European Journal of Operational Research 209 (2011) 37–50

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Discrete Optimization

An integer linear programming-based heuristic for scheduling heterogeneous, part-time service employees

Mehran Hojati *, Ashok S. Patil

Edwards School of Business, University of Saskatchewan, Saskatoon, Canada S7N 5A7

ARTICLE INFO

Article history:
Received 3 May 2008
Accepted 1 September 2010
Available online 7 September 2010

Keywords:
Scheduling
Assignment
Heuristics
Integer programming
Rostering

ABSTRACT

Scheduling of heterogeneous, part-time, service employees with limited availability is especially challenging because employees have different availability and skills, and work different total work hours in a planning period, e.g., a week. The constraints typically are to meet employee requirements during each hour in a planning period with shifts which have a minimum & maximum length, and do not exceed 5 work days per week for each employee. The objectives typically are to minimize over staffing and to meet the target total work hours for each employee during the planning period. We decompose this problem into (a) determining good shifts and then (b) assigning the good shifts to employees, and use a set of small integer linear programs to solve each part. We apply this method to the data given in a reference paper and compare our results. Also, several random problems are generated and solved to verify the robustness of our solution method.

1. Introduction

We consider the scheduling of heterogeneous, part-time employees of a service organization, e.g., a restaurant, bank, call center, or a large retail store. There is usually a pool of active employees who are available, but only during part of the time on some days, and only have some of the skills (i.e., can perform only some of the tasks). Also, they tend to work different total work hours in the planning period (e.g., a week). The demand (work load) typically fluctuates throughout a day and across days, although it follows similar patterns, such as lunch or dinner peaks.

There are typically three steps in performing employee scheduling for a service organization. The first step is to forecast the demand (or hourly sales) for each working hour of each day during the next planning period (e.g., a week). (Some services may use half-hour or quarter-of-hour as the unit time interval). The second step is to translate the demand (or hourly sales) forecasts into employee requirements for each working hour of the planning period in order to provide a given customer service level. The third step is to meet the employee requirements for each working hour of the planning period by determining the work days, and a shift on each work day, for each employee. There are usually some government employment standards which must be met, such as a maximum length for a shift and a minimum number of days off in a week. In addition, there are some organizational policies/practices. In this paper, for illustration, we use the general practice of most McDonald's restaurants: a shift should be between 3 and 8 hours long; the number of work days in a week should be limited to 5, but the work days need not be consecutive.

For simplicity, in the remainder of this paper we consider a week as the planning period, but the results can be applied to other intervals, e.g., 2 weeks or a month.

The major objectives of weekly employee scheduling are (a) to cover the hourly employee requirements during the week with a minimum number of total employee hours (i.e., to minimize over staffing), and (b) to meet the target weekly work hours for each employee. The latter is determined either by agreement between the manager and the employee or is based on a fair share scheme, e.g., as a percentage of total employee availabilities. It is reasonable to assume that the sum of target weekly work hours for all employees equals the sum of hourly employee requirements during the week.

This employee scheduling problem is hard to solve optimally: it requires solving a very large integer linear program. Unfortunately, there are not many references in the literature that solve this problem. Of the methods available, to the best of our knowledge none can guarantee a close-to-optimal feasible solution in a reasonable amount of time.

* Corresponding author. Tel.: +1 306 966 8428; fax: +1 306 966 2515.
E-mail address: Hojati@Edwards.usask.ca (M. Hojati).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.09.004

In this paper, our objective is to determine a simple and practical solution method (i.e., one which can be solved on a personal computer using a spreadsheet software such as Excel) for scheduling heterogeneous, part-time service employees with limited availability. We will decompose the problem into (a) determining good shifts and (b) assigning the shifts to the employees, and use a set of small integer linear programs to solve each part. We apply this method to the data given in Loucks (1987) and Loucks and Jacobs (1991), and compare our results. Also, we generate several random problems and test our solution method.

The remainder of this paper is as follows. Section 2 provides literature review. The problem is formulated as a large integer linear program in Section 3. Section 4 contains our solution method, which is applied to a specific problem in Section 5. Section 6 describes how computational experiments were performed to validate our solution method, whereas Section 7 describes how our method can be applied to some more general environments. Finally, Section 8 concludes the paper.

2. Literature review

As it can be seen in the review paper by Ernst et al. (2004), there are only a limited number of articles on weekly (or "tour") scheduling of heterogeneous, part-time employees. These are reviewed below. The model used in Loucks (1987) and Loucks and Jacobs (1991) is identical to ours, while their solution technique differs from ours as follows: Instead of decomposing the problem into (a) determining good shifts and (b) assigning them to employees, they tried to solve the whole problem in two phases. They focussed on an hour, instead of a shift, as the unit of work interval. In the first phase, called the construction phase, the task-hour with the lowest number of available and eligible employees minus requirements is identified and then an available and eligible employee is chosen according to a set of rules and is assigned to it (the maximum 5 work days constraint is relaxed in this phase). If an available and eligible employee cannot be found, an exchange mechanism is used between two employees. In the second phase, called the improvement phase, any excess above the 5 workdays is eliminated by assigning the excess shift to other employees. Also, attempts are made to reduce over staffing, deviations from target weekly work hours for each employee, and the changes in tasks within multi-segmented shifts (each segment involving a different task). This method produces excessive number of multi-segmented shifts. Also, it may not find a feasible solution (i.e., one or more shifts may be left unassigned at the termination of the solution method).

Thompson (1988)'s model differs from ours as follows: There was no distinction made between different skills, i.e., the requirements were aggregated across skills. Also, there was no target for weekly work hours of each employee, but a distinction was made between full time and part-time employees. Another difference is that a quarter-of-an-hour was used as the unit interval of time, and that half-an-hour of break time for 5 hours shifts and one hour of meal break for 6 to 8 hours shifts were scheduled. Two different methods were used.

