RESIDENT SCHEDULING PROBLEM

by

Muhannad Hasan Ramahi

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE
IN
INDUSTRIAL AND SYSTEMS ENGINEERING

APPROVED:

Hanif D. Sherali, Chairman

Subhash C. Sarin

John E. Kobza

September, 1998

Blacksburg, Virginia

Resident Scheduling Problem

by

Muhannad H. Ramahi

Dr. Hanif D. Sherali, Chairman

Industrial and Systems Engineering

(ABSTRACT)

This thesis is concerned with the Resident Scheduling Problem (RSP) in which a good schedule is desired that will meet both departmental requirements and residents' preferences. Three scenarios that represent most situations and account for various departmental requirements and needs are described. Although similar scheduling problems are considered in the literature, no analysis exists that adequately deals with this specific problem. The problem is modeled as a mixed-integer program (MIP) and heuristic solution procedures are developed for the different identified scheduling scenarios. These procedures exploit the network structure of the problem which is an important feature that enhances problem solvability. For the sake of comparison, the problem is also solved exactly via the CPLEX-MIP package. The contribution of this work is important since many hospitals are still utilizing manual techniques in preparing their own schedules, expending considerable effort and time with less scheduling flexibility.

Acknowledgments

I would like to thank my advisor, Dr. Hanif D. Sherali for his invaluable support, guidance and patience. His remarkable knowledge, enthusiasm, and dedication inspired me to join this marvelous world of Operations Research and obtain my Masters degree in Industrial Engineering (Operations Research).

I would also like to thank Dr. Subhash C. Sarin and Dr. John E. Kobza for serving on my committee and for their support.

My sincere thanks go to my parents, my grandmother, my sisters, my uncle, and the rest of my family for their love, support, and encouragement throughout my life.

Finally, I would like to thank and dedicate this thesis to my wife Omayma for her love, support, and patience, and to my three lovely daughters Dana, Renad and Lianne for the joy, love and happiness they brought into my life.

Table of Contents

Chapter 1
Introduction: Problem Description

The Resident Scheduling Problem (RSP) attempts to find a best (optimal) match between residents' preferences and departmental requirements in assigning night shifts to residents during a typical scheduling horizon. In particular, the residents express their preferences, for example, by filling out a form indicating their most and least desirable choices for work-nights over the horizon. On the other hand, management tries to assign the residents to shifts according to departmental needs for each night, complying with residents' expressed preferences as closely as possible. In attaining these goals certain constraints must be satisfied as follows:

- Residents must work a certain minimum number of nights.
- If needed, residents must work extra nights (say, up to three nights) with a certain penalty imposed on this requirement.
- Residents must have some specified off-nights between successive work-nights.
- Residents are usually classified into different groups according to seniority levels. A minimum number of each group must be present each night. Also, a certain number of backup/moonlighter residents (outside the department) might be required to be available each night to allow for special contingencies, with a penalty incurred when such residents are scheduled.

One resident mix requirement that will be used in this thesis is obtained from *St. John Hospital and Medical Center*, Detroit, Michigan. In general, hospitals have different requirements and mix restrictions. In our case study, the number of residents is 50 (1st year - 3rd year). Residency programs are typically 3-years to 5-years in duration, depending on the specialty.

The goal of this research effort is to devise a method for effectively scheduling the given set of residents based on their preferences to the required shifts, taking into consideration hospital and union requirements. To accomplish this, we formulate this problem as a mixed-integer programming problem (MIP) in which the objective is to minimize the total cost that reflects the penalties ascribed to each shift by each resident. In addition, hospital requirements and policies must be satisfied in the form of constraints specified in the problem. Additional union-imposed constraints are formulated to ensure that residents have off-nights gap between their work stretches; also, some off-night weekends have to occur. In some cases, the above requirements cannot be attained internally, and therefore, residents from other departments are needed to fill the gap. These residents are classified as extraneous residents, and their usage will incur a penalty attached to the objective function to reflect the undesirability of this occurrence.

We explore the solution of the mixed-integer program as well as the design of a suitable heuristic procedure based on a restructuring of the problem to reveal an embedded network structure. It should be emphasized that the parameters and costs prescribed in the problem can significantly impact the solution obtained. These parameters need to be therefore selected carefully, in a manner that reflects the relative priorities imposed by the hospital administration and staff.

Chapter 2

Literature Review

Manpower scheduling, in general, involves the allocation of workers to specific shifts (time-slots) in order to minimize costs in such a way that demand is met and certain requirements and policies are satisfied. Such problems arise in many aspects of life including hospitals, police departments, banks, hospitals, airlines, etc. A first formal approach to this problem appears to have been presented by Edie (1954), and since then, many researchers have been attracted to this problem. A first integer programming formulation was introduced by Dantzig (1954) as follows:

$$\text{minimize} \sum_{k \in K} c_k X_k,$$

$$\text{subject to} \sum_{k \in K} a_{kt} X_k \geq b_t \qquad \text{for all } t \in T,$$

$$X_k \geq 0 \text{ and integer,}$$

where K is the set of available shifts, T is the set of planning periods that the shift schedule covers, b_t is the number of employees needed in period t, c_k is the cost of assigning an employee to shift k, a_{kt} is equal to one if period t is a work period for shift k and zero otherwise, and X_k is an integer variable defined as the number of employees assigned to shift k, $k \in K$.

It is noted that for complex scheduling problems encompassing different types and lengths of shifts, different start and end times, the existence of several break-windows such as relief (rest) breaks or lunch breaks, a standard integer programming model tends to be less useful as the number of variables can become quite large. A new method for tackling such problems was introduced by Aykin (1996), who further elaborates on these difficulties in the context of determining an optimal shift schedule having multiple break-windows. He presents a new formulation for this problem in which he reduces the number of variables, but at the expense of adding more constraints. The model proposed is applicable to a 24-hour or less duration operation, and can flexibly accommodate various combinations of shifts. Aykin was able to solve several problems that included between 1,728 and 8,640 shift variations, and five demand patterns, thus proving the efficiency of the method for large problems.

The difficulties mentioned above have induced researchers to develop heuristic methods in search for efficient solutions. For example, Segal (1974) considers a telephone operator scheduling problem in which each shift can include up to 15-minute relief and one 30-minute lunch break. Break-windows are specified as 1.5 hours in duration so that each 15-minute relief break may start at six different times. The problem is formulated as a network flow problem and the shift scheduling problem is solved heuristically over three stages. Many other researchers utilize heuristic procedures such as Keith (1979) who models the same problem but with less flexibility by pre-setting the break-windows. In this approach, he first solves the linear programming relaxation of the integer model and employs a rounding heuristic to obtain a feasible shift schedule. Break-windows are examined subsequently in an attempt to improve the schedule thus obtained. Henderson and Berry (1976) solve the shift scheduling problem with multiple breaks heuristically, considering from 7,120 up to 15,000 different shift variations.

Vohra (1985) studies the cyclic staffing problem which considers minimizing the number of workers needed per day in a 7 day cycle schedule. The problem insures that there are 5 consecutive work days for each worker and that the remaining 2 days are idle.

Another type of scheduling problem examined in the literature is concerned with evenly distributing the workload among drivers in a mass transit system. This problem was studied by Bianco and Bielli (1992). To accomplish this task, rosters are developed which compose daily work-loads into a set of weekly or monthly plans. A roster is assigned to each driver, and sometimes, a complete roster rotation is required to ensure the balance of workloads among all drivers. Earlier work in this area was performed by many researchers such as Baker (1974 and 1976), Baker and Magazine (1977), and Bartholdi *et al*. (1980), who considered the problem of minimizing the maximum workload. They formulated the problem as a Multilevel Bottleneck Assignment (MBA) problem and proved that it was NP-hard. A heuristic approach was developed providing asymptotically optimal solutions. Bianco and Bielli's work involved minimizing the cost of different types of roster combinations, controlling the starting and ending times of rosters, and ensuring a minimum of two days rest between rosters. They devised a heuristic procedure to iteratively solve the problem of balancing the workload among rosters, solving an MBA problem at each iteration.

A large-scale tour scheduling problem was studied by Jarrah *et al*. (1994). The tour scheduling problem involves solving three related problems. The first one involves finding the days-off assignment; the second deals with the daily shift assignment, and finally, the third problem produces the weekly tour. The problem without any demand variations during the week days, and no demand on the weekends, and with fixed start and end shifts is relatively easy to solve. On the other hand, the problem becomes very complex when daily patterns change, demand exists on the weekends, and part-time workers constitute a major portion of the workforce such as in the airline industry, and many public service industries. The methodology suggested by Jarrah *et al.* involved combining the shift and days-off for a less than 24-hours a day shift. The problem was formulated as an integer programming problem, and aggregate variables and related cuts were introduced. As such, the original problem yielded seven subproblems representing each day of the week.

A heuristic approach along with an enumeration strategy was utilized that produced lower and upper bounds that were shown to converge to near optimal solutions. The proposed methodology was applied to a United States Postal Service Facility that required many constraints to be fulfilled such as half-hour breaks, minimum full-time to part-time ratios, variable start times, four and five work days, among others.

Another important aspect of the scheduling problem is adherence to worker's preferences, which may be ignored in order to satisfy job due dates or job requirements. Yura (1994) studies a problem that incorporates this consideration, and formulates it as a linear goal programming problem based on different scenarios of satisfying worker's preferences with respect to days-off and over-time assignments, without violating job due-date constraints. It is noted that in previous studies, this objective was not fully accomplished, especially under heavy workload conditions. Also, in this study, the author illustrates his method using a very small example (2 workers and 8 jobs), and therefore, the performance of the method on larger problems having different variations of workers and jobs is questionable.

Beasley and Cao (1996) solve a crew scheduling problem (CSP) which involves assigning *K* crews to *N* tasks, given fixed starting and ending shifts, without exceeding the workload limit on each worker. The crew scheduling problem arises in many situations such as in the airline, and mass transit industries. Beasley and Cao formulate the problem as a zero-one integer linear programming problem, and apply a Lagrangian relaxation method to produce a lower bound on the problem. They propose a tree search procedure (branch-and-bound) in case the Lagrangian dual optimization fails to produce an optimal solution to the original CSP. This approach is generic with respect to its ability to accommodate other constraints and specific situations/instances.

An interesting crew scheduling problem that has received considerable attention arises in the airline industry. Graves *et al.* (1993) solve such a flight crew scheduling problem for United Airlines.

The purpose here is to develop a speedy responsive system to schedule changes that result in cost reductions. The developed system generates efficient solutions for medium and large problems and results in significant savings to the company (around $16 million dollars annually). The system is comprised of two components; a generator that creates potential crew pairings and an optimizer that feeds off from the generator to create a set of pairings that cover all the flights at a minimum total cost. Once a solution is obtained by the optimizer, several cycles between the generator and the optimizer may be performed to further refine the solution. Barnhart *et al.* (1993) formulate a mixed-integer programming problem differently by tightening the linear programming relaxation in an attempt to find an efficient solution. The method developed was applied to a real problem to validate the efficiency of their proposed formulation thus proving the importance of problem formulation in practical mixed-integer programming problems. Chu *et al.* (1997) solve large-scale crew scheduling problems formulated as zero-one integer problems by generating crew pairings satisfying duty rules. Dual variables obtained from successively solving these relaxed problems were fed into a graph-based branching heuristic to further refine the solution. Vance *et al.* (1997) deviate from the traditional set partitioning problem used to model airline crew scheduling problems. Specifically, a new formulation is utilized that has a tight linear programming relaxation, and a decomposition approach is used to solve the problem. In this methodology, the problem is solved over two stages. The first stage generates the duty periods that cover flights in the schedule. The second stage then produces pairings using the previously obtained duty periods.

Narasimhan (1996) examines the problem of scheduling shifts of a hierarchical workforce. Earlier, this problem was studied by Emmons and Burns (1991). Problems of this type are encountered in health care personnel scheduling, job shop employees, and maintenance crew scheduling scenarios. The fundamental task is to derive a seven day schedule using a single shift per day. Job requirements must be met through the scheduling of workers, and a maximum of five consecutive days a week with two days off every week should be ensured, including a set proportion of off weekends. Furthermore, variations in demand occur for each class of workers and between weekends and weekdays.

The objective is to determine a most economical mix of employees to satisfy the required daily demand and the above mentioned constraints. The nature of this problem permits a certain class of workers to substitute for others, and not vice versa, hence the term *hierarchical* in the classification of the problem.

Beaumont (1997) describes a case study for a staff scheduling problem using mixed-integer programming (MIP). The problem involves finding the number of employees needed, including permanent employees and outside contractors, for a given shift based on demand, along with the start and end time of shifts. According to this study and similar studies done by others, the MIP formulation tends to be large and intractable, and the computer time needed to solve the problem is prohibitively expensive. In addition, the goal was not simply to find a solution to this problem, but also to be able to develop a flexible procedure that permitted the client to make various what-if explorations and sensitivity analyses with minimum effort and time. Such sensitivity runs were also required in case there were changes in the scheduling requirements and costs. One of the main issues treated in this study involved estimating cost parameters since the solution obtained was very much dependent on these values. Thus, many parameter variations along with many scenarios suggested by the client were also studied and documented. .

A similar problem to the one considered in this thesis is the nurse scheduling problem. This problem involves generating a schedule of working days and days off for each nurse in a hospital staff. In general, there exist three typical daily shifts: day shift (08:00-16:00), evening shift (16:00-24:00), and night shift (00:00-08:00). There exist three different approaches that have been used to deal with this problem. The first approach uses a decision support system to assist the scheduler (head-nurse) in assigning shifts using the computer in an interactive way. See for example Okada (1988) and Smith *et al*. (1977). The second approach uses mathematical programming techniques, and can itself be divided into different categories. One category involves the optimization of a single objective.

A typical objective would be to maximize total satisfaction while meeting with certain constraints relating to administrative issues and availability of workers. In general, these types of problem formulations are case specific and computationally intensive. See for example, Baker (1974), Bartholdi *et al.* (1980), and Burns (1978). Another category is to use multi-objective mathematical programming techniques. Examples include the work of Musa *et al.* (1984) and Ozkarahan *et al.* (1988). Efficient solutions are sought based on a prioritization of the objectives. The third approach is a hybrid method that utilizes an expert system coordinated with mathematical programming techniques. Chen and Yeung (1993) develop such a system, producing schedules using zero-one programming models, while generating nurse rotations via an expert system.

