0	0	0	0	0	0	0	0
130	0	0	0	1	1	1	0	1	0	0	0	0	0	0	0	0
131	0	0	1	0	0	1	1	1	0	0	0	0	0	0	0	0
132	0	0	1	0	1	0	1	1	0	0	0	0	0	0	0	0
133	0	0	1	0	1	1	0	1	0	0	0	0	0	0	0	0
134	0	0	1	1	0	0	1	1	0	0	0	0	0	0	0	0
135	0	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0
136	0	0	1	1	1	0	0	1	0	0	0	0	0	0	0	0
137	0	1	0	0	0	1	1	1	0	0	0	0	0	0	0	0
138	0	1	0	0	1	0	1	1	0	0	0	0	0	0	0	0
139	0	1	0	0	1	1	0	1	0	0	0	0	0	0	0	0
140	0	1	0	1	0	0	1	1	0	0	0	0	0	0	0	0
141	0	1	0	1	0	1	0	1	0	0	0	0	0	0	0	0
142	0	1	0	1	1	0	0	1	0	0	0	0	0	0	0	0
143	0	1	1	0	0	0	1	1	0	0	0	0	0	0	0	0
144	0	1	1	0	0	1	0	1	0	0	0	0	0	0	0	0
145	0	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0
146	0	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0
147	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
148	1	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
149	1	0	0	0	1	1	0	1	0	0	0	0	0	0	0	0
150	1	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0
151	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
152	1	0	0	1	1	0	0	1	0	0	0	0	0	0	0	0
153	1	0	1	0	0	0	1	1	0	0	0	0	0	0	0	0
154	1	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0
155	1	0	1	0	1	0	0	1	0	0	0	0	0	0	0	0

[illegible]

216	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
217	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
218	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
219	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
220	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
221	1	1	1	1	1	0	1	0	0	0	0	0	0	0	0
222	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
223	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0
224	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0
225	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0
226	0	1	1	1	1	1	1	0	0	0	0	0	0	0	0
227	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
228	0	0	0	0	0	0	0	1	1	1	1	0	1	0	0
229	0	0	0	0	0	0	0	1	1	1	1	0	0	1	0
230	0	0	0	0	0	0	0	1	1	1	0	1	1	0	0
231	0	0	0	0	0	0	0	1	1	1	0	1	0	1	0
232	0	0	0	0	0	0	0	1	1	1	0	0	1	1	0
233	0	0	0	0	0	0	0	1	1	0	1	1	1	0	0
234	0	0	0	0	0	0	0	1	1	0	1	1	0	1	0
235	0	0	0	0	0	0	0	1	1	0	1	0	1	1	0
236	0	0	0	0	0	0	0	1	1	0	0	1	1	1	0
237	0	0	0	0	0	0	0	1	0	1	1	1	1	0	0
238	0	0	0	0	0	0	0	1	0	1	1	1	0	1	0
239	0	0	0	0	0	0	0	1	0	1	1	0	1	1	0
240	0	0	0	0	0	0	0	1	0	1	0	1	1	1	0
241	0	0	0	0	0	0	0	1	0	0	1	1	1	1	0
242	0	0	0	0	0	0	0	0	1	1	1	1	1	0	0
243	0	0	0	0	0	0	0	0	1	1	1	1	0	1	0
244	0	0	0	0	0	0	0	0	1	1	1	0	1	1	0
245	0	0	0	0	0	0	0	0	1	1	0	1	1	1	0
246	0	0	0	0	0	0	0	0	1	0	1	1	1	1	0
247	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0
248	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
249	0	0	0	0	0	0	0	1	1	1	1	1	1	0	0
250	0	0	0	0	0	0	0	1	1	1	1	1	0	1	0
251	0	0	0	0	0	0	0	1	1	1	1	0	1	1	0
252	0	0	0	0	0	0	0	1	1	1	0	1	1	1	0
253	0	0	0	0	0	0	0	1	1	0	1	1	1	1	0
254	0	0	0	0	0	0	0	1	0	1	1	1	1	1	0
255	0	0	0	0	0	0	0	0	1	1	1	1	1	1	0
256	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
257	1	0	0	0	0	0	0	0	1	1	0	0	0	0	0
258	1	0	0	0	0	0	0	0	1	0	1	0	0	0	0
259	1	0	0	0	0	0	0	0	1	0	0	1	0	0	0
260	1	0	0	0	0	0	0	0	1	0	0	0	1	0	0
261	1	0	0	0	0	0	0	0	1	0	0	0	0	1	0
262	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0
263	1	0	0	0	0	0	0	0	0	1	0	1	0	0	0
264	1	0	0	0	0	0	0	0	0	1	0	0	1	0	0
265	1	0	0	0	0	0	0	0	0	1	0	0	0	1	0
266	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0
267	1	0	0	0	0	0	0	0	0	0	1	0	1	0	0
268	1	0	0	0	0	0	0	0	0	0	1	0	0	1	0
269	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
270	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
271	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
272	0	1	0	0	0	0	0	0	0	1	1	0	0	0	0
273	0	1	0	0	0	0	0	0	0	1	0	1	0	0	0
274	0	1	0	0	0	0	0	0	0	1	0	0	1	0	0
275	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0

