



ELSEVIER

Available online at www.sciencedirect.com

SCIENCE @ DIRECT®

European Journal of Operational Research 164 (2005) 510–534

 EUROPEAN
 JOURNAL
 OF OPERATIONAL
 RESEARCH

www.elsevier.com/locate/dsw

O.R. Applications

Preference scheduling for nurses using column generation [☆]

Jonathan F. Bard ^{*}, Hadi W. Purnomo

*Graduate Program in Operations Research & Industrial Engineering, The University of Texas, ETC 5.160,
C2200, Austin, TX 78712-1063, USA*

Received 24 February 2003; accepted 11 June 2003

Available online 18 March 2004

Abstract

The purpose of this paper is to present a new methodology for scheduling nurses in which several conflicting factors guide the decision process. Unlike manufacturing facilities where standard shifts and days off are the rule, hospitals operate 24 hours a day, 7 days a week and face widely fluctuating demand. A more flexible arrangement for working hours and days off is needed, especially in light of the growing nursing shortage. To improve retention, management must now take into account individual preferences and requests for days off in a way that is perceived as fair, while ensuring sufficient coverage at all times. This multi-objective problem is solved with a column generation approach that combines integer programming and heuristics. The integer program is formulated as a set covering-type problem whose columns correspond to alternative schedules that a nurse can work over the planning horizon. A double swapping heuristic is used to generate the columns. The objective coefficients are determined by the degree to which the individual preferences of a nurse are violated. As part of the computational scheme, feasible solutions are refined to minimize the use of outside nurses, but when gaps in coverage exist, the outside nurses are distributed as evenly as possible over the shifts. The methodology was tested on a series of problems with up to 100 nurses using data provided by a large hospital in the US. The results indicate that high-quality solutions can be obtained within a few minutes in the majority of cases.

© 2004 Elsevier B.V. All rights reserved.

Keywords: Nurse scheduling; Column generation; Integer programming; Self-scheduling

1. Introduction

The current crisis in health care is forcing hospital executives to run their organizations in a more business-like manner. The constant challenge is to provide high-quality service at ever reduced cost. This problem is exacerbated by an acute shortage of nurses, said to be 120,000 today and expected to grow to 808,000 by 2020 in the United States (US) alone (USDHHS, 2002). Because the nursing service is one of the largest cost components in a hospital's budget, it is essential for every manager to develop an efficient

[☆] This work was supported in part by the National Science Foundation under Grant No. DMI-0218701.

^{*} Corresponding author. Fax: +1-512-471-3076.

E-mail addresses: jbard@mail.utexas.edu (J.F. Bard), hadiwaskito@yahoo.com (H.W. Purnomo).

operational plan that makes the best use of available resources. The decision is complicated by a welter of factors including hospital policies, labor laws, different nursing skill categories, the mix of part-timers and full-timers, random fluctuations in demand, and the desire to accommodate individual preferences.

Nursing skills are typically divided into at least four different categories: registered nurses (RN), licensed practical nurses (LPN), nurse aides, and technical nurses. Registered nurses are the most versatile and generally preferred because they can provide the widest range of care. LPNs are less flexible, while nurse aids have only limited training and skills. A technical nurse is usually needed to operate certain medical instruments specific to a unit, such as cardiology. For purposes of matching skills with requirements, some percentage of the average daily demand is assigned to each nurse category. The specification of this percentage is a management decision that represents a tradeoff between cost, availability, and quality of service.

While there are several levels of staff planning in the hospital environment, our focus is on the mid-term problem associated with weekly assignments. Decisions at this level are made on a regular basis to account for planned leave and expected departures from average demand. To accomplish this task in an equitable manner, information about demand, nurse availability, and the rules concerning priorities and individual requests must be gathered. The decision process must resolve the conflicting viewpoints of the hospital and the nursing personnel. Hospitals are required to provide some minimum level of care (in terms of staff by skill category) during each shift, while nurses want individualized schedules that take into account requests for days off, the exclusion of undesirable work patterns, and other personal considerations. The primary goal is to provide the nursing staff with high-quality schedules subject to demand requirements and cost considerations. This is known as *preference scheduling*.

The purpose of this paper is to present a robust methodology for solving the preference scheduling problem that can accommodate both the quantitative and qualitative detail that nurse managers must address. Rather than trying to minimize cost alone, our objective function is designed to balance contractual agreements and management prerogatives with the use of outside nurses (primarily floaters and agency nurses). The output is a set of rosters for the nursing staff that trades off individual preferences with personnel costs without violating any of the hard constraints in the system. Critical to the success of the procedure is the idea of fairness and the transparency of the resultant schedules. The short-term goal is adequate coverage at minimum cost, the long-term goal is staff retention.

In the next section, the scheduling environment is described, followed by a review of the nurse scheduling literature. We then present the optimization model and the solution approach. Because the full model was too big to solve with a commercial code, an iterative scheme based on column generation was developed, and then tested using data provided by a US hospital. The results indicate that problem instances with up to 100 nurses can be solved within 10 minutes for a 4-week planning horizon and within an hour for a 6-week planning horizon.

2. Background and literature review

There are three different approaches to solving the mid-term nurse scheduling problem. The first is called *rotational* or *cyclical scheduling* (Howell, 1998). In this approach, several sets of schedules are generated that collectively satisfy the demand requirements. Nurses are then rotated from one set of schedules to another in consecutive planning horizons. An exact solution of the cyclical scheduling problem can be found by using a simplified mathematical model. In this model, cyclic work patterns are generated to minimize the size of the nursing staff needed to cover the demand. Burns (1978) studied the case of 10 days on in a 14-day planning horizon with every second weekend off and up to six consecutive days on. Although easy to implement, cyclical schedules have become the bane of the profession because of the rigidity they impose. Many nurses view flexibility as an entitlement that comes with the job.

In contrast, *self-scheduling* attempts to solve the problem in a more flexible way. Having determined the number of nurses required for each period, nurses are asked to sign up for any shifts that they want to work during the planning horizon as long as they are within their contractual bounds. To avoid an over-supply in any given period, an upper limit is placed on the number permitted to sign up in each block of time. When conflicts occur, further adjustments are made through consensus. Griesmer (1993) reported a successful application involving 32 nurses. Although the approach is appealing, it is quite difficult to implement in practice because of the impracticality of holding meetings to resolve conflicts, especially for large units. Moreover, the results may not necessarily be perceived as fair. Those who are savvy enough to game the system will always have an advantage over the procrastinators. Controlling the sign-up order and rotating it over the year is a partial solution.

The third approach—*preference scheduling*—is actually a combination of the first two. The general idea is to accommodate individual preferences when creating schedules. Preferences can be measured in terms of requests to work specific shifts or to be given specific days off, and by other rules such as the number of working hours, shift sequence patterns or even nurse to patient ratios. Like self-scheduling, nurses are asked to sign up for shifts prior to the beginning of the planning horizon. At that time, they usually submit a list of requests to the nurse manager who compiles them and decides which to approve immediately in light of the coverage requirements for the upcoming planning horizon. When generating the final rosters, she tries to satisfy as many preferences and pending requests as possible.

Exact methods, heuristics, and constraint satisfaction have all been applied to the preference scheduling problem over the last 25 years. Exact methods primarily involve the use of set covering-type models. In this approach, alternative schedules are generated by any number of methods and one is selected for each nurse. The typical objective is to minimize a weighted combination of costs and preference penalties. What has limited the usefulness of this approach is the size of the integer program (IP) that must be solved and the determination of the penalty coefficients. The longer the planning horizon, the more alternatives that can be generated, and as a result, the harder the IP is to solve. The penalty coefficients must be defined so that 'small' values imply a 'good' schedule.

Warner (1976) was the first to develop a methodology for solving the set covering model for the case in which all nurses work 8-hour shifts. To overcome the unmanageable size of the full IP, only 50 good schedules obtained by a greedy method were included in the model for each nurse. A block pivoting strategy was used to find feasible solutions, which were then improved with a post-processor. The methodology was implemented at two hospitals with staff sizes ranging from 19 to 47 nurses.

Jaumard et al. (1998) extended the basic IP model to include both 8- and 12-hour shifts, and different levels of nursing skills. This extension adopted the concept of demand periods, a unit time bucket that alleviates the problem of overlapping working hours in two different shifts. The modified problem was then solved using Dantzig–Wolfe decomposition with the necessary adjustments for integrality restrictions (Wolsey, 1998). Columns were generated at each iteration by solving a resource-constrained shortest path subproblem, one for each nurse.

While the set covering methodology focuses on strategies for generating candidate schedules in the form of columns, heuristic approaches use a different strategy. They are designed to first obtain an initial solution, and then improve upon it with neighborhood search methods. Miller et al. (1976) used a simplified version of a rotation heuristic for a 4-week problem. The objective function was a weighted combination of personnel costs and preference penalties. A greedy heuristic with feasible neighborhood swaps was adopted to solve a real instances with up to 12 nurses.

Meta-heuristics have also been used to solve the preference scheduling problem. Dowsland (1998) developed a tabu search approach with strategic oscillation. The algorithm starts with an initial roster obtained with a greedy heuristic that ignores the minimum coverage requirement and treats each nurse separately. In the next phase, three swap moves are used to try to satisfy the coverage constraints: shift swaps for a single nurse, two-nurse chain swaps, and rotation swaps. The quality of each schedule produced

by the algorithm is measured by a weighted sum of the penalty coefficients associated with the preference violations.