The quality of the schedule produced is based on how well the assigned shifts relate to what the employees (nurses in this case) desire. The solution must also conform with departmental requirement policies and union contracts. Warner (1976) and others such as Schuette (1973) delineate certain important issues related to the nurse scheduling problem as noted below.

1. *Coverage:* This basically states the number of nurses required on a certain shift as a proportion of the total staff required on each shift.
2. *Quality and stability of the schedule:* This is expressed by each nurse in regard to preference towards the schedule with respect to the number of working days, work stretch, days off, adherence to weekend policy, rotation, etc.
3. *Flexibility of schedules:* This denotes how well the schedule can handle changes and contingencies resulting from absentees, vacation requests, etc.
4. *Fairness:* This describes how the schedule of each nurse compares with that of other nurses.
5. *Cost:* This represents an associated measure of undesirability attached to each schedule.

One of the first approaches toward solving the nurse scheduling problem was the cyclical scheduling method. This procedure describes a schedule for a certain number of days, say a week, and then extrapolates it over several following weeks. This approach produces high stability schedules since the work span is very long. On the other hand, the method is somewhat rigid toward permitting any changes, even slight ones. Warner (1976) proposes a method to overcome this inflexibility toward change by dissecting the problem into two phases. The first phase incorporates policy decisions into the schedules such as proper coverage, rotation, weekends, days off, etc. Consequently, these policies are interpreted into a set of parameters and decision variables that will constitute the mathematical programming formulation (second phase). An input from each nurse is also solicited with respect to the desirability of a schedule. This input is then evolved into a set of possible schedules for each nurse. The objective of the model is to maximize nurses' preferences, taking into consideration the system constraints. Due to the existence of many constraints, not many feasible schedules are generated by this method, hence restricting the overall feasibility of the model. The solution algorithm used is that developed by Blantify and Blackburn (1969) with some minor adjustments. This procedure attempts to minimize total constraint violation in the first phase, and then maximizes the original objective (nurses' preferences) while restricting the total violation of constraints to that obtained in the first phase. A somewhat similar approach is adopted by Miller *et al.* (1976). Here, the constraints are divided into two sets. The first set represents the feasibility set which contains all the hard constraints that must be satisfied. The second set represents soft constraints that may be violated, such as special requests by nurses, but with a penalty incurred due to this violation. The objective then minimizes the penalty incurred from the soft constraints.

Rosenbloom and Goertzen (1987) present another study of this problem in which they consider the daily changes in shift requirements and the many constraints imposed by the hospital and labor unions. Their approach focuses on making the problem as generic as possible so that it is applicable to a variety of situations. In particular, they develop an algorithm that constructs a nurse schedule by dividing the problem into three phases.

The first phase generates a set of feasible solutions (schedules) to choose from. Relatively few solutions are generated since the problem is assumed to be cyclic. Here, feasibility describes schedules that conform to the constraints satisfying labor requirements and administration policies. The second stage formulates an integer programming problem to optimally impose the individual nurse schedules generated in phase one. This also gives the number of nurses required for each shift. Finally, the third stage produces the final schedule for each nurse. It is worth noting that the authors overcome the difficulties in solving such a large integer programming problem through dynamic and static elimination techniques, which simply eliminates any schedule from being considered in a consequent cycle based on a constraint violation, thus keeping the problem as small as possible.

Kostreva and Jennings (1991) develop an approach that strongly focuses on nurses' preferences. In this study, nurses provide aversion "hate" points to various aspects of the system such as work stretch, working shifts, days off, rotation pattern, etc., using a questionnaire developed for this purpose. The main goal then is to minimize the total system aversion. This hate point concept was originally suggested by Passini (1978), using emotions to derive a computerized scheduling system. The method proposed by Kostreva and Jennings involves solving two subproblems that alternatively generate a feasible set of schedules and optimize them in order to reach the final prescribed solution.

Another attempt to solve the nurse scheduling problem is presented by Weil *et al.* (1995) using a constraint programming approach. This method combines operations research and logic programming techniques. Specifically, the authors use object oriented programming in which they define an object class called "nurse" that encapsulates all the required information pertaining to nurses such as name, title, identity, etc. Other object classes are used to define the problem constraints. A very nice feature of object-oriented programming is its flexibility to accommodate any changes in the system. For example, if new nurses are added or deleted, this involves a simple add/delete step applied to the class "nurse". Constraints that apply to all nurses can be defined once and then associated with the entire "nurse" class, instead of having to impose these constraints for each nurse separately, which saves much needed computer memory. Any modification to the system is done with a minimal effort compared to traditional methods. A 30 single-skill-nurse schedule having 420 variables and 1470 constraints was solved using this method in 12 seconds on a DecStation 3000/200 computer.

In a more recent approach for solving the nurse rostering problem, Cheng *et al*. (1997) propose a constraint satisfaction and redundant modeling technique. They justify the need for their method since the problem at hand involves 25 to 28 nurses of different skill classes, with 11 possible shift types, thereby creating a huge solution space. The nurses were also permitted to make pre-assignment shifts which could possibly create a tremendous reduction in the work force available in a specific shift if a substantial proportion of nurses decided not to work on a specific day/shift. A constraint satisfaction approach was adopted, which is defined in Mackworth (1977) as follows: "We are given a set of variables, a domain of possible values for each variable, and a conjunction of constraints. Each constraint is a relation defined over a subset of the variable, limiting the combination of values that the variables in this subset can take. The goal is to find a consistent assignment of values to the variables so that all the constraints are satisfied simultaneously." As mentioned above, the pre-assignment of shifts creates problems. This is overcome using redundant modeling. This involves modeling a single problem in more than one manner, and then connecting these different models to speed up solution time. The authors modeled their problem in two different ways. In the first model, nurses were the variables and shifts were the domain, while in the second model, shifts were the variables and nurses were the domain. The connection is done via special types of constraints (channeling constraints) that were used to link these two models. Their work was implemented on the Accident and Emergency Unit of the Tang Shiu Kin Hospital, Hong Kong.

The literature review has revealed many related problems and models dealing with similar scheduling contexts, but none for the specific Resident Scheduling Problem (RSP) at hand. As we shall see, our problem shares some similarities with some of the above mentioned studies, but there are also notable differences and variances that need to be specifically treated. We now proceed to describe our problem in more precise terms, and to develop a mathematical programming model for this problem.

Chapter 3
Model Development

3.1 *Problem Formulation*

We are given some m residents of various classes, such as senior residents, junior residents and rotators (residents from other departments). Let us denote $M = \{1, 2, ..., m\}$ to be the set containing all the residents, and in particular, let M_1 be the set of senior residents, M_2 be the set of junior residents, and M_3 be the set of rotator residents. (Other classes are also possible.) Also, define $|M_t| = m_t \ \forall \ t = 1, ..., T$, where T is the number of groups that the residents are partitioned into. In addition, extraneous (*backup/moonlighter*) residents (borrowed from other departments) may be used (with a high associated penalty.) Let the index $(m + 1)$ represent the *entire collection* of these backup residents, and let $T + 1$ denote this type of residents, with $M_{T+1} \equiv \{m + 1\}$.

It is required that a schedule be prepared for some n nights, where we denote $N = \{1, ..., n\}$ as the set of nights under consideration. A typical value for n is 30 - 40 nights. The set of nights N is partitioned into weekday nights (N_W) and weekend nights (N_E). In addition, let us define A_i to be the set of nights for which resident i is available for work, i.e.,

A_i = {nights $j \in N$: resident i is available to work on night j}. We assume that $A_{m+1} \equiv N$.

Now, define the decision variables x_{ij} as follows:

$$x_{ij} = \begin{cases} 1 \text{ if resident } i \in M \text{ is scheduled to work on night } j \in A_i, \\ 0 \text{ otherwise.} \end{cases} \qquad (1a)$$

Furthermore, for the collective type of resident $T + 1$ (resident index $m + 1$), define an integer variable

$x_{m+1,\,j}$ = number of backup/moonlighter residents that are selected to work on night j, $\forall\, j \in N$. (1b)

3.2 *Constraints*

(C1) As specified above, resident i is required to work for some minimum number of nights n_i , but could be required to work for some extra nights (up to three extra nights). Hence, define

$$s_{iq} = \begin{cases} 1 \text{ if resident } i \text{ works for a } q\text{th extra night,} \\ 0 \text{ otherwise, } \forall\ i\ \in\ M,\ q\ =\ 1,\ 2,\ 3. \end{cases} \tag{2a}$$

A penalty is attached on the use of s_{iq}, namely μ_{iq}, such that:

$$0 < \mu_{i1} < \mu_{i2} < \mu_{i3},\ \forall\ i \in M. \tag{2b}$$

This essentially means that the penalty for scheduling resident i for a second extra night is higher than that for the first extra night, etc. As will be seen below, s_{iq} can be treated as a continuous variable on [0, 1] and will automatically turn out to be integral at optimality. Moreover, the penalty structure (2b) will automatically ensure that $s_{i2} = 1 \Rightarrow s_{i1} = 1$ and that $s_{i3} = 1 \Rightarrow s_{i1} = s_{i2} = 1$. The workload constraint can then be stated as follows.

$$\sum_{j \in A_i} x_{ij} - \sum_{q=1}^{3} s_{iq} = n_i \quad \forall\ i\ \in\ M, \text{ with } 0 \le s_{iq} \le 1\ \forall\ i, q. \tag{3}$$

(C2) A restriction is also imposed on the number of weekend nights that a resident is scheduled for work, i.e., denoting n_{Ei} as a limit on the weekend nights that resident i can be required to work, we have,

$$\sum_{j \in A_i \cap N_E} x_{ij} \le n_{Ei} \qquad \forall\ i \in M. \tag{4}$$

(C3) Different departments have different requirements on the mix of residents required to work each night. Let G_k, $k = 1, \ldots, K$, be groups of residents (possibly overlapping), composed of unions of selected sets M_t such that $\bigcup_k G_k = M \bigcup \{m+1\}$.

The nightly resident composition requirement is then specified via a set of constraints which state that at least g_k residents from group G_k must be scheduled each night, for each $k = 1, ..., K$. This leads to the constraints:

$$\sum_{\substack{i \in G_k \\ \ni j \in A_i}} x_{ij} \geq g_k \quad \forall\ k = 1, ..., K, \text{ for each } j \in N. \tag{5}$$

Example 1. This example presents the nightly composition structure at the *St. John Hospital and Medical Center*, Detroit, Michigan. Suppose that $T = 3$ with M_1, M_2, and M_3 being the senior, junior, and rotator residents, respectively, and as before, let $M_4 \equiv \{m + 1\}$ denote the extraneous residents. Then, we might have:

$$\left.\begin{array}{l} G_1 = M_1 \text{ with } g_1 = 1 \\ G_2 = M_1 \cup M_4 \text{ with } g_2 = 2 \\ G_3 = M_2 \text{ with } g_3 = 1 \\ G_4 = M_2 \cup M_3 \text{with } g_4 = 2. \end{array}\right\} \tag{6}$$

Hence, since the objective function will discourage the use of more residents than required for each night (except perhaps if constraint (C1) enforces otherwise), we would essentially consider (at least) the following sets of residents, one set of which would be working at any given night. These sets are composed from combinations of {either two seniors, or 1 senior and one extraneous resident} (based on groups G_1 and G_2) and {two juniors, or 1 junior and 1 rotator} (based on groups G_3 and G_4).

Set 1: {2 seniors, 2 juniors}

Set 2: {1 senior, 1 extraneous, 2 juniors}

Set 3: {2 seniors, 1 junior, 1 rotator}

Set 4: {1 senior, 1 extraneous, 1 junior, 1 rotator}

Note that if we also want to permit the combinations:

Set 5: {2 seniors, 1 junior}

Set 6: {1 senior, 1 extraneous, 1 junior} we can include a “dummy” resident among the rotator set where this resident is available each night (perhaps at an associated cost).

If the sets of the type constructed in Example 1 are specified by the department and are more general in form than can be represented by the constraints of type (5), we can use an alternative representation of constraints (C3) as described in (C3′) below.

(C3′) Suppose that in lieu of (C3) we are given possible combination groups $k = 1, \ldots, K$, where for combination k, it is required to have $\alpha_{tk} \geq 0$ residents of type t, for $t = 1, \ldots, T + 1$.

To model this, we define the binary decision variables

$$z_{jk} = \begin{cases} 1 \text{ if combination } k \text{ of residents is used for night } j, \\ 0 \text{ otherwise}, \forall\, j \in N,\ k = 1, \ldots, K. \end{cases} \tag{7}$$

Then, we can impose the set of constraints

$$\sum_{k=1}^{K} z_{jk} = 1 \ \forall\, j \in N \tag{8a}$$

$$\sum_{\substack{i \in M_t \\ \ni j \in A_i}} x_{ij} = \sum_{k=1}^{K} \alpha_{tk} z_{jk} \quad \forall\, t = 1, \ldots, T + 1, \forall\, j \in N. \tag{8b}$$

Remark 1. Whenever possible, it is preferable to represent these workforce composition or combination constraints via (C3) rather than (C3′). However, this may not always be possible, as for example, if we are required to have either {2 seniors, 2 juniors} or
{1 senior, 2 juniors, 2 rotators} in Example 1 above.

(C4) Another restriction requires a minimum number of off-nights between any successive work-nights. Thus, for each resident $i \in M_t$ of type t, $t = 1, \ldots, T$, suppose that between any two successive work-nights, there must be at least some h_t off-nights. This yields the constraints

$$\sum_{\substack{\rho=j \\ \ni \rho \in A_i}}^{j+h_t} x_{i\rho} \leq 1 \qquad \text{for each } j = 1, ..., n\text{-}h_t, \ \forall\ i \in M_t,\ t = 1,..., T. \tag{9}$$

(Note that we assume $h_t \leq n$-1 without loss of generality, else, the resident would be permitted to work at most one night over the horizon.)