276	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	0
277	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1	0
278	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	1
279	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1	0
280	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	1
281	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	1
282	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0	0
283	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0
284	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1	0
285	0	0	1	0	0	0	0	0	1	0	0	0	0	0	0	1
286	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0	0
287	0	0	1	0	0	0	0	0	0	0	0	1	0	1	0	0
288	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	1
289	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1	0
290	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	1
291	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	1
292	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0	0
293	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0	0
294	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1	0
295	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1
296	0	0	0	1	0	0	0	0	0	1	0	0	1	0	0	0
297	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1	0
298	0	0	0	1	0	0	0	0	0	1	0	0	0	0	0	1
299	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0
300	0	0	0	1	0	0	0	0	0	0	0	0	0	1	0	1
301	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
302	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0	0
303	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0	0
304	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0
305	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	1
306	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0	0
307	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1	0
308	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	1
309	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1	0
310	0	0	0	0	1	0	0	0	0	0	1	0	0	0	0	1
311	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1
312	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0	0
313	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0	0
314	0	0	0	0	0	1	0	0	1	0	0	1	0	0	0	0
315	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1
316	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0	0
317	0	0	0	0	0	1	0	0	0	1	0	1	0	0	0	0
318	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1
319	0	0	0	0	0	1	0	0	0	0	1	1	0	0	0	0
320	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	1
321	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0	1
322	0	0	0	0	0	0	1	1	1	1	0	0	0	0	0	0
323	0	0	0	0	0	0	1	1	0	1	0	0	0	0	0	0
324	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0
325	0	0	0	0	0	0	1	1	0	0	0	0	1	0	0	0
326	0	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0
327	0	0	0	0	0	0	1	0	1	0	1	0	0	0	0	0
328	0	0	0	0	0	0	1	0	1	0	0	1	0	0	0	0
329	0	0	0	0	0	0	1	0	0	1	1	0	0	0	0	0
330	0	0	0	0	0	0	1	0	0	1	0	1	0	0	0	0
331	0	0	0	0	0	0	1	0	0	0	1	1	0	0	0	0
332	1	0	0	0	0	0	0	0	0	1	1	1	0	0	0	0
333	1	0	0	0	0	0	0	0	0	1	1	0	1	0	0	0
334	1	0	0	0	0	0	0	0	0	1	1	0	0	1	0	0
335	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	1

336	1	0	0	0	0	0	0	0	0	1	0	1	1	0	0
337	1	0	0	0	0	0	0	0	0	1	0	1	0	1	0
338	1	0	0	0	0	0	0	0	0	1	0	1	0	0	1
339	1	0	0	0	0	0	0	0	0	1	0	0	1	1	0
340	1	0	0	0	0	0	0	0	0	1	0	0	1	0	1
341	1	0	0	0	0	0	0	0	0	1	0	0	0	1	1
342	1	0	0	0	0	0	0	0	0	0	1	1	1	0	0
343	1	0	0	0	0	0	0	0	0	0	1	1	0	1	0
344	1	0	0	0	0	0	0	0	0	0	1	1	0	0	1
345	1	0	0	0	0	0	0	0	0	0	1	0	1	1	0
346	1	0	0	0	0	0	0	0	0	0	1	0	1	0	1
347	1	0	0	0	0	0	0	0	0	0	1	0	0	1	1
348	1	0	0	0	0	0	0	0	0	0	0	1	1	1	0
349	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1
350	1	0	0	0	0	0	0	0	0	0	0	1	0	1	1
351	1	0	0	0	0	0	0	0	0	0	0	0	1	1	1
352	0	1	0	0	0	0	0	0	0	0	1	1	1	0	0
353	0	1	0	0	0	0	0	0	0	0	1	1	0	1	0
354	0	1	0	0	0	0	0	0	0	0	1	1	0	0	1
355	0	1	0	0	0	0	0	0	0	0	1	0	1	1	0
356	0	1	0	0	0	0	0	0	0	0	1	0	1	0	1
357	0	1	0	0	0	0	0	0	0	0	1	0	0	1	1
358	0	1	0	0	0	0	0	0	0	0	0	1	1	1	0
359	0	1	0	0	0	0	0	0	0	0	0	1	1	0	1
360	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1
361	0	1	0	0	0	0	0	0	0	0	0	0	1	1	1
362	0	0	1	0	0	0	0	0	1	0	0	1	1	0	0
363	0	0	1	0	0	0	0	0	1	0	0	1	0	1	0
364	0	0	1	0	0	0	0	0	1	0	0	1	0	0	1
365	0	0	1	0	0	0	0	0	1	0	0	0	1	1	0
366	0	0	1	0	0	0	0	0	1	0	0	0	1	0	1
367	0	0	1	0	0	0	0	0	1	0	0	0	0	1	1
368	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0
369	0	0	1	0	0	0	0	0	0	0	0	1	1	0	1
370	0	0	1	0	0	0	0	0	0	0	0	1	0	1	1
371	0	0	1	0	0	0	0	0	0	0	0	0	1	1	1
372	0	0	0	1	0	0	0	0	1	1	0	0	1	0	0
373	0	0	0	1	0	0	0	0	1	1	0	0	0	1	0
374	0	0	0	1	0	0	0	0	1	1	0	0	0	0	1
375	0	0	0	1	0	0	0	0	1	0	0	0	1	1	0
376	0	0	0	1	0	0	0	0	1	0	0	0	1	0	1
377	0	0	0	1	0	0	0	0	1	0	0	0	0	1	1
378	0	0	0	1	0	0	0	0	0	1	0	0	1	1	0
379	0	0	0	1	0	0	0	0	0	1	0	0	1	0	1
380	0	0	0	1	0	0	0	0	0	1	0	0	0	1	1
381	0	0	0	1	0	0	0	0	0	0	0	0	1	1	1
382	0	0	0	0	1	0	0	0	1	1	1	0	0	0	0
383	0	0	0	0	1	0	0	0	1	1	0	0	0	1	0
384	0	0	0	0	1	0	0	0	1	1	0	0	0	0	1
385	0	0	0	0	1	0	0	0	1	0	1	0	0	1	0
386	0	0	0	0	1	0	0	0	1	0	1	0	0	0	1
387	0	0	0	0	1	0	0	0	1	0	0	0	0	1	1
388	0	0	0	0	1	0	0	0	0	1	1	0	0	1	0
389	0	0	0	0	1	0	0	0	0	1	1	0	0	0	1
390	0	0	0	0	1	0	0	0	0	1	0	0	0	1	1
391	0	0	0	0	1	0	0	0	0	0	1	0	0	1	1
392	0	0	0	0	0	1	0	0	1	1	1	0	0	0	0
393	0	0	0	0	0	1	0	0	1	1	0	1	0	0	0
394	0	0	0	0	0	1	0	0	1	1	0	0	0	0	1
395	0	0	0	0	0	1	0	0	1	0	1	1	0	0	0