Valouxis and Housos (2000) used a combination of integer programming and heuristics to find solutions to a limited version of the preference scheduling problem. Instead of solving the classic set covering model, they solved a relaxation that minimized the total number of shifts needed to satisfy demand subject to several preferences constraints. Arthur and Ravindran (1981) were the first to apply goal programming to the preference scheduling problem. They considered the following four goals: contractual requirements, preferences, requests, and staffing requirements. In the first phase of their two-phase approach, a small IP is solved to decide the day on which each nurse is to work. Shift assignments are made in the second phase. In a similar vein, Berrada et al. (1996) proposed a constraint satisfaction model that viewed demand coverage and the number of working days as the hard constraints and compliance to shift patterns, daily requirements for supervisory personnel, and the grouping of days off and weekends off as the soft constraints.

For the same problem, Ferland et al. (2001) developed a tabu search methodology that included a diversification strategy and an adaptive memory structure. Similarly, Burke et al. (1999) used tabu search to schedule nurses at several Belgium hospitals using a code implemented in the Plane software system. Subsequent refinements allowed for multiple objectives and more equitable weekend assignments.

Working independently, Cheng et al. (1997) proposed about 20 heuristic rules to build rosters and to recursively fill in the uncovered shifts. Following a different course, Kawanaka et al. (2001) used a genetic algorithm to also find solutions. Their method was a bit restrictive, though, because it was designed around six hard constraints specific to the hospital environment in which they were working. Other heuristics include the use of language theory, artificial intelligent techniques and expert systems. The general idea is to represent problem constraints as clauses or strings that can be combined to form feasible schedules with the help of an inference engine like Prolog (e.g., see Okada, 1992).

3. Problem statement and model formulation

Hospitals typically employ nurses to work either 8- or 12-hour shifts, giving rise to five standard shift types: three 8-hour shifts called Day (7 a.m.–3 p.m.), Evening (3 p.m.–11 p.m.), Night (11:00 p.m.– 7 a.m.) and two 12-hour shifts called AM (7 a.m.–7 p.m.) and PM (7 p.m.–7 a.m.). An AM shift starts at the same time as a Day shift and ends mid-way into the Evening shift. A similar interpretation exists for a PM shift. The problem will be first stated for a single nurse category using these shift types and then extended to allow for downward substitution of skill levels.

For modeling efficiency, it is convenient to divide the day into a set of contiguous (hourly) periods of uneven length such that each shift consists of one or more periods. By doing so, we can express the demand in terms of periods rather than shifts and hence reduce the size (number of rows) of the model. Fig. 1 depicts the relationship between shifts and periods for our problem. As can be seen, only four periods are needed to cover 24 hours. The first demand period coincides with the Day shift, the second period covers

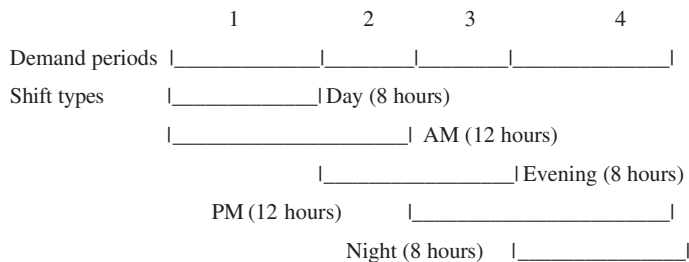


Fig. 1. Shift types and demand periods.

the first half of the Evening shift, the third covers the second half of the Evening shift, and the last coincides with the Night shift.

In the developments, we make use of the following notation.

Indices and sets

i	index for nurses; $i \in N$
j	index for schedules; $j \in S_i$
p	scheduling period (portion of a day); $p \in P$
d	day of the week; $d \in D$
N	set of nurses to be scheduled
S_i	set of schedules considered for nurse i
D	set of days in the planning horizon (typically 28 or 42)
P	set of periods in a day

Parameters and data

c_{ij}	penalty “cost” of assigning schedule j to nurse i (this value is a function of the number and severity of preference and request violations in the schedule)
a_{ijdp}	1 if schedule j for nurse i contains period p on day d , 0 otherwise
LD_{dp}	lower bound on demand for nurses on day d in period p
UD_{dp}	upper bound on demand for nurses on day d in period p
M	large number representing the cost of either an outside nurse or under-coverage in a period

Decision variables

x_{ij}	(binary) 1 if nurse i is assigned to schedule j , 0 otherwise
y_{dp}	number of outside nurses used on day d in period p
s_{dp}	slack variable for the “left” constraint on day d in period p

The basic optimization model is designed to minimize the cost of meeting the demand for a particular nurse category in each of the $m \equiv |P| \times |D|$ periods over the planning horizon, where $|X|$ denotes to the cardinality of the set X . The general formulation is as follows:

$$\text{Minimize } \sum_{i \in N} \sum_{j \in S_i} c_{ij} x_{ij} \quad (1a)$$

$$\text{subject to } LD_{dp} \leq \sum_{i \in N} \sum_{j \in S_i} a_{ijdp} x_{ij} \leq UD_{dp} \quad \text{for all } d \in D, p \in P, \quad (1b)$$

$$\sum_{j \in S_i} x_{ij} = 1 \quad \text{for all } i \in N, \quad (1c)$$

$$x_{ij} = 0 \text{ or } 1 \quad \text{for all } i \in N, j \in S_i. \quad (1d)$$

The first constraint (1b) ensures that the number of nurses scheduled in each period falls between some minimum and maximum value, and is really two separate constraints. In many situations, it might only be necessary to include the left-hand portion of this constraint, but it is generally desirable to place a maximum limit on the number of nurses assigned in each period. Constraint (1c) along with (1d) guarantee that each nurse in the unit is assigned one and only one schedule. Note that for the sake of compactness, it is possible to combine d and p into a single index that references the time period over the planning horizon.

Model (1a)–(1d) can be thought of as column oriented. The j th schedule for nurse i over a planning horizon of $|D|$ days can be represented by a column vector of size $|P| \times |D|$ as follows:

$$A_{ij} = (a_{ij1}, a_{ij2}, \dots, a_{ij,|P|}, a_{ij,|P|+1}, \dots, a_{ij,2|P|+1}, \dots, a_{ij,|P| \times |D|})^T. \quad (2)$$

In preference scheduling, we initialize the model with $|N|$ sign-up schedules, one for each nurse. The difficulty comes in constructing the set S_i . An effective algorithm is needed to identify good feasible schedules for each nurse i , where “good” means few violations of the nurse’s preferences and requests. A method for generating these schedules is discussed presently, but first we introduce several adjustments to the model that are needed to achieve a better representation of the actual scheduling environment.

3.1. Insufficient coverage

Many hospitals and clinics face a shortage of nurses, implying that under coverage or gaps are inevitable in some periods. To compensate for these shortages, several options are available. The first is to plan for the use of travelers—nurses who are hired through outside firms for approximately 12 weeks at a time. The second is to use agency nurses who work through temporary agencies for a negotiated number of days or weeks. The third is to commandeer nurses from other units in the hospital that may have an excess supply in a particular period. A nurse that is re-assigned under these conditions is called a *float*er and is not part of the mid-term planning process. To deal with shortages in our model we introduce the variable y_{dp} corresponding to the number of outside nurses used in period p on day d . These may be agency nurses, travelers, or any other type of nurse assigned to fill a gap in the schedule. The appropriate modification to constraint (1b) is

$$LD_{dp} \leq \sum_{i \in N} \sum_{j \in S_i} a_{ijdp} x_{ij} + y_{dp} \leq UD_{dp} \quad \text{for all } d \in D, p \in P. \tag{1b'}$$

To avoid pathological situations, it will be assumed that a feasible solution always exists to (1b’).

3.2. Two-sided constraints

The efficiency of any linear programming-based code is a function of the number of structural constraints in the model. Two-sided constraints such as those in (1b) can be reduced to a single-sided constraint by introducing an equivalent number of bounded variables. Let s_{dp} be the slack variable for the “left” constraint in (1b’). That is,

$$LD_{dp} = -s_{dp} + \sum_{i \in N} \sum_{j \in S_i} a_{ijdp} x_{ij} + y_{dp}. \tag{3}$$

If we impose the following upper bound on s_{dp} , the two-sided constraint (1b) will always be satisfied.

$$0 \leq s_{dp} \leq UD_{dp} - LD_{dp}. \tag{4}$$

We can now substitute constraints (3) and (4) for (1b’) in our model.

Full Model

$$\text{Minimize } \sum_{i \in N} \sum_{j \in S_i} c_{ij} x_{ij} + M \sum_{d \in D} \sum_{p \in P} y_{dp} \tag{5a}$$

$$\text{subject to } -s_{dp} + \sum_{i \in N} \sum_{j \in S_i} a_{ijdp} x_{ij} + y_{dp} = LD_{dp} \quad \text{for all } d \in D, p \in P, \tag{5b}$$

$$\sum_{j \in S_i} x_{ij} = 1 \quad \text{for all } i \in N, \tag{5c}$$

$$x_{ij} = 0 \text{ or } 1 \quad \text{for all } i \in N, j \in S_i, \tag{5d}$$

$$0 \leq s_{dp} \leq UD_{dp} - LD_{dp}, y_{dp} \geq 0 \quad \text{for all } d \in D, p \in P. \tag{5e}$$

The objective (5a) is to minimize the weighted sum of the costs associated with the schedules assigned to the regular nurses and the cost attending the use of outside nurses to cover gaps. The choice of the parameter M implicitly defines the tradeoff between satisfying the collective preferences of the nursing staff and incurring additional costs by calling on outside nurses to meet unfilled demand. In general, we want $M \gg \max\{c_{ij} : i \in N, j \in S_i\}$.