Remark 2. We can soften constraint (9) by permitting the slack therein to be possibly negative, but then penalizing such a violation. Hence, we can write (9) alternatively as

$$\sum_{\substack{\rho=j \\ \ni \rho \in A_i}}^{j+h_t} x_{i\rho} + s_{ij}^{+} - s_{ij}^{-} = 1 \quad \forall\ j = 1, ..., n\text{-}h_t,\ i \in M_t,\ t = 1, ..., T, \tag{10}$$

and incorporate a penalty term in the objective function of the following type, where $\mu_i^h > 0$, $i \in M$, is an associated penalty parameter.

$$\sum_{t=1}^{T} \sum_{i \in M_t} \mu_i^h \left[\sum_{j=1}^{n-h_t} s_{ij}^{-} \right]. \tag{11}$$

Note that (11) aggregates the penalized violations in (9) over all time-windows of width 1 + h_t for each $i \in M_t$, t = 1, ..., T. In particular, the lesser the gap between consecutive work nights, the greater the penalty. For example, suppose that n = 7 and h_t = 3 for any resident $i \in M$. Consider the table below where x_{ij} = 1 if the resident i is scheduled on the jth night (j = 1, …, 7).

$x_{i3} = x_{i5} = 1$	—	—	1	—	1	—	—
$x_{i3} = x_{i4} = 1$	—	—	1	1	—	—	—
Nights (j)	1	2	3	4	5	6	7

For $x_{i3} = x_{i5} = 1$, we incur a penalty of $2\mu_i^h$ in (11) since $s_{i2}^{-} = s_{i3}^{-} = 1$. However, if $x_{i3} = x_{i4} = 1$, then we would have $s_{i1}^{-} = s_{i2}^{-} = s_{i3}^{-} = 1$, resulting in a penalty of $3\mu_i^h$ in (11). This can be seen clearly from expanding the corresponding constraints in (10) which state the following:

$$x_{i1} + x_{i2} + x_{i3} + x_{i4} + s_{i1}^{+} - s_{i1}^{-} = 1 \qquad (j = 1)$$

$$x_{i2} + x_{i3} + x_{i4} + x_{i5} + s_{i2}^{+} - s_{i2}^{-} = 1 \qquad (j = 2)$$

$$x_{i3} + x_{i4} + x_{i5} + x_{i6} + s_{i3}^{+} - s_{i3}^{-} = 1 \qquad (j = 3)$$

$$x_{i4} + x_{i5} + x_{i6} + x_{i7} + s_{i4}^{+} - s_{i4}^{-} = 1 \qquad (j = 4)$$

Also, the term $\sum_{j=1}^{n-h_t} s_{ij}^{-}$ in this example expands to the expression $s_{i1}^{-} + s_{i2}^{-} + s_{i3}^{-} + s_{i4}^{-}$.

Now, for the case $x_{i3} = x_{i5} = 1$, this will yield $s_{i2}^{-} = 1$ and $s_{i3}^{-} = 1$, to satisfy the above constraints, specifically for $j = 2$ and $j = 3$. This will incur a penalty of $2\mu_i^h$ in (11). Similarly, $x_{i3} = x_{i4} = 1$ will yield $s_{i1}^{-} = 1$, $s_{i2}^{-} = 1$, and $s_{i3}^{-} = 1$ from the constraints for $j = 1$, $j = 2$, and $j = 3$, respectively. This will incur a penalty of $3\mu_i^h$ in (11).

3.3 *Objective Function*

The following terms comprise the objective function.

(a) Each resident $i \in M$ specifies a cost factor c_{ij} between 1 and 5, indicating his/her desirability to work on each night $j \in A_i$, where $c_{ij} = 1$ if this assignment is most desirable, and $c_{ij} = 5$ if it is most undesirable, though feasible. Furthermore, we can assign priority factors ρ_t for each type of resident $t \in \{1, ..., T\}$, where a relatively higher value of ρ_t reflects a higher (seniority-based) priority for accommodating indicated desirabilities. This gives the following term in the objective function:

$$\sum_{t=1}^{T} \rho_t \sum_{i \in M_t} \sum_{j \in A_i} c_{ij} x_{ij} . \qquad (12)$$

(b) Commensurate with (12), let c_{m+1} be a cost factor for each extraneous resident used over the horizon N. This yields the term:

$$c_{m+1} \sum_{j \in N} x_{m+1,j.} \qquad (13)$$

(c) Based on constraint (C1), there exists a penalty for making each resident work some extra night. This gives the term:

$$\sum_{i\in M}\sum_{q=1}^{3}\mu_{iq}s_{iq}. \tag{14}$$

(d) Finally, we include the penalty term (11) based on constraint (C4) (if the softened constraints (10) are used in lieu of (9); else, we can simply assume that $\mu_i^h = \infty$ so that $s_{ij}^{-} \equiv 0$ $\forall$ i, j at optimality).

Note that all these penalty parameters must be made commensurate with each other. Putting the objective terms (a)-(d) together, along with the constraints (C1)-(C4), we obtain the following model for the **Resident Scheduling Problem (RSP).**

RSP: Minimize:

$$\sum_{t=1}^{T}\rho_t\sum_{i\in M_t}\sum_{j\in A_i}c_{ij}x_{ij}+c_{m+1}\sum_{j\in N}x_{m+1,j}+\sum_{i\in M}\sum_{q=1}^{3}\mu_{iq}s_{iq}+\sum_{t=1}^{T}\sum_{i\in M_t}\mu_i^h\left[\sum_{j=1}^{n-h_t}s_{ij}^{-}\right] \tag{15a}$$

subject to:

$$\sum_{j\in A_i}x_{ij}-\sum_{q=1}^{3}s_{iq}=n_i \qquad \forall\ i\in M \tag{15b}$$

$$\sum_{j\in A_i\cap N_E}x_{ij}\le n_{Ei} \qquad \forall\ i\in M \tag{15c}$$

$$\sum_{\substack{i\in G_k\\ \ni\, j\in A_i}}x_{ij}\ge g_k \qquad \forall\ k=1,\ldots,K, j\in N \tag{15d}$$

$$\sum_{\substack{\rho=j\\ \ni\,\rho\in A_i}}^{j+h_t}x_{i\rho}+s_{ij}^{+}-s_{ij}^{-}=1 \qquad \forall\ j=1,\ldots,n\text{-}h_t,\ i\in M_t,\ t=1,\ldots,T \tag{15e}$$

x_{ij} binary, for $i\in M, j\in A_i$, $x_{m+1,j}$ integer $\forall\ j\in N$, $0\le s_{iq}\le 1\ \forall\ i\in M$, $q = 1, 2, 3$,

$s_{ij}^{\pm}\ge 0\ \ \forall\ i\in M_t$, $t = 1, \ldots, T$, $j = 1, \ldots, n\text{-}h_t$. (15f)

Note that if (C3′) is used in lieu of (C3), then constraints (15d) would get replaced by (8a, b), where z_{jk} are binary variables.

Let us now consider Example 1 with the following scenario:

Let $m = 8$, $M_1 = \{1, 2, 3\}$, $M_2 = \{4, 5, 6\}$, and $M_3 = \{7, 8\}$. The index $m + 1 = 9$ will represent the backup/moonlighter residents, as before, so that $M_4 = \{9\}$.

Let $n = 4$ and $N_W = \{1, 2\}$ and $N_E = \{3, 4\}$.

Also, we have the following set of available nights for each resident i:

$A_1 = \{1, 2, 3, 4\}$,

$A_2 = \{1, 2, 3\}$,

$A_3 = \{1, 2, 4\}$,

$A_4 = \{1, 3, 4\}$,

$A_5 = \{1, 2, 3\}$,

$A_6 = \{2, 4\}$,

$A_7 = \{1, 2, 3\}$, and

$A_8 = \{1, 4\}$.

Define also n_i and n_{Ei} for each resident i as follows:

$n_1 = 4$, and $n_{E1} = 2$,

$n_2 = 3$, and $n_{E2} = 1$,

$n_3 = 2$, and $n_{E3} = 2$,

$n_4 = 2$, and $n_{E4} = 1$,

$n_5 = 2$, and $n_{E5} = 1$,

$n_6 = 2$, and $n_{E6} = 2$,

$n_7 = 2$, and $n_{E7} = 2$, and

$n_8 = 1$, and $n_{E8} = 1$.

Let $h_t = 1 \; \forall \; t = 1, 2, 3$.

Let us also assume the following c_{ij} values for the various residents:

Group	Resident i	$j = 1$	$j = 2$	$j = 3$	$j = 4$
M_1	1	1	2	3	5
	2	5	2	4	—
	3	1	3	—	4
M_2	4	1	—	3	5
	5	1	4	2	—
	6	—	5	—	4
M_3	7	2	5	3	—
	8	2	—	—	1
M_4	9	50	50	50	50

In addition, let the priority parameters ρ_t be as follows:

$\rho_1 = 5$, $\rho_2 = 3$, and $\rho_3 = 1$.

Also, set the penalty parameters $\mu_{i1} = 10, \mu_{i2} = 20,$ and $\mu_{i3} = 30$. Finally, put $\mu_i^h = 40$ for all $i \in M$.

The problem formulation and solution are given below:

Minimize $z = 5x_{11} + 10x_{12} + 15x_{13} + 25x_{14} + 25x_{21} + 10x_{22} + 20x_{23} + 5x_{31} + 15x_{32} + 20x_{34} + 3x_{41} + 9x_{43} + 15x_{44} + 3x_{51} + 12x_{52} + 6x_{53} + 15x_{62} + 12x_{64} + 2x_{71} + 5x_{72} + 3x_{73} + 2x_{81} + x_{84} + 10s_{11} +20s_{12} +30s_{13} + 10s_{21} + 20s_{22} + 30s_{23} + 10s_{31} + 20s_{32} + 30s_{33} + 10s_{41} + 20s_{42} + 30s_{43} + 10s_{51} + 20s_{52} + 30s_{53} + 10s_{61} + 20s_{62} + 30s_{63} + 10s_{71} + 20s_{72} + 30s_{73} + 10s_{81} + 20s_{82} + 30s_{83} + 50x_{91} + 50x_{92} + 50x_{93} + 50x_{94} + 40s^-_{11} + 40s^-_{12} + 40s^-_{13} + 40s^-_{21} + 40s^-_{22} + 40s^-_{23} + 40s^-_{31} + 40s^-_{32} + 40s^-_{33} + 40s^-_{41} + 40s^-_{42} + 40s^-_{43} + 40s^-_{51} + 40s^-_{52} + 40s^-_{53} + 40s^-_{61} + 40s^-_{62} + 40s^-_{63} + 40s^-_{71} + 40s^-_{72} + 40s^-_{73} + 40s^-_{81} + 40s^-_{82} + 40s^-_{83}$

subject to

$x_{11} + x_{12} + x_{13} + x_{14} - s_{11} - s_{12} - s_{13} = 4$

$x_{21} + x_{22} + x_{23} - s_{21} - s_{22} - s_{23} = 3$

$x_{31} + x_{32} + x_{34} - s_{31} - s_{32} - s_{33} = 2$

$x_{41} + x_{43} + x_{44} - s_{41} - s_{42} - s_{43} = 2$

$x_{51} + x_{52} + x_{53} - s_{51} - s_{52} - s_{53} = 2$

$x_{62} + x_{64} - s_{61} - s_{62} - s_{63} = 2$

$x_{71} + x_{72} + x_{73} - s_{71} - s_{72} - s_{73} = 2$

$x_{81} + x_{84} - s_{81} - s_{82} - s_{83} = 1$

$x_{13} + x_{14} \leq 2$

$x_{23} \leq 1$

$x_{34} \leq 2$

$x_{43} + x_{44} \leq 1$

$x_{53} \leq 1$

$x_{64} \leq 2$

$x_{73} \leq 2$

$x_{84} \leq 1$

$x_{11} + x_{21} + x_{31} \geq 1$

$x_{12} + x_{22} + x_{32} \geq 1$

$x_{13} + x_{23} \geq 1$

$x_{14} + x_{34} \geq 1$

$x_{11} + x_{21} + x_{31} + x_{91} \geq 2$

$x_{12} + x_{22} + x_{32} + x_{92} \geq 2$

$x_{13} + x_{23} + x_{93} \geq 2$

$x_{14} + x_{34} + x_{94} \geq 2$

$x_{41} + x_{51} \geq 1$

$x_{52} + x_{62} \geq 1$

$x_{43} + x_{53} \geq 1$

$x_{44} + x_{64} \geq 1$

$x_{41} + x_{51} + x_{71} + x_{81} \geq 2$

$x_{52} + x_{62} + x_{72} \geq 2$

$x_{43} + x_{53} + x_{73} \geq 2$

$x_{44} + x_{64} + x_{84} \geq 2$

$x_{11} + x_{12} + s^{+}_{11} - s^{-}_{11} = 1$

$x_{12} + x_{13} + s^{+}_{12} - s^{-}_{12} = 1$

$x_{13} + x_{14} + s^{+}_{13} - s^{-}_{13} = 1$

$x_{21} + x_{22} + s^{+}_{21} - s^{-}_{21} = 1$

$x_{22} + x_{23} + s^{+}_{22} - s^{-}_{22} = 1$

$x_{23} + s^{+}_{23} - s^{-}_{23} = 1$

$x_{31} + x_{32} + s^{+}_{31} - s^{-}_{31} = 1$

$x_{32} + s^{+}_{32} - s^{-}_{32} = 1$

$x_{34} + s^{+}_{33} - s^{-}_{33} = 1$

$x_{41} + s^{+}_{41} - s^{-}_{41} = 1$

$x_{43} + s^{+}_{42} - s^{-}_{42} = 1$

$x_{43} + x_{44} + s^{+}_{43} - s^{-}_{43} = 1$

$x_{51} + x_{52} + s^{+}_{51} - s^{-}_{51} = 1$

$x_{52} + x_{53} + s^{+}_{52} - s^{-}_{52} = 1$

$x_{53} + s^{+}_{53} - s^{-}_{53} = 1$

$x_{62} + s^{+}_{61} - s^{-}_{61} = 1$

$x_{62} + s^{+}_{62} - s^{-}_{62} = 1$

$x_{64} + s^{+}_{63} - s^{-}_{63} = 1$

$x_{71} + x_{72} + s^{+}_{71} - s^{-}_{71} = 1$

$x_{72} + x_{73} + s^{+}_{72} - s^{-}_{72} = 1$

$x_{73} + s^{+}_{73} - s^{-}_{73} = 1$

$x_{81} + s^{+}_{81} - s^{-}_{81} = 1$

$s^{+}_{82} - s^{-}_{82} = 1$

$x_{84} + s^{+}_{83} - s^{-}_{83} = 1$

Bounds

x_{11}, x_{12}, x_{13}, x_{14}, x_{21}, x_{22}, x_{23}, x_{31}, x_{32}, x_{34}, x_{41}, x_{43}, x_{44}, x_{51}, x_{52}, x_{53}, x_{62}, x_{64}, x_{71}, x_{72}, x_{73}, x_{81}, x_{84} are binary and ≥ 0,