396	0	0	0	0	0	1	0	1	0	1	0	0	0	1
397	0	0	0	0	0	1	0	1	0	0	1	0	0	1
398	0	0	0	0	0	1	0	0	1	1	1	0	0	0
399	0	0	0	0	0	1	0	0	1	1	0	0	0	1
400	0	0	0	0	0	1	0	0	1	0	1	0	0	1
401	0	0	0	0	0	1	0	0	0	1	1	0	0	1
402	0	0	0	0	0	0	1	1	1	1	0	0	0	0
403	0	0	0	0	0	0	1	1	1	0	1	0	0	0
404	0	0	0	0	0	0	1	1	1	0	0	1	0	0
405	0	0	0	0	0	0	1	1	0	1	1	0	0	0
406	0	0	0	0	0	0	1	1	0	1	0	1	0	0
407	0	0	0	0	0	0	1	1	0	0	1	1	0	0
408	0	0	0	0	0	0	1	0	1	1	1	0	0	0
409	0	0	0	0	0	0	1	0	1	1	0	1	0	0
410	0	0	0	0	0	0	1	0	1	0	1	1	0	0
411	0	0	0	0	0	0	1	0	0	1	1	1	0	0

Table 15 below presents the list of shifts used in this study.

Table 15. List of Shifts

[illegible]

The demand generation step mainly focuses on the distribution of demand over the periods to satisfy the load factor requirement specified by the user. To accomplish that goal, available and required nurse hours are calculated. Available nurse hours is the sum of FTE times 40 hours (weekly full time load) while required nurse hours is available nurse hours times load factor. So, the nurses will be utilized over the scheduling period with the specified load factor.

Distribution over the days and periods will be next step of the demand generation. One important assumption made in this step is the daily demand is uniform within days of the week and follows the same distribution. So, the required nurse hours are sampled from a discrete uniform distribution. Then using a uniform or binomial function, this daily required nurse hours are distributed over 24 periods.

The final step is to check feasibility and consistency metrics for the generated data set. One of the feasibility metrics is the number of nurses in a period of the day cannot exceed the number of available nurses. If this happens, demand in these periods is redistributed to other periods in the adjustment step. A consistency metric is the load factor. If the load factor is above or below the specified number because of rounding in the calculation steps, again the adjustment step will increase or decrease the demand in some periods to bring it to the specified level.

Output

The problem instance includes number of nurses, demand information per period, and the nurse information. Nurse information is composed of the nurse id, FTE, grade, feasible day pattern upper and lower bounds, preferences on the feasible day patterns, feasible intraday lower and upper bounds, and preferences on the feasible intraday patterns. The data generator outputs the generated data set to different file formats for future analysis. The first format is .out format which is used by the algorithm. The second format is OPL Studio format where the small test problems can be tested. The third format is .csv format which allows us to analyze the data set using a spreadsheet.

APPENDIX E

PROBLEM GENERATOR SOURCE CODE

This appendix presents C++ code of the problem generator that is discussed in detail in Appendix D. This code includes the header and cpp files. Further explanations of the implementation are provided as comments within the code.

```

/*-----
Class of Nurse
Each nurse has a number, grade, fte level, lower and upper bound on the patterns
that they can work for, and preferences

Revisions:
    created by Hasim TURHAN, 12/25/2010
-----*/
#include <vector>

using namespace std;

class Nurse{
public:
    short i;      //nurse number
    short grade;  //grade of the nurse
    short fte;    //fte level of the nurse
    short pi;     //p(i) nurses' preference over day and night shifts
    short d_lb;   //lower bound of day patterns
    short d_ub;   //upper bound of day patterns
    short n_lb;   //lower bound of night patterns
    short n_ub;   //upper bound of night patterns
    short m_lb;   //lower bound of mixed patterns
    short m_ub;   //upper bound of mixed patterns
    short intra_lb; //lower bound of intra-day shift patterns
    short intra_ub; //upper bound of intra-day shift patterns
    short p_assign; // assigned pattern number
    short intra_assign; // assigned intra_day pattern

    vector <float> d_preference; //nurses preference on day shifts
    vector <float> n_preference; // nurses preference on night shifts
    vector <float> m_preference; // nurses preference on mixed shifts
    vector <short> intra_preference; // nurses preference on intra-day patterns

    Nurse(short);
};

