Near-Optimal MIP Solutions for Preference Based Self-Scheduling

Eyjólfur Ingi Ásgeirsson · Guðríður Lilla Sigurðardóttir

Received: date / Accepted: date

Abstract Making a high quality staff schedule is both difficult and time consuming for any company that has employees working on irregular schedules. We formulate a mixed integer program (MIP) to find a feasible schedule that satisfies all hard constraints while minimizing the soft constraint violations as well as satisfying as many of the employees' requests as possible. We present the MIP model and show the result from four real world companies and institutions. We also compare the results with those of a local search based algorithm that is designed to emulate the solution strategies when the schedules are created manually.

The results show that the using near-optimal solutions from the MIP model, with a relative MIP gap of around 0.01-0.1 allows us to find very good solutions in a reasonable amount of time that compare favorably with both the manual solutions and the solutions found by the local search based algorithm.

Keywords Staff Scheduling · Rostering · Mixed Integer Programming · Local Search

Work done in collaboration with Vaktaskipan ehf.

E. I. Ásgeirsson
Reykjavík University
Tel.: +354-599-6385
Fax: +354-599-6201
E-mail: eyjo@ru.is

G. L. Sigurðardóttir
Reykjavík University

1 Introduction

Staff scheduling is a problem that is well known to all companies that have employees working on irregular schedules. Usually, the problem is to determine which employees should cover which shifts so that the demand for manpower is met at every time period and without breaking any regulations or contracts. Additionally, the schedule should be good, i.e. it should meet with the employees' approval and satisfy as many of their requests as possible.

The nurse rostering problem has been studied by personnel managers, operations researchers and computer scientists for over 50 years. Variations of the problem are NP-hard [21,10,29,25]. Because the nurse rostering problem is well known and since it can include many types of constraints and cover a large set of staff scheduling problems, a large part of the research on staff scheduling is focused on nurse rostering and the terms nurse rostering and nurse scheduling have been used over the years to cover several types of personnel scheduling problems [14].

In nurse rostering there are three key approaches: cyclical scheduling, preference scheduling and self scheduling [9]. A cyclical scheduling problem is a scheduling problem in which several sets of schedules are generated that cover a certain period of time i.e. a month or three months. Then the staff is assigned to a schedule that best fits their preferences so that all demands for manpower are met. The schedules are then repeated for each period. Cyclical scheduling is somewhat inflexible and therefore not able to adjust rapidly to changes in the environment [27]. The main advantage of the cyclical scheduling is that the employees know their schedule a long time in advance.

In preference scheduling, the employees list their preferences for the staff manager who then creates schedules, trying to fulfill as many preferences as possible but also makes sure that all demands for manpower and all work restrictions are met. Thus the personnel manager has a great deal of responsibility for the quality of the schedules. The preference scheduling has many advantages, the major ones being its flexibility and its individual tailoring. Preferences of the staff have become a vital feature of any successful scheduling system. Kellogg and Walczak [24] state that any academic nurse rostering model that does not include some opportunity for preference scheduling will probably not be implemented. The major downside to preference scheduling is the time it takes for the personnel manager to create a good schedule.

In self scheduling the employees themselves become responsible for creating the schedule, instead of the staff manager. These schedules are created by each employee signing up for their preferred shifts knowing the minimum and maximum number of staff needed for each shift with the requirement that the resulting schedule must be a feasible one. The biggest advantages of the self scheduling, beside possible time savings, are the potentially greater staff satisfaction, more commitment and reduced staff turnover, since the employees are empowered by making the schedules themselves. However until recently self scheduling has not been a good approach since it was too difficult to execute this method fairly [22,8], the order in which the personnel sign up

did matter, there was a possibility that the system might get manipulated by some personnel, new employers were unfamiliar with the system and might therefore be disadvantaged and some employees might not sign up for any shifts at all. With the advent of the Internet it has become easier to implement self scheduling fairly, putting self scheduling back on the map as a viable approach. However, having the employees involved in the decision process will always bring some risk of game playing, where employees try to manipulate the system for their own gain 14].

A good way to implement self scheduling is by mixing preference scheduling and pure self scheduling. Here the staff signs up for shifts, making a draft that the personnel manager then turns into feasible schedule. The personnel manager makes sure that the demand for manpower is met at every time and that no work regulations are broken. In this approach the personnel is responsible for creating a good preliminary schedule but the final responsibility of creating the schedule lies with the personnel manager, making this approach better and more fair than either pure preference scheduling or pure self scheduling. This is the approach used in this paper.

Due to the multiple and often changing objectives and goals of staff scheduling, the research on staff scheduling has included many different methods 119]. Mathematical programming techniques, such as column generation 15,23] and branch and price methods 130], have shown good results. The research on staff scheduling has also focused on more flexible metaheuristic approaches such as genetic algorithms 11,2,18,28] and variable neighborhood search 113], with Tabu-Search 112,17] and Simulated Annealing 111] being particularly successful 120]. Good overviews of staff scheduling are 13,20,26].

The paper is structured as follows: Section 2 the problem is defined, in Section 3 the model is introduced while Section 4 contains the result of the model using real data, as well as a comparison to a local search based heuristic from 16]. Finally, Section 5 contains conclusions and suggestions for further work.

2 Problem Definition

Staff scheduling problems have a large number of constraints that need to be satisfied. Those constraints can be divided into two groups, hard constraints and soft constraints. Hard constraints must always be satisfied in order to have a feasible schedule. Hard constraints are often a result of physical resource restrictions and legislations. Soft constraints are requirements that are desirable but not obligatory and therefore allowed to be violated if necessary but it will result in a penalty in the model. Soft constraints violations are often used to evaluate the quality of feasible schedules.