$0 \leq s_{11} \leq 1$

$0 \leq s_{12} \leq 1$

$0 \leq s_{13} \leq 1$

$0 \leq s_{21} \leq 1$

$0 \leq s_{22} \leq 1$

$0 \leq s_{23} \leq 1$

$0 \leq s_{31} \leq 1$

$0 \leq s_{32} \leq 1$

$0 \leq s_{33} \leq 1$

$0 \leq s_{41} \leq 1$

$0 \leq s_{42} \leq 1$

$0 \leq s_{43} \leq 1$

$0 \leq s_{51} \leq 1$

$0 \leq s_{52} \leq 1$

$0 \leq s_{53} \leq 1$

$0 \leq s_{61} \leq 1$

$0 \leq s_{62} \leq 1$

$0 \leq s_{63} \leq 1$

$0 \leq s_{71} \leq 1$

$0 \leq s_{72} \leq 1$

$0 \leq s_{73} \leq 1$

$0 \leq s_{81} \leq 1$

$0 \leq s_{82} \leq 1$

$0 \leq s_{83} \leq 1$

x_{91}, x_{92}, x_{93}, x_{94} are integers, and ≥ 0,

$s^{\pm}_{11}$, $s^{\pm}_{12}$, $s^{\pm}_{13}$, $s^{\pm}_{21}$, $s^{\pm}_{22}$, $s^{\pm}_{23}$, $s^{\pm}_{31}$, $s^{\pm}_{32}$, $s^{\pm}_{33}$, $s^{\pm}_{41}$, $s^{\pm}_{42}$, $s^{\pm}_{43}$, $s^{\pm}_{51}$, $s^{\pm}_{52}$, $s^{\pm}_{53}$, $s^{\pm}_{61}$, $s^{\pm}_{62}$, $s^{\pm}_{63}$, $s^{\pm}_{71}$, $s^{\pm}_{72}$, $s^{\pm}_{73}$, $s^{\pm}_{81}$, $s^{\pm}_{82}$, $s^{\pm}_{83} \geq 0$.

Solution:

$s^{-}_{11} = 1$

$s^{-}_{12} = 1$

$s^{-}_{13} = 1$

$s^{-}_{21} = 1$

$s^{-}_{22} = 1$

$s^{-}_{71} = 1$

$x_{11} = 1$

$x_{12} = 1$

$x_{13} = 1$

$x_{14} = 1$

$x_{21} = 1$

$x_{22} = 1$

$x_{23} = 1$

$x_{31} = 1$

$x_{34} = 1$

$x_{41} = 1$

$x_{43} = 1$

$x_{51} = 1$

$x_{53} = 1$

$x_{62} = 1$

$x_{64} = 1$

$x_{71} = 1$

$x_{72} = 1$

$x_{84} = 1$

$s^{+}_{32} = 1$

$s^{+}_{73} = 1$

$s^{+}_{81} = 1$

$s^{+}_{82} = 1$

All other variables are zero.

A summary of the scheduled assignments is given below.

Group	Resident i	j = 1	j = 2	j = 3	j = 4
M_1	1	▨ 1	▨ 2	▨ 3	▨ 5
	2	▨ 5	▨ 2	▨ 4	—
	3	▨ 1	3	—	▨ 4
M_2	4	▨ 1	—	▨ 3	5
	5	▨ 1	4	▨ 2	—
	6	—	▨ 5	—	▨ 4
M_3	7	▨ 2	▨ 5	3	—
	8	2	—	—	▨ 1
M_4	9	50	50	50	50

▨ : Assigned Shift

Chapter 4

Analysis of the Structure of the Resident Scheduling Problem (RSP)

The Resident Scheduling Problem (RSP) has a predominant network structure that we can exploit in devising exact or heuristic solution procedures. To see this, let us rewrite Problem RSP as follows, by introducing the following dependent variables.

$X_{Ei} \equiv \sum_{j \in A_i \cap N_E} x_{ij} =$ number of weekend nights assigned to resident i, $\forall\ i \in M$. (16a)

$X_{Wi} \equiv \sum_{j \in A_i \cap N_W} x_{ij} =$ number of weekday nights assigned to resident i, $\forall\ i \in M$. (16b)

$y_{jt} \equiv \sum_{\substack{i \in M_t \\ \ni j \in A_i}} x_{ij} =$ number of residents of type t that have been assigned to work on night j,

for $t = 1, ..., T+1,\ j \in N$. (16c)

RSP: Minimize

$$\sum_{t=1}^{T} \rho_t \sum_{i \in M_t} \sum_{j \in A_i} c_{ij} x_{ij} + c_{m+1} \sum_{j \in N} x_{m+1,j} + \sum_{i \in M} \sum_{q=1}^{3} \mu_{iq} s_{iq} + \sum_{t=1}^{T} \sum_{i \in M_t} \mu_i^h \left[\sum_{j=1}^{n-h_t} s_{ij}^- \right] \tag{17a}$$

subject to

$$X_{Ei} + X_{Wi} - \sum_{q=1}^{3} s_{iq} = n_i \qquad \forall i \in M \qquad (17b)$$

$$-X_{Ei} + \sum_{j\in A_i \cap N_E} x_{ij} = 0 \qquad \forall i \in M \qquad (17c)$$

$$-X_{Wi} + \sum_{j\in A_i \cap N_W} x_{ij} = 0 \qquad \forall i \in M \qquad (17d)$$

$$-\sum_{\substack{i\in M_t \\ \ni j\in A_i}} x_{ij} + y_{jt} = 0 \qquad \forall j \in N, \quad t = 1, \ldots, T+1 \quad (17e)$$

$$-\sum_{t: M_t \subseteq G_k} y_{jt} \le -g_k \qquad \forall k = 1, \ldots, K, j \in N \qquad (17f)$$

$$\sum_{\substack{\rho\ =j \\ \ni\ \rho\ \in\ A_i}}^{j+h_t} x_{i\rho} + s_{ij}^{+} - s_{ij}^{-} = 1 \ \forall\, j = 1, \ldots, n\text{-}h_t,\ i \in M_t,\ t = 1, \ldots, T \qquad (17g)$$

$0 \le X_{Ei} \le n_{Ei} \quad \forall\, i \in M,\ X_{Wi} \ge 0 \ \ \forall\, i \in M,$

$0 \le x_{ij} \le 1 \quad \forall\, i \in M, j \in A_i,\ \ x_{m+1,\,j} \ge 0 \ \forall\, j \in N,$

$0 \le s_{iq} \le 1 \quad \forall\, i \in M, q = 1, 2, 3,\ \ s_{ij}^{\pm} \ge 0 \ \forall\, i \in M_t, t = 1,\ldots,T, j = 1, \ldots, n\text{-}h_t$ (17h)

x_{ij} binary, for $i \in M, j \in A_i$, and $x_{m+1,\,j}$ integer $\forall\, j \in N$. (17i)

Observe that in order to reveal a network substructure, we have split constraints (15b) into (17b, c, d), we have accommodated (15c) into (17h) as simple upper bounding restrictions, and we have split (15d) into (17e, f). Furthermore, if we use the restrictions (C3′) in lieu of (C3) (as given by Equations (8a, b)), then we would replace (17f) by the constraints

$$\sum_{k=1}^{K} z_{jk} = 1 \ \forall\, j \in N \qquad (17f\,')$$

$$\sum_{k=1}^{K} \alpha_{tk} z_{jk} - y_{jt} = 0 \ \ \forall\, t = 1, \ldots, T+1, \forall\, j \in N, \qquad (17f\,'')$$

where $z_{jk}\ \forall\, j, k$ are restricted to be binary valued (within (17i)).

Now, consider the following result.

Theorem 1. Define a relaxed version **RRSP** of Problem RSP by omitting constraints (17g) and (17i). Then, under any of the following scenarios S1, S2, or S3, the problem RRSP is a bounded variables linear network flow programming problem for which at an extreme point optimum, (17i) will automatically hold true.

(S1). G_k, k = 1, …, K, are disjoint sets.

(S2). The sets G_k are as specified by (6), for k = 1, ..., $K \equiv 4$, and with T = 3 as in Example 1.

(S3). Constraints $(17f',17f'')$ are used in lieu of (17f), but the nightly combination selections as given by the variables z_{jk} $\forall$ k = 1, ..., K, $j \in N$, are prespecified.

Proof. Observe that for problem RRSP, each of the columns of the variables X_{Ei} $\forall$ i, X_{Wi} $\forall$ i, and x_{ij} $\forall$ $i \in M$, $j \in A_i$ have exactly two nonzeros, and these nonzeros are a +1 and a -1. Furthermore, the columns of the variables s_{iq} $\forall$ $i \in M$, q = 1, 2, 3 have exactly one nonzero which is a -1. Hence, by Bazaraa, Jarvis and Sherali (1990), for example, in order to establish the required result, we need to exhibit the same network structure with respect to the y-variables in the problem. Note that under scenario (S1), each variable y_{jt} has a +1 in the corresponding constraint in (17e) and a -1 for that constraint in (17f) which corresponds to night j and group k for which $t \in G_k$. Since each t belongs to no more than one group by scenario (S1), each y-variable conforms to a network structure. Similarly, by $(17f', \ 17f'')$, since the y-variables are essentially fixed under scenario (S3), this case also possesses a network structure.

Finally, let us consider scenario (S2). Noting (6), in this case, constraints (17f) are given as follows

$$-y_{j1} \leq -g_1, \ \ -y_{j1} - y_{j4} \leq -g_2, \ \ -y_{j2} \leq -g_3, \ \ -y_{j2} - y_{j3} \leq -g_4, \ \ \forall \, j \in N. \tag{18}$$

By accommodating the restrictions $y_{j1} \geq g_1$ and $y_{j2} \geq g_3$ $\forall$ $j \in N$ within the variable bounding constraints (17h), and retaining the other two constraints in (18), for each $j \in N$, as structural constraints within (17f), we observe that each y_{jt} variable has a +1 and a -1 in its corresponding constraint in (17e) and (17f), respectively, and zero elsewhere within the structural constraints. Hence, we again deduce a network structure for Problem RRSP, and this completes the proof. ■

As evident from the proof of Theorem 1, and the structure of (17), the embedded network structure in Problem RRSP appears as follows.

Nodes:

Construct the following nodes:

(a) Node *i* to represent each corresponding constraint in (17b)

(b) Node *Ei* to represent each corresponding constraint in (17c)

(c) Node *Wi* to represent each corresponding constraint in (17d)

(d) Node (*j*, *t*) to represent each corresponding constraint in (17e)

(e) Node (*j*, *k*) to represent each corresponding constraint in (17f).

Arcs:

Construct the following arcs. (In each of the following illustrations, the quantities shown against each arc are {cost, [lower bound, upper bound], variable}.

(a) Surplus arcs s_{iq}, $q = 1, 2, 3$, coming into each node $i \in M$, having a flow bounded on [0, 1] and a cost of μ_{iq} (see Figure 1).

(b) Arcs X_{Ei} from node *i* to node *Ei*, for each $i \in M$, having a flow bounded on [0, n_{Ei}] and a cost of 0 (see Figure 1).

(c) Arcs X_{Wi} from node *i* to node *Wi*, for each node $i \in M$, having a flow bounded on [0, ∞] (or equivalently, [0, $n_i + 3$] due to (17b) and (17h)), and a cost of 0 (see Figure 1).

(d) Arcs x_{ij} for each $i \in M, j \in A_i$, starting at node Ei if $j \in N_E$ or at node Wi if $j \in N_W$, and ending at node (j, t) where t is such that $i \in M_t$. This is bounded on the interval [0, 1] and has a cost of c_{ij}, $\forall\ i \in M, j \in A_i$ (see Figure 2).

(e) Surplus arcs $x_{m+1,\, j}$ for each $j \in N$ coming into the corresponding node (j, T+1), having an interval bound of [0, ∞] and a cost of c_{m+1} (see Figure 2).

(f) Arcs y_{jt} $\forall\ j \in N$, $t = 1, \ldots, T + 1$, starting at the corresponding node (j, t), and terminating as follows, based on the scenarios (S1) and (S2) of Theorem 1. (For the case of scenario (S3), this is a fixed quantity, given the binary z-value variables.)

Case of (S1): The termination node for arc y_{jt} is the corresponding node (j, k) where k is such that $M_t \subseteq G_k$ (see Figure 3). Each such y_{jt} is bounded on [0, ∞] and has a cost of 0.

Case of (S2): Examining (18), the termination node for each y_{j1} and y_{j4} is the corresponding node (j, k = 2), and that for y_{j2} and y_{j3} is the corresponding node (j, k = 4) for each $j \in N$ (see Figure 4). Furthermore, y_{j1} is bounded on [g_1, ∞], y_{j2} on [g_3, ∞], and y_{j3} and y_{j4} on [0, ∞], with all associated costs being zero.

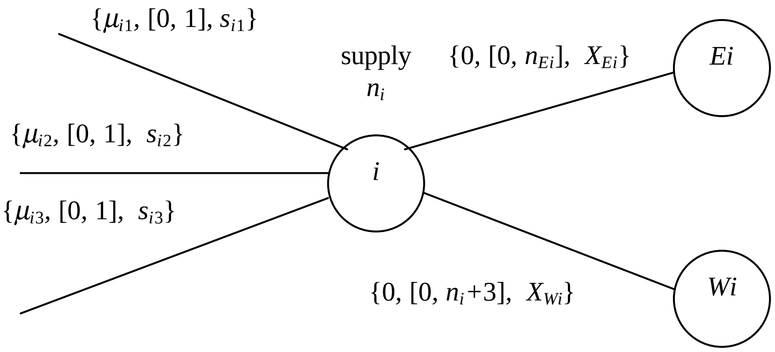

{μ_{i1}, [0, 1], s_{i1}}
supply
n_i
{0, [0, n_{Ei}], X_{Ei}}
Ei
{μ_{i2}, [0, 1], s_{i2}}
i
{μ_{i3}, [0, 1], s_{i3}}
{0, [0, n_i+3], X_{Wi}}
Wi

Figure 1. Conservation of Flow for Each Node $i \in M$.