```

```

/*-----
    This is a data generator for the nurse scheduling problem with intra-day
    demand variability and nurse preferences over shift and day patterns

Revisions:
    created by Hasim TURHAN, 01/31/2011
    functions for file output, 06/29/2011
License:
    RandNumGen files are Copyrighted and licensed under the
    Gnu General Public License License. Programs come with ABSOLUTELY
    NO WARRANTY. See Library folders for details.
-----*/

#include <iostream>
#include <stdlib.h>
#include <fstream>
#include <iomanip>
#include <vector>
#include <string>
#include <sstream>
#include <math.h>
#include "nurse.h"
using namespace std;

// Application headers
#include "random.hpp"

#define PERIOD 24
#define PATTERN 411
#define SHIFT 41
#define DAY 7
#define NFTE 6
#define GRADE 3

double Factorial(double val);
double Combination(double N, double R);
string getFileName_out(int N, int mixed, float loadf, short d_dist, float p, long
seed, float d_fte[NFTE]);
string getFileName_dat(int N, int mixed, float loadf, short d_dist, float p, long
seed, float d_fte[NFTE]);
int output(ofstream &nurseoutput, int N, short demand[7][24], vector <Nurse*>
*nurse);
int calc_demand_param(short demand[7][24]);

Rand *X; // Create a pointer to Rand(om) process
Rand *Y; // will be used for continuous uniform

// == Can change seeds
DefaultSeed *MyDefaultSeed = new DefaultSeed( (long) 0 ); // built in default
long seed = 12345; // seed for the run.

int main(int argc, char *argv[]){

```

```

int N; // number of available nurses
int unavail; // number of unavailable nurses for that week
int i, j, k; // general purpose index
int mixed; //0: Mixed 1:Fixed
short demand[DAY][PERIOD];
float daydemand[DAY];
short id;
short z;
float x,y;
short fte[NFTE] = {40, 32, 24, 20, 16, 8}; // full time equivalents of
fte's
float d_fte[NFTE]; // fte distribution
float d_grade[GRADE];
short sumfte = 0;
short d_dist; // demand distribution 1:triangular, 2: exponential
int a, b, c; // triangular distribution parameters
int n = 24; // binomial parameter = number of periods
float p, prob; // binomial parameter
float reqnurse_hrs = 0;
float loadf;
string nurseinput, filename_out, filename_dat, directory;
string directory1 = "Nursedata/";
string directory2 = "Nursedata_rev/";
ofstream convert("convert.txt");
//cout << "number of arguments: " << argc << endl;
//for(i=0; i<argc ; i++)
//    cout << "Argument #" << i << " = " << argv[i] << endl;
nurseinput = "nurse.nur";

if ( argc > 1 ){
    nurseinput = argv[1];
    seed = atol ( argv[2] );
    filename_out = nurseinput + argv[2]+ ".out" ;
}

MyDefaultSeed->SetSeed(seed);
vector<Nurse*> *nurse;
vector<Nurse*> *unavailNurse;

X = new RandUniform(0.0,1.0);
Y = new RandUniform(0.0,2.0);

nurse = new vector<Nurse*>;
unavailNurse = new vector<Nurse*>;

// input # of nurses, load factor, demand distribution, parameters etc
ifstream nursefile (nurseinput);
if(nursefile.is_open()){
    nursefile >> N; // read in number of nurses
    nursefile >> mixed; // read fixed shifts or mixed?
    nursefile >> loadf; // read in load factor
    for(i=0; i<NFTE; i++){
        nursefile >> d_fte[i]; // read in fte dist
    }
}

```

```

nursefile >> d_dist; // read in demand distribution
if(d_dist == 1)
    nursefile >> a >> b >> c; // read in triangular dist
parameters.
if(d_dist == 4)
    nursefile >> p;
for(i=0; i<GRADE; i++)
    nursefile >> d_grade[i];
}
else{
    cout<< "ERROR in opening the file" << endl;
    exit(1);
}

filename_out = getFileName_out(N,mixed,loadf,d_dist,p,seed, d_fte);

ofstream nurseoutput (filename_out);

//cout << filename_out << endl;

// assign fte's to nurses
for(i=0; i<N; i++){
    nurse->push_back(new Nurse(i+1));
    x = X->RandValue();
    if(x<d_fte[0])
        (*nurse)[i]->fte = 1;
    else if(x>=d_fte[0] && x<d_fte[1])
        (*nurse)[i]->fte = 2;
    else if(x>=d_fte[1] && x<d_fte[2])
        (*nurse)[i]->fte = 3;
    else if(x>=d_fte[2] && x<d_fte[3])
        (*nurse)[i]->fte = 4;
    else if(x>=d_fte[3] && x<d_fte[4])
        (*nurse)[i]->fte = 5;
    else
        (*nurse)[i]->fte = 6;
}

// assignment of nurses to pattern sets according to ftes
for(i=0; i<N; i++){
    if((*nurse)[i]->fte ==1){
        (*nurse)[i]->d_lb = 1;
        (*nurse)[i]->d_ub = 21;
        (*nurse)[i]->n_lb = 228;
        (*nurse)[i]->n_ub = 247;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
    if((*nurse)[i]->fte ==2){
        (*nurse)[i]->d_lb = 57;
        (*nurse)[i]->d_ub = 91;
        (*nurse)[i]->n_lb = 22;
        (*nurse)[i]->n_ub = 56;
    }
}