2.1 Hard constraints

The hard constraints are mainly based on contracts with the employees and union contracts and must therefore be satisfied at all times. Not all the constraints are the same for all employees although usually the main constraints are the same. The constraints that are usually not the same for all employees are the work limit constraints. The labels in parenthesis after each constraint in the following lists refer to the classification introduced in F7]. In the model the following hard constraints are considered:

- **Restrictions on working hours and rest periods from union regulations and employee contracts.** The union regulations about rest periods, maximum lengths of continuous work within a day, maximum number of continuous days worked, minimum length of continuous rest between shifts and other limits have to be met. Employee contracts can include restrictions on when employee can work, for example an employee that will never work nights or weekends. (R1,R4,R5,R7,R8)
- **Vacation request.** Vacations are considered to be a hard constraint to ensure that no employee will be assigned to a shift while on a vacation. (R2)
- **Requests for time off.** Each employee has a right to some time off, how many hours depending on the company and the employee contract. In our settings this needs to be a hard constraint so these requests won't be violated. (R1)
- **Working weekends.** There can be limits on how many weekends employees are allowed to work in each scheduling period. Each employee has to receive at least A out of every B weekends off, where $A \leq B$. These are limits like 2 or 3 weekends off out of every 4 consecutive weekends. (R3)
- **Special shifts, training sessions or meetings.** Employees often have work related duties that are not flexible and are often not included in the number of employees on duty. Since training sessions and meetings are not flexible these constraints must be satisfied. (R6,O6)
- **Other limits on shifts or working hours, for example split shifts.** Split shifts are defined as two separate shifts within the same day, where the time between the shifts is less than the minimum resting period between shifts. It can differ between companies whether splits shifts are allowed or not. Splits shifts are only hard constraints when split shifts are not allowed. (R9)

Each company is different when it comes to number of employees, contracts, habits and regulations, therefore the constraints differs from one company to another. Each company wants to be able to quickly generate a high quality schedule that satisfies all hard constraints and as many of the soft constraints as possible.

2.2 Soft constraints

Each time a soft constraint is violated the schedule receives a penalty that appears in the objective function. How high the total penalty is depends on which constraints are violated and how often they are violated. The penalties have different weight factors, depending on how serious a violation of the relevant constraint would be. In the model the following soft constraints are considered:

- Minimum and maximum staff level. An estimate of the demand for manpower at every time slot over the whole period the schedule is supposed to cover is necessary. This estimate can vary greatly between companies depending on how good their forecast for the demand of manpower is. Some companies use minimum and maximum staff level for every time slot while others give exact number of employees needed for every time slot. We want the on-duty employees in the schedule to be between the minimum and maximum staff level or as close as possible to the exact number of required on-duty employees, otherwise the schedule will be penalized. (C2,C3)
- Minimum and maximum number of on-duty hours for each employee. In every employee contract a number of required on-duty hours are given. However since the employees are often working irregular hours, there must be some flexibility in required on-duty hours for each scheduling period. Therefore the required on-duty hours are interpreted as minimum and maximum number of on-duty hours for each employee. Minimum and maximum numbers of on-duty hours for each employee are calculated based on monthly working hours given in the contracts and accumulated deviations from the required on-duty hours from the previous period. (R1)
- Employee requests for shifts. The first step in making a schedule is to make each employee signs up for their preferred shifts knowing the minimum and maximum number of staff needed for each shift. This encourages employees to create their own work schedule and makes the schedule more acceptable for the employees. It is therefore important to meet as many requests as possible. (P3,P4)
- Employees assigned to shifts on weekends before or after their vacations. If an employee is finishing his vacation on Friday or beginning his vacation on Monday it is unlikely he wants to work the adjacent weekend. So unless otherwise requested we will try to have the adjacent weekend free. (E8)

3 The MIP Model

The model contains five sets of binary variables to keep track of the staff allocation. The binary variable y_{itk} determines if employee $i \in I$ is working in timeslot $t \in T$ on day $k \in K$, where I is the set of employees, T is the set of timeslots within a single day and K is the set of days in the scheduling period. Each day is partitioned into timeslots, which, in our examples are

usually 30 minutes, although some companies use 15 minutes or whole hours. The employees are assigned to shifts, where a shift is subset of the timeslots. The timeslots within a single shift are usually contiguous, although this is not necessary. The binary variable x_{ijk} determines whether employee i E I is assigned to shift j E J on day k E K, where J is the set of allowed shifts. A shift can include timeslots from two consecutive days, so we say that shift j belongs to day k if the first timeslot in shift j is within day k.

Additionally, we have binary variables d_{ik} that determine if employee i E I is working on day k E K. Since some regulations concern weekends, we use binary variables ω_{iw} that denote if employee i is working on the w-th weekend, where 1 < w < |W| and W = [W1, W2, ...] C (KXK) is the set of all weekends, i.e. Saturdays and Sundays, in K, and W_w is a set containing the corresponding days for the w-th weekend.

Every employee contract we've seen so far contains the requirement that the employee must have a specific number of contiguous hours of rest in every 24 hour period. We generate so called rest-shifts to ensure that the employees get their contiguous rest. For each day, we generate all possible rest-shifts starting within that day of length equal to the required rest. Each employee is then assigned to one such rest shift every day. The binary variables z_{ilk} determine if employee i E I is assigned to rest-shift l E L on day k E K, where L is the set of all possible contiguous timeslots of the required length, i.e. the rest-shifts.

The model also contains a number of variables to determine the soft constraint violations. These variables are called p^{α}_{γ} where α corresponds to the equation number where the penalty applies and γ denotes the indices over which the penalty variable is defined. The penalties are weighted, the constant c_{α} is the weight of penalty p^{α}_{γ}. One such penalty is a binary variable, while the other penalties are continuous.

The objective of the mixed integer model is to minimize the total weighted sum of the penalties that correspond to the soft constraint violations.

$$\min c_2 \sum_{t \in T} \sum_{k \in K} p2_{tk} + c_3 \sum_{t \in T} \sum_{k \in K} \cdots + c_9 \sum_{i \in I} p_i^9 + c_{10} \sum_{i \in I} p_i^{10} + c_{11} \sum_{i \in I} \cdots + c_{12} \sum_{i \in I} \sum_{k \in K} p_{ik}^{12} + c_{13} \sum_{i \in I} \sum_{t \in T} \sum_{k \in K} \cdots + c_{14} \sum_{i \in I} \sum_{k \in K} \cdots + c_{18} \sum_{i \in I} p^{18} \quad (1)$$

$$\text{s.t.} \sum_{i \in I} y_{itk} - p^2_{tk} \geq \text{demand}_{tk} \quad \forall t \in T, k \in K \quad (2)$$

$$y_{itk} < \text{demand}^{max}_{tk} + p3_{tk} \quad \forall t \in T, k \in K \qquad (3)$$

$$\sum_{i \in I}$$

$$X y_{itk} = \sum_{j \in shifts} x_{ijk} \quad \forall i \in I, t \in T, k \in K \quad (4)$$

$$y_{itk} < 1 - z_{ilk} \quad \forall i \in I, t \in T, l \in L, k \in K \text{ where } t \in L \quad (5)$$