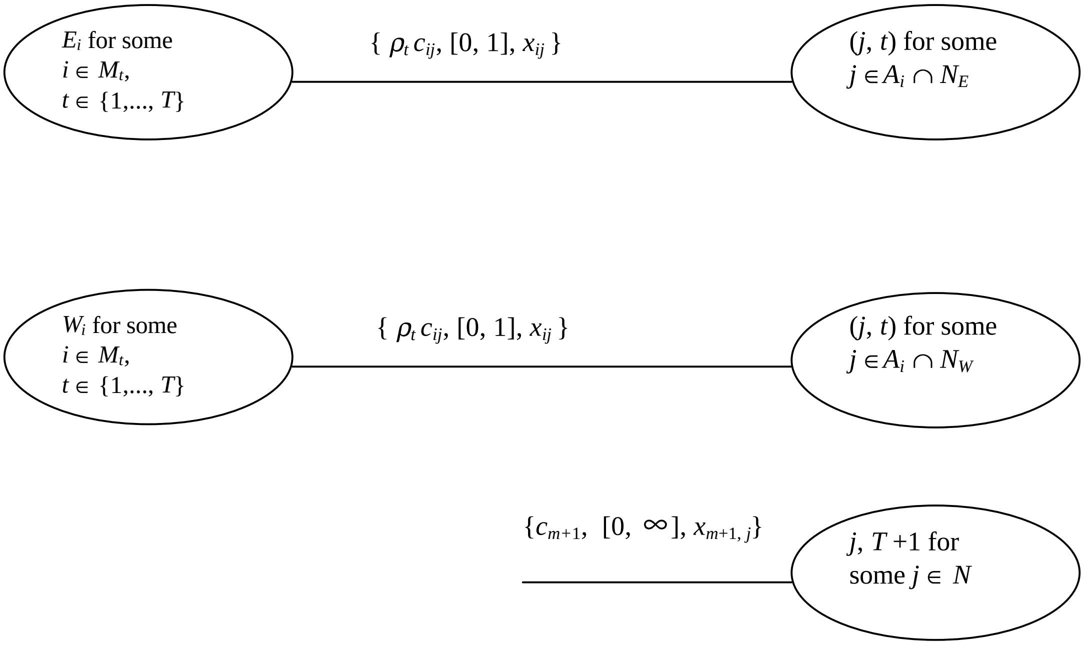


Figure 2. Illustration of Arcs x_{ij} and $x_{m+1,j}$.

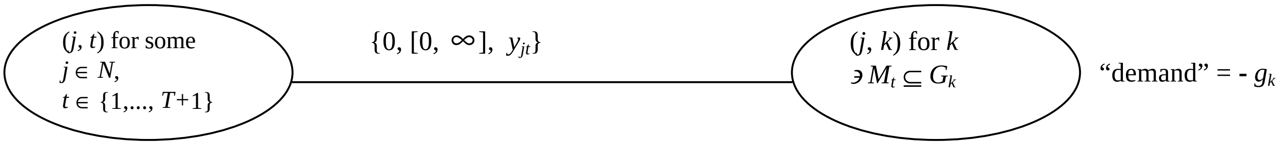


Figure 3. Illustration of Arcs y_{jt} under Scenario (S1).

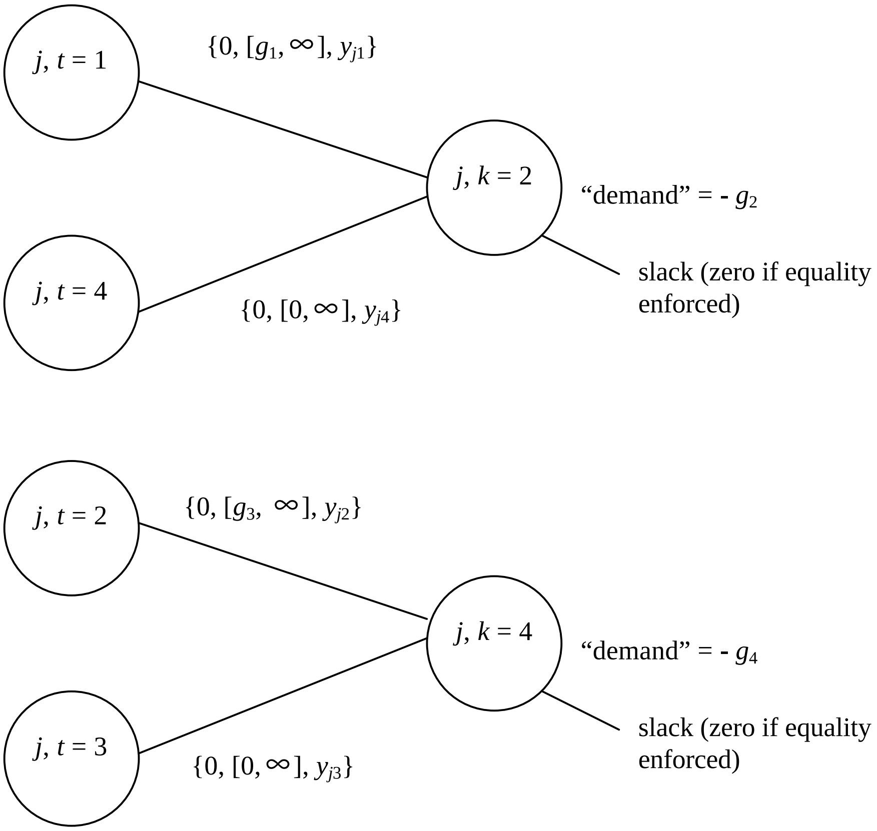
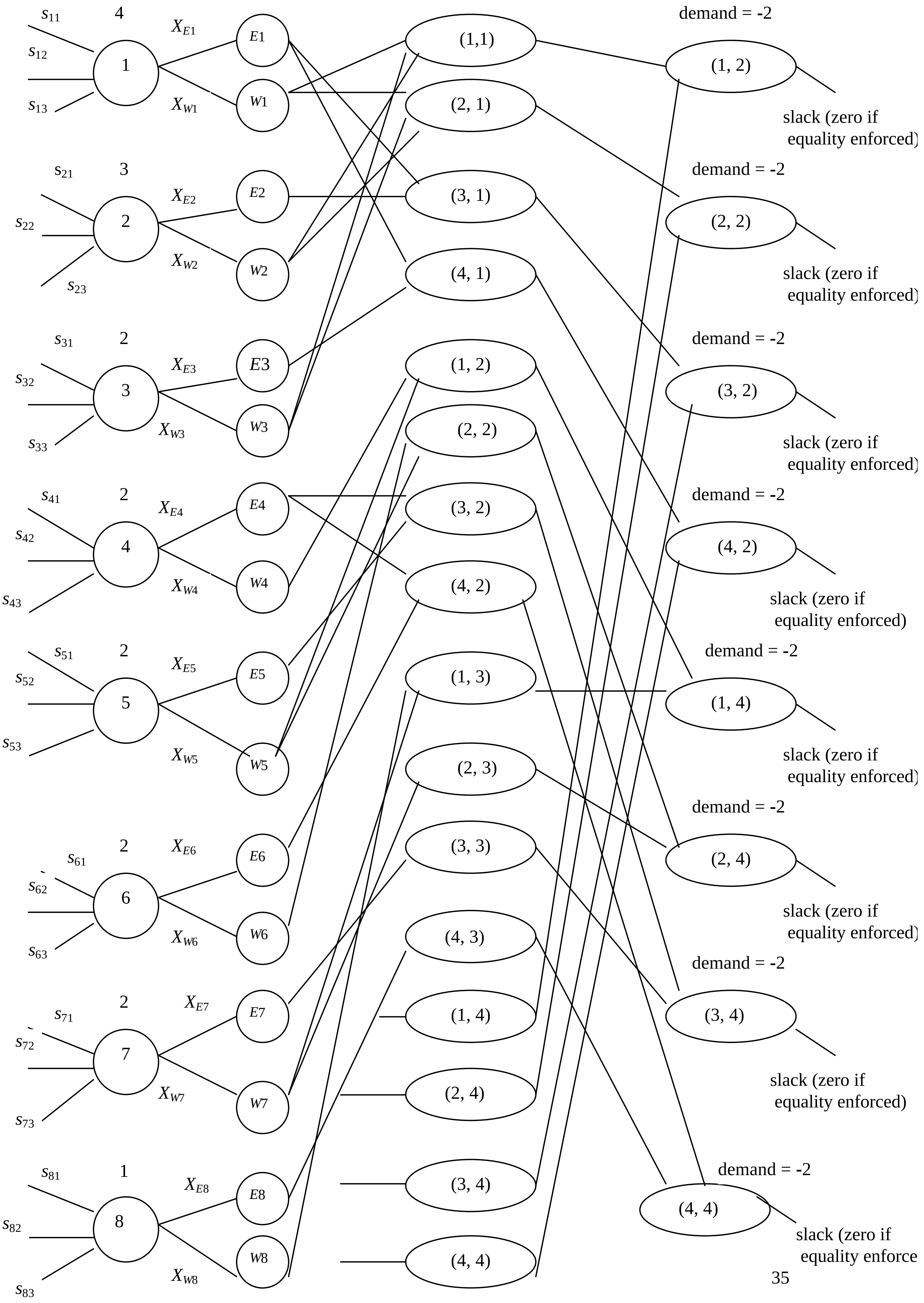

$j, t = 1$
$\{0, [g_1, \infty], y_{j1}\}$
$j, k = 2$
“demand” = - g_2
slack (zero if equality enforced)
$j, t = 4$
$\{0, [0, \infty], y_{j4}\}$
$j, t = 2$
$\{0, [g_3, \infty], y_{j2}\}$
$j, k = 4$
“demand” = - g_4
slack (zero if equality enforced)
$j, t = 3$
$\{0, [0, \infty], y_{j3}\}$

Figure 4. Illustration of Arcs y_{jt} under Scenario (S2).

Hence, in the case of Example 1, where exactly one of the six sets is required, Figure 4 yields the required network structure where the slack variables shown are fixed at zero (eliminated). The union of Figures 1, 2, and 4 represent constraints (15b, c, d, f) for the formulation (15) of this problem. The complicating side constraints (15e), along with the corresponding penalty on s_{ij}^{-} in (15a), must be handled separately. These constraints examine each $i \in M$, and for the collection of arcs emanating jointly from the nodes *Ei* and *Wi*, require the stated gaps between any successive night’s arcs. The following graphs and tables represent Example 1 as a network structured problem as described above.


s_{11}
4
X_{E1}
$E1$
(1,1)
demand = -2
s_{12}
1
(1, 2)
s_{13}
X_{W1}
$W1$
(2, 1)
slack (zero if equality enforced)
s_{21}
3
X_{E2}
$E2$
(3, 1)
demand = -2
s_{22}
2
(2, 2)
X_{W2}
$W2$
(4, 1)
s_{23}
slack (zero if equality enforced)
s_{31}
2
demand = -2
X_{E3}
$E3$
(1, 2)
s_{32}
3
(3, 2)
X_{W3}
$W3$
(2, 2)
s_{33}
slack (zero if equality enforced)
s_{41}
2
X_{E4}
$E4$
demand = -2
(3, 2)
s_{42}
4
(4, 2)
X_{W4}
$W4$
(4, 2)
s_{43}
slack (zero if equality enforced)
s_{51}
2
X_{E5}
$E5$
demand = -2
(1, 3)
s_{52}
5
(1, 4)
s_{53}
X_{W5}
$W5$
(2, 3)
slack (zero if equality enforced)
demand = -2
s_{61}
2
X_{E6}
$E6$
(3, 3)
(2, 4)
s_{62}
6
X_{W6}
$W6$
(4, 3)
s_{63}
slack (zero if equality enforced)
demand = -2
s_{71}
2
X_{E7}
$E7$
(1, 4)
(3, 4)
s_{72}
7
X_{W7}
$W7$
(2, 4)
s_{73}
slack (zero if equality enforced)
s_{81}
1
X_{E8}
$E8$
(3, 4)
demand = -2
s_{82}
8
(4, 4)
slack (zero if equality enforce
X_{W8}
$W8$
(4, 4)
s_{83}

The following tables specify the properties attached to each arc as shown in the network displayed above.