In Method 1, Thompson (1988) first determined the days-on (a maximum of 5 days) for each employee in order to meet the daily employee requirements and to maximize the total daily availability of employees. Then, shifts were constructed for each day using three different techniques: (a) an iterative adding and dropping heuristic, (b) a linear program which considers the work breaks explicitly, and (c) a linear program without work breaks, followed by a break insertion method. In both linear programming techniques (b) and (c), availability of employees are taken into account by determining the intersection of availability periods (called regions) of every subset of employees and making sure that the number of shifts in each region is not larger than the number of employees available in the region. Finally, shifts are assigned first to full time employees using a ranking procedure, and then to part-time employees as follows: (i) find the day with least total availability hours, (ii) solve the linear programming formulation of the standard assignment problem with the objective of assigning those employees who have least availability outside of the day, and (iii) continue until all days are considered.

In Method 2, Thompson (1988) first determined the shifts for each day using the three techniques given in Method 1 above. Then, the day with most shift hours still unassigned is chosen. Next, a shift with least number of employees available during it is chosen on that day. Finally, the employee with closest fit of availability and the (chosen) shift hours (whose number of work days does not exceed 5) is assigned the shift. It is not clear if either method guarantees a feasible solution.

Glover and McMillan (1986) considered a more general employee scheduling problem where the unit of work interval is the set of possible shifts (called a tour) for an employee during one week. They used minimum and maximum weekly work hours for each employee instead of target weekly work hours. The use of tours increases the number of variables substantially (billions). Therefore, they used Tabu Search to solve their problem. The authors report good solutions for most of the problems they tested. However, like other search heuristics, the performance of Tabu search cannot be guaranteed.

Love and Hoey (1990) considered employee scheduling in four Burger King restaurants. Their model differed from ours as follows: They used a half-hour as unit interval of time. Instead of a target for weekly work hours for each employee, they used a target for the number of days to work for each employee. Each number-of-days-to-work-per-week by each employee had a coefficient in the objective function. These coefficients also prevented the employee from being assigned to more than 5 work days. Also, the availability and skill of each employee was represented as coefficients of the shifts being assigned to the employees in the objective function. They formulated and solved each of (a) determining shifts and (b) assigning them to employees as a network flow problem. However, problem (b) was very large and needed a special network flow solution method and software.

Gopalakrishnan et al. (1993) considered the problem of determining part time employee requirements and scheduling them in the distribution department of a newspaper publisher. The start times of the 21 shifts in a week were fixed, but their lengths varied between 4 and 8 hours. Workers had different availabilities and preferences for shifts. Each worker was guaranteed a minimum of 24 hours a week. The solution method used by the authors was a two phase heuristic: in phase 1 each worker, starting from most senior, received close to but not exceeding 24 hours of work per week. If the requirement of some shifts was still not satisfied, they were assigned to employees, again starting from most senior, up to the maximum hours determined for that employee. The authors tested this simple greedy heuristic only on 3 very small problems. Our problem deals with a more general situation where the employees and shifts have different skills and skill requirements. Also, we consider the more general case where each employee may have a different target weekly work hours.

Litchfield et al. (2003) considered a restaurant whose employees had the same skills. The shifts were 6 hours long and pre-determined. Hence, there was no need to determine good shifts first. They used a minimum and a maximum number of shifts per week for each employee rather than target weekly work hours. The integer linear program (ILP) of the assignment-of-shifts-to-employees had approximately 600 binary variables and 1,400 constraints. This is normally solvable by a good ILP software. However, the authors decided to use a two phase method: (a) a construction heuristic and (b) a Tabu Search improvement method. The construction phase is as follows: at each step, the shift with least number of available employees is assigned to the employee who has the least number of shifts available to him/her. The solution method was programmed in Excel using Visual Basic for Applications.

Eveborn and Ronnqvist (2004) used the tours of work for employees during one or more weeks as the unit work interval. In addition to cost of over staffing and under staffing, they minimized the penalty of the worst tour among the employees (this penalty includes cost of receiving a different weekly work hours than the target for the employee). Given that the formulated integer linear program is very large, they first generated one feasible tour for each employee, and then improved the solution by generating more feasible tours as needed (a column generation method). The solution procedure appears complex and may take some time before a good solution is obtained. For example, for the six test problems considered, after 20 minutes the average gap between the best solution obtained and its lower bound was 28%.

Gordon and Erkut (2004) considered scheduling gate volunteers for the Edmonton Folk Festival which occurs annually over a 4 days period. This is a smaller version of our problem and therefore solvable by an integer linear programming solver. Most shifts are determined by coordinators while a small set-covering model determines the shifts during the weekend days. The assignment of volunteers to the determined shifts is formulated as an integer linear program and solved by Premium Solver in Excel. Each volunteer works 20 hours during the 4 days and his/her availability is reflected in his/her preference for each shift which is maximized over all volunteers. The constraints include restrictions on morning & afternoon shifts and percentage requirements for having experienced and skilled volunteers on each shift.

Zolfaghari et al. (2007) considered a very general version of our problem, and formulated it as a large integer linear programming problem. The formulation contains both hard and soft constraints, and the violations of the latter are reflected in the objective function as penalties. However, they do not solve this problem, but only propose various heuristics for reducing the number of shifts that should be used under various demand scenarios. These heuristics are not needed if the problem is decomposed, as we have done in this paper, into (a) determining good shifts for each skill & day and then (b) assigning these good shifts to the employees. The reason is that the size of each sub-problem will be small enough in our method.

The solution methods used in the above articles show the diversity of methods that can be used to solve our employee scheduling problem. For a classification of these and other methods, see Alfares (2004).

There are several workforce management software packages that claim to perform optimized employee scheduling. Some of these, having revenues of more than $5 million per year, were reviewed in Degnan Manning et al. (2006). Although most of these software claim to use ILP and/or search heuristics, the details of their solution method are unknown due to trade secrecy. Also, it is not clear if they use targets for weekly work hours for each employee as an objective. ESP by Thoughtworks Inc is a specialized employee scheduling software for fast food restaurants (used by McDonald's restaurants in Canada). It uses linear programming for determining preferred shifts and Simulated Annealing for assigning these shifts to the employees.

None of the above references explicitly consider the objective of target weekly work hours for each employee and actually solve the problem. Also, using a tour as the unit work interval makes the problem more complicated, without adding any advantages, because tours for part-time employees are very flexible (the shifts do not have to start at the same time each work day).