$$\sum_{l \in L} z_{ilk} = 1 \quad \forall i \in I, k \in K \quad (6)$$

$$y_{itk} = 0 \quad \forall (i, t, k) \in NotAvailable \quad (7)$$

$$x_{ijk} = 0 \quad \forall (i, j, k) \in NotAvailableShift \quad (8)$$

$$\sum_{t \in T} \sum_{k \in K} y_{itk} > time^{min}_i - p^9_i \quad \forall i \in I \quad (9)$$

$$\sum_{t \in T} \sum_{k \in K} y_{itk} > time^{min}_i * (1 - p^{10}_i) \quad \forall i \in I \quad (10)$$

$$\sum_{t \in T} \sum_{k \in K} y_{itk} < time^{max}_i + p^{11}_i \quad \forall i \in I \quad (11)$$

$$\sum_{t \in T} y_{itk} < timeperday^{max}_{ik} + p^{12}_{ik} \quad \forall i \in I, k \in K \quad (12)$$

$$y_{itk} = 1 - p^{13}_{itk} \quad \forall (i, t, k) \in Requests \quad (13)$$

$$\sum_{j \in J} x_{ijk} = d_{ik} + p^{14}_{ik} \quad \forall i \in I, k \in K \quad (14)$$

$$x_{ijk} < d_{ik} \quad \forall i \in I, j \in J, k \in K \quad (15)$$

$$\sum_{\delta=0}^{d_{max}} d_{i(k+\delta)} < d_{max} \quad \forall i \in I, k \in \{K : k < |K| - d_{max}\} \quad (16)$$

$$d_{ik} < \omega_{iw_{max}} \quad \forall i \in I, 1 < w < |W|, k \in W_w \quad (17)$$

$$\sum_{\delta=0} \omega_{i(w+\delta)} < W^{max} + p^{18}_i \quad \forall i \in I, 1 < w < |W| - W^{max} \quad (18)$$

$$x_{ijk} \in \{0, 1\} \quad \forall i \in I, j \in J, k \in K \quad (19)$$

$$y_{itk} \in \{0, 1\} \quad \forall i \in I, t \in T, k \in K \quad (20)$$

$$z_{ilk} \in \{0,1\} \quad \forall i \in I, l \in L, k \in K \quad (21)$$

$$d_{ik} \in \{0,1\} \quad \forall i \in I, k \in K \quad (22)$$

$$w_{i\omega} \in \{0, 1\} \quad \forall i \in I, \omega \in W \quad (23)$$

$$p^{10}_i \in \{0,1\} \quad \forall i \in I \quad (24)$$

$$p^9_i, p^{11}_i, p^{18}_i, p^{12}_{ik}, p^{14}_{ik} > 0 \quad \forall t \in T, k \in K \quad (25)$$

$$p^{13}_{itk} > 0 \quad \forall i \in I \quad (26)$$

$$p^{1}_{i1k} > 0 \quad \forall i \in I, k \in K \quad (27)$$

$$p^{13}_{itk} > 0 \quad \forall i \in I, t \in T, k \in K \quad (28)$$

Constraints 2 and 3 handle the number of on-duty employees at all times, demand$^{min}_{tk}$ and demand^{max}

$_{tk}$ denote the minimum and maximum estimated demand for manpower during timeslot t in day k. The penalty variable p_{2tk} counts how many employees are needed to achieve the minimum number of

employees in each timeslot, while p^3_{tk} count how many employees are above the maximum number of required on-duty employees. By summing $p_{2\,tk}$ over all $t \in T$ and $k \in K$, we get the total number of man-hours that are understaffed, and similarly for the total number of overstaffed man-hours using p_{3tk}. We use constraint 4 to connect timeslots to shifts. Constraint 5 ensures that the employees are not working during their mandatory daily rest while constraint 6 ensures that each employees gets a rest period exactly once every day.

The availability of an employee is determined by various factors, such as contracts (e.g. no night shifts), vacations or requests for time off. We use constraints 7 and 8 and the sets NotAvailable and NotAvailableShift to limit the availability of employees. Constraints 9 and 11 make sure that the total working hours for each employee is within given limits, while constraint 10 is used to count how many employees are below the minimum required working hours over the planning period. Constraint 12 ensures that the number of working hours within a single day.

Employees sign up for shifts, but we translate those into timeslots, and use constraint 13 to figure out which requested timeslots are fulfilled. This method of fulfilling requests is the same as the one used in [6]. Constraints 14 and 15 are used to determine if an employee is working on a specific day, while the penalty associated with constraint 14 is used to determine if split shifts are allowed or not. Most employee contracts state the maximum number of consecutive days that the employee is allowed to work. Constraint 16 ensures that this is not violated. The constant d_{max} is the maximum allowed consecutive working days. Finally, constraint 17 is used to determine if an employee is working on a weekend while constraint 18 handles the maximum number of consecutive working weekends that are allowed, the constant W^{max} denotes the maximum number of consecutive working weekends.

Since the scheduling period is not isolated from the day to day running of the company or institution, we need to be careful with the boundary conditions, e.g. if an employee is working on the last shift before the start of the scheduling period, we cannot assign him/her to a shift at the start of the scheduling period. These boundary conditions can be encoded into the sets that we use to determine availability and preprocessed as a fixed assignment in the instances where an employee is working on a shift that straddles the boundary of the scheduling period.

3.1 Model limitations

One of the major limitations of the MIP model is the issue of fairness. The model does not include any notion of fairness to employees, so we might get a solution where no requests are granted for one employee while other employees have all their wishes fulfilled. We could improve the fairness, e.g. by adding a bound on the fraction of fulfilled requests, but that would also require additional penalties and weights.

Other limitation is that, if the workforce doesn't match the requirements so that the company or institution is forced to have understaffing or overstaffing, our partners would prefer if this is spread somewhat equally, instead of having a massive under- or overstaffing on a single shift and no problems at other times. Our model does not include any mechanism for leveling out any potential deviations. However, by adjusting the required manpower based on the workforce, this leveling could be achieved.

For some of the cases that we tried, the running time it took the solver to find an optimal solution was not within reasonable limits for the optimal solution. Instead of finding the optimal solution, we settle for a near optimal solution, i.e. a solution that is provably within some fraction of the optimal solution. Using our test cases, we found a solution that was within 1% of the optimal solution in a reasonable timeframe. By increasing the relative MIP gap, i.e. the difference between our solution and the bound on the optimal solution, we can speed up the solution time, but at the cost of a higher objective value. However, since a large part of the input, such as the manpower estimates, is often only based on a best guess it is debatable whether finding the optimal solution is actually worthwhile, especially if the time required is orders of magnitude larger than the time it takes to find a solution within 1%-5% of the optimum.