Arc	{cost = μ_{ij}, [LB, UB], s_{ij}}
s_{11}	10, [0, 1], s_{11}
s_{12}	20, [0, 1], s_{12}
s_{13}	30, [0, 1], s_{13}
s_{21}	10, [0, 1], s_{21}
s_{22}	20, [0, 1], s_{22}
s_{23}	30, [0, 1], s_{23}
s_{31}	10, [0, 1], s_{31}
s_{32}	20, [0, 1], s_{32}
s_{33}	30, [0, 1], s_{33}
s_{41}	10, [0, 1], s_{41}
s_{42}	20, [0, 1], s_{42}
s_{43}	30, [0, 1], s_{43}
s_{51}	10, [0, 1], s_{51}
s_{52}	20, [0, 1], s_{52}
s_{53}	30, [0, 1], s_{53}
s_{61}	10, [0, 1], s_{61}
s_{62}	20, [0, 1], s_{62}
s_{63}	30, [0, 1], s_{63}
s_{71}	10, [0, 1], s_{71}
s_{72}	20, [0, 1], s_{72}
s_{73}	30, [0, 1], s_{73}
s_{81}	10, [0, 1], s_{81}
s_{82}	20, [0, 1], s_{82}
s_{83}	30, [0, 1], s_{83}

Arc	{cost, [LB, UB], X_{Ei} or X_{Wi}}
X_{E1}	0, [0, 2], X_{E1}
X_{W1}	0, [0, 7], X_{W1}
X_{E2}	0, [0, 1], X_{E2}
X_{W2}	0, [0, 6], X_{W2}
X_{E3}	0, [0, 2], X_{E3}
X_{W3}	0, [0, 5], X_{W3}
X_{E4}	0, [0, 1], X_{E4}
X_{W4}	0, [0, 5], X_{W4}
X_{E5}	0, [0, 1], X_{E5}
X_{W5}	0, [0, 5], X_{W5}
X_{E6}	0, [0, 2], X_{E6}
X_{W6}	0, [0, 5], X_{W6}
X_{E7}	0, [0, 2], X_{E7}
X_{W7}	0, [0, 5], X_{W7}
X_{E8}	0, [0, 1], X_{E8}
X_{W8}	0, [0, 4], X_{W8}

Arc	{cost, [LB, UB], x_{ij}}
*E*1, (3, 1)	15, [0, 1], x_{13}
*E*1, (4, 1)	25, [0, 1], x_{14}
*W*1, (1, 1)	5, [0, 1], x_{11}
*W*1, (2, 1)	10, [0, 1], x_{12}
*E*2, (3, 1)	20, [0, 1], x_{23}
*W*2, (1, 1)	25, [0, 1], x_{21}
*W*2, (2, 1)	10, [0, 1], x_{22}
*E*3, (4, 1)	20, [0, 1], x_{34}
*W*3, (1, 1)	5, [0, 1], x_{31}
*W*3, (2, 1)	15, [0, 1], x_{32}
*E*4, (3, 2)	9, [0, 1], x_{43}
*E*4, (4, 2)	15, [0, 1], x_{44}
*W*4, (1, 2)	3, [0, 1], x_{41}
*E*5, (3, 2)	6, [0, 1], x_{53}
*W*5, (1, 2)	3, [0, 1], x_{51}
*W*5, (2, 2)	12, [0, 1], x_{52}
*E*6, (4, 2)	12, [0, 1], x_{64}
*W*6, (2, 2)	15, [0, 1], x_{62}
*E*7, (3, 3)	3, [0, 1], x_{73}
*W*7, (1, 3)	2, [0, 1], x_{71}
*W*7, (2, 3)	5, [0, 1], x_{72}
*E*8, (4, 3)	1, [0, 1], x_{84}
*W*8, (1, 3)	2, [0, 1], x_{81}

Arc	{cost, [LB, UB], x_{m+1j}}
→ (1, 4)	50, [0, ∞], x_{91}
→ (2, 4)	50, [0, ∞], x_{92}
→ (3, 4)	50, [0, ∞], x_{93}
→ (4, 4)	50, [0, ∞], x_{94}

Arc: (j, t), (j, k)	{cost, [LB, UB], y_{jt}}
(1, 1), (1, 2)	0, [1, ∞], y_{11}
(2, 1), (2, 2)	0, [1, ∞], y_{21}
(3, 1), (3, 2)	0, [1, ∞], y_{31}
(4, 1), (4, 2)	0, [1, ∞], y_{41}
(1, 2), (1, 4)	0, [0, ∞], y_{12}
(2, 2), (2, 4)	0, [0, ∞], y_{22}
(3, 2), (3, 4)	0, [0, ∞], y_{32}
(4, 2), (4, 4)	0, [0, ∞], y_{42}
(1, 3), (1, 4)	0, [0, ∞], y_{13}
(2, 3), (2, 4)	0, [0, ∞], y_{23}
(3, 3), (3, 4)	0, [0, ∞], y_{33}
(4, 3), (4, 4)	0, [0, ∞], y_{43}
(1, 4), (1, 2)	0, [2, ∞], y_{14}
(2, 4), (2, 2)	0, [2, ∞], y_{24}
(3, 4), (3, 2)	0, [2, ∞], y_{34}
(4, 4), (4, 2)	0, [2, ∞], y_{44}

4.1 *Heuristic 1: Network Based Heuristic for Scenarios S1, S2 or S3*

Step 1. Examine constraints (15b, c, d (or C3′), f) under scenario (S1), (S2), or (S3), and represent them as network constraints as discussed above (see Theorem 1). Ignore (15e) and solve the resulting problem RRSP (rewritten via (17) and (18)) as a network flow problem. If the resulting solution is acceptable with respect to (15e) (See Equation (22) below), then stop. Otherwise, proceed to Step 2.

Step 2. Since, the current solution is not acceptable with respect to (15e), identify the set of "violated" constraints that need to be penalized in the objective function as

$$V = \{(i, t, j) : s_{ij}^{-} = \sum_{\substack{\rho = j \\ \ni \rho \in A_i}}^{j+h_t} x_{i\rho} - 1 \geq 1 \text{ in constraint (15e) for the current solution}\}. \tag{19}$$

Augment (15a) with the term

$$\sum \sum\sum_{(i,t,j)\in V} \mu_i^h (\sum_{\substack{\rho=j \\ \ni \rho \in A_i}}^{j+h_t} x_{i\rho}), \tag{20}$$

and update the optimum to the network flow problem with respect to this new objective function. If the resulting solution is acceptable with respect to off-nights gaps, then stop. Else, proceed to Step 3 with $(\hat{x}, \hat{s}, \hat{s}^{-})$ as the current solution, and record $\hat{x}$ as the incumbent solution.

Step 3. Remove the extra term (20) from the objective function (15a) (i.e., reset the objective coefficients of the *x*-variables to their original values). For each $i \in M$, compute the objective contribution due to resident *i* as

$$\theta_i = \sum_{j \in A_i} c_{ij} \rho_{t(i)} \hat{x}_{ij} + \sum_{q=1}^{3} \mu_{iq} \hat{s}_{iq} + \sum_{j=1}^{n-h_{t(i)}} \mu_i^h \hat{s}_{ij}^{-} \tag{21}$$

where $t(i)$ is the type of resident i, $\forall\, i \in M$. Construct a list L of residents $i \in M$ arranged in nonincreasing order of θ_i.

Step 4. Proceeding in order within L, find the first resident i for which there exists a pair of successive nights p , q in A_i, at least one of which is unflagged, for which $\hat{x}_{ip} = \hat{x}_{iq} = 1$ while the gap between p and q is unacceptable. If none exists, then stop with $\hat{x}$ as the prescribed solution. Otherwise, let Δ_p (and similarly Δ_q) be the decrease in the last term in the objective function (15a) from its current value if $\hat{x}_{ip}$ (alternatively, $\hat{x}_{iq}$) is put equal to zero. If p or q is flagged, set its corresponding value Δ_p or Δ_q to zero. Pick the larger of Δ_p and Δ_q and set the corresponding value c_{ip} (or c_{iq}) to "∞ ".

Step 5. Update the solution $\hat{x}$ to the network flow problem. If any variable having an "∞" cost coefficient is 1 in the solution $\hat{x}$, reset the coefficient c_{ip} or c_{iq} that was set to ∞ at Step 4 to its original value, flag the corresponding p or q, and return to Step 4. On the other hand, if $\hat{x}$ is acceptable to the off-nights gap restrictions, stop with this solution as the prescribed solution. Otherwise, update the incumbent solution, if necessary, based on (15a). If some limit (say, three times per resident under study) on the visits to this step has been reached, stop with the incumbent as the prescribed solution. Otherwise, return to Step 4.

Remark 3. At the end of Step 5 above, we could transfer instead to Step 3 and recompute the ordered list L based on the most recent solution obtained. This is open to future empirical investigation.

Remark 4. The above algorithm is similar to a depth-first branch-and-bound search having a one level backtrack step whenever infeasibility occurs, and which is terminated once a feasible (acceptable) solution is detected. The acceptability of a solution with respect to (15e) can be specified according to the minimal gap between successive work nights for each resident and/or a minimum number of violations in this minimal gap. This needs to be prespecified as an input. For example, if

$$\sum_{j=1}^{n-h_t} s_{ij}^- \leq h_t \quad \forall\ i \in M_t,\ t = 1,\ldots T \tag{22}$$

we can consider the corresponding solution to be acceptable. Note that as explained before, (22) considers both the number of violations and the extent of violations simultaneously. Also, note that if the horizon of n nights is relatively large, we could permit a violation of h_t for every τ nights at an average (e.g. τ = 20 - 30), and hence use $h_t\left\lceil n/\tau \right\rceil$ in the right-hand side of (22).

4.2 *Heuristic 2: Modification of Heuristic 1 for Case of General Constraints (C3′)*

Let Heuristic 1($\bar{y}$) denote Heuristic 1 operated with y fixed at $\bar{y}$. Then, this heuristic proceeds as follows.

Step 1. Solve the LP relaxation of (17) where (17f) is replaced by $(17\text{f}', 17\text{f}'')$ and (17i) is relaxed. Maintain a variable -e_{jt} on the left-hand side of $(17\text{f}'')$ to ensure feasibility, where $e_{jt} \geq 0$ and is given a high penalty of

$$10^6 . \max_{i \in M_t} \mu_{i3} \quad \forall\ j, t \tag{23}$$

in the objective function. Let $(\bar{x}, \bar{y}, \bar{z})$ be the solution obtained. If this is the first iteration and $\bar{x}$ and $\bar{z}$ are integral, stop with this solution as optimal.

Step 2. If $\bar{z}$ is binary, perform Heuristic 1($\bar{y}$) to obtain the prescribed solution, and stop. Otherwise, consider the sets

$F_1 = \{(j,k) : \bar{z}_{jk} = 1\}$ and $F_2 = \{(j,k) : 0.5 \leq \bar{z}_{jk} \leq 1\}$.

If $F_2 = \varnothing$, let $F_2 = \{(\bar{j},\bar{k})\}$ where $(\bar{j},\bar{k}) \in \underset{(j,k)\notin F_1}{\arg\max}\{\bar{z}_{jk}\}$.

Fix $z_{jk} \equiv 1 \ \forall \ (j, k) \in F_1 \cup F_2$ and return to Step 1.

4.3 *Heuristic 3: Modification of Heuristic 1 for Case of General Constraints (C3)*

Define Heuristic $1(\bar{y})$ as in Section 4.2, and similar to Heuristic 2, add an extraneous resident supply variable e_{jt} to each variable y_{jt} in (17f), where e_{jt} is given an objective penalty as specified by (23) $\forall$ j, t.

Step 1. Solve the LP relaxation of (17) by omitting (17i) (and including the variables e_{jt} $\forall$ j, t as described above). If (17i) is satisfied, then stop with this solution as optimal. Otherwise, let $(\bar{x}, \bar{y}, \bar{e})$ be the solution obtained and denote by π_{jt} the absolute value of the optimal dual variable associated with the constraint (17e) $\forall$ j, t.

Step 2. Let N_{jt} be the number of times that y_{jt} appears in the constraint set (17f) and let

$$F = \{(j,t) : \bar{y}_{jt} \text{ is fractional}\}.$$

Denote $\hat{y} = \lfloor \bar{y} \rfloor$. Proceeding in the order of nondecreasing values of π_{jt}/N_{jt}, increase $\hat{y}_{jt}$ by 1 for each $(j, t) \in F$ in turn until (17f) is satisfied (with e fixed at $\bar{e}$ throughout this process). Then, examining each positive $\hat{y}_{jt}$ variable and the constraints in (17f) where it appears, in the order of nonincreasing values of π_{jt}/N_{jt}, decrease $\hat{y}_{jt}$ by integral amounts to the extent possible while not violating any constraints. (Note that this will perhaps be possible for variables that appear only in (currently) nonbinding constraints.) Given the resulting solution $\hat{y}$ thus obtained, perform Heuristic $1(\hat{y})$ to derive the prescribed solution and stop.

Remark 5. Note that positive values of the extraneous resident variables (e) obtained upon applying Heuristic 2 or 3 should be reported as an output, and would need to be handled by the decision-maker in an interactive fashion. Similarly, infeasiblities in the off-night constraints with respect to the final prescribed solution should be appropriately reported. Note also that to accommodate the case when the original problem might be infeasible, we can include such variables e_{jt} to accompany the corresponding variables y_{jt} in the network relaxations solved in Heuristic 1, along with the associated objective penalties (23). This is implemented in our approach.

4.4 *User Specified Options*

The software developed for the Resident Scheduling Problem permits the use of the following user-specified options. In each of these options, the user can further specify partial fixings of the x, y, or z variables in the input as appropriate, and the corresponding procedure will fix these particular variables at the specified values throughout the process. This permits the user to interactively employ the developed procedures to sequentially construct a schedule.

Option 1. Scenario S1: use Heuristic 1.

Option 2. Scenario S2: use Heuristic 1.

Option 3. y-variables fixed at $\bar{y}$ (including the case of Scenario S3): use Heuristic $1(\bar{y})$.

Option 4. Case of general constraints (C3′): use Heuristic 2.

Option 5. Case of general constraints (C3): use Heuristic 3.

Chapter 5
Computational Results and Conclusions

In this chapter, a comparison between the two solution methods described earlier in this thesis is presented. The first method represents an exact solution approach using the CPLEX-MIP software, whereas the second method is a library of three approximate (heuristic) solution procedures based on the scheduling requirements as discussed in the previous section.

Solving MIP exactly requires initially transforming the problem into a format acceptable by CPLEX-MIP, and then employing the subroutines of this package to yield the required solution. A C++ code is written for the above mentioned purpose. A brief description of the process is given below.

CPLEX-MIP requires the problem data to be entered via ten matrices, namely, MATBEG, MATCNT, MATIND, MATVAL, LB, UB, CTYPE, OBJ, RHS, and SENSE. Furthermore, objective function coefficients, constraint matrix, and problem variable types must be entered using these matrices. The required matrices are defined as follows.

matbeg[i]: contains the index of the beginning of column i

matcnt[i]: contains the number of entries in column i

matval[i]: contains the value of each entry i in the constraint matrix

matind[i]: indicates the row number of the corresponding coefficient, matval[i]

lb[i]: contains the lower bound of variable i

ub[i]: contains the upper bound of variable i

ctype[i]: represents the integer type of variable i

obj[i]: contains the objective coefficient i

rhs[i]: contains the right-hand side coefficient of row i, and

sense[i]: represents the sense of each constraint in the constraint matrix.