```

```

        (*nurse)[i]->m_lb = 332;
        (*nurse)[i]->m_ub = 411;
    }
    if((*nurse)[i]->fte ==3){
        if(X->RandValue() <= 0.5){
            (*nurse)[i]->d_lb = 127;
            (*nurse)[i]->d_ub = 161;
            (*nurse)[i]->n_lb = 92;
            (*nurse)[i]->n_ub = 126;
            (*nurse)[i]->m_lb = 257;
            (*nurse)[i]->m_ub = 331;
        }
        else{
            (*nurse)[i]->d_lb = 183;
            (*nurse)[i]->d_ub = 203;
            (*nurse)[i]->n_lb = 162;
            (*nurse)[i]->n_ub = 182;
            (*nurse)[i]->m_lb = 0;
            (*nurse)[i]->m_ub = 0;
        }
    }
}
if((*nurse)[i]->fte ==4){
    x = X->RandValue();
    if(x <= 0.33){
        (*nurse)[i]->d_lb = 57;
        (*nurse)[i]->d_ub = 91;
        (*nurse)[i]->n_lb = 22;
        (*nurse)[i]->n_ub = 56;
        (*nurse)[i]->m_lb = 332;
        (*nurse)[i]->m_ub = 411;
    }
    else if(x > 0.33 && x <= 0.67){
        (*nurse)[i]->d_lb = 1;
        (*nurse)[i]->d_ub = 21;
        (*nurse)[i]->n_lb = 228;
        (*nurse)[i]->n_ub = 247;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
    else{
        (*nurse)[i]->d_lb = 183;
        (*nurse)[i]->d_ub = 203;
        (*nurse)[i]->n_lb = 162;
        (*nurse)[i]->n_ub = 182;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
}
}
if((*nurse)[i]->fte ==5){
    if(X->RandValue() <= 0.5){
        (*nurse)[i]->d_lb = 57;
        (*nurse)[i]->d_ub = 91;
        (*nurse)[i]->n_lb = 22;
        (*nurse)[i]->n_ub = 56;
        (*nurse)[i]->m_lb = 332;
    }
}

```

```

        (*nurse)[i]->m_ub = 411;
    }
    else{
        (*nurse)[i]->d_lb = 183;
        (*nurse)[i]->d_ub = 203;
        (*nurse)[i]->n_lb = 162;
        (*nurse)[i]->n_ub = 182;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
}
if((*nurse)[i]->fte ==6){
    x = X->RandValue();
    if(x <= 0.33){
        (*nurse)[i]->d_lb = 57;
        (*nurse)[i]->d_ub = 91;
        (*nurse)[i]->n_lb = 22;
        (*nurse)[i]->n_ub = 56;
        (*nurse)[i]->m_lb = 332;
        (*nurse)[i]->m_ub = 411;
    }
    else if(x > 0.33 && x <= 0.67){
        (*nurse)[i]->d_lb = 211;
        (*nurse)[i]->d_ub = 217;
        (*nurse)[i]->n_lb = 204;
        (*nurse)[i]->n_ub = 210;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
    else{
        (*nurse)[i]->d_lb = 183;
        (*nurse)[i]->d_ub = 203;
        (*nurse)[i]->n_lb = 162;
        (*nurse)[i]->n_ub = 182;
        (*nurse)[i]->m_lb = 0;
        (*nurse)[i]->m_ub = 0;
    }
}
}

// assignment of nurses preferences over patterns
for(i=0; i<N; i++){
    x= X->RandValue();
    for(j=0; j<(((*nurse)[i]->d_ub-(*nurse)[i]->d_lb +1); j++){
        //y = floor(X->RandValue()*2+0.5);
        y= Y->RandValue();
        (*nurse)[i]->d_preference.push_back(y);
    }
    for(j=0; j<(((*nurse)[i]->n_ub-(*nurse)[i]->n_lb +1); j++){
        //y = floor(X->RandValue()*2+0.5);
        y = Y->RandValue();
        (*nurse)[i]->n_preference.push_back(y);
    }
    for(j=0; j<(((*nurse)[i]->m_ub-(*nurse)[i]->m_lb +1); j++){
        //y = floor(X->RandValue()*4 +0.5);