4 Experimental Results

To evaluate the performance of our algorithm, we use actual data from four companies and institutions. These companies and institutions include a nursing home, call centers and an airport service company. We will present the details of each problem instance and show examples of the preliminary schedule and the near-optimal solution. To get a better feeling for the complexity of the problem and the quality of the solutions, we also compare the near-optimal MIP solution to the solution of the local search algorithm introduced in [6]. The local search algorithm is designed to emulate the behavior of a typical staff manager. The local search algorithm iterates through multiple stages, with different objectives and different neighborhoods for all the stages, each of which is designed to emulate a specific action taken by the staff manager. For our examples, the scheduling period is usually 6 weeks, but we will plot the preliminary schedule and the improved schedule for only a single week for each problem instance. We tried to select a typical week for each instance. The data for the problem instances is available online [31]. Most of the instances include employees that have predefined or fixed schedules. For the purpose of the the MIP problem, we preprocess these employees, so they are not included in the MIP problem, and adjust all parameters such as staffing levels accordingly.

Figure 1 shows a log-scale of the relative MIP gap as a function of running time for two of the examples that we have. The problems were all solved using Gurobi 4.6.1 on a laptop with a 2.5 GHz Intel Core 2 Duo processor and 4 GB of memory, running Mac OS X 10.6. As can be seen from Figure 1, the

Table 1 Default penalty weights for soft constraint violations.

c2	c3	c9	c10	c11	c12	c13	c14	c18
15	2	10	100	10	10000	1	10000	10000

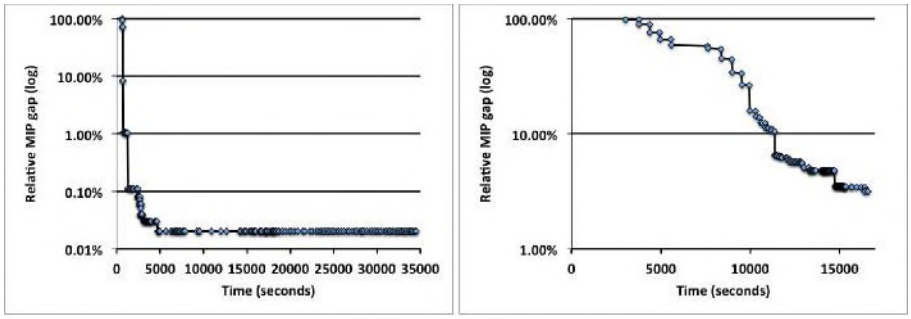

Fig. 1 The logarithm of the relative MIP gap for two instances as a function of the running time. These instances are of the two call centers from our datasets.

solver manages to find a solution within 1% of optimum after 21 minutes in the first instance. After running the solver for over 8.5 hours for the first instance, the solver still had not found an optimal solution. The second instance is significantly harder, there we were only below 10% after around 3 hours and around 3% after more than 4.5 hours. The easiest problem in our problem sets was however solved to optimality within 2 minutes. Since these schedules are usually created for periods of 4-6 weeks, a few hours of computation are usually acceptable. However, the difference between the two graphs in Figure 1 emphasizes the unreliability of the MIP approach with regards to the running time. When presented with a new instance, it can be difficult to guess if the solution will be ready in minutes or if we will have to wait a few hours until we have a good solution.

The default penalty weights that we used are shown in Table 1. The weights for maximum hours per day for an employee, more than one shift per day and consecutive working weekends is set very high, so these constraints are effectively hard constraints. Our partners prefer overstaffing to understaffing, so overstaffing has a penalty of 2 while understaffing carries a penalty of 15 for each timeslot. To minimize the number of people below their minimum duty hours, the corresponding penalty weight of 100 is relatively high, while the penalty for each timeslot over or under the duty hour limits carries a penalty of 10. Finally, the penalty for each unfulfilled requested timeslot carries a penalty of 1. By adjusting these penalties, we can tailor our model to different company cultures, e.g. put more emphasis on satisfying employee requests by increasing the corresponding penalty.

Using the notation introduced by De Causmaecker [15,16], we can describe the following problem instances as (AS|TVNO|PLGO).

Table 2 Results for the nursing home instance.

	Preliminary	Local Search	**MIP**
Scheduled hours	4669	5198	**4774**
Man-hours overstaffed	478	220	0
Man-hours understaffed	777	21	**109**
Employees below minimum duty hours	3	0	0
Requested hours granted		0.97	**0.98**

4.1 Problem instance: Nursing home

The first problem instance comes from a nursing home with 55 employees. The scheduling period is 6 weeks with 30 minute intervals. Each day contains of 18 shifts. The length of each shift ranges from 4 hours up to 12 hours. After preprocessing 5 employees with fixed schedules, we're left with 50 employees. If we sum up the total maximum working hours over the scheduling period, we get that the maximum number of hours that we can assign, without any overstaffing, is 5217 hours. However, the total duty hours of the employees is 5333 hours, so unless we violate the overstaffing constraint, we can never satisfy all duty hour requirements for the employees. Table 2 shows the soft constraint violations for the preliminary schedule, the local search algorithm from [6] and the near-optimal MIP solution.

The nursing home has the following hard constraints. An employee cannot work on more than 6 consecutive days, the maximum length of a shift is 9 hours while the minimum length of a shift is 4 hours. In any 24 hour period, each employee must get at least 8 consecutive hours of rest, while the maximum number of working hours in any 24 hour period is 9 hours. Initially, the problem had 1.204.482 rows, 351.032 columns and 3.378.221 nonzeros. After presolve the problem had 50.116 rows, 57.409 columns and 591.422 nonzeros. Gurobi managed to solve the problem to optimality within 2 minutes.

Figure 2 shows the preliminary schedule, the local search solution and the MIP solution for a typical week in the scheduling period. We see that the MIP solution has almost the same fraction of satisfied requests as the local search solution and the plan fits better into the manpower limits, although we're still left with a slight understaffing problem. All employees are within their duty-hours limits. We can see the slight understaffing in the MIP solution in Figure 2 over the last two days, i.e. the solution is a couple of employees short during the morning shift on the weekend.