The heuristic solution procedures employ a network flow program (RELAXT-II) written by Bertsekas and Tseng to solve the required minimum cost network flow problems. This network problem is converted within the code into a format acceptable by RELAXT-II. Eleven test problems are solved for each of the three scenarios described earlier in this thesis, using CPLEX-MIP and the heuristic procedures for comparison purposes. These two sets of runs are conducted on a SUN ULTRA 1 workstation having 256 Megabytes of RAM, and the outputs are documented and analyzed. The tables report problem rows, columns, nonzero coefficients, nodes, arcs, objective function value, violations in the specified number of nights available (abbreviated as total workload violation), off-night gap violations, total iterations (including visits to Step 5 given in parenthesis), and CPU time (seconds). The objective function value in the tables represents the sum of the first two terms in Equation (15a). Furthermore, the third term in Equation (15a) is represented by the total workload violations without being multiplied by the μ_{iq} factor. Similarly, the fourth term is represented by the off-nights violations without inflating the term by the μ^h_i factor.

5.1 *Test Problems for Scenario 1*

Problem No. 1

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	8768	9892
Workload violations	0	0
Off-nights violations	0	643
Iterations	6227	3700 (100)
CPU (seconds)	17	48

Problem No. 2

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21305	
Nodes		581
Arcs		4315
Objective value	8883	9914
Workload violations	0	0
Off-nights violations	0	652
Iterations	6480	3700 (100)
CPU (seconds)	21	43

Problem No. 3

Solution method	CPLEX-MIP	Heuristic 1
Rows	3780	
Columns	10835	
Non-zero coefficients	21172	
Nodes		581
Arcs		4315
Objective value	9173	10240
Workload violations	0	0
Off-nights violations	17	658
Iterations	6744	4168 (113)
CPU (seconds)	18	48

Problem No. 4

Solution method	CPLEX-MIP	Heuristic 1
Rows	3780	
Columns	10835	
Non-zero coefficients	21172	
Nodes		581
Arcs		4315
Objective value	Infeasible	5010212
Workload violations		0
Off-nights violations		652
Iterations		4168 (113)
CPU (seconds)		49

Problem No. 5

Solution method	CPLEX-MIP	Heuristic 1
Rows	3780	
Columns	10835	
Non-zero coefficients	21172	
Nodes		581
Arcs		4315
Objective value	9247	10122
Workload violations	31	31
Off-nights violations	17	707
Iterations	7288	4096 (111)
CPU (seconds)	20	71

Problem No. 6

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21255	
Nodes		581
Arcs		4315
Objective value	14760	16678
Workload violations	50	50
Off-nights violations	1	623
Iterations	9438	10828 (298)
CPU (seconds)	35	69

Problem No. 7

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	10525	11577
Workload violations	25	148
Off-nights violations	0	717
Iterations	> 953106	3700 (100)
CPU (seconds)	> 5905	105

Problem No. 8

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	9276	10278
Workload violations	78	78
Off-nights violations	0	696
Iterations	10609	3700 (100)
CPU (seconds)	125	81

Problem No. 9

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	8341	9314
Workload violations	0	0
Off-nights violations	0	636
Iterations	8830	3700 (100)
CPU (seconds)	87	61

Problem No. 10

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	20669	21850
Workload violations	0	0
Off-nights violations	1436	1907
Iterations	10082	16552 (457)
CPU (seconds)	223	159

Problem No. 11

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		581
Arcs		4315
Objective value	12104	13155
Workload violations	70	70
Off-nights violations	0	834
Iterations	11369	3772 (102)
CPU (seconds)	109	80

The results indicate that the relative solution times for CPLEX-MIP and the heuristic procedure are comparable. For half the test problems, the solutions are obtained faster using CPLEX-MIP whereas for the other half, the heuristic procedure is faster. On the other hand, the number of off-nights gap violations are higher for the heuristic procedure while CPLEX-MIP usually finds solutions having a negligible number of violations. However, the total number of workload violations are identical in almost all the cases tested for the heuristic procedure and CPLEX-MIP. It should be also noted, however, that whenever a rather tight problem in which the schedule needed requires most of the available residents is solved, the heuristic procedure compares very well (with respect to time or objective function value) as seen in test problems 7, 8, 9, 10, and 11. Furthermore, it is found that a problem that is infeasible using CPLEX-MIP has yielded a heuristic solution. As an example, test problem 6 is infeasible as detected by CPLEX-MIP because the problem requires that resident 1 (in this particular case) be available for work for more than his/her pre-determined maximum available nights (this resident is not available for the entire horizon), causing the infeasiblity to occur. This situation can be simply resolved by increasing this particular resident’s availability nights over the required horizon. On the other hand, the heuristic solution managed to overcome this infeasibility by borrowing the needed nights from other residents within the same group but with an increase in the off-nights violations. Also, when CPLEX-MIP required a long time to solve the problem, the heuristic procedure obtained a reasonable solution in far less time as seen with Problem 7.

5.2 *Test Problems for Scenario 2*

Problem No. 1

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		511
Arcs		4245
Objective value	22512	23867
Workload violations	41	41
Off-nights violations	0	883
Iterations	6764	3700 (100)
CPU (seconds)	15	83

Problem No. 2

Solution method	CPLEX-MIP	Heuristic 1
Rows	3875	
Columns	10940	
Non-zero coefficients	21545	
Nodes		511
Arcs		4245
Objective value	13081	14689
Workload violations	35	35
Off-nights violations	0	820
Iterations	9213	3700 (100)
CPU (seconds)	35	48

Problem No. 3

Solution method	CPLEX-MIP	Heuristic 1
Rows	3740	
Columns	10805	
Non-zero coefficients	21275	
Nodes		511
Arcs		4245
Objective value	27979	30475
Workload violations	0	0
Off-nights violations	2017	2120
Iterations	10113	10936 (301)
CPU (seconds)	42	91

Problem No. 4

Solution method	CPLEX-MIP	Heuristic 1
Rows	3813	
Columns	10764	
Non-zero coefficients	21199	
Nodes		511
Arcs		4245
Objective value	29590	31673
Workload violations	0	0
Off-nights violations	2217	2246
Iterations	9780	10936 (301)
CPU (seconds)	32	66

Problem No. 5

Solution method	CPLEX-MIP	Heuristic 1
Rows	3813	
Columns	10764	
Non-zero coefficients	21199	
Nodes		511
Arcs		4245
Objective value	Infeasible	33128
Workload violations		0
Off-nights violations		2251
Iterations		10900 (300)
CPU (seconds)		70

Problem No. 6

Solution method	CPLEX-MIP	Heuristic 1
Rows	3805	
Columns	10905	
Non-zero coefficients	21405	
Nodes		511
Arcs		4245
Objective value	13585	15268
Workload violations	70	70
Off-nights violations	0	810
Iterations	13081	3700 (100)
CPU (seconds)	45	70

Problem No. 7

Solution method	CPLEX-MIP	Heuristic 1
Rows	3743	
Columns	10729	
Non-zero coefficients	21059	
Nodes		511
Arcs		4245
Objective value	29590	31658
Workload violations	0	0
Off-nights violations	2217	2253
Iterations	10119	10900 (300)
CPU (seconds)	54	71

Problem No. 8

Solution method	CPLEX-MIP	Heuristic 1
Rows	3743	
Columns	10729	
Non-zero coefficients	21059	
Nodes		511
Arcs		4245
Objective value	29590	31658
Workload violations	0	0
Off-nights violations	2217	2253
Iterations	10119	10900 (300)
CPU (seconds)	59	74

Problem No. 9

Solution method	CPLEX-MIP	Heuristic 1
Rows	3743	
Columns	10729	
Non-zero coefficients	21059	
Nodes		511
Arcs		4245
Objective value	29590	31694
Workload violations	0	0
Off-nights violations	2217	2241
Iterations	9476	10936 (301)
CPU (seconds)	43	70

Problem No. 10

Solution method	CPLEX-MIP	Heuristic 1
Rows	37778	
Columns	10764	
Non-zero coefficients	21129	
Nodes		511
Arcs		4245
Objective value	29397	31389
Workload violations	0	0
Off-nights violations	2217	2264
Iterations	10820	10900 (301)
CPU (seconds)	21	61

Problem No. 11

Solution method	CPLEX-MIP	Heuristic 1
Rows	3840	
Columns	10940	
Non-zero coefficients	21475	
Nodes		511
Arcs		4245
Objective value	29492	31475
Workload violations	0	0
Off-nights violations	2400	2330
Iterations	10034	12880 (355)
CPU (seconds)	21	75

For this scenario, it is seen that the heuristic procedure produced solutions of quality comparable with that obtained by CPLEX-MIP. In particular, the workload violations coincide for the two methods in all cases, and the off-nights violations are almost identical for the two solution methods in most cases. Moreover, objective values reported for both solution methods are off by a very small margin (around 6%). On the other hand, the CPU time for the heuristic procedure exceeds that for CPLEX-MIP in all the test problems. Similar to the case of Scenario 1, an infeasible problem encountered by CPLEX-MIP has offered a heuristic solution with the same explanation presented before in Scenario 1.

5.3 *Test Problems for Scenario 3*

Problem No. 1

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	26029	26531
Workload violations	40	40
Off-nights violations	0	730
Iterations	6853	3700 (100)
CPU (seconds)	25	18

Problem No. 2

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22064	
Nodes		441
Arcs		4175
Objective value	26044	26567
Workload violations	40	40
Off-nights violations	0	719
Iterations	7220	3700 (100)
CPU (seconds)	24	92

Problem No. 3

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22064	
Nodes		441
Arcs		4175
Objective value	46701	47589
Workload violations	0	0
Off-nights violations	2402	2376
Iterations	10033	12808 (353)
CPU (seconds)	27	227

Problem No. 4

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22064	
Nodes		441
Arcs		4175
Objective value	33235	32801
Workload violations	15	15
Off-nights violations	2433	2376
Iterations	10965	12412 (342)
CPU (seconds)	26	104

Problem No. 5

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22064	
Nodes		441
Arcs		4175
Objective value	48517	48679
Workload violations	0	0
Off-nights violations	2402	2256
Iterations	10318	12664 (349)
CPU (seconds)	28	178

Problem No. 6

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	136996	136746
Workload violations	50	50
Off-nights violations	2499	2287
Iterations	12733	12592 (347)
CPU (seconds)	33	125

Problem No. 7

Solution method	CPLEX-MIP	Heuristic 1
Rows	3875	
Columns	11045	
Non-zero coefficients	21516	
Nodes		441
Arcs		4175
Objective value	123673	123627
Workload violations	0	0
Off-nights violations	801	1176
Iterations	8487	13780 (380)
CPU (seconds)	22	283

Problem No. 8

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	1070492	1070937
Workload violations	0	0
Off-nights violations	400	1160
Iterations	8020	13204 (364)
CPU (seconds)	25	195

Problem No. 9

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	1070492	1070937
Workload violations	0	0
Off-nights violations	400	1160
Iterations	8020	13204 (355)
CPU (seconds)	26	184

Problem No. 10

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	1070492	1071117
Workload violations	0	0
Off-nights violations	400	1162
Iterations	8127	12736 (351)
CPU (seconds)	24	138

Problem No. 11

Solution method	CPLEX-MIP	Heuristic 1
Rows	3980	
Columns	11045	
Non-zero coefficients	22070	
Nodes		441
Arcs		4175
Objective value	1070524	1073217
Workload violations	0	0
Off-nights violations	410	1176
Iterations	8142	12801 (373)
CPU (seconds)	26	142

For the case of Scenario 3, the CPU time is considerably higher for the heuristic procedure, but it finds solutions having a comparable value for the objective function, in contrast with CPLEX-MIP. Also, the off-nights gap violations are higher than those for the CPLEX-MIP solutions in most cases, although in some cases, they are comparable to the CPLEX-MIP values. The workload violations are again identical for both the methods for this scenario runs.

5.4 *Test Problems for Heuristic 2*

Problem No. 1

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	26029	25444
Workload violations	40	40
Off-nights violations	0	0
Iterations	6853	5885
CPU (seconds)	25	24

Problem No. 2

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3702
Columns	11045	7194
Non-zero coefficients	22064	17585
Nodes		
Arcs		
Objective value	26044	25414
Workload violations	40	40
Off-nights violations	0	0
Iterations	7220	5892
CPU (seconds)	24	22

Problem No. 3

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3702
Columns	11045	7194
Non-zero coefficients	22064	17585
Nodes		
Arcs		
Objective value	46701	45297
Workload violations	0	0
Off-nights violations	2402	2403
Iterations	10033	10153
CPU (seconds)	27	25

Problem No. 4

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3665
Columns	11045	7133
Non-zero coefficients	22064	17429
Nodes		
Arcs		
Objective value	33235	31831
Workload violations	15	15
Off-nights violations	2433	2433
Iterations	10965	10130
CPU (seconds)	26	24

Problem No. 5

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3702
Columns	11045	7194
Non-zero coefficients	22064	17585
Nodes		
Arcs		
Objective value	48517	40113
Workload violations	0	0
Off-nights violations	2402	2403
Iterations	10318	11361
CPU (seconds)	28	26

Problem No. 6

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	136996	135592
Workload violations	50	50
Off-nights violations	2499	2500
Iterations	12733	11361
CPU (seconds)	33	33

Problem No. 7

Solution method	CPLEX-MIP	Heuristic 2
Rows	3875	3512
Columns	11045	7044
Non-zero coefficients	21516	13768
Nodes		
Arcs		
Objective value	123673	122739
Workload violations	0	0
Off-nights violations	801	801
Iterations	8487	8382
CPU (seconds)	22	21

Problem No. 8

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	1070492	1069558
Workload violations	0	0
Off-nights violations	400	400
Iterations	8020	8364
CPU (seconds)	25	24

Problem No. 9

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	1070492	1069558
Workload violations	0	0
Off-nights violations	400	400
Iterations	8020	8364
CPU (seconds)	26	24

Problem No. 10

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	1070492	1069558
Workload violations	0	0
Off-nights violations	400	400
Iterations	8127	8333
CPU (seconds)	24	24

Problem No. 11

Solution method	CPLEX-MIP	Heuristic 2
Rows	3980	3705
Columns	11045	7200
Non-zero coefficients	22070	17600
Nodes		
Arcs		
Objective value	1070502	1069562
Workload violations	0	0
Off-nights violations	410	400
Iterations	8239	8471
CPU (seconds)	28	27

5.5 *Test Problems for Heuristic 3*

Problem No. 1

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		
Arcs		
Objective value	8768	8768
Workload violations	0	0
Off-nights violations	0	0
Iterations	6227	5569
CPU (seconds)	17	15