```

```

        y= Y->RandValue();
        (*nurse)[i]->m_preference.push_back(y);
    }
    if(x<0.33){ // if this nurse prefers day shifts
        (*nurse)[i]->pi =1;
        // for(j=0; j<((*nurse)[i]->d_ub-(*nurse)[i]->d_lb +1); j++){
        //     y = floor(X->RandValue()*50);
        //     (*nurse)[i]->d_preference.push_back(y);
        // }
        // for(j=0; j<((*nurse)[i]->n_ub-(*nurse)[i]->n_lb +1); j++){
        //     y = 50 + floor(X->RandValue()*50);
        //     (*nurse)[i]->n_preference.push_back(y);
        // }
        // for(j=0; j<((*nurse)[i]->m_ub-(*nurse)[i]->m_lb +1); j++){
        //     y = floor(X->RandValue()*100);
        //     (*nurse)[i]->m_preference.push_back(y);
        // }
    }
    else if(x>=0.33 && x<0.67){// if this nurse prefers night shifts
        (*nurse)[i]->pi = 2;
        // for(j=0; j<((*nurse)[i]->d_ub-(*nurse)[i]->d_lb +1); j++){
        //     y = 50+floor(X->RandValue()*50);
        //     (*nurse)[i]->d_preference.push_back(y);
        // }
        // for(j=0; j<((*nurse)[i]->n_ub-(*nurse)[i]->n_lb +1); j++){
        //     y = floor(X->RandValue()*50);
        //     (*nurse)[i]->n_preference.push_back(y);
        // }
        // for(j=0; j<((*nurse)[i]->m_ub-(*nurse)[i]->m_lb +1); j++){
        //     y = floor(X->RandValue()*100);
        //     (*nurse)[i]->m_preference.push_back(y);
        // }
    }
    else{// if this nurse doesn't have any preference
        (*nurse)[i]->pi = 3;
        // for(j=0; j<((*nurse)[i]->d_ub-(*nurse)[i]->d_lb +1); j++){
        //     (*nurse)[i]->d_preference.push_back(0);
        // }
        // for(j=0; j<((*nurse)[i]->n_ub-(*nurse)[i]->n_lb +1); j++){
        //     (*nurse)[i]->n_preference.push_back(0);
        // }
        // for(j=0; j<((*nurse)[i]->m_ub-(*nurse)[i]->m_lb +1); j++){
        //     (*nurse)[i]->m_preference.push_back(0);
        // }
    }
}
if(mixed == 0){
    // intra-day shift patterns
    for(i=0; i<N; i++){ // for each nurse
        if((*nurse)[i]->fte == 1){
            (*nurse)[i]->intra_lb = 1;
            (*nurse)[i]->intra_ub = 5;
        }
        if((*nurse)[i]->fte == 2){
            (*nurse)[i]->intra_lb = 1;

```

```

        (*nurse)[i]->intra_ub = 5;
    }
    if((*nurse)[i]->fte == 3){
        if((*nurse)[i]->d_lb == 127){
            (*nurse)[i]->intra_lb = 1;
            (*nurse)[i]->intra_ub = 5;
        }
        else{
            (*nurse)[i]->intra_lb = 10;
            (*nurse)[i]->intra_ub = 10;
        }
    }
    if((*nurse)[i]->fte == 4){
        if((*nurse)[i]->d_lb == 57){
            (*nurse)[i]->intra_lb = 14;
            (*nurse)[i]->intra_ub = 21;
        }
        else if((*nurse)[i]->d_lb == 1){
            (*nurse)[i]->intra_lb = 22;
            (*nurse)[i]->intra_ub = 30;
        }
        else{
            (*nurse)[i]->intra_lb = 11;
            (*nurse)[i]->intra_ub = 13;
        }
    }
    if((*nurse)[i]->fte == 5){
        if((*nurse)[i]->d_lb == 183){
            (*nurse)[i]->intra_lb = 1;
            (*nurse)[i]->intra_ub = 5;
        }
        else{
            (*nurse)[i]->intra_lb = 22;
            (*nurse)[i]->intra_ub = 30;
        }
    }
    if((*nurse)[i]->fte == 6){
        if((*nurse)[i]->d_lb == 211){
            (*nurse)[i]->intra_lb = 1;
            (*nurse)[i]->intra_ub = 5;
        }
        else if((*nurse)[i]->d_lb == 183){
            (*nurse)[i]->intra_lb = 22;
            (*nurse)[i]->intra_ub = 30;
        }
        else{
            (*nurse)[i]->intra_lb = 31;
            (*nurse)[i]->intra_ub = 41;
        }
    }
}
}
else{
    // intra-day shift patterns
    for(i=0; i<N; i++){ // for each nurse

```

```

if((*nurse)[i]->fte == 1){
    (*nurse)[i]->intra_lb = 1;
    (*nurse)[i]->intra_ub = 2;
}
if((*nurse)[i]->fte == 2){
    (*nurse)[i]->intra_lb = 1;
    (*nurse)[i]->intra_ub = 2;
}
if((*nurse)[i]->fte == 3){
    if((*nurse)[i]->d_lb == 127){
        (*nurse)[i]->intra_lb = 1;
        (*nurse)[i]->intra_ub = 2;
    }
    else{
        (*nurse)[i]->intra_lb = 5;
        (*nurse)[i]->intra_ub = 5;
    }
}
if((*nurse)[i]->fte == 4){
    if((*nurse)[i]->d_lb == 57){
        (*nurse)[i]->intra_lb = 8;
        (*nurse)[i]->intra_ub = 10;
    }
    else if((*nurse)[i]->d_lb == 1){
        (*nurse)[i]->intra_lb = 11;
        (*nurse)[i]->intra_ub = 13;
    }
    else{
        (*nurse)[i]->intra_lb = 6;
        (*nurse)[i]->intra_ub = 7;
    }
}
if((*nurse)[i]->fte == 5){
    if((*nurse)[i]->d_lb == 183){
        (*nurse)[i]->intra_lb = 1;
        (*nurse)[i]->intra_ub = 2;
    }
    else{
        (*nurse)[i]->intra_lb = 11;
        (*nurse)[i]->intra_ub = 13;
    }
}
if((*nurse)[i]->fte == 6){
    if((*nurse)[i]->d_lb == 211){
        (*nurse)[i]->intra_lb = 1;
        (*nurse)[i]->intra_ub = 2;
    }
    else if((*nurse)[i]->d_lb == 183){
        (*nurse)[i]->intra_lb = 11;
        (*nurse)[i]->intra_ub = 13;
    }
    else{
        (*nurse)[i]->intra_lb = 14;
        (*nurse)[i]->intra_ub = 19;
    }
}