4.2 Problem instance: Call center A

The second problem instance is the first of the two call centers in our datasets. Call center A has 92 employees and the scheduling period is 6 weeks in 30 minute intervals. After preprocessing, i.e. removing employees that have fixed

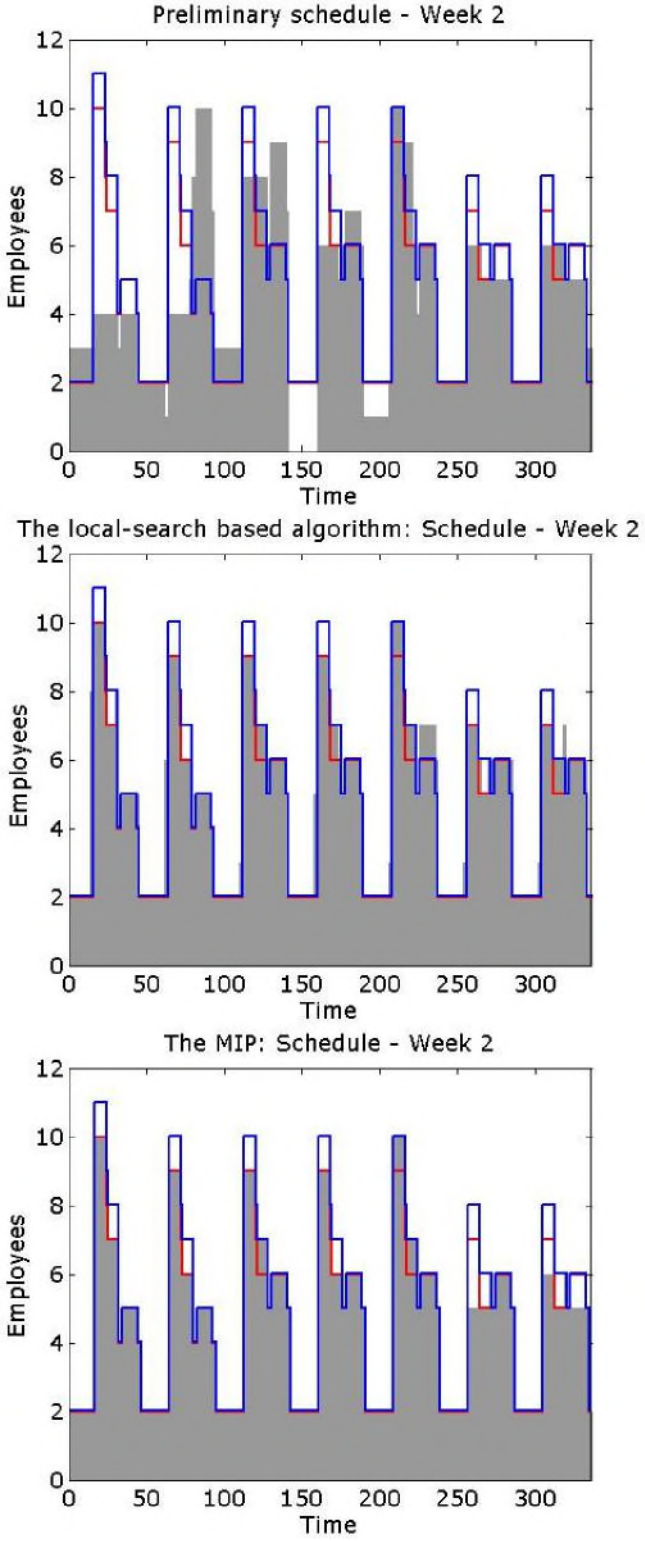

Fig. 2 Staffing levels for the nursing home problem instance. The gray area denotes the number of employees on duty while the two lines denote the minimum and maximum required staff on duty at each time.

Table 3 Results for the call center A problem instance.

	Preliminary	Local Search	**MIP**
Scheduled hours	9424	11920	**10578**
Man-hours overstaffed	390	791	**38**
Man-hours understaffed	1560	14	77
Employees below minimum duty hours	19	5	8
Requested hours granted		0.96	**0.79**

schedules, the number of employees drops down to 74. Each day consists of 97 shifts, whose lengths vary from 4 hours up to 11 hours. The total maximum required on-duty employees is 11582, while the total duty hours for all employees is 12054, so it will be impossible to satisfy both the duty hour constraints and the overstaffing constraints. The maximum number of consecutive working days is 6, the maximum number of working hours in each 24 hour period is 9 hours. In any 24 hour period, each employee must get at least 11 consecutive hours of rest. Table 3 shows the data from the preliminary schedule, the results of the local search and the results of the near-optimal MIP solution.

The MIP problem for call center A had 2.419.592 rows, 763.124 columns and 9.794.776 nonzeros. After presolve we were left with 118.445 rows, 267.630 columns and 3.277.415 nonzeros. Gurobi managed to solve the problem to within 1% of optimality within 22 minutes. The solution we show here is within 0.2% of optimum, after 8.5 hours of computations.

We see from Table 3 that the near-optimal MIP solution manages to satisfy the manpower requirements extremely well, with just a few hours of over/understaffing. However, this comes at the cost of satisfying employee requests, which is down do 79%. We could improve the request ratio by modifying the manpower requirements or decreasing the penalty of overstaffing. However, for these instances, we would recommend that the unfulfilled requests would be better handled manually, i.e. by allowing the staff manager to decide whether to accept a request or to have the number of on-duty staff within limits. The solution to the MIP problem along with a list of unsatisfied requests would make such a task relatively easy.

Figure 3 shows a typical week for call center A. We see that the preliminary schedule has problem with both under- and overstaffing. The local search puts emphasis on satisfying employee requests at the expense of overstaffing, while the near-optimal MIP solution focuses on the staffing levels while sacrificing a larger share of the employee requests.

4.3 Problem instance: Call center B

Call center B does the planning for only 4 weeks in advance, specifying exactly how many should be on duty in every 15 minute interval. The total number of employees at call center B is 62, which, after preprocessing drops down to 46.