Finally, note that as long as LD_{dp} and UD_{dp} are integral, an optimal solution will always exist with s_{dp} and y_{dp} integral so it is not necessary to declare them as such. A related point is that the complementarity condition $s_{dp} \times y_{dp} = 0$ will always hold at optimality.

3.3. Specification of cost coefficients

A critical measure of success in the use of preference scheduling is the perceived fairness or balance in the posted rosters. If some nurses feel that their preferences and requests are continually being ignored the approach is likely to fail. Bickering, increased absenteeism, low morale, poor job performance, and high turnover rates may become the norm. To avoid this situation, the model must be driven to select schedules that are in balance, i.e., have about the same number of violations per nurse across the board. This can be achieved by defining each cost coefficient c_{ij} in (5a) as an increasing function of the number and severity of preference violations. Thus each schedule j for nurse i , as represented by the column vector in (2), will have a cost that increases at an increasing rate.

To implement this idea, let v be the equivalent number of penalty points (cost) associated with a violation for a particular nurse. Table 1 lists the degree or severity of a violation in qualitative terms, the corresponding value of v , and the cost coefficient $c_{ij}(v)$. We have found that the exponential function $c_{ij}(v) = 2^{v-1}$ works best in model (5).

In specifying the entries in Table 1, the aim was for a universal measure that would demand little effort on the part of the user to calibrate the model. The only thing required is that each violation be assigned to one of the four enumerated categories. Most violations are expected to be “simple” and incur a penalty of 1. Examples might include assigning a nurse a Night shift when Evening shifts are the norm, having to work an on-off-on pattern, and working a Night shift the day before the start of a vacation. More serious violations might be associated with split weekends (one weekend day on each of two consecutive weekends), and working a PM shift right after returning from vacation. An “extreme” violation is considered the worst possible case and, according to our scheme, is eight times as severe as a simple violation.

Of course, other methods of quantifying violations have been proposed. A common approach is to enumerate all possible violations and ask each nurse to allocate 100 point among them, the more points the more severe the violation (e.g., see Randhawa and Sitompul, 1993). This provides individualized cost coefficients but can lead to gaming by those who are savvy, and to frustration for those who cannot rationalize the process in their minds. Burke et al. (2001) developed an approach called *numbering* that is more comparable to ours. Their major contribution relates to the data structure used to compute violations of the soft constraints. Experience has shown, however, that procedures that place too much of a burden on the user to quantify preferences are either not accepted or are soon abandoned. Our approach tries to circumvent these pitfalls by simplifying the user input without sacrificing the need to customize the process.

Table 1
Quantification of preference violations

Degree of violation	Equivalent no. of penalty points, v	Cost coefficient, $c_{ij}(v) = 2^{v-1}$
Simple	1	1
Serious	2	2
Severe	3	4
Extreme	4	8

4. Model input

To a large extent, input requirements for model (5) are a function of the algorithmic approach. Specific parameters, such as the maximum number of columns to be included in the formulation and the minimum number of requests permitted per nurse, are discussed in the next section where the column generation algorithm and its various components are presented. At the general level, we must specify the length of the planning horizon, the number and types of shifts to be assigned, the demand per shift, and a host of rules. The latter can be categorized as either hard or soft constraints, reflecting institutional policies, contracts, individual agreements, and legal restrictions.

From a requirements point of view, the rules that must be followed in mid-term scheduling can be divided into four categories: (i) hard constraints at the system level, (ii) hard constraints associated with an individual, (iii) soft constraints at the system level, and (iv) soft constraints associated with an individual. Hard constraints are those which define feasibility while soft constraints are those which can be violated under certain circumstances, but at a cost. This cost may be an explicit dollar amount, such as an agreement to pay overtime for certain shifts, or qualitative, as reflected by a preference violation that leads to low morale. Unfortunately, there is no consensus amongst hospitals, or even units within a hospital, as to which constraints are hard and which are soft.

The set of rules and the classification scheme that we used in our implementation are enumerated in Table 2. Also shown are the penalty values associated with a violation of the soft constraints. This

Table 2
Rules and penalty structure for preference violations

No.	Rule	Hard constraint	Soft constraint (violation)	Penalty assessment	
				Severity	Points
1	Illegal work patterns—there must be an eight hour break between shifts	yes			
2	Number of different shift types a nurse can work in 2-week period when contract calls for single shift type	≤ 2	Works 2 different shifts	Simple	1/violation
3	Consecutive working days (<i>workstretch</i>)	≤ 6 days ^a	6	Simple	1/violation
4	Consecutive days off				
	8-hour shift (5 working days)	N/A	N/A		
	12-hour shift (3 working days)	<6	≥ 4	Simple	1/violation
5	Weekend penalties				
	Number of working weekends	2 weekends in 4 weeks			
	Number of working weekend days		3 days	Serious	2
	Contract for Day/AM shift	N/A	Assigned N/PM shift on week-end	Simple	1
	Contract for Evening/Night/PM shift	N/A	Assigned D/AM shift on week-end	Simple	1
6	Pattern violations				
	Off-active-off (0 1 0) pattern (only for full-time 8-hour shifts)	N/A	When happens	Simple	1/violation
	Active-off-active (1 0 1)	N/A	When happens	(not included)	

(continued on next page)

Table 2 (continued)

No.	Rule	Hard constraint	Soft constraint (violation)	Penalty assessment	
				Severity	Points
7	Days-off shift No AM/ Day shift after requested day off/ holiday	N/A	Counted only for nurses working on E, N, PM shift	Simple	1/violation
	No PM/Night shift before requested day off/holiday	N/A	Only counted for nurses working on D/ AM	Simple	1/violation
8	Rotation schedules Day/AM shift	Must work hired shift for 50% of assigned schedule			
	Evening/Night/PM shift				
9	Number of transitions for rotational nurse during each workstretch (e.g., schedule Day-Day-Evening-Day-Day has two transitions)	≤ 6 in 2 weeks	3 in 2 weeks	Simple	1
			4 in 2 weeks	Serious	2
			5 in 2 weeks	Severe	3
			6 in 2 weeks	Extreme	4
10	Personal requests	'MinReq' requests guaranteed over planning horizon			
11	Overtime Full-time/part-time (8-hour shift)	Patterns allowed (8-hour shift) Day→AM, Night→PM, Evening→AM/PM	>40 hour a week (full-time)	Simple	1
	Full-time (12-hour shift)	Patterns allowed (8-hour extension for 12-hour shift) AM→Day, PM→Night	28 hour a week (part-time) >36 hour a week	Simple	1
		Maximum working hours allowed ≤ 44 per week (for full-time nurses) ≤ 88 per 2 weeks (for full-time) ≤ 32 per week (for part-time)		Simple	1
12	Outside nurses (fill in gaps)				50
13	Minimum hours a nurse must work in a 2-week period (based on contractual agreement; varies among full-time and part-time nurses)	Yes			

^a Only applicable for full-time nurses that are hired to work 8-hour shifts.

table does not show individual constraints that are handled on a case-by-case base in the implementation.

5. Solution methodology

The major component of the solution methodology is the procedure for generating candidate schedules. To initialize the problem, we start with either the sign-up schedule provided by each nurse or with individualized templates based on their contract. This gives us $|N|$ columns for the structural constraints in (5b). If the full roster satisfies the demand this might be all that is needed; however, many of the nurses may not have been able to sign up for their desired shifts, implying that one or more preference violations will exist in their schedule. Requests must also be considered.

An important factor in setting up the model is the balance between the number of candidate schedules and the computation time. The number of potential columns per nurse grows exponentially with the number of shift types and days in the planning horizon. It is critical that the size of the model be managed intelligently. In particular, we would only like to generate columns that are legal (satisfy the hard constraints), are attractive from the nurses point of view (few preference violations), accommodate requests, and biased towards periods in which shortages exist. The latter is a key point because of the huge number of potential columns.

The overall scheme is to generate a subset of the legal columns for each nurse by modifying one or two shifts in his or her initial schedule (hereafter called the *base* schedule) and adding them to the formulation. This will be done for each of the $|N|$ nurses, one at a time, in the outer loop of the algorithm. We refer to the outer loop as the *major* iterations and reference it by the index α . To limit the number of columns generated per nurse at iteration α we introduce the parameter $V_i^{\max}(\alpha)$, the maximum number of (equivalent) preference violations (see Table 1) permitted for nurse i , and the parameter $\text{MaxCol}(\alpha)$, the maximum number of new columns per nurse to be included in the model. Logically, $V_i^{\max}(\alpha)$ is a nondecreasing function of α , and may be different for each nurse i depending on his or her seniority, previous schedules, contract, or personal agreement with management. $\text{MaxCol}(\alpha)$ is also specified as a nondecreasing function of α . The generation of columns for an individual nurse is done in the inner loop of the algorithm and referenced by the index i .