3. Problem formulation

As with most of the methods reviewed above, we will use the task-shift, as opposed to task-hour or a tour, as our unit of work interval. This will make the problem easier to solve. Let

h = index of working hours during a day, e.g., $h = 1,\ldots,18$
d = index of days in a week, $d = 1,\ldots,7$
t = index of tasks/skills, e.g., t = Grill (Gr), Drive Through (DT), Counter (Co)
e = index of employees, e.g., $e = 1,\ldots,40$
s = index of possible shifts (within a day), e.g., $s = 1,\ldots,81$, where $s = 1$ consists of hours (1,2,3), $s = 2$ consists of hours $(1,\ldots,4)$, etc. We assume shifts are contiguous, with no work breaks.
y_{std} = number of employees scheduled on shift s for task t on day d
$a_{hstd} = 1$ if hour h is in shift s for task t on day d; 0 otherwise
r_{htd} = number of employees needed during hour h for task t on day d
$x_{estd} = 1$ if shift s for task t on day d is assigned to employee e; 0 otherwise
$b_{estd} = 1$ if employee e is available and eligible to perform on shift s for task t on day d; 0 otherwise
hrs_s = number of hours in shift s (i.e., its duration)
$Trgt_e$ = number of hours per week targeted for employee e; we assume $\sum_e Trgt_e = \sum_h \sum_t \sum_d r_{htd}$
λ_e^u = number of hours that the assigned hours to employee e is under the target $Trgt_e$
λ_e^o = number of hours that the assigned hours to employee e is over the target $Trgt_e$

The weekly heterogeneous, part-time service employee scheduling problem with limited availability can be formulated as the following bi-criteria integer linear program:

$$\text{Minimize} \quad \sum_d \sum_t \sum_s hrs_s \cdot y_{std} \tag{1}$$

$$\text{Minimize} \quad \sum_e \lambda_e^u + \sum_e \lambda_e^o \tag{2}$$

$$\text{Subject to :} \quad \sum_s a_{hstd} \cdot y_{std} \geqslant r_{htd} \quad \text{for each hour } h, \text{task } t, \text{day } d, \tag{3}$$

$$\sum_e b_{estd} \cdot x_{estd} = y_{std} \quad \text{for each shift } s, \text{task } t, \text{day } d, \tag{4}$$

$$\sum_s \sum_t b_{estd} x_{estd} \leqslant 1 \quad \text{for each employee } e, \text{day } d, \tag{5}$$

$$\sum_s \sum_t \sum_d b_{estd} \cdot x_{estd} \leqslant 5 \quad \text{for each employee } e, \tag{6}$$

$$\sum_s \sum_t \sum_d hrs_s \cdot b_{estd} \cdot x_{estd} + \lambda_e^u - \lambda_e^o = Trgt_e \quad \text{for each employee } e, \tag{7}$$

$$y_{std} = \text{integer} \quad \text{for each shift } s, \text{task } t, \text{day } d, \tag{8}$$

$$x_{estd} = \text{binary} \quad \text{for each employee } e, \text{shift } s, \text{task } t, \text{day } d, \tag{9}$$

$$\lambda_e^u,\ \lambda_e^o \geqslant 0 \quad \text{for each employee } e. \tag{10}$$

Objective (1) minimizes the total employee hours used (which is equivalent to minimizing over staffing). Objective (2) minimizes the total deviation from target weekly work hours for the employees. For each task and day, Constraint (3) requires that for each hour, the total number of employees on all shifts which include the hour meet or exceed the employee requirements for the hour. For each task and day, Constraint (4) ensures that each shift is assigned to the right number of available and eligible employees. Constraint (5) is the requirement that each employee may only work one shift per day. Constraint (6) is the requirement that each employee can work a maximum of 5 days a week. Constraint (7) ensures that total weekly work hours assigned to each employee plus any underage minus any overage equals the target weekly work hours for the employee. Constraint (8) ensures that a whole number is used for the number of employees scheduled for each shift. Constraint (9) ensures that the variables assigning each shift to each employee are zero/one. Finally, Constraint (10) ensures that the deviations from the target weekly work hours for each employee are non-negative.

In order to be able to use a single-objective integer linear programming software to solve this problem, we replace objectives (1) and (2) with the "weighted" sum of the two:

$$\text{Minimize} \ \sum_d \sum_t \sum_s hrs_s \cdot y_{std} + \alpha \left(\sum_e \lambda_e^u + \sum_e \lambda_e^o \right) \tag{11}$$

By varying α, one can generate different Pareto-optimal solutions. We will use $\alpha = 1$ in this paper.

The above model is very large. For example, for 40 workers, 6 a.m. to 12 a.m. operating hours, 3 to 8 hours shifts, and typical density of matrix b_{estd}, there will be approximately 40,000 binary & integer variables and 4,000 constraints. This problem cannot be solved in a reasonable amount of time on any personal computer. We entered the problem given in (Loucks, 1987 and Loucks and Jacobs, 1991) in CPLEX using the MPL modeling language. After 9 hours of computation, we received a memory overflow error message.

As it may be seen from the coefficient matrix of Constraints (3)–(7), this matrix is sparse. Note that most of the rows are due to Constraints (4), relating y_{std} to x_{estd}. Therefore, it is reasonable to decompose the problem into sub-problems (a) determining good shifts and then (b) assigning them to the employees. The transition between (a) and (b) can be programmed using e.g., Excel's Visual Basic for Applications. Sub-problem (a) is presented in Section 4.1 and sub-problem (b) in Section 4.2 below.

4. Our solution method

4.1. Determining good shifts

Despite the decomposition, we will not totally ignore the employees in determining good shifts. To do so may result in shifts that cannot be assigned to any employee. Therefore, for each possible shift, we will determine how many employees are available and eligible for it, and choose those shifts that maximize the total number of available and eligible employees as well as minimize overstaffing.

Also, we will limit the number of employee-shifts $(\sum_s y_{std})$ for each task on each day, because the total number of employee-shifts should be less than five times the number of employees or else the problem will be infeasible.