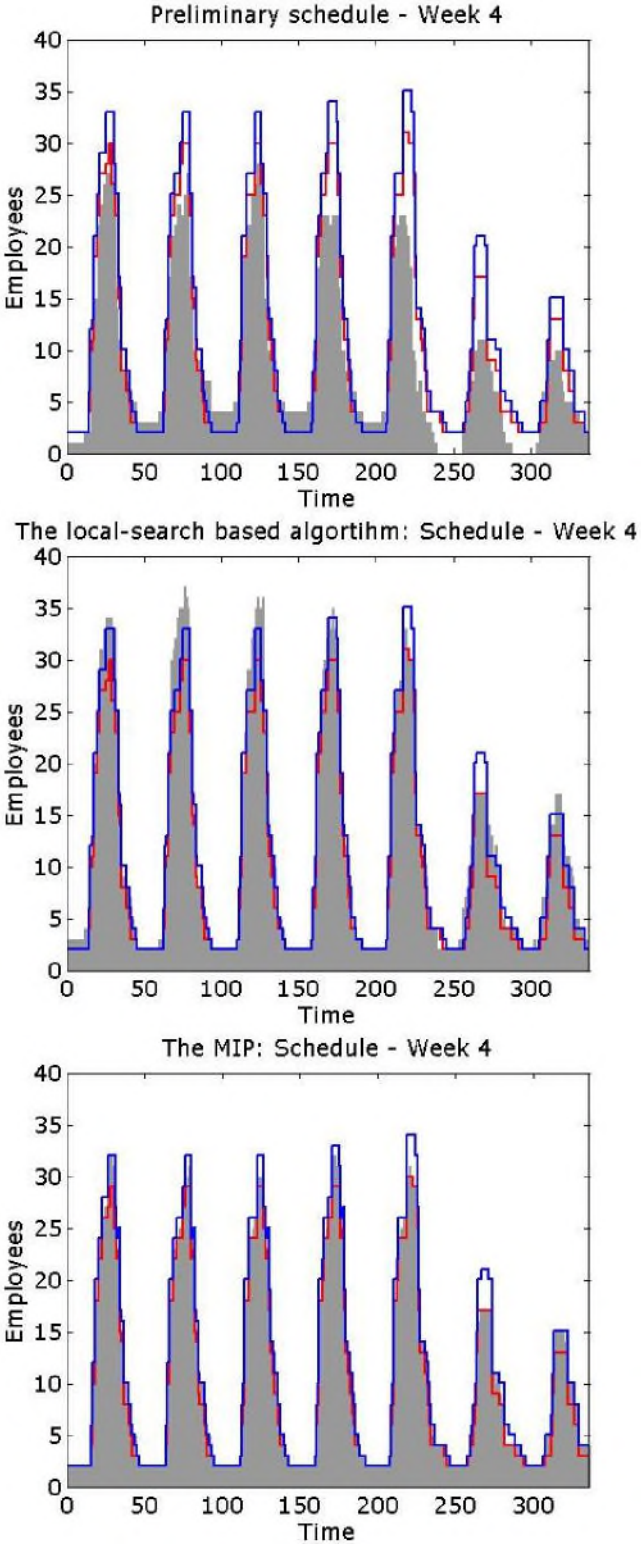

Fig. 3 Staffing levels for call center A. The gray area denotes the number of employees on duty while the two lines denote the minimum and maximum required staff on duty at each time.

Table 4 Results for the call center B problem instance.

	Preliminary	Local Search	**MIP**
Scheduled hours	6623	7554	**6071**
Man-hours overstaffed	795	609	**234**
Man-hours understaffed	1306	189	**37**
Employees below minimum duty hours	15	7	17
Requested hours granted		0.86	**0.65**

The call center is overstaffed, the total available man-hours for the scheduling period is 8134 hours while the total required man-hours over the same period is 7134 hours. There are 115 possible shifts for each day. The hard constraints that must be satisfied for call center B are that employees cannot be working on more than 6 consecutive days, in every 24 hour period there must be at least 11 consecutive hours of rest and at most 11 hours of work. The maximum length of a shift is 11 hours while the length of a shift must be at least 4 hours.

The MIP problem consists of 5.110.509 rows, 585.322 columns and 10.399.996 nonzeros. After presolve, the problem has 126.839 rows, 218.628 columns and 4.598.686 nonzeros. Call center B was by far the most challenging instance that we tried. The Gurobi solver didn't find a feasible solution until after 50 minutes and we had to wait for more than 3 hours before the value of solution was within 10% of optimum.

Table 4 shows the results for the local search procedure, the near optimal MIP solution as well as the preliminary schedule.

Figure 4 shows the staffing levels for a typical week. The three figures contain the preliminary schedule, the local search solution and the solution of the MIP problem. The required staffing level for the MIP solution has a different shape than the other two due to the fixed assignments that were preprocessed in the MIP solution, but kept as a part of the input in the preliminary schedule and the local search solution.

Since there is no flexibility in the staffing level limits, both algorithms have some problems with both under- and overstaffing. As seen in Figure 4 and Table 4, the near-optimal MIP solution manages to stay close to the required staffing levels, but at the expense of employee requests. Since the call center is overstaffed, it is not surprising that both solutions have some employees that are below the minimum required duty hours as well as overstaffing problems.

4.4 Problem instance: Airport ground service.

The fourth problem instance is an airport ground service company. The scheduling period is six weeks in 30 minute intervals. The demand for on-duty employees depends on the flight schedules at the airport. In this particular instance, there are many flights that leave during the early morning, and then there is another concentration of flights in the afternoon. Since there are almost no