```

```

    }
}

// intra-day shift preferences
for(i=0; i<N; i++){

    //(*nurse)[i]->intra_preference.resize((*nurse)[i]->intra_ub-
    (*nurse)[i]->intra_lb +1));
    for(j=(*nurse)[i]->intra_lb; j<=(*nurse)[i]->intra_ub; j++){
        //y= floor(X->RandValue()*10 + 0.5);
        y=0;
        (*nurse)[i]->intra_preference.push_back(y);
    }
}

// calculate available nurse hours (sum of fte's)
for(i=0; i<N; i++){
    sumfte += fte[(*nurse)[i]->fte-1];
}

// calculate required nurse hours (sumfte* load factor)
reqnurse_hrs = floor(sumfte * loadf + 0.5);

// distribute required nurse hours over days (uniform)
for(i=0; i<DAY; i++){
    daydemand[i] = floor(reqnurse_hrs/DAY+0.5);
}
//distribute total daily required hours to periods
float alpha = 1.0;
float r;
for(i=0; i<DAY; i++){
    for(j=0; j<PERIOD; j++){
        demand[i][j] = 0;
    }
}
int difference;

// demand distribution triangular case!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!FIX THE BUG
HERE!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
if(d_dist == 1)
{
    for(i=0; i<DAY; i++){
        for(j=0; j<PERIOD; j++){
            if(j==0){
                if(j>=a && j<=c){
                    demand[i][j] = daydemand[i]*(((j-a)^2)/
((b-a)*(c-a)));
                    difference = daydemand[i]- demand[i][j];
                }
                else if(j>=c && j<=b){
                    demand[i][j] = daydemand[i]*(1-(((b-
j)^2)/((b-a)*(b-c))));
                    difference = daydemand[i]- demand[i][j];
                }
            }
        }
    }
}

```

```

        else{
            demand[i][j] = 0;
        }
    }
    else{
        if(j>=a && j<=c){
            //cout<< daydemand[i]<<endl;
            //cout<< ((float)((j-a)^2)/ ((b-a)*(c-
a))) << endl;
            //cout<< daydemand[i]*((float)((j-a)^2)/
((b-a)*(c-a)));
            demand[i][j] = difference -
(daydemand[i]- floor(daydemand[i]*((float)((j-a)^2)/ ((b-a)*(c-a)))));
            difference -= demand[i][j];
        }
        else if(j>=c && j<=b){
            demand[i][j] = floor(daydemand[i]*(1-
((float)((b-j)^2)/((b-a)*(b-c)))));
            difference -= demand[i][j];
        }
        else{
            demand[i][j] = 0;
        }
    }
}
}
}
// demand distribution exponential decrease
if(d_dist == 2){
    for(i=0; i<DAY; i++){
        for(j=0; j<PERIOD; j++){
            if(j==0){
                demand[i][j] = floor((1-(exp((-
1)*alpha*(j+1))))*daydemand[i]);
                difference = daydemand[i]- demand[i][j];
            }
            else{
                demand[i][j] = difference - (daydemand[i]-
floor((1-(exp((-1)*alpha*(j+1))))*daydemand[i]));
                difference -= demand[i][j];
            }
        }
    }
}
// demand distribution uniform (same) in all periods
if(d_dist == 3)
{
    for(i=0; i<DAY; i++){
        for(j=0; j<PERIOD; j++){
            demand[i][j] = daydemand[i]/PERIOD;
        }
    }
}
// demand distribution binomial
if(d_dist == 4){

```

```

        for(i=0; i<DAY; i++){
            for(j=0; j<PERIOD; j++){
                prob = Combination(n,j) * (pow(p,j)) * (pow((1-p),(n-
j))));
                demand[i][j] = floor((daydemand[i]*prob) + 0.5);
            }
        }

// exponential random decrease
//for(i=0; i<DAY; i++){
//    for(j=0; j<daydemand[i]; j++){
//        r= X->RandValue();
//        y = ceil(log(1-r)/(-1* alpha));
//        demand[i][y] = demand[i][y]+1;
//    }
//}

int demand_param;
int p_max;
int p_min;
int max_iter =0;
demand_param = calc_demand_param(demand);

// think about a control for maximum number of improvements!!!!!!
while(demand_param >= N && max_iter <50){// if does not meet the
requirement re-distribute demand
    // find the lowest and highest demand periods
    p_max = 0;
    p_min = 0;
    // this takes care of the first half
    for(j=0; j<PERIOD/2; j++){
        if(demand[0][p_min] > demand[0][j])
            p_min = j;
        if(demand[0][p_max] < demand[0][j])
            p_max = j;
    }

    // update demand for the first half
    for(i=0; i<DAY; i++){
        demand[i][p_min]++;
        demand[i][p_max]--;
    }
    // second half
    // find the lowest and highest demand periods
    p_max = 12;
    p_min = 12;
    for(j=0; j<PERIOD/2; j++){
        if(demand[0][p_min] > demand[0][j+12])
            p_min = j+12;
        if(demand[0][p_max] < demand[0][j+12])
            p_max = j+12;
    }
}