At each major iteration, an attempt is made to solve (5a)–(5e) with a commercial code (CPLEX in our case). Note that this problem is always feasible because of the inclusion of outside nurses represented by the variable y_{dp} ; however, depending on the effectiveness of CPLEX and the stopping criteria specified in the algorithm, there may be a gap between the solution reported and the (unknown) optimum. Assuming for the moment that optimality has been achieved to within an acceptable percentage, two situations can arise.

- (i) The solution includes outside nurses to cover shortages or gaps.
- (ii) All periods are covered by the regular staff.

With regard to (i), if a 0% optimality gap results, then the current columns in the model may not be sufficiently diverse to achieve full coverage with only staff nurses. If the demand exceeds the supply then it might not be possible to obtain a better solution. In any case, either more columns should be considered or more nurses need to be added to the unit. If the optimality gap is greater than zero, we cannot be sure whether or not a solution exists with no outside nurses. To verify this, it would be necessary to adjust the stopping criteria to allow more of the feasible region to be explored. Either way, the solution represents an implicit tradeoff between the quality of the individual schedules and the cost of using outside nurses. To achieve the second situation (ii), more columns must be added to the model. According to the generation scheme,

these columns would likely include more preference violations than the set used at the current iteration. Nevertheless, due to legal restrictions and explicitly imposed limits on the number of preference violations per nurse, it may not be possible to obtain a solution in which no outside nurses are required.

We now present the main algorithm. The individual routines referenced at a particular step are either detailed in Appendix A or can be found in Purnomo (2002).

Preference Scheduling Algorithm

Input: Sign-up schedule, requests, nurse attributes, rules, demand requirements per period, maximum number of new columns $\text{MaxCol}(\alpha)$ per nurse, and maximum number of preference violations $V_i^{\max}(\alpha)$ permitted for nurse i at iteration α ($i \in N, \alpha = 1, \dots, \alpha^{\max}$).

Output: Roster for every nurse for the planning horizon.

Step 1. (Initialization) Read input data files and set counters $\alpha = 1$ and $i = 1$.

1a. Convert demand requirements per shift to demand requirements per period.

1b. Set the sign-up schedule as the base schedule σ_{base} .

1c. Build initial A -matrix ($m \times n$) for IP model (5a)–(5e) from the base schedule σ_{base} .

1d. Add m outside nurse variables (y_{dp}) and m slack variables (s_{dp}) to convert the two-sided inequality constraints (1b) into single-sided equality constraints (5b).

1e. Set first ‘MinReq’ as hard constraints for each nurse.

1f. Call **Maximum_Violation_Algorithm** to get V_i^{\max} for all nurses. Set $V_i^{\max}(1) = \min\{V_i^{\max}(1), V_i^{\max}\}$.

1g. Set upper bound on objective function $\bar{z}(0) = \infty$.

Step 2. (Optimality of Base Schedule) Determine whether or not the base schedule has the following characteristics: all demand is satisfied, no hard constraints are violated, and no preference violations exist for any nurse. If so, go to Step 6 with an optimal schedule; if not, go to Step 3.

Step 3. (Column Generation) For current values of $V_i^{\max}(\alpha)$ and $\text{MaxCol}(\alpha)$, call **Column_Generation_Algorithm** to generate as many candidate schedules (S_i^{new}) as possible for each nurse i from the base schedule σ_{base} . Randomly select $n(\alpha) = \min\{\text{MaxCol}(\alpha), |S_i^{\text{new}}|\}$ and place them in $S(\alpha)$. Put

$$S_i \leftarrow S_i \cup S(\alpha),$$

$$n \leftarrow n + n(\alpha),$$

$$i \leftarrow i + 1 \text{ and repeat Step 3 until } i = |N| + 1.$$

Step 4. (Update Formulation) Add new columns to model, update SOS list and data structure used to store full A -matrix. [The size of IP is $(m + |N|)(n + 2m)$.]

Step 5. Try to solve the IP with upper bound $\bar{z}(\alpha)$ as input.

If (no solution is found) and ($n < \text{MaxColTot}$), then

put $\alpha \leftarrow \alpha + 1$, $\bar{z}(\alpha) \leftarrow \bar{z}(\alpha - 1)$, and go to Step 3;

If (no solution is found) and ($n \geq \text{MaxColTot}$), then go to Step 6.

Otherwise, let objective function value be $z(\alpha)$ and the new schedule be $\sigma(\alpha)$.

Reduce the upper bound by putting $\bar{z}(\alpha) \leftarrow \theta z(\alpha)$, and

Update base schedule by putting $\sigma_{\text{base}} \leftarrow \sigma(\alpha)$.

If (number outside nurses > 0) or ($n < \text{MaxColTot}$), then

put $\alpha \leftarrow \alpha + 1$ and go to Step 3;

If (number outside nurses $= 0$) and ($n < \text{MaxColTot}$) and ($\alpha < \alpha^{\max}$), then

delete m columns associated with outside nurses,

put $\alpha \leftarrow \alpha + 1$,

set $V_i^{\max}(\alpha) = \min\{V_i^{\max}(\alpha - 1), V_i^{\max}\}$ and go to Step 3.

Else go to Step 6.

Step 6. Report final schedule.

Solving the IP at Step 5 was less time consuming than originally expected because most search trees were small. If a solution is found such that $z(\alpha) \leq \bar{z}(\alpha)$, the incumbent is updated. Before adding new columns to the model and re-solving though, we reduce the upper bound by the fractional value θ in an effort to obtain a visible improvement at the next major iteration. If no solution is found and the maximum number of columns (MaxColTot) has not yet be reached, the algorithm returns to Step 3; otherwise we go to Step 6 and quit. At Step 4, the SOS list and the A -matrix are updated. Although CPLEX has a feature that obviates the need to include the SOS constraints explicitly, we did not use this feature because the results were much better without it; i.e., the objective values associated with the LP solutions at node 0 of the search trees were much greater when the SOS constraints were omitted from the model.

The most computationally intensive and time-consuming component of the algorithm is column generation at Step 3. It is here where the many rules and requests must be checked. To reduce the overall effort it is desirable to introduce additional stopping criteria. For example, we have restricted the number of major iterations to the parameter α^{\max} . Also, we terminate a major iteration when the search tree reaches some maximum number of nodes set by the parameter MaxNodes or after TimeLim seconds are reached on the clock.

5.1. Generating columns

Candidate schedules in the form of columns are generated for each nurse $i \in N$ in the inner loop of the algorithm by modifying the base schedule. The goal is to construct candidate schedules that not only have low objective function coefficients but also permit us to build a roster over the planning horizon with a minimum number of outside nurses (gaps). The proposed methodology consists of three major components: identifying a set of periods in which there is under-coverage and identifying a set of periods in which there is over-coverage; performing swaps between these two periods to produce a new candidate schedule; and checking the legality and redundancy of this schedule.

The idea of swapping shifts is similar to the swap neighborhood heuristics commonly used to solve combinatorial optimization problems. The same principles were used by Nonobe and Ibaraki (1998) to solve the constraint satisfaction problem. In model (5), recall that a column A_{ij} consists of a string of 0's and 1's, where a 1 indicates that the nurse is assigned to work in the corresponding period. A standard shift neighborhood heuristic produces a new column by interchanging a 1 and a 0 in the string. As a simple example consider the first three days of a schedule for nurse A: (1000 0001 0001). Looking at Fig. 1, we can see that this string corresponds to an assignment of Day–Night–Night for days 1, 2, and 3, respectively. Our heuristic generates a new column by moving an active period (denoted by 1 in the string) to an inactive period (denoted by 0 in the string). One possibility is to move the first active period on day 1 to the fourth period which is inactive. The result is the string (0001 0001 0001) translated as Night–Night–Night shifts. Another possibility is move active period 8 to inactive period 6, a four-hour period. In this case, we also have to make period 7 active in order to get an 8-hour shift. The result is the string (1000 0110 0001) translated as Day–Evening–Night shifts for the first three days.

Because the goal of the heuristic is to generate enough varied columns so that the required number of outside nurses is minimum, it is desirable to bias the interchanges towards demand periods in which there is insufficient coverage. At each major iteration, the column generation algorithm uses the current IP solution (base schedule for nurse i) to determine the periods in which there are staff shortages and staff surpluses. To implement this idea, every period is scanned and classified into one of two sets when appropriate.

- $A(i)$ set of originating periods for swap. It includes periods in which there is more than the average number of nurses signed up and nurse i is scheduled to work,
- $B(i)$ set of target periods for swap. It includes periods in which there are fewer than the average number of nurses signed up and nurse i is not scheduled to work.

Depending on the characteristics of the base schedule for nurse i , these sets are derived using one of two procedures called *All_Columns_Classification* and *Average_Classification*, respectively. In the first procedure, each working period for nurse i is a candidate for membership in the set $A(i)$; the remaining periods are candidates for membership in $B(i)$. Generating columns using these definitions, however, can be very time consuming because the number of possible columns grows exponentially with the number of time periods in the planning horizon. To alleviate this problem, a further restriction is applied in classifying periods based on the average nurse availability. This is the rationale of the second procedure which uses a threshold value in the construction of $A(i)$ and $B(i)$ based on the average number of nurses scheduled per period over the planning horizon. In our implementation, we start with the *Average_Classification* scheme and switch to the *All_Columns_Classification* scheme when an insufficient number of columns have been generated for a particular nurse. In general, a column is generated when it is possible to switch the working pattern of a nurse from the period with surplus demand to a period with a shortage without violating any of the embedded hard constraints. Both single and double swaps are considered.