We will use the following three step solution method:

1. Determine a target number of employee-shifts for each task on each day. We assume that an average employee will work 4 days a week. Thus, for say 40 employees, the total target number of employee-shifts will be 160 per week. We will split the 160 shifts in proportion to total employee hours needed for each task on each day, i.e., $\sum_h r_{htd}$.
2. Determine the number of employees available and eligible for each shift.
3. Formulate an integer linear program for each task on each day and solve it. First let
   $Trgt_{td}$ = target number of employee-shifts for task t on day d
   E_{std} = number of employees available and eligible for shift s, task t, day d
   λ_{td}^u = number of employee-shifts under the target $Trgt_{td}$
   λ_{td}^o = number of employee-shifts over the target $Trgt_{td}$
   Solve the following integer linear program for each task t on each day d:

$$\text{Minimize} \quad \sum_s hrs_s \cdot y_{std} + \delta\lambda^o_{td} - \varepsilon \sum_s E_{std} \cdot hrs_s \cdot y_{std} \tag{12}$$

$$\text{Subject to :} \quad \sum_s a_{hstd} \cdot y_{std} \geqslant r_{htd} \qquad \text{for each hour } h, \tag{13}$$

$$\sum_s y_{std} + \lambda^u_{td} - \lambda^o_{td} = Trgt_{td} \tag{14}$$

$$y_{std} = \text{integer} \qquad \text{for each shift } s, \tag{15}$$

$$\lambda^u_{td},\ \lambda^o_{td} \geqslant 0. \tag{16}$$

The objective function (12) is a variation of (1) with two other terms added: (i) the second term $\delta\lambda^o_{td}$ minimizes the number of employee-shifts over the target, which will allow shift overage but tries to keep it to a minimum, (ii) the third term, $-\varepsilon\sum_s E_{std} \cdot hrs_s \cdot y_{std}$, tries to maximize the selection of those shifts with larger number of available and eligible employees. The coefficients δ and ε are small numbers which are determined by trial and error. We used $\delta = 2$ and $\varepsilon = .01$.

The above integer linear programs are easy to solve, because their linear programming relaxations have mostly integer optimal solutions (due to the consecutive 1s in the columns of Constraints 13). Also, the number of variables is small enough so that Excel's standard Solver can be used. For example, if shifts start and end on the hour, are between 3 & 8 hours long, and daily operating hours are between 6 a.m. and 11 p.m., there will be 83 variables. The computation time for each integer linear program is approximately a second.

4.2. Assigning the good shifts to the employees

We could formulate this problem as an integer linear program (ILP) as well. First, let $\{\hat{s}\}$ = the subset of shifts with positive optimal values for y_{std}, obtained from solving (12)–(16) in Section 4.1 above. If for any shift $\hat{s}$ the optimal $y_{\hat{s}td} = n \geqslant 2$, then n copies of the shift $\hat{s}$ are produced and used. This will make the assignment problem easier to solve.

Solve the following integer linear program:

$$\text{Minimize} \quad \sum_e \lambda^u_e + \sum_e \lambda^o_e \tag{17}$$

$$\text{Subject to :} \quad \sum_e b_{e\hat{s}td} \cdot x_{e\hat{s}td} = 1 \qquad \text{for each shift } \hat{s},\ \text{task } t, \text{day } d, \tag{18}$$

$$\sum_s \sum_t b_{e\hat{s}td} \cdot x_{e\hat{s}td} \leqslant 1 \qquad \text{for each employee } e, \text{day } d, \tag{19}$$

$$\sum_s \sum_t \sum_d b_{e\hat{s}td} \cdot x_{e\hat{s}td} \leqslant 5 \qquad \text{for each employee } e, \tag{20}$$

$$\sum_s \sum_t \sum_d hrs_{\hat{s}} \cdot b_{e\hat{s}td} \cdot x_{e\hat{s}td} + \lambda^u_e - \lambda^o_e = Trgt_e \qquad \text{for each employee } e, \tag{21}$$

$$x_{e\hat{s}td} = \text{binary} \qquad \text{for each } e, \hat{s}, t, d, \tag{22}$$

$$\lambda^u_e,\ \lambda^o_e \geqslant 0 \qquad \text{for each employee } e. \tag{23}$$

The objective (17) is identical to (2). The constraints (18)–(21) are identical to (4)–(7), but only the selected shifts $\hat{s}$ from Section 4.1 are used.

The above integer linear program is much smaller than the original ILP. For the problem given in Loucks (1987) and Loucks and Jacobs (1991), the number of binary variables is 2,170 and the number of constraints is 485. This problem was solved on a personal computer using CPLEX in 32 minutes. However, a randomly generated problem took more than one hour to solve. Because the solution is not guaranteed in a reasonable amount of time, we have devised the following heuristic for solving this problem.

4.2.1. Integer linear programming-based heuristic for assignment of good shifts to employees

We will determine all the (good) shifts to be assigned to an employee at once (using a small integer linear program), one employee at a time. This method differs from methods of Loucks (1987) and Loucks and Jacobs (1991), Thompson (1988), and ESP software which assign one shift at a time to an employee. Our method has similarities with the Lagrangian relaxation and (interior point) barrier methods in that some of constraints are put in the objective function with weights (penalties) that are increased as the solution gets close to violating them.

When assigning a (good) shift to an employee, it is important to realize that by doing so, this employee may not be able to take on another shift later during the solution procedure for which he/she will be the only available and eligible employee. To avoid problems such as this, we will explicitly keep track of and use:

- The number of remaining available and eligible employees for each remaining shift
- The number of shifts remaining during each day and the number of available and eligible employees remaining during each day
- The total number of shifts remaining and the total number of available and eligible employee-shifts remaining for the week.