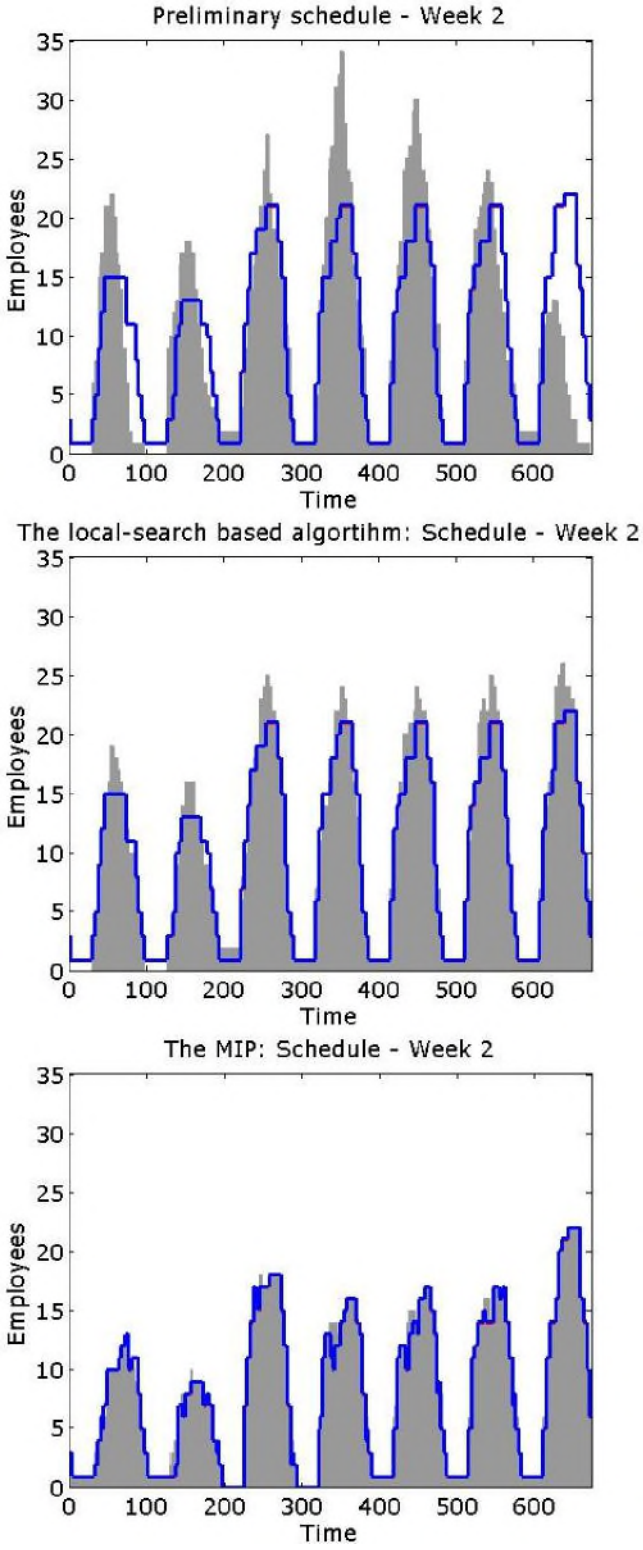

Fig. 4 Staffing levels for call center B. The gray area denotes the number of employees on duty while the solid line denotes the exact number of employees that are required to be on duty at each time.

Table 5 Results for the airport ground services problem instance.

	Preliminary	Local Search	**MIP**
Scheduled hours	3997	6670	**6608**
Man-hours overstaffed	192	641	**50**
Man-hours understaffed	2464	517	11
Employees below minimum duty hours	23	0	0
Requested hours granted		0.94	**0.89**

flights scheduled at any time apart from the morning and afternoon busy periods, the requirements for employees peaks during the two busy periods but drops sharply during other times. Each day contains 53 different shifts and the airport ground service has 53 employees, one of which has a fixed schedule. Due to the structure of the manpower requirements, the employees often work a short morning shift and then another short afternoon shift with a few hour break in-between. This means that split shifts are allowed so we change the weight c_{14} to zero. This problem instance is understaffed compared to the previous examples, here the total maximum required man-hours is 8152 hours over the scheduling period while the available man-hours is only 6350. Table 5 shows the preliminary schedule, the results of the local search method and the near-optimal MIP solution.

The hard constraints for the airport ground service are that there must be a minimum continuous rest of 11 hours in any 24 hour period, each employee can not work more than 5 consecutive days, employees cannot work more than 12 consecutive hours while the number of working hours in any 24 hour period is also 12 hours.

The MIP problem consists of 1.716.477 rows, 441.352 columns and 5.342.624 nonzeros. After presolve, the problem has 113.047 rows, 170.851 columns and 2.132.307 nonzeros. It took Gurobi around 25 minutes until the value of solution was within 1% of optimum.

As we can see in Figure 5, the near-optimal MIP solution manages to fit the staffing levels almost perfectly, while still satisfying the majority of the employees requests. For comparison, the local search method has a slightly higher ratio of fulfilled requests, but both the over- and understaffing is at least an order of magnitude larger than for the MIP solution.

5 Conclusions

In this paper we have introduced a mixed-integer programming (MIP) formulation to create a high quality feasible staff schedule from a partial staff schedule based on requests from employees. The results from the four different companies and institutions show that it is possible to find high quality schedules in a reasonable amount of time by using mixed integer programming. The data for these problem instances is available online [31]. Our MIP model al-

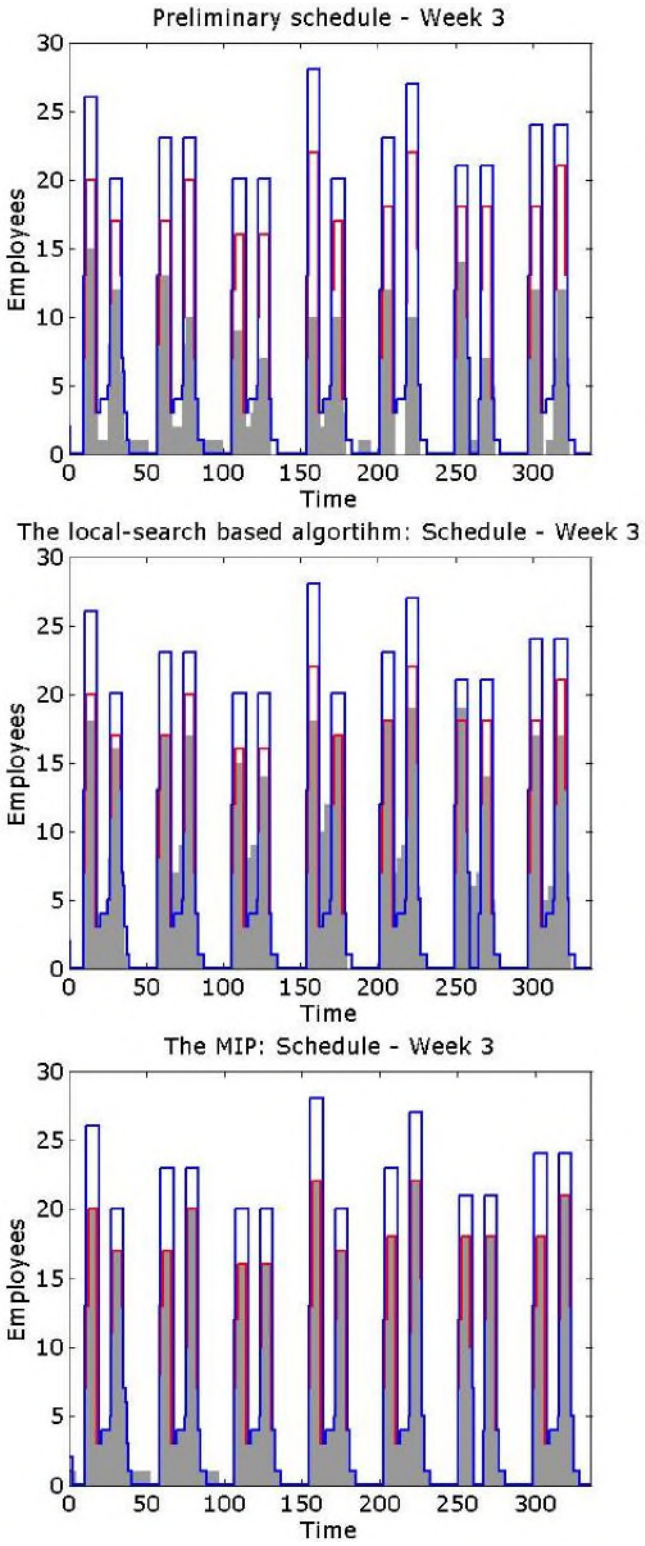

Fig. 5 Staffing levels for the airport ground services. The gray area denotes the number of employees on duty while the two lines denote the minimum and maximum required staff on duty at each time.

lows flexibility in terms of shifts lengths and shifts starting times, as well as handling all the constraints and requirements of our real-life instances. Due to time constraints, we only solved one of the four instances to optimality. The other three were solved until we had a solution that was within 1% of optimum for the easier problems and within 10% for the most difficult problem. The longest time it took for such a solution was around 3-4 hours, which, is acceptable in these cases, since these scheduling problems are only solved every 4-6 weeks.