Column_Generation_Algorithm

Input: Base schedule for every nurse i ; maximum number of hours $V_i^{\max}(\alpha)$ allowed for nurse i without overtime (MaxHr _{i}).

Output: Set of new columns S_i^{new} for every nurse i .

Step 1. Calculate total hours (TH_i) for all $i \in N$. If $TH_i < \text{MaxHr}_i$, run **Overtime_Generation_Algorithm**.

Step 2. Let TC_i be the total number of columns in the model for nurse $i \in N$; calculate $\text{AvgCol} = \frac{1}{|N|} \sum_{i=1}^{|N|} TC_i$. Set $i = 1$.

Step 3. Set $S_i^{\text{new}} = \emptyset$. If $TC_i > \text{AvgCol}$ then

Run **Average_Classification_Algorithm** to build sets $A(i)$ and set $B(i)$. Run single-swap move for each member of set $A(i)$ and set $B(i)$.

Else

Run **All_Columns_Classification_Algorithm** to build set $A(i)$ and set $B(i)$. Go to Step 5 (generate new columns using double swap move).

Step 4. Run **Check_Legality_Algorithm** to see if swap is allowed. If legal then go to Step 5, else go to Step 7.

Step 5. Call **CalcPenalty_Algorithm** to calculate the cost coefficient c_i^{new} . Let v_i^{new} be the number of equivalent preference violations associated with c_i^{new} . If $v_i^{\text{new}} > V_i^{\max}(\alpha)$ discard column and go to Step 7; else go to Step 6.

Step 6. Check redundancy using **Hash_Table_Algorithm**. If is redundant, discard column and go to Step 7; else put $S_i^{\text{new}} \leftarrow S_i^{\text{new}} \cup A_i^{\text{new}}$ and go to Step 7.

Step 7. If $i < |N|$, put $i \leftarrow i + 1$ and go to Step 3; else stop and return.

In practice, before incrementing the nurse counter at Step 7 we check to see if $|S_i^{\text{new}}| > \text{MaxCol}(\alpha)$. If indeed this condition is met, we randomly reorder the elements in S_i^{new} and select the first $\text{MaxCol}(\alpha)$. This is referenced in Step 3 of the *Preference Scheduling Algorithm*. The algorithms for checking the legality and redundancy of a schedule are presented in Appendix A.

5.2. Determining the maximum number of permitted violations

In preference scheduling, one of the measures used to judge the quality of a schedule is its “fairness” as perceived by the nursing staff. To some degree, fairness can be measured by how closely an individual’s schedule matches the schedule for which he or she was hired, and by the severity of the preference violations in the derived schedule. Of course, some overlap exists between these two measures.

An additional consideration relates to the historical trend or the quality of successive schedules assigned to a nurse. Even treatment might mean that if a nurse received a bad schedule in one month, then she would

receive a good schedule in the next month. We have implemented this idea by limiting the number of equivalent violations in any new schedule to a maximum of V_i^{\max} . (In the *Preference Scheduling Algorithm*, the actual value is limited to $\min\{V_i^{\max}(\alpha), V_i^{\max}\}$ at iteration α .) To determine V_i^{\max} , we first compute the average number of violations in the previous scheduling period (call it \bar{v}) and the accompanying standard deviation (call it $\bar{\sigma}$). If the schedule in the previous period included more than $\bar{v} + \bar{\sigma}$ equivalent violations, the number of violations in the current period is restricted to be less than or equal to $\max\{1, \bar{v} - \bar{\sigma}\}$. The algorithmic details follow.

Maximum_Violations_Algorithm

Input: v_i^{old} , equivalent number of preference violations for each nurse i in previous period.

Output: V_i^{\max} , maximum number of preference violations allowed in current period.

Step 1. Compute average and standard deviation associated with the number of preference violations in the previous scheduling period.

$$\bar{v} = \frac{1}{|N|} \sum_{i=1}^{|N|} v_i^{\text{old}} \quad \text{and} \quad \bar{\sigma} = \sqrt{\frac{1}{|N| - 1} \sum_{i=1}^{|N|} (v_i^{\text{old}} - \bar{v})^2}.$$

Set $i = 1$, $V_i^{\max} = \infty$, and go to Step 2.

Step 2. If $v_i^{\text{old}} > \bar{v} + \bar{\sigma}$, set $V_i^{\max} \leq \lfloor \max\{1, \bar{v} - \bar{\sigma}\} \rfloor$, else go to Step 3.

Step 3. If $i = |N|$, stop; else put $i \leftarrow i + 1$ and go to Step 2.

6. Model enhancements

Several additional features have been incorporated in the algorithmic structure to ensure that the results are realistic and that the methodology is robust. The features considered were strongly influenced by the need to balance the computational effort with the quality of schedules produced.

6.1. Distribution of gaps in the roster

Most hospitals prefer that gaps in the schedule be distributed evenly over the planning horizon. In qualitative terms, this means that shortages should be spread as uniformly as possible among the shifts. The simplest way of achieving this pattern of coverage is to impose an upper bound, *MaxGap*, on the number of outside nurses, y_{dp} , in each period. In our implementation, *MaxGap* is initially set to an arbitrarily large value to avoid an infeasible instance of (5a)–(5e). After the first solution is obtained, it is reset to $\max\{y_{dp} : d \in D, p \in P\} - 1$, and then reduced incrementally at subsequent iterations if an improvement is realized. The process continues until one of the stopping criteria is met or no feasible solution has been found in three consecutive iterations. The results highlight the relationship between the average number of preference violations in a schedule and the use of outside nurses.

A more efficient way of dealing with this issue might be to modify the objective function (5a) so that y_{dp} appears as a quadratic term.

$$\text{Minimize } \sum_{i \in N} \sum_{j \in S_i} c_{ij} x_{ij} + M \sum_{d \in D} \sum_{p \in P} y_{dp}^2. \tag{5a'}$$

This formulation penalizes the use of outside nurses in each period at an increasing rate, and so should have a leveling effect. However, the resultant model is a mixed-integer quadratic program and may be much harder to solve than its linear counterpart.

6.2. Requirements for charge nurses

The charge nurse in a unit is the person responsible for the shift. This role is typically filled by the head nurse when he or she is on duty, but at other times by the available staff. Because most RNs are qualified to be charge nurses, it is unlikely that a roster produced by the preference scheduling algorithm will have any uncovered periods. Nevertheless, to ensure compliance, this requirement should be included explicitly in the model. This can be done by adding the following set covering constraints:

$$\sum_{i \in N_c} \sum_{j \in S_i} a_{ijdp} x_{ij} \geq 1 \quad \text{for all } d \in D, p \in P, \quad (6)$$

where $N_c \subseteq N$ is the set of RNs that are qualified to act as charge nurses. Note that any demand for RNs with various types of certification, bilingual skills, cross-training and so on, can be treated in the same manner as charge nurses; i.e., by identifying the appropriate subsets of N and adding constraints similar to (6) to the model.

6.3. Multi-skilled workforce

The discussion so far has centered on RNs, but is generally applicable to all skill categories in a hospital. This follows because most units treat each category separately permitting the scheduling problem to be decomposed accordingly. In those cases where it is desirable to assign temporarily idle higher skilled workers to shifts that require lesser skills, model (5) can be extended to include two or more categories of nurses (e.g., RNs and LPNs). This process is often called *downgrading* and has been applied successfully by a few airlines in scheduling their flight crews (see Dawid et al., 2001).

For two skill categories, the necessary modifications require the use of two sets of demand constraints. The following additional notation is used to formulate the extended model.

N^H	set of nurses with higher level skills (RNs),
N^L	set of nurses with lower level skills (PLNs),
LD_{dp}^H	lower bound on demand on day d in period p for nurses with higher level skills,
UD_{dp}^H	upper bound on demand on day d in period p for nurses with higher level skills,
LD_{dp}^L	lower bound on demand on day d in period p for nurses with lower level skills,
UD_{dp}^L	upper bound on demand on day d in period p for nurses with lower level skills.

Now, instead of (1b) we have

$$LD_{dp}^H \leq \sum_{i \in N^H} \sum_{j \in S_i} a_{ijdp} x_{ij} \leq UD_{dp}^H \quad \text{for all } d \in D, p \in P, \quad (8a)$$

$$LD_{dp}^H + LD_{dp}^L \leq \sum_{i \in N^H \cup N^L} \sum_{j \in S_i} a_{ijdp} x_{ij} \leq UD_{dp}^H + UD_{dp}^L \quad \text{for all } d \in D, p \in P, \quad (8b)$$

where constraint (8a) ensures that a sufficient number of higher skilled nurses are selected to cover the corresponding demand, while (8b) allows either type to provide coverage at the lower level. To avoid duplicate coverage by the higher skilled nurses, it is necessary to include constraint (8a) in constraint (8b). Note that this formulation is independent of the relative pay scales because each nurse must be assigned a schedule; however, unless the cost coefficients in (5a) are weighted, there is an implicit assumption that preference violations and the use of outside nurses are treated equally regardless of skill category.