The heuristic is simple, but unfortunately the required notation makes it appear complicated. Let,

$\widehat{S}^r_{ed}$ = number of remaining shifts that employee e is available and eligible for on day d
$\widehat{S}^r_e$ = total number of remaining shifts that employee e is available and eligible for in the week; $\widehat{S}^r_e = \sum_d \widehat{S}^r_{ed}$
$E^r_{ed} = 1$ if employee e is available and eligible for at least one remaining shift during day d, i.e., if $\widehat{S}^r_{ed} \geqslant 1$; 0 otherwise
E^r_d = total number of remaining employees available and eligible for at least one shift during day d; $E^r_d = \sum_{e \in \text{remaining employees}} E^r_{ed}$
E^r = total number of remaining employee-days available and eligible for at least one shift during a day (up to a maximum of 5 days a week); $E^r = \sum_{e \in \text{remaining employees}} \min\left(\sum_d E^r_{ed}, 5\right)$

$E^r_{\hat{s}td}$ = number of remaining employees available and eligible for shift $\hat{s}$, for task t, on day d
E^r_{Ld} = minimum number of remaining employees available and eligible for any L-hour shift on day d; $E^r_{Ld} = \min_{\hat{s},t}\{E^r_{\hat{s}td} : hrs_{\hat{s}} = L\}$
$\widehat{S}^r_d$ = number of shifts remaining on day d
Λ^r_u = total number of hours under the target weekly work hours so far
Λ^r_o = total number of hours over the target weekly work hours so far
d^u_e = days of work under 5 for employee e
x_{Ld} = 1 if a shift of L-hour length is chosen to be assigned on day d; 0 otherwise

Our heuristic for assigning the good shifts to employees is as follows. Initially, let $\Lambda^r_u = 0$ and $\Lambda^r_o = 0$.

1. Calculate the total number of remaining shifts each employee is available and eligible for ($\widehat{S}^r_e$). Find that employee with the smallest $\widehat{S}^r_e$, say employee e_i.
2. For employee e_i,
   i. for each remaining shift $\hat{s}$, determine the total number of remaining employees (including e_i) who are available and eligible for it ($E^r_{\hat{s}td}$)
   ii. for each length of shift L for remaining shifts on each day d, determine the minimum number of remaining employees (including e_i) available and eligible (E^r_{Ld})
   iii. for each day d, determine the total number of remaining employees (including e_i) who are available and eligible for at least one shift (E^r_d)
   iv. determine the total number of remaining employee-days available and eligible for at least one shift during a day, up to a maximum of 5 days a week (E^r)
   v. for each day d, determine the number of shifts remaining ($\widehat{S}^r_d$)
   vi. Solve the following integer linear program for employee e_i:

Table 1
Employee capabilities (1 means employee can perform the task) and weekly target hours for the Example.

Worker	Tasks					Targets
	Gr	DT	FF	BC	Co	
	1	2	3	4	5	
1					1	6
2		1	1	1	1	12
3			1		1	8
4	1					10
5	1					10
6	1	1	1	1	1	18
7	1					15
8	1		1			18
9	1					12
10			1	1	1	16
11	1	1	1	1	1	20
12		1	1	1	1	15
13		1	1	1	1	18
14		1	1	1	1	14
15	1	1			1	15
16	1	1	1	1	1	25
17		1	1	1	1	24
18	1		1			16
19	1		1			20
20		1	1	1	1	20
21		1	1	1	1	26
22		1	1	1	1	20
23	1	1		1	1	24
24				1	1	24
25	1					20
26	1		1			24
27	1			1	1	25
28		1	1	1	1	22
29	1					25
30		1	1	1	1	30
31	1	1	1	1	1	35
32			1		1	25
33		1	1	1	1	35
34	1					32
35					1	35
36		1	1		1	32
37	1					35
38	1	1	1	1	1	35
39	1	1		1		30
40	1		1			32

$$\text{Minimize} \quad (1+\varepsilon\Lambda_u^r)\lambda_e^u + (1+\varepsilon\Lambda_o^r)\lambda_e^o + \frac{\delta_1 d_e^u}{E^r - \sum_d \widehat{S}_d^r + \varepsilon} - \delta_2\left(\sum_d \sum_L \frac{x_{Ld}}{E_{Ld}^r - (1-\varepsilon)}\right) - \delta_3 \sum_d \left(\frac{\sum_L x_{Ld}}{E_d^r - \widehat{S}_d^r + \varepsilon}\right) \tag{24}$$

$$\text{Subject to}: \quad \sum_d \sum_L x_{Ld} + d_e^u = 5 \tag{25}$$

$$\sum_d \sum_L x_{Ld} L + \lambda_e^u - \lambda_e^o = Trgt_e \tag{26}$$

$$\sum_L x_{Ld} \leqslant 1 \qquad \text{for each } d \tag{27}$$

$$x_{Ld} \leqslant E_{Ld}^r \qquad \text{for each } L \text{ and } d \tag{28}$$

$$x_{Ld} = \text{binary} \qquad \text{for each } L \text{ and } d \tag{29}$$

$$d_e^u,\ \lambda_e^u, \lambda_e^o \geqslant 0. \tag{30}$$

Let the optimal solution to the above ILP be $\{x_{Ld}^*\}$. Note that $\{x_{Ld}^*\}$ can have a maximum of 5 members (i.e., a maximum of 5 x's can be 1).

vii. Assign each shift in $\{x_{Ld}^*\}$ to employee e_i and delete the shift. Remove employee e_i. Update $\Lambda_u^r = \Lambda_u^r + \lambda_e^u$ or $\Lambda_o^r = \Lambda_o^r + \lambda_e^o$. Go back to Step 1 unless all shifts have been assigned, in which case terminate.

In Step 1, we find the employee with the smallest number of remaining available and eligible shifts, because finding shifts for this employee in order to reach his/her target weekly work hours $Trgt_e$ appears to be most difficult. In (24), the coefficient of λ_e^u or λ_e^o is chosen so that as the total deviation (underage or overage) from the target weekly work hours so far, Λ_u^r or Λ_o^r, increases, the coefficient increases as well. This forces the solution toward balancing Λ_u^r and Λ_o^r, and minimizes the total deviation from weekly target work hours. The coefficient ε is a small number, and is determined by trial and error. We used $\varepsilon = .1$. Also in (24), the coefficient of d_e^u is chosen so that as E^r gets close to $\sum_d \widehat{S}_d^r$ from above, this coefficient increases sharply. Similarly in (24), the coefficient of x_{Ld} is chosen so that as E_{Ld}^r gets close to 1 from above, this coefficient becomes a large negative number, thus making the x_{Ld} more attractive to select. Finally in (24), the coefficient of $\sum_L x_{Ld}$ is chosen so that as E_d^r approaches $\widehat{S}_d^r$ from above, this coefficient becomes a large negative number, thus making this $\sum_L x_{Ld}$ more

Table 2
Availabilities of each employee for the Example.