```

```

        // update demand for the second half
        for(i=0; i<DAY; i++){
            demand[i][p_min]++;
            demand[i][p_max]--;
        }

        // recalculate the demand parameter
        demand_param = calc_demand_param(demand);
        max_iter++;
    }

    output(nurseoutput,N,demand,nurse);

    //PARAMETERS SUMMARY
    cout << filename_out <<",";
    if(mixed==0)
        printf("**Parameter Summary**, %s", "V" );
    if(mixed==1)
        printf("**Parameter Summary**, %s", "F" );
    printf(",%10.4d", seed );
    printf(",%1d", N );
    printf(",%10.4f", loadf );
    if(d_dist==3)
        printf(",%10.4s", "U," );
    if(d_dist==4)
        printf(",%s%f", "B,",p );
    for(i=0; i<NFTE; i++){
        printf(",%10.4f", d_fte[i]); // read in fte dist
    }
    printf("\n");

    nurse->clear();
    nursefile.close();
    nurseoutput.close();
    return 0;
} // end main()

double Factorial(double val)
{
    double Result = 1;
    for(double i = 2; i <= val; i++)
    {
        Result *= i;
    }
    return Result;
} // end Factorial()

double Combination(double N, double R)
{
    return (Factorial(N)/(Factorial(N-R)*Factorial(R)));
} // end Combination()

// generate file names including # of nurses, distribution etc.

```

```

string getFileName_out(int N, int mixed, float loadf, short d_dist, float p, long
seed, float d_fte[NFTE]){
    ostringstream oss;
    oss << "N_" << mixed << "_" << seed << "_" << N << "_" << loadf << "_";
    if(d_dist==3)
        oss << "U_";
    if(d_dist==4)
        oss << "B" << p << "_";
    for(int i=0;i<NFTE;i++)
        oss << d_fte[i];
    oss << ".out";
    return oss.str();
} //end getFileName_out()

// generate file names including # of nurses, distribution etc.
string getFileName_dat(int N, int mixed, float loadf, short d_dist, float p, long
seed, float d_fte[NFTE]){
    ostringstream oss;
    oss << "N_" << mixed << "_" << seed << "_" << N << "_" << loadf << "_";
    if(d_dist==3)
        oss << "U";
    if(d_dist==4)
        oss << "B" << p;
    for(int i=0;i<NFTE;i++)
        oss << d_fte[i];
    oss << ".dat";
    return oss.str();
} //end getFileName_dat()

int output(ofstream &nurseoutput, int N, short demand[7][24], vector <Nurse*>
*nurse){
    int i, j;
    nurseoutput << N << endl; // output # of nurses

    nurseoutput << endl;
    //output demand
    for(i=0; i<DAY; i++){
        for(j=0; j<PERIOD; j++){
            nurseoutput << demand[i][j]<< " ";
        }
        nurseoutput << endl;
    }
    nurseoutput << endl;
    for(i=0; i<N; i++){ // for each available nurse

        nurseoutput << (*nurse)[i]->i << "\t" << (*nurse)[i]->fte << "\t"<<
(*nurse)[i]->pi << endl;
        nurseoutput << (*nurse)[i]->d_lb << "\t" << (*nurse)[i]->d_ub <<
"\t" << (*nurse)[i]->n_lb << "\t" << (*nurse)[i]->n_ub << "\t" << (*nurse)[i]-
>m_lb << "\t" << (*nurse)[i]->m_ub << endl;
        nurseoutput << (*nurse)[i]->intra_lb << "\t" << (*nurse)[i]-
>intra_ub << endl;

        for (j=0; j<(((*nurse)[i]->d_ub-(*nurse)[i]->d_lb+1)); j++){// output
day pattern preferences

```

```

        nurseoutput << (*nurse)[i]->d_preference[j] << " ";
    }
    for (j=0; j<((*nurse)[i]->n_ub-(*nurse)[i]->n_lb+1); j++){ // output
night pattern preferences
        nurseoutput << (*nurse)[i]->n_preference[j] << " ";
    }
    for (j=0; j<((*nurse)[i]->m_ub-(*nurse)[i]->m_lb+1); j++){ // output
mixed pattern preferences
        nurseoutput << (*nurse)[i]->m_preference[j] << " ";
    }
    nurseoutput << endl;
    for (j=0; j<(*nurse)[i]->intra_preference.size(); j++){ // output
intra-day pattern preferences
        nurseoutput << (*nurse)[i]->intra_preference[j] << " ";
    }
    nurseoutput << endl;
}
return 0;
}

int calc_demand_param(short demand[7][24]){
    // demand re-evaluation
    int max_day=0;
    int max_night =0;
    int demand_param =0;

    // calculate the demand parameter
    for(int j=0; j<PERIOD/2; j++){
        if(max_day < demand[0][j])
            max_day = demand[0][j];
        if(max_night < demand[0][j+12])
            max_night = demand[0][j+12];
    }
    demand_param = 2*(max_day + max_night);

    return demand_param;
}

```