We also compared the near-optimal solution to a solution from a local search based method [6]. The results show that the MIP solution does an extremely good job of keeping the number of on-duty personnel within the required limits, at the expense of employees requests, while the local search had more emphasis on satisfying requests. For the easier problems, the local search method does quite well, so that there is not much of a difference between the solutions. However, when the problems became more difficult, with complicated shift structures, and in one case, split shifts, the MIP solver produced solutions that were vastly superior to the local search solutions.

References

1. Ahmad, J., Yamamoto, M., Ohuchi, A., Evolutionary Algorithms for Nurse Scheduling Problem. Proceedings of CEC00, San Diego, 196–203 (2000).
2. Aickelin, U., Dowsland, K., Exploiting problem structure in a genetic algorithm approach to a nurse rostering problem. Journal of Scheduling 3(3), 139–153 (2000).
3. Alfares, H.K., Survey, categorization and comparison of recent tour scheduling literature. Annals of Operations Research 127, 145–175 (2004).
4. Alsheddy, A., Edward, T., Empowerment scheduling for a field workforce. Journal of Scheduling, 1–16 (2011).
5. Al-Yakoob, S. M., Sherali, H. D., A column generation approach for an employee scheduling problem with multiple shifts and work locations. Journal of the Operational Research Society, 59(1), 34–43 (2008).
6. Ásgeirsson, E. I., Bridging the gap between self schedules and feasible schedules in staff scheduling, Annals of Operations Research (2012)
7. Ásgeirsson, E.I., Kyngäs, J., Nurmi, K. and Stølevik, M., A Framework for Implementation-Oriented Staff Scheduling, In Proc of the 5th Multidisciplinary Int. Scheduling Conf.: Theory and Applications (MISTA), Phoenix, USA, (2011).
8. Bailyn, L., Collins, R., Song, Y., Self-scheduling for hospital nurses: an attempt and its difficulties, Journal of Nursing Management, 15(1),72–77 (2007).
9. Bard, J. F., Purnomo, H. W., Preference Scheduling for Nurses Using Column Generation. European Journal of Operational Research, 164, 510–534 (2005).
10. Bartholdi, J.J., A Guaranteed-Accuracy Round-off Algorithm for Cyclic Scheduling and Set Covering. Operations Research 29, 501–510 (1981).
11. Brusco, M. J., Jacobs, L. W., Cost analysis of alternative formulations for personnel scheduling in continuously operating organisations. European Journal of Operational Research, 86, 249–261 (1995).
12. Burke, E. K., De Causmaecker, P., Vanden Berghe, G. (1999). A Hybrid Tabu Search Algorithm for the Nurse Rostering Problem. SEAL'98, LNCS 1585, 187–194.
13. Burke, E. K., De Causmaecker, P., Petrovic, S., Vanden Berghe, G., Variable neighborhood search for nurse rostering problems. Metaheuristics: computer decision-making, 153–172 (2004).
14. Burke, E. K., De Causmaecker, P., Vanden Berghe, G., Van Landeghem, H., The State of the Art of Nurse Rostering. Journal of Scheduling, 7, 441–499 (2004).

15. De Causmaecker, P., Vanden Berghe, G. (2011). Towards a reference model for timetabling and rostering. Annals of Operations Research.
16. De Causmaecker, P., Vanden Berghe, G., A categorisation of nurse rostering problems. Journal of Scheduling, 14, 3–16 (2011).
17. Dowsland, K., Nurse scheduling with Tabu Search and Strategic Oscillation. European Journal of Operations Research, 106, 393–407 (1998).
18. Easton, F., Mansour, N., A Distributed Genetic Algorithm for Employee Staffing and Scheduling Problems. Conference on Genetic Algorithms, San Mateo, 360–367 (1993).
19. Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D., An Annotated Bibliography of Personnel Scheduling and Rostering. Annals of Operations Research, 127, 21–144 (2004).
20. Ernst, A. T., Jiang, H., Krishnamoorthy, M., Sier, D., Timetabling and Rostering. European Journal of Operational Research, 153(1), 3–27 (2004).
21. Garey, M.R., Johnson, D.S. ,Computers and Intractability. A Guide to the Theory of NP-Completeness, Freeman (1979).
22. Hung, R., Improving Productivity and Quality through Workforce Scheduling. Industrial Management, 34(6) (1992).
23. Jeroen, B., Demeulemeester, E., On the trade-off between staff-decomposed and activity-decomposed column generation for a staff scheduling problem. Annals of Operations Research, 155(1), 143–166 (2007).
24. Kellogg D.L., Walczak S., Nurse Scheduling: From Academia to Implementation or Not. Interfaces 37(4), 355–369 (2007).
25. Lau, H. C., On the Complexity of Manpower Shift Scheduling. Computers and Operations Research 23(1), 93-102 (1996).
26. Meisels, A., Schaerf, A., Modelling and solving employee timetabling problems. Annals of Mathematics and Artificial Intelligence 39, 41–59 (2003).
27. Rowland, H. S., Rowland, B. L., Nursing administration handbook 4th ed., Gaithersburg, Maryland, (1997).
28. Tanomaru, J., Staff Scheduling by a Genetic Algorithm with Heuristic Operators. Proceedings of CEC95, 456–461 (1995)..
29. Tien J, Kamiyama A., On Manpower Scheduling Algorithms. SIAM Rev. 24 (3), 275–287 (1982).
30. van der Veen E., Veltman B., Rostering from staffing levels: a branch-and-price approach. Proceedings of the 35th International Conference on Operational Research Applied to Health Services (ORAHS), 1–10 (2009).
31. Datasets for experimental results: http://www.ru.is/kennarar/eyjo/staffscheduling.html