Worker	Sunday		Monday		Tuesday		Wednesday		Thursday		Friday		Saturday	
	Start	End	Start	End	Start	End	Start	End	Start	End	Start	End	Start	End
1					4 p.m.	8 p.m.					8 a.m.	4 p.m.		
2					6 a.m.	5 p.m.			6 a.m.	5 p.m.			6 a.m.	5 p.m.
3	5 p.m.	10 p.m.	6 p.m.	10 p.m.	6 p.m.	10 p.m.	6 p.m.	10 p.m.	6 p.m.	10 p.m.	6 p.m.	12 a.m.	6 p.m.	12 a.m.
4			7 a.m.	2 p.m.	7 a.m.	2 p.m.			7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	4 p.m.
5			7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.
6	6 a.m.	5 p.m.			12 p.m.	5 p.m.			12 p.m.	5 p.m.	9 a.m.	5 p.m.	6 a.m.	1 a.m.
7	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.	7 a.m.	2 p.m.
8	6 a.m.	12 a.m.	5 p.m.	8 p.m.			5 p.m.	8 p.m.			5 p.m.	1 a.m.	6 a.m.	1 a.m.
9			5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	1 a.m.	6 a.m.	1 a.m.
10			6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.		
11			6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.	6 a.m.	5 p.m.		
12	6 a.m.	12 a.m.	8 a.m.	3 p.m.	8 a.m.	3 p.m.			8 a.m.	3 p.m.			6 a.m.	1 a.m.
13	6 a.m.	5 p.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	1 a.m.	6 a.m.	5 p.m.
14			4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	1 a.m.	6 a.m.	1 a.m.
15	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	1 a.m.	6 a.m.	1 a.m.
16	6 a.m.	3 p.m.	6 a.m.	3 p.m.	6 a.m.	3 p.m.	6 a.m.	3 p.m.	6 a.m.	3 p.m.	6 a.m.	3 p.m.	6 a.m.	3 p.m.
17	6 a.m.	7 p.m.	7 a.m.	3 p.m.	7 a.m.	3 p.m.	7 a.m.	3 p.m.	7 a.m.	3 p.m.	7 a.m.	5 p.m.	6 a.m.	7 p.m.
18	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	12 a.m.	4 p.m.	1 a.m.	6 a.m.	1 a.m.
19	6 a.m.	10 p.m.	1 p.m.	10 p.m.	4 p.m.	10 p.m.	4 p.m.	10 p.m.	4 p.m.	10 p.m.	4 p.m.	1 a.m.	6 a.m.	1 a.m.
20			7 a.m.	7 p.m.	7 a.m.	7 p.m.	7 a.m.	7 p.m.	7 a.m.	7 p.m.	7 a.m.	7 p.m.	7 a.m.	7 p.m.
21	6 a.m.	12 a.m.			6 a.m.	12 a.m.	6 a.m.	12 a.m.					6 a.m.	1 a.m.
22	6 a.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	1 a.m.	6 a.m.	1 a.m.
23	6 a.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	12 a.m.	5 p.m.	1 a.m.	6 a.m.	1 a.m.
24	6 a.m.	6 p.m.	6 a.m.	6 p.m.	6 a.m.	6 p.m.	6 a.m.	6 p.m.	6 a.m.	6 p.m.	6 a.m.	6 p.m.	6 a.m.	6 p.m.
25	1 p.m.	12 a.m.	1 p.m.	12 a.m.	1 p.m.	12 a.m.	1 p.m.	12 a.m.	1 p.m.	12 a.m.	1 p.m.	1 a.m.	6 a.m.	1 a.m.
26	6 a.m.	12 a.m.			12 p.m.	12 a.m.	5 p.m.	12 a.m.	12 p.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
27	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.
28	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.	6 a.m.	7 p.m.
29	6 a.m.	12 a.m.	6 a.m.	12 a.m.			6 a.m.	12 a.m.			6 a.m.	1 a.m.	6 a.m.	1 a.m.
30	6 a.m.	12 a.m.	8 a.m.	12 a.m.	8 a.m.	12 a.m.	8 a.m.	12 a.m.	8 a.m.	12 a.m.	8 a.m.	1 a.m.	5 p.m.	1 a.m.
31			6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
32	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	4 p.m.	6 a.m.	12 a.m.	6 a.m.	4 p.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
33	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
34	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
35	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
36	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
37	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
38	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
39	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.
40	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	12 a.m.	6 a.m.	1 a.m.	6 a.m.	1 a.m.

Table 3
The hourly employee requirements for a given week (1 = 6–7 a.m., 2 = 7–8 a.m.,...) for the Example.

	Sunday					Monday					Tuesday					Wednesday					Thursday					Friday					Saturday					
	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	Gr	DT	FF	BC	Co	
1	2	1			1	2	1			1	2	1			1	2	1			1	2	1			1	2	1			1	2	1			1	6 a.m.
2	2	1			1	2	1		1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	1		1	1	7 a.m.
3	2	1		1	1	2	1		1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	1	2	8 a.m.
4	2	1	1	1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1	1	1	1	3	2	1	1	2	9 a.m.
5	2	2	1	1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	3	2	1	1	2	10 a.m.
6	2	2	1	1	1	3	2	1	1	2	3	3	1	1	2	3	3	1	1	2	3	3	1	1	2	3	3	1	1	3	4	3	1	1	3	11 a.m.
7	3	3	1	1	2	4	3	1	1	3	4	3	1	2	3	4	3	1	2	3	4	3	1	2	3	4	4	1	2	3	4	4	2	2	4	12 p.m
8	2	2	1	1	2	3	2	1	1	2	2	2	1	1	2	3	2	1	1	2	3	2	1	1	2	3	3	1	1	2	4	4	2	2	4	1 p.m.
9	2	2	1	1	2	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	2	1	1	1	1	3	3	1	1	3	2 p.m.
10	2	2	1	1	2	2	2	1	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	1	1	2	2	1	1	1	3	2	1	1	2	3 p.m.
11	2	2	1	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	1	1	2	2	1	1	1	2	2	1	1	1	3	2	1	1	2	4 p.m.
12	2	1	1	1	1	3	2	1	1	2	2	2	1	1	2	3	2	1	1	2	3	2	1	1	2	3	3	1	1	2	2	2	1	1	2	5 p.m.
13	2	2	1	1	1	3	2	1	1	2	3	3	1	1	2	3	3	1	1	2	3	3	1	1	2	3	3	1	1	3	2	2	1	1	2	6 p.m.
14	2	1	1	1	1	2	2	1	1	1	2	1	1	1	1	2	1	1	1	1	2	2	1	1	2	3	2	1	1	2	2	1		1	1	7 p.m.
15	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	1	1	1	1	2	1		1	1	8 p.m.
16	2	1			1	2	1		1	1	2	1		1	1	2	1		1	1	2	1		1	1	2	2	1	1	1	2	1		1	1	9 p.m.
17	2	1			1	2	1			1	2	1			1	2	1			1	2	1			1	2	2	1	1	1	2	1			1	10 p.m.
18	2	1			1	2	1			1	2	1			1	2	1			1	2	1			1	2	1		1	1	2	1			1	11 p.m.
19																									2	1			1	2	1			1	12 p.m.	