It is straightforward to transform (8a) and (8b) into single-sided constraints as was done with (1b). The new model would have twice as many variables and up to twice as many constraints. If the corresponding

integer program were too big to solve, a two-stage approach could be used. At the first stage, the higher level problem would be solved and any idle time due to excess coverage, identified. This idle time (or equivalently, the number of surplus nurses in each period) would then be applied (or assigned) to cover as much lower level demand as possible. At the second stage, the problem associated with the lower skilled nurses would be solved, but with the adjusted demand rather than the original demand.

7. Computational results

The column generation approach described in Section 5 was implemented in C++ and linked to the CPLEX 7.1 libraries to solve the IPs that arise at each major iteration. Run times were obtained through the clock function in C++. All computations were performed on a 1.1 GHz PC.

To test the effectiveness and usefulness of the approach, three sets of experiments were conducted on problems that included between 20 and 100 nurses. The first set was designed to see how well the algorithm performed on real problem instances. The second set was aimed at evaluating performance for a more difficult class of problems, and the third set was designed to help understand the influence that some of the model parameters have on the ability of the algorithm to find high-quality solutions in a timely manner.

To conduct the experiments, we used both real and randomly generated data. The real data were provided by Seton Hospital in Austin, Texas, and involved the scheduling of 20 and 68, respectively, over 4 weeks. These data were used in the first set of experiments referred to as the baseline. To work with larger problem instances in the second set of experiments, sign-up schedules were duplicated randomly to increase the number of available nurses, as were the demand requirements to extend the planning horizon. Also included were some special features like the requirement for charge nurses in every period.

The characteristics of the 12 problem instances investigated are displayed in Table 3. The entries in the ‘total demand’ column were computed by converting the demand for every period into its equivalent

Table 3
Summary of problems used in computational experiments

Problem no.	Nurses	Total demand (hours)	Total supply (hours)	Sign-up schedule (Δ hours)	Average no. of violations	Average no. of outside nurses	Feature
1	20	1824	1840	12	0	0.02	–
2	68	9824	8076	48	0	2.64	–
3	58	7138	7748	480	0	0.71	15% charge nurses
4	68	8068	8076	760	0	1.13	–
5	68	9824	8076	40	0	2.64	15% charge nurses
6	100	11,840	12,112	528	0	0.86	–
7	100	13,856	12,112	312	0	3.06	–
8	100	13,856	12,112	312	0	3.06	10% charge nurses
9	68	14,736	12,140	56	0	2.63	6-week horizon
10	100	20,784	18,186	456	0	3.00	6-week horizon
11	100	20,784	18,186	456	0	3.00	6-weeks horizon, 10% charge nurses
12	68	14,736	12,140	56	0	2.63	6-weeks horizon, 15% charge nurses

Table 4
Parameter settings for algorithm

Parameter	Value
Maximum number of columns in IP, MaxColTot	20,000
Maximum number of columns per nurse per iteration, MaxCol(α)	50
Maximum number of nodes in B&B tree, MaxNodes	1000
Maximum CPU time for IP per iteration (seconds)	1000
Maximum number of iterations, α^{\max}	20

number hours and summing over the planning horizon. Depending on his or her contract, a nurse is required to work for a certain number of hours during this period. The ‘total supply’ column gives the corresponding number of hours available. The difference between the two indicates the minimum number of hours that must be made up by outside nurses. However, when the total supply exceeds the total demand, it does not mean that we can always produce a solution without outside nurses.

The ‘sign-up schedule’ column indicates the difference (Δ) between the number of hours that must be made up by outside nurses as determined by the sign-up schedule, and the number of hours that must be made up as determined by the difference between the ‘total demand’ and ‘total supply’ columns. Excess coverage is not included in the calculation. Empirically, the larger the value of Δ , the more difficult a problem is to solve. The next two columns in Table 3 give the average number of violations per nurse and the average number of outside nurses needed per period, respectively, as determined from the sign-up schedule. The last column identifies two scenarios beyond the 4-week baseline that were investigated. Because the data did not indicate which employees were qualified to act as charge nurses, we randomly selected 10% to 15% of the total for a given run. In fact, this is a conservative estimate for the hospitals participating in the study. The actual number is closer to 25%.

Each problem instance was run using the parameter settings given in Table 4 unless otherwise stated. The results are summarized in the next subsections. The two main criteria used to judge the quality of a schedule are the number of preference violations and the number of outside nurses required. The implementation is judged in part by the amount of time spent in finding solutions. The detailed schedules are available from the authors.

7.1. Experiments with real data

A summary of the results for the first set of experiments is given in Table 5. The third and fourth columns indicate the average number of violations per nurse and the corresponding standard deviation. The maximum number of violations for any nurse is given in the fifth column. The next two columns report the average number of outside nurses required per period and the maximum number needed in any one period over the planning horizon. These values give an indication of how evenly distributed the outside nurses are.

Table 5
Summary of results for real problem instances

Problem	No. of nurses	No. of violations			Outside nurses		Columns in model	No. of iterations	Time (seconds)
		Average	Std. dev.	Max.	Average	Max.			
1	20	0.11	0.1	2	0	0	40	5	5
2	68	0.25	0.56	3	2.60	4	7347	6	215
3	58	0.16	0.49	3	0.04	1	5555	6	198

The ‘average’ value was computed by summing the number of outside nurses required per period and then dividing by the number of 8-hour shifts in the planning horizon (84 periods for 4-week baseline). The heading ‘columns in model’ refers to the total number of columns that were in the model when the computations were terminated, including those associated with the sign-up schedule which are used at the first iteration. Duplicate columns and columns whose penalty weight was above the threshold V_i^{\max} were discarded, and hence not included in this entry.

All problems were solved in less than 4 minutes. To a large extent, solution quality depends on the contractual agreements, the sign-up schedules, and the difference between total supply and demand. In all cases, the results indicate only a small number of preference violations and minimal need for outside nurses. The largest problem with 68 nurses required a total of 7347 columns or 108 columns per nurse before converging. In contrast, the smallest problem converged with the addition of only 20 new columns. As expected, computation times grew sharply with the size of the staff, but are still reasonable. Detailed results for problems 2 and 6 are given in Appendix B.

Several additional observations follow. These were corroborated in the second set of experiments.

1. Regarding the size of the IP, the greater the number of nurses, the more columns needed per nurse. The degree of difficulty increases with the number of nurses at an increasing rate that appears to be exponential.
2. The difference between the number of outside nurse hours needed in theory and the number of hours determined from the sign-up schedule provides an indication of how difficult a problem will be to solve. This difference is denoted by Δ in Table 3. Considering problem 2, for example, the difference between the total demand and total supply is $9824 - 8076 = 1748$ hours. The sign-up schedule, however, indicates that a total of 1796 additional hours are needed so $\Delta = 48$ hours. Thus the best the algorithm can do is to find a schedule that calls for 48 fewer outside nurse hours. Solution times tend to grow linearly with Δ .
3. The value of the slack variables and outside nurse variables in the sign-up schedule also affect the difficulty of a problem. When the number of periods that require outside nurses and the number of periods in which there is slack is about the same, the problem is more difficult to solve. This is due to the fact that the swapping heuristic has to generate more columns to reallocate the nurses from the slack periods to the period in which there is under-coverage.
4. When the charge nurse constraints are included in the model, the problem becomes more difficult to solve as the percentage of qualified nurses declines. This can be easily explained by the fact that the feasible region becomes increasingly tighter as the percentage falls. For the data sets investigated, a majority of the problems were infeasible when this parameter was below 15%.
5. The number of redundant columns generated may contribute significantly to the length of time required to solve a problem. As the iterations proceed, the number of redundant columns increases at a proportional rate. For a fixed time limit, the effort spent to weed out redundancy is wasted, compromising the effort to find more attractive columns and better solutions.

7.2. Experiments with derived data

The original data sets were used to generate additional problem instances with a slant towards making them more difficult. The following steps were taken.

- (a) Demand data were modified to obtain larger values of Δ while maintaining a relatively even balance between periods in which there was over- and under-coverage.
- (b) A certain number of existing nurse profiles were duplicated to increase the roster.
- (c) Sign-up schedules were duplicated in part to lengthen the planning horizon.

Table 6
Summary of results for derived problem instances

Problem	No. of nurses	No. of violations			Outside nurses		Columns in model	No. of iterations	Time (seconds)
		Average	Std. dev.	Max.	Average	Max.			
4	68	0.22	0.59	3	0.60	6	5491	7	256
5	68	0.04	0.27	2	2.60	5	8070	8	466
6	100	0.15	0.45	2	0.02	4	15,586	8	513
7	100	0.30	0.69	3	2.66	4	15,687	8	287
8	100	0.03	0.17	1	2.65	4	12,948	9	607
9	68	0.78	0.91	3	2.57	5	6836	7	1091
10	100	0.58	0.83	3	2.60	7	16,427	9	3178
11	100	0.20	0.55	3	2.60	4	18,293	9	3496
12	68	0.04	0.46	1	2.58	5	10,213	7	1100

With regard to point (c), the reason why we did not randomly generate sign-up schedules is because it is difficult to do so and make them realistic. Problems 5 and 8 were designed to study the effect of including charge nurses in the computations. The number of nurses that could serve in this capacity was determined randomly and limited to 15% of the total. The results are summarized in Table 6. For the 6-week instances, no time limit was imposed.