Problem No. 2

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3624
Columns	10905	7169
Non-zero coefficients	21305	17357
Nodes		441
Arcs		4175
Objective value	8883	9850
Workload violations	0	5
Off-nights violations	0	571
Iterations	6480	3700 (100)
CPU (seconds)	21	37

Problem No. 3

Solution method	CPLEX-MIP	Heuristic 3
Rows	3780	3624
Columns	10835	7169
Non-zero coefficients	21172	17357
Nodes		441
Arcs		4175
Objective value	9173	110236
Workload violations	0	23
Off-nights violations	17	565
Iterations	6744	3700 (100)
CPU (seconds)	18	37

Problem No. 4

Solution method	CPLEX-MIP	Heuristic 3
Rows	3780	3623
Columns	10835	7167
Non-zero coefficients	21172	17352
Nodes		441
Arcs		4175
Objective value	Infeasible	11586
Workload violations		0
Off-nights violations		89
Iterations		2174
CPU (seconds)		8

Problem No. 5

Solution method	CPLEX-MIP	Heuristic 3
Rows	3780	3624
Columns	10835	7169
Non-zero coefficients	21172	17357
Nodes		441
Arcs		4175
Objective value	9247	9247
Workload violations	31	31
Off-nights violations	17	17
Iterations	7288	5937
CPU (seconds)	20	16

Problem No. 6

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3563
Columns	10905	7083
Non-zero coefficients	21255	17124
Nodes		441
Arcs		4175
Objective value	14760	12108
Workload violations	50	181
Off-nights violations	1	629
Iterations	9438	4312 (117)
CPU (seconds)	35	48

Problem No. 7

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		441
Arcs		4175
Objective value	10525	11731
Workload violations	25	148
Off-nights violations	0	706
Iterations	> 953106	3916 (106)
CPU (seconds)	> 5905	59

Problem No. 8

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		441
Arcs		4175
Objective value	9276	11304
Workload violations	78	78
Off-nights violations	0	714
Iterations	10609	10900 (300)
CPU (seconds)	125	70

Problem No. 9

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		441
Arcs		4175
Objective value	8341	9325
Workload violations	0	1
Off-nights violations	0	587
Iterations	8830	3700 (100)
CPU (seconds)	87	45

Problem No. 10

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		441
Arcs		4175
Objective value	20669	20509
Workload violations	0	0
Off-nights violations	1436	1430
Iterations	10082	9926
CPU (seconds)	223	72

Problem No. 11

Solution method	CPLEX-MIP	Heuristic 3
Rows	3805	3705
Columns	10905	7300
Non-zero coefficients	21405	17700
Nodes		441
Arcs		4175
Objective value	12104	11889
Workload violations	70	69
Off-nights violations	0	0
Iterations	11369	9958
CPU (seconds)	109	33

As described above, Heuristics 2 and 3 are processed in two stages. The first stage utilizes CPLEX-LP in solving a relaxed version of Problem RSP. In some cases, an optimal solution could be found by this first stage itself. In other cases, Heuristic 1($\bar{y}$) is used to complete the process. From above, it can be seen that in most cases, Heuristics 2 and 3 yield an improvement over Heuristic 1, and prove to be also compatible to CPLEX-MIP. In particular, test problems solved using Heuristics 2 and 3 have better CPU time, objective function value, workload violations, and off-nights violations as compared with Heuristic 1. Furthermore, in some cases, these procedures even superseded CPLEX-MIP, reaching a solution using fewer iterations, and yielding lesser off-nights and workload violations, the latter occurring because some of the constraints present in the original problem are relaxed in Heuristics 2 and 3.

As documented above, two approaches are proposed to solve the resident scheduling problem. Three scenarios are suggested for this problem that capture many possibilities and situations facing hospitals. The first approach is concerned with formulating and solving an MIP problem using CPLEX-MIP subroutines. An exact solution is obtained by this method, with no compromise on infeasiblities. The second approach utilizes a heuristic methodology that exploits the underlying network structure to compose a solution. Note that the resident scheduling problem is formulated using cost factors provided by residents involved in the scheduling process that reflect his/her desirability for working during a particular shift. Any change in these coefficients may lead to a different solution (schedule). The schedule-maker and residents have to work closely so that a good schedule is obtained that satisfies both departmental requirements and residents' preferences.

The test problems used in this chapter represent situations where schedules need to be prepared for 100 residents over a five-week horizon. As seen above, the heuristic methodology tends to provide good solutions that are comparable to the exact solutions with more control on the computational effort. In some cases, heuristic solutions are prescribed where no exact solutions could be reached. This is an advantage of utilizing the heuristic procedure in solving the problem; the resulting infeasibilities could be resolved based on resource availabilities and requirements.

Chapter 6
Summary, Conclusions, and Recommendations for Future Research

This thesis has considered the Resident Scheduling Problem (RSP) in which a good schedule is sought that meets both departmental needs and residents' preferences, and that can be produced quickly, reliably, and with greater flexibility than the manual technique employed presently by hospitals. Three scenarios are described that correspond to most cases encountered by hospitals. Models are also formulated for other more general situations. The RSP is solved via two methods, an exact and an approximate (heuristic) approach. The exact solution method utilizes the CPLEX-MIP package, while the heuristic procedures developed exploit the network structure of the problem to construct schedules in a robust and reliable fashion. These heuristic procedures accommodate special scenarios as well as general scheduling requirements. To test the efficiency of the heuristic procedures, eleven problems are solved for Scenarios 1, 2 and 3 utilizing Heuristic 1. In addition, eleven more generally constrained sets of test problems of two types are also solved via the appropriate Heuristics 2 and 3. For comparison purposes, all these test problems are solved using CPLEX-MIP package as well. Detailed results are presented in Chapter 5. It is found that in most cases, the heuristic procedures' results are comparable to the exact solution, and in some cases, a good solution is produced while the exact method exceeded computational limits. In addition, for some tightly restricted problems for which the exact method simply declared infeasiblity, the heuristic procedure offered a compromising solution.

This research effort has attempted to accommodate several cases in modeling the resident scheduling problem. Obviously, there might exist departments and requirements that have not been covered. Amendments can be made on the model developed in this thesis that will account for such other circumstances and conditions. In this case, the input data format would need to be changed to be compatible with the RELAXT-II subroutine, assuming that these new constraints would still maintain the partial network structure of the problem. Otherwise, the new requirements must be added to the model and new solution methods might need to be devised.

It would be interesting to see how this model behaves using real data obtained from hospitals, and to compare the solutions obtained with the hospital’s manual schedules constructed for large groups of residents over a horizon of several weeks. In particular, feedback from residents can be used to evaluate the effectiveness of the schedules obtained by this model versus the hospital’s own schedules. This is proposed as a next step following this research.

Bibliography

Aykin, T., “Optimal Shift Scheduling with Multiple Break Windows,” *Management Science*, 42(4), pp. 591-602, 1996.

Baker, K., “Scheduling a Full-time Workforce to Meet Cyclic Staffing Requirements,” *Management Science*, 20, pp. 1561-1568, 1974.

Barnhart, C., E. Johnson, G. Nemhauser, G. Sigismondi, and P. Vance, “Formulating a Mixed Integer Programming Problem to Improve Solvability,” *Operations Research*, 41(6), pp. 1013-1019, 1993.

Bartholdi, J., J. Orlin, and H. Ratliff, “Cyclic Scheduling Via Integer Programs with Circular Ones, ”*Operations Research*, 28, pp. 1074-1085, 1980.

Bazaraa, M., J. Jarvis, and H. Sherali, *Linear Programming and Network Flows*, (2nd Edition), John Wiley & Sons, New York, 1990.

Beasley, J. and B. Cao, “A Tree Search Algorithm for the Crew Scheduling Problem,” *European Journal of Operational Research*, 94(3), pp. 517-526, 1996.

Bertsekas, D., “A Unified Framework for Primal-Dual Methods in Minimum Cost Network Flow Problems, “ *Mathematical Programming*, 32, pp. 125-145, 1985.

Bertsekas, D. and P. Tseng, “Relaxation Methods for Minimum Cost Ordinary and Generalized Network Flow Problem,” *Operations Research*, 36, pp. 93-114, 1988.

Beaumont, N., “Scheduling Staff using Mixed Integer Programming,” *European Journal of Operational Research*, 98(3), pp. 473-484, 1997.

Bianco L. and M. Bielli, “A Heuristic Procedure for the Crew Rostering Problem,” *European Journal of Operational Research*, 58(2), pp. 272-283, 1992.

Blantify, J. and C. Blackburn, “General Purpose Multiple Choice Programming by Truncated Block Enumeration,” presented at the 36th National Meeting of ORSA, Miami Beach, Florida, November, 1969.

Burns, R., “Manpower Scheduling with Variable Demands and Alternate Weekends Off,” *Information*, 16, pp. 101-111, 1978.

Chen, J. and T. Yeung, “Hybrid Expert-System Approach to Nurse Scheduling,” *Comput. Nursing*, 11, pp. 183-190, 1993.

Cheng, B., J. Lee, and J. Wu, "A Nurse Rostering System Using Constraint Programming and Redundant Modeling," *IEEE Transaction Technology in Biomedicine*, 1(1), pp. 44-54, 1997.

Chu, H., E. Gelman, and E. Johnson, "Solving Large Scale Crew Scheduling Problems," *European Journal of Operational Research*, 97(2), pp. 260-268, 1997.

Dantzig, G., "A Comment on Edie's 'Traffic Delays at Toll Booths'," *Operations Research*, 2(3), pp. 339-341, 1954.

Edie, L. C., "Traffic Delays at Toll Booths," *Operations Research*, 2(2), pp. 107-138, 1954.

Emmons, H. and R. Burns, "Off-day Scheduling with Hierarchical Worker Categories," *Operations Research*, 39(3), pp. 484-495, 1991.

Graves G., and R. McBride, I. Gershkoff, D. Anderson, And D. Mahidhara, "Flight Crew Scheduling," *Management Science*, 39(6), pp. 736-745, 1993.

Henderson, W. B. and W. L. Berry, "Heuristic Methods for Telephone Operator Shift Scheduling: An Experimental Analysis," *Management Science*, 22(12), pp. 1372-1380, 1976.

Jarrah, A., J. Board, and A. DeSilva, "Solving Large-scale Tour Scheduling Problems," *Management Science*, 40(9), pp. 1124-1144, 1994.

Keith, E. G., "Operator Scheduling," *AIIE Transactions*, 11, pp. 37-41, 1979.

Kostreva, M., M. Leszcyski and F. Passini, "The Nurse Scheduling Decision Via Mixed-Integer Programming," *Proceedings of the American Hospital Association Forum on Nurse Scheduling*, pp. 291-305, 1978.

Kostreva, M. and K. Jennings, "Nurse Scheduling on a Microcomputer," *Computers and Operations Research*, 18(8), pp. 731-739, 1991.

Mackworth, A., "Consistency in Networks of Relations," *AI Journal*, 8(1), pp. 99-118, 1977.

Miller, H., W. Pierskalla and G. Rath, "Nurse Scheduling Using Mathematical Programming," *Operations Research*, 24(5), pp. 857-870, 1976.

Musa, A. and U. Saxena, "Scheduling Nurses Using Goal Programming Techniques," *IIE Transactions*, 16, pp. 216-221, 1984.

Narasimhan, R. “An Algorithm for Single Shift Scheduling of Hierarchical Workforce,” *European Journal of Operational Research*, 96(1), pp. 113-121, 1997.

Okada, M. and M. Okada, “Prolog-based System for Nursing Staff Scheduling Implemented on Personal Computer,” *Computers and Biomedical Research*, 21, pp. 53-63, 1988.

Ozkarahan, I. and J. Baily, “Goal Programming Model Subsystem of a Flexible Nurse Scheduling Support System,” *IIE Transactions*, 20, pp. 306-316, 1988.

Rosenbloom, E. and N. Goertzen, “Cyclic Nurse Scheduling,” *European Journal of Operational Research*, 31, pp. 19-23, 1987.

Schuette, L., “Development of Criteria for Evaluating Nurse Scheduling,” Program in Hospital Administration, University of Michigan, Ann Arbor, Michigan, 1973.

Segal, M., “The Operator-Scheduling Problem: A Network Flow Approach,” *Operations Research*, 22, pp. 808-823, 1974.

Smith, L. and Wiggins A., “A Computer Based Nurse Scheduling System,” *Computers and Operations Research*, 4, pp. 195-212, 1977.

Vance, P., C. Barnhart, E. Johnson, and G. Nemhauser, “Airline Crew Scheduling: A New Formulation and Decomposition Algorithm,” *Operations Research*, 45(2), pp. 188-200, 1997.

Vohra, R. V., “The Cost of Consecutivity in the (5, 7) Cyclic Staffing Problem,” *IIE Transactions*, 19(3), pp. 296-299, 1987.

Warner, D., “Scheduling Nursing Personnel According to Nursing Preference: A Mathematical Programming Approach,” *Operations Research*, 24, pp. 842-856, 1976.

Weil, G., K. Heus, P. Francois, and M. Poujade, “Constraint Programming for Nurse Scheduling, ”*IEEE Engineering in Medicine and Biology*, 14(4), pp. 417-422, 1995.

Yura K., “Production Scheduling to Satisfy Worker’s Preferences for Days Off and Overtime under Due-date Constraints,” *International Journal of Production Economics*, 33(1-3), pp. 265-270, 1994.

Vita

Muhannad Hasan Ramahi was born on July 30, 1965 in Kuwait. He received his B.Sc. in Civil Engineering from Kuwait University in January 1987. He worked from August 1987 to July 1989 as a Civil Engineer for Campenon Bernard on the Jahra/Ghazalli Expressway project in Kuwait City.

In August 1989, he attended University of Michigan, Ann Arbor and was awarded an M.Sc. degree in Electrical Engineering in December 1992. Afterwards, he enrolled at Ohio State University and obtained a Master of Science degree in Civil Engineering in September 1994. Finally, he attended Virginia Polytechnic Institute and State University (Virginia Tech) in the Fall of 1995. He plans to graduate this Fall of 1998 with a Master of Science degree in Industrial Engineering (Operations Research).

He would be pursuing a career as an Operations Research Consultant with THE SABRE GROUP.