attractive to select. In (24), $\delta_1,\ \delta_2,\ \delta_3$ are small numbers that are determined by trial and error. Too small a value for them result in an infeasible solution but too large a value results in a large total deviation. We used $\delta_1 = \delta_2 = \delta_3 = 3$. Eq. (25) is just the maximum 5 work day constraint. Eq. (26) is just the target weekly work hours constraint. Inequality (27) will force a maximum of one shift length to be selected per day. Inequality (28) will force the variable for a given shift length to be 0 if employee e_i is unavailable or ineligible for any shift of that length on that day.

Each integer linear program above has 42 binary variables, but a maximum of five of them can be 1. Therefore, the computation time for solving each ILP is usually just a few seconds.

Although most instances of the problem are solved feasibly using the integer linear programming-based heuristic above, some instances require the following special adjustments.

4.2.2. Special adjustments required

1. The total number of remaining employees available and eligible for at least one shift during day d, E^r_d, is sometimes inflated. This happens when more than m "similar" employees, $m \geqslant 1$, are available and eligible for a set of m shifts on a day and those shifts are the only possible shifts on that day for these employees. Let $e_{\widehat{S}^r_{ed}\in M}$ = set of similar employees who are only available and eligible for a set of m shifts, M, on day d. Then, compute and subtract $(|e_{\widehat{S}^r_{ed}\in M}| - m)$ from E^r_d, where $|\ |$ denotes the cardinality (i.e., number of members) of a set.

   In practice (i.e., based on our experience with random problem experiments, described in Section 6 below), we have found that investigating subsets of employees who are available and eligible for up to m shifts, $1 \leqslant m \leqslant 12$, is sufficient.
2. If the number of shifts remaining on day d, $\widehat{S}^r_d$, equals the number of employees available and eligible for at least one shift on the day, E^r_d, possibly after adjustment 1 above, then it is important that only members of $e_{\widehat{S}^r_{ed}\in M},\ 1 \leqslant |\ \mathrm{M}\ | \leqslant 12$, be assigned to their available and eligible shifts. Otherwise, we will end up with more shifts than employees on day d. Therefore, remove all the other employees associated with these shifts.
3. If there are two or more similar shifts (in terms of availability and eligibility for a subset of size m, $m \geqslant 2$, of the remaining employees) with $E^r_{std} \leqslant m$, then E^r_{std} of each of these shifts should be reduced by the number of these shifts -1. This is because assigning one of these shifts to an available and eligible employee will automatically reduce E^r_{std} for the rest of these shifts. More formally, let

   $$\hat{s}^r_d(similar : E^r_{std} \leqslant m) = \text{set of shifts on day } d \text{ with } E^r_{std} \leqslant m \text{ which are available and eligible for a subset of size } m \text{ of the remaining employees}$$

   Then, for each member of $\hat{s}^r_d(similar : E^r_{std} \leqslant m)$, reduce E^r_{std} by $|\hat{s}^r_d(similar) : E^r_{std} \leqslant m)| - 1$. In practice, we have found that investigating subsets of shifts with E^r_{std} up to m, $2 \leqslant m \leqslant 7$ is sufficient.
4. If the number of shifts remaining on day d, $\widehat{S}^r_d$, equals the number of employees available and eligible for at least one shift on the day, E^r_d, possibly after adjustment 1 above, then it is important that only members of $\hat{s}^r_d(similar : E^r_{std} \leqslant m),\ 2 \leqslant m \leqslant 7$, be assigned to their available and eligible employees. Otherwise, we will end up with more shifts than employees on day d. Therefore, remove all the other shifts associated with the employees who are available and eligible for $\hat{s}^r_d(similar : E^r_{std} \leqslant m)$. This also applies for shifts with $E^r_{std} = 1$.

5. Example

We will use the problem given in Loucks (1987) and Loucks and Jacobs (1991), which is from a McDonald's restaurant, as a preliminary test problem. This enables comparison of performance of our and their solution methods.

Table 4
The desired number of shifts per day for the Example.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Total
Total employee hours required	110	116	117	120	123	141	151	878
Number of shifts desired	20	21	21	22	22	26	28	160

Table 5
The desired number of daily shifts for each task for the Example.

	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Total
Hours required	110	116	117	120	123	141	151	878
Number of shifts	20	21	21	22	22	26	25	160
Hours required by task:								
Gr	37	42	40	42	42	45	49	
DT	27	26	26	27	29	35	36	
FF	11	9	11	11	11	15	13	
BC	13	15	16	16	16	18	17	
Co	22	24	24	24	25	28	36	
Number of shifts by task:								
Gr	7	8	7	8	8	8	9	
DT	5	5	5	5	5	6	7	
FF	2	2	2	2	2	3	2	
BC	2	3	3	3	3	3	3	
Co	4	4	4	4	4	5	7	

There are 40 employees. Some have the skill for one or two of the tasks, and the others know most or all the five tasks. These and weekly target hours for a specific week are given in Table 1. Some employees are available some of the days and a part of the day, and the others are available all days and all operating hours during the day. These are given in Table 2. For each day and each task, the hourly employee requirements are derived from Appendix B of Loucks (1987). These are given in Table 3.