The presence of the charge nurse constraints and a longer planning horizon has the expected effect of increasing computation times. Comparing problem 2 with problem 5, we see that the running time increases by a factor of 2, while the number of columns in the final model only increases by about 700. In fact, during the iterations, about twice as many columns were generated for problem 5, but many were discarded because their penalty weights were too high. Comparing problem 7 with problem 8, we see an approximate twofold increase in running time, but a decrease in the number of columns by about 1600. The increase in running time in both cases is attributed mainly to the twofold increase in problem size due to the inclusion of charge nurses in the model. In the case of problems 6 and 7, the twofold increase in running time is due to the values of Δ . For problem 7, $\Delta = 324 (= 2068 - 1744)$ which is only about 55% of the value for problem 6.

As the planning horizon goes from 4 to 6 weeks, the problems become much more difficult to solve. Problem 10, for example, took about 45 minutes to converge. This order of magnitude increase is due to the quadratic increase in the number of swaps that have to be considered in the column generation algorithm. Upon closer examination, we found that most of the time was spent generating and evaluating columns, rather than solving the IP. This was true, regardless of the parameter settings or problem size.

With respect to the quality of the solutions, we see that the average number of violations is always less than or equal to 0.91, and all standard deviations are comparable. This implies that most nurses have very good schedules with no violations, but a few have schedules with two or three violations. In all cases, the average number of outside nurses per period is between two and three, which reflects the fact that for the most part, the demand is greater than the supply—the more difficult situation. Nevertheless, it was always possible to reduce the maximum number of outside nurses required to no more than twice the average, a reasonable result.

7.3. Parametric analysis

In the presence of shortages, the better the schedules with respect to preference violations, the more outside nurses that are likely to be needed to fill in the gaps. To investigate this tradeoff, we studied the impact of limiting the number of violations permitted per nurse (i.e., maximum column weight) on the average number of outside nurses in the final schedule. Data sets 4 and 7 were used in this experiment

because the total number of outside nurse hours needed (determined from the solutions of problems 4 and 7) was considerably greater than the minimum number of hours specified by the total demand in Table 3. In the analysis, the maximum penalty allowed per column was varied from 1 to 32. Recall that a penalty of 1 is equivalent to one violation, and a penalty of 32 to 6 violations. This compares with a penalty of 50 points for each 8-hour outside shift used.

In general, including columns in the model with ever more violations increases the number of alternatives for the IP solver, and so should yield solutions with fewer and fewer outside nurses. At some

Table 7
Statistics for parametric analysis

	Maximum penalty allowed					
	1	2	4	8	16	32
<i>Total columns generated</i>						
68 nurses (problem 4)	2761	4583	4788	6532	8618	9030
100 nurses (problem 7)	7384	8204	9757	11,221	14,865	15,798
<i>High penalty columns discarded</i>						
68 nurses	66,252	29,694	4804	4755	160	78
100 nurses	89,150	53,955	4043	349	125	105
<i>Average number of violations</i>						
68 nurses	0	0.16	0.46	0.49	0.38	0.38
100 nurses	0	0.17	0.27	0.27	0.31	0.22
<i>Standard deviation of violations</i>						
68 nurses	0	0.61	0.85	0.92	0.88	0.88
100 nurses	0	0.61	0.79	0.77	0.76	0.79
<i>Average number of outside nurses</i>						
68 nurses	0.36	0.20	0.04	0.01	0	0.01
100 nurses	0.29	0.11	0.02	0	0	0
<i>Computation time</i>						
68 nurses	259	236	253	258	>1000	211
100 nurses	465	485	479	525	>1000	>1000

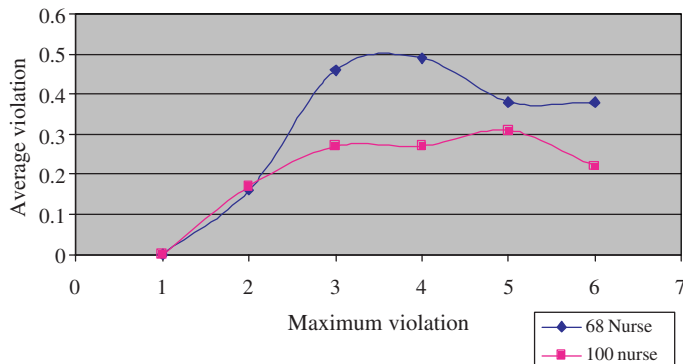


Fig. 2. Parametric analysis of average violations as maximum penalty varies.

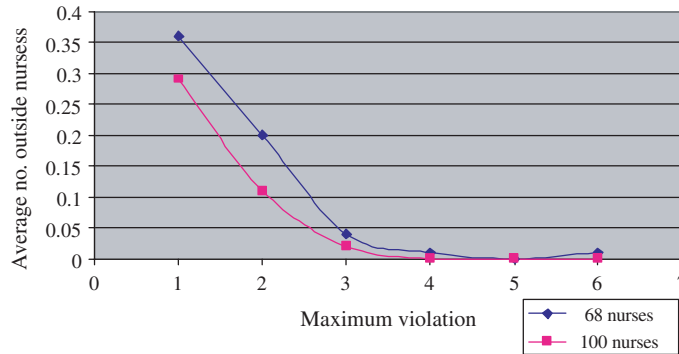


Fig. 3. Parametric analysis of outside nurses as maximum penalty varies.

point, though, diminishing returns will set in with the consequence that the marginal improvement will go to zero.

Table 7 summarizes the results for six runs. As can be seen, when only a small number of violations are permitted, an overwhelming number of columns are rejected and the average number of violations is relatively low. Beyond 3 violations (penalty of 4), though, the solution quality does not improve much so this might be a good cutoff point.

A portion of these results is plotted in Figs. 2 and 3 to highlight the influence of the penalty parameter. We see, for example, that the average number of violations is extremely low when the maximum penalty allowed is low, increasing steadily and then tapering off. This follows because the initial feasible sign-up schedules are good. When fewer columns are generated for lower penalty values, it is difficult to find schedules that use fewer outside nurses. On the other hand, an increase in the maximum penalty decreases the need for outside nurses to obtain a balanced schedule. After a value of 3, though, the number of outside nurses required drops to near zero.

8. Summary and conclusions

In this paper, a robust column generation procedure was presented for solving the nurse preference scheduling problem. At the center of the algorithm is a heuristic that identifies attractive schedules by adaptively swapping single and double periods. Although the computations were initiated with a sign-up schedule, other possibilities include the use of rotational schedules or an initialization procedure based on, say, the previous month's rosters. Solutions for problems with up to 100 nurses and planning horizons up to 6 weeks were found within a matter of minutes in most cases. These results provided the confidence for full implementation in a new health care management system now being tested at several hospitals.

An important feature included in the methodology is the requirement for a charge nurse in every period. Modeling this option, however, increased the run time by a factor of two to three. On a related point, there may be several types of nurses that must be scheduled in a hospital unit. The approach discussed here can be extended to deal with this situation either by adding constraints (8a) and (8b) to the model, or by solving each problem in turn, beginning with the highest skill category (i.e., RNs). Once a solution is found for a specific skill category, any excess coverage can be used to satisfy demand at the next lower category.

Other areas in which generalizations or improvements are possible relate to the shift structure and the generation of columns. Many hospitals use more than four starting times and may have shifts that are other than 8 and 12 hours in length. An efficient way to deal with the general case is needed, particularly with regard to the swapping procedure and the elimination of redundant columns.

Acknowledgements

This work was supported in part by the National Science Foundation under Grant No. DMI-0218701.

Appendix A. Legality checking and redundancy

Each new column obtained from the neighborhood swap procedure must be checked to see if it violates any of the hard constraints. This is the most tedious task in the overall methodology because many of the rules may be quite intricate and not all of them will necessarily be applicable to all nurses (e.g., nurse i must have every other Monday or Tuesday off and can only work four days a week). For a specific nurse, it is often possible to include some of the unique rules in the construction of sets $A(i)$ and $B(i)$ so no columns are generated that lead to a violation. We do this for the first ‘MinReq’ requests submitted by a nurse.

In general, the checking procedure should be modularized and flags used for each nurse so only the appropriate rules are checked. The algorithm presented below is designed specifically for the scheduling problem associated with the shift set {D, E, N, AM, PM} and corresponding demand periods of length 8, 4, 4, 8 hour, as shown in Fig. 1. The first four steps are aimed at uncovering illegal work patterns of a general nature. The computational effort for these checks is minimal. The remaining steps address the other hard constraints in Table 2. For example, Step 6 considers the maximum workstretch and Step 7 checks the weekend constraints.

Legality_Check_Algorithm

Input: New column A_i^{new} with periods a and b swapped.

Output: <true> if swap is legal, <false> otherwise.

Notation: *workstretch* = maximum number of consecutive working days in a schedule associated with A_i^{new} .

Step 1. Set $idx = b \bmod 4$ \ \ current period in the day.

Step 2. Depending on the value of idx , check one of the following four conditions.

- 0: If $b - 1$ is active or $b + 2$ is active in A_i^{new} then go to Step 9, else go to Step 3. (This test identifies the illegal patterns 0001 1000, 0011 1100, 1110, and 1111.)
- 1: If both $b - 1$ and $b + 1$ are active or both $b - 1$ and $b + 1$ are inactive in A_i^{new} then go to Step 9, else go to Step 3. (These conditions identify the illegal patterns 1110, 1111, 1101 and 0100.)
- 2: If both $b - 1$ and $b + 1$ are active or both $b - 1$ and $b + 1$ are inactive in A_i^{new} , then go to Step 9, else go to Step 3. (This test identifies the illegal patterns 0111, 1111, 1011 and 0010.)
- 3: If $b + 1$ is active or $b - 2$ is active in A_i^{new} , then go to Step 9, else go to Step 3. (This test identifies the illegal patterns 0001 1000, 0011 1100, 0111, and 1111.)

Step 3. Set $idx = a \bmod 4$. Now check for the illegal patterns 0100 and 0010.

Step 4. If ($idx = 0$ or $idx = 1$) and ($a + 1$ is active) then go to Step 9, else go to Step 6.

Step 5. If ($idx = 2$ or $idx = 3$) and ($a - 1$ is active) then go to Step 9, else go to Step 6.

Step 6. If (*workstretch* > 6) go to Step 9, else go to Step 7.

Step 7. If (weekend constraints violated) go to Step 9, else go to Step 8.

Step 8. If (other hard constraints violated) go to Step 9, else return <true>.

Step 9. Return <false>.

Once a column passes the legality test, a check must be made to ensure that it does not duplicate an existing schedule for the nurse being considered (of course, two nurses can have the same schedule without causing a problem). Although the sets $A(i)$ and $B(i)$ may change at every iteration, they are likely to overlap to an extent that makes duplication unavoidable. This is especially true when the base schedule is the same for a particular nurse from one iteration to the next. To check for redundancy, we compute an index

(*ColumnIndex*) for each column and store it in a hashing table, one table per nurse. When a new column is generated, its index is computed and the hashing table checked to see if the index is listed. In our implementation, we use a table with 50 rows.

Hash Table Algorithm

Input: New column A_i^{new} , objective function coefficient c_i^{new} , swap periods a and b .

Output: <true> if A_i^{new} is not redundant, <false> otherwise.

Notation:

$s(d)$ shift type nurse is assigned on day d ,

$\Psi(s(d))$ random number associated with shift type the nurse is assigned on day d ,

$\Phi(d)$ random number associated with day d ,

\hat{d} last day of the planning horizon,

ColumnIndex index for new column A_i^{new} ,

ArrayIndex row index in hashing table associated with A_i^{new} .

Step 1. Calculate $\text{ColumnIndex} = \sum_{d=1}^{\hat{d}} [\Psi(s(d)) + \Phi(d)][\Psi(s(d+1)) + \Phi(d+1)]$.

Step 2. Calculate $\text{ArrayIndex} = [c_i^{\text{new}} + (a \times b)] \bmod 49$.

Step 3. Go to *ArrayIndex* in hashing table and see whether or not *ColumnIndex* has been generated previously. If (yes) then discard A_i^{new} and return with <false>; otherwise, return with <true>.

Appendix B. Algorithmic performance

The data in Tables 8 and 9 give an indication of how the column generation algorithm (CGA) performed on two representative problems. In most instances, the IP was solved at either the root node or within a few nodes for each major iteration α . Although the relaxed LP solutions were not always integral, CPLEX's feasibility heuristic was extremely effective in finding integer solutions that indeed were optimal. In all but a few cases did the branch and bound tree grow beyond a handful of nodes. This usually happened when the solution contained a small number of outside nurses per shift, as in problem 8 where 14 nodes were needed at iteration 5.

The vast majority of time was spent on the column generation aspect of the algorithm—performing swaps, testing legality, and checking redundancy. Roughly speaking, 15% of the time went to solving the IPs and 85% to generating columns. When the IP was difficult to solve, though, about 50% of the computational effort was spent in branch and bound. Of the four stopping criteria used, the infeasibility criterion dominated. Recall that in this case, the computations are halted when no feasible solution is found in

Table 8
Detailed computational results for problem 2 (68 nurses)

Major iteration (α)	Nodes in B&B tree	Node best solution found	%LP gap at node 0	% optimality gap	Total solution time (seconds)	Solution time for IP (seconds)	Solution time for CGA (seconds)
1	0	0	0	0	~0	~0	~0
2	0	0	0	0	1	~0	1
3	1	1	0	0	41	2	39
4	0	0	0	0	29	5	24
5	0	0	–	Infeasible	49	11	38
6	0	0	–	Infeasible	47	12	35
7	0	0	–	infeasible	48	14	34
Total					215	44	171

Table 9
Detailed computational results for problem 6 (100 Nurses)

Major iteration (z)	Nodes in B&B tree	Node best solution found	% LP gap at node 0	% optimality gap	Total solution time (seconds)	Solution time for IP (seconds)	Solution time for CGA (seconds)
1	0	0	0	0	~0	~0	~0
2	0	0	0	0	1	~0	1
3	0	0	0	0	66	3	63
4	1000	10	5.16	4.53	157	117	40
5	1000	14	25	1.71	180	119	61
6	0	0	–	Infeasible	67	29	38
7	0	0	–	Infeasible	52	26	26
8	0	0	–	Infeasible	84	25	59
Total					607	319	288

three consecutive iterations for the given value of the parameter MaxGap—the maximum number of outside nurses permitted per period.

References

- Arthur, J.L., Ravindran, A., 1981. A multiple objective nurse scheduling model. *AIIE Transactions* 13 (1), 55–60.
- Berrada, I., Ferland, J.A., Michelon, P., 1996. A multi-objective approach to nurse scheduling with both hard and soft constraints. *Socio-Economic Planning Science* 30 (3), 183–193.
- Burke, E.K., Causmaecker, P.D., Berghe, G.V., 1999. A hybrid tabu search algorithm for the nurse rostering problem. In: *Simulated Evolution and Learning 1998* McKay, B. et al. (Eds.), Lecture Notes in Artificial Intelligence, 1585. Springer, Berlin, pp. 187–194.
- Burke, E.K., Causmaecker, P.D., Petrovic, S., Berghe, G.V., 2001. Fitness evaluation for nurse scheduling problem. In: *Proceedings of the Congress on Evolutionary Computation*, Seoul, South Korea, vol. 2. IEEE Press, pp. 1139–1146.
- Burns, R.N., 1978. Manpower scheduling with variable demands and alternate weekends off. *INFOR* 16 (2), 101–111.
- Cheng, B.M.W., Lee, J.M.L., Wu, J.C.K., 1997. A nurse rostering system using constraint programming and redundant modeling. *IEEE Transactions in Information Technology in Biomedicine* 1 (1), 44–54.
- Dawid, H., Konig, J., Strauss, C., 2001. An enhanced rostering model for airline crews. *Computers & Operations Research* 28 (7), 671–688.
- Dowsland, K.A., 1998. Nurse scheduling with tabu search and strategic oscillation. *European Journal of Operational Research* 106, 393–407.
- Ferland, J.A., Berrada, I., Nabli, I., Ahiod, B., Michelon, P., Gascon, V., Gagné, E., 2001. Generalized assignment type goal programming problem: Application to nurse scheduling. *Journal of Heuristics* 7, 391–413.
- Griesmer, H., 1993. Self-scheduling turned us into a winning team. *Management Decisions* 56 (12), 21–23.
- Howell, J.P., 1998. Cyclical scheduling of nursing personnel. *Hospital JAHA* 40, 77–85.
- Jaumard, B., Semet, F., Vovor, T., 1998. A generalized linear programming model for nurse scheduling. *European Journal of Operational Research* 107, 1–18.
- Kawanaka, H., Yamamoto, K., Yoshikawa, T., Shinogi, T., Tsuruoka, S., 2001. Genetic algorithm with constraints for the nurse scheduling problem. In: *Proceedings of Congress on Evolutionary Computation*, Seoul, South Korea, vol. 2. IEEE Press, pp. 1123–1130.
- Miller, H.E., Pierskalla, W.P., Rath, G.J., 1976. Nurse scheduling using mathematical programming. *Operations Research* 24 (5), 857–870.
- Nonobe, K., Ibaraki, T., 1998. A tabu search approach to the constraint satisfaction problem as a general problem solver. *European Journal of Operational Research* 106, 599–623.
- Okada, M., 1992. An approach to the generalized nurse scheduling problem—generation of a declarative program to represent institution-specific knowledge. *Computers & Biomedical Research* 28, 417–434.
- Purnomo, H.W., 2002. Solving nurse scheduling problem using column generation approach. Ms Thesis. Graduate Program in Operations Research and Industrial Engineering, The University of Texas, Austin, TX.
- Randhawa, S.U., Sitompul, D., 1993. A heuristic-based computerized nurse scheduling system. *Computer & Operations Research* 20 (8), 837–844.

- USDHHS, 2002. Projected Supply, Demand and Shortages of Registered Nurses: 2000–2020. National Center for Health Workforce Analysis. US Department of Health and Human Services, Rockville, MD.
- Valouxis, C., Housos, E., 2000. Hybrid optimization techniques for the workshift and rest assignment of nursing personnel. *Artificial Intelligence in Medicine* 20, 155–175.
- Warner, D.M., 1976. Scheduling nursing personnel according to nursing preference: A mathematical programming approach. *Operations Research* 24 (5), 842–856.
- Wolsey, L.A., 1998. *Integer Programming*. John Wiley & Sons, New York.