Our shift scheduling method starts by determining the number of employee-shifts desired for each task on each day. We assume that an average employee will work 4 days a week, thus 40 employees will be assigned 160 employee-shifts. Splitting 160 in proportion to total

Table 6
The determined good shifts for each day (1 = 6–7 a.m., 2 = 7–8 a.m.,...) for the Example.

	Gr		DT		Co		BC		FF	
	Start	End	Start	End	Start	End	Start	End	Start	End
Sunday										
	1	3	1	7	1	8	3	9	4	11
	1	7	5	9	7	10	10	15	12	14
	4	8	7	11	9	11				
	7	11	10	13	12	18				
	9	11	13	18						
	12	18								
	12	18								
Monday										
	1	3	1	7	1	7	2	8	6	11
	1	7	6	8	6	8	9	11	12	14
	4	8	7	10	7	13	12	16		
	6	8	10	14	12	18				
	7	11	12	18						
	9	13								
	12	18								
	12	18								
Tuesday										
	1	7	1	7	1	7	2	7	2	4
	1	7	6	8	6	8	7	11	6	11
	6	8	6	13	7	13	12	16	12	14
	7	11	12	14	12	18				
	9	13	13	18						
	12	18								
	13	18								
Wednesday										
	1	3	1	7	1	7	2	7	2	4
	1	7	6	8	6	8	7	11	6	11
	4	8	6	13	7	13	12	16	12	14
	6	8	11	13	12	18				
	7	11	13	18						
	9	13								
	12	18								
	12	18								
Thursday										
	1	3	1	7	1	7	2	7	2	4
	1	7	6	8	6	8	7	11	6	11
	4	8	6	13	7	14	12	16	12	14
	6	8	10	14	12	18				
	7	11	13	18						
	9	13								
	12	18								
	12	18								
Friday										
	1	3	1	7	1	7	2	7	2	4
	1	7	6	8	6	8	7	11	6	11
	4	8	6	8	6	13	12	18	12	17
	6	11	7	13	12	14				
	7	11	10	14	13	19				
	12	14	12	17						
	12	19	16	19						
	12	19								
Saturday										
	1	3	1	3	1	3	2	8	3	8
	1	8	3	9	3	9	7	11	7	13
	4	8	4	9	4	9	12	16		
	4	8	6	8	6	8				
	6	11	7	11	7	11				
	9	13	10	13	10	13				
	9	13	12	19	12	19				
	14	19								
	14	19								

employee hours required per day results in the desired number of daily shifts, which are given in Table 4. We rounded the numbers to the nearest integer, but when the total did not sum to 160, we rounded the number closest to .5 the other way. For example, Sunday should have 20 shifts because (110/878)(160) = 20.0456, which is rounded to 20.

For each day, these target (desired) number of employee-shifts are further subdivided in proportion to employee hours required for each task. Again, we rounded the numbers to the nearest integer, but when the total did not sum right, we rounded the number closest to .5 the other way. The results are shown in Table 5. For example, Sunday Grill should have 7 shifts because (37/110)(20) = 6.727, which is rounded to 7.

Then, for each possible shift (on each day and for each task) the number of employees available and eligible for it is computed using Excel's VBA. For example, for the 6 a.m.–9 a.m. Sunday Grill shift, using the information given in Tables 1 and 2, there are 13 available and eligible employees.

For each day and task, using Excel's standard Solver, we solved the integer linear program (12)–(16) to determine the good shifts. There were 35 ILPs, one for each of 5 tasks on each of 7 days. Each ILP for Sunday–Thursday had 83 variables and 19 constraints, and each ILP for Friday–Saturday had 89 variables and 20 constraints. The results are displayed in Table 6. Each ILP took approximately 2 seconds to find an optimal solution.

Next, we assigned the good shifts to employees using the target hours given in Table 1. This involved solving 40 ILPs (24)–(30), one for each of the 40 employees. Each ILP had 45 variables and 93 constraints. Our heuristic resulted in the assignments given in Table 7, where

Table 7
The assignments of good shifts to employees for the Example.

Iteration	Worker	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Target	Worked	Over	Under
1	wkr1						*d*6 *t*3 6–8		6	3		3
2	wkr3	*d*1 *t*5 12–14							8	3		5
3	wkr4						*d*6 *t*1 4–8	*d*7 *t*1 4–8	10	10		
4	wkr5				*d*4 *t*1 4–8			*d*7 *t*1 4–8	10	10		
5	wkr7	*d*1 *t*1 4–8	*d*2 *t*1 4–8			*d*5 *t*1 4–8			15	15		
6	wkr9			*d*3 *t*1 13–18			*d*6 *t*1 12–14	*d*7 *t*1 1–3	12	12		
7	wkr8		*d*2 *t*5 12–14		*d*4 *t*5 12–14		*d*6 *t*5 12–17	*d*7 *t*1 14–19	18	18		
8	wkr19	*d*1 *t*1 1–3	*d*2 *t*1 9–13	*d*3 *t*5 12–14		*d*5 *t*5 12–14		*d*7 *t*1 14–19	20	20		
9	wkr18						*d*6 *t*1 12–19	*d*7 *t*1 1–8	16	16		
10	wkr25	*d*1 *t*1 9–11	*d*2 *t*1 12–18		*d*4 *t*1 9–13	*d*5 *t*1 9–13			20	20		
11	wkr29	*d*1 *t*1 12–18	*d*2 *t*1 7–11		*d*4 *t*1 7–11		*d*6 *t*1 1–3	*d*7 *t*1 9–13	25	25		
12	wkr26	*d*1 *t*1 7–11		*d*3 *t*1 9–13		*d*5 *t*1 7–11	*d*6 *t*5 2–4	*d*7 *t*1 6–11	24	24		
13	wkr34	*d*1 *t*1 12–18			*d*4 *t*1 12–18	*d*5 *t*1 12–18	*d*6 *t*1 6–11	*d*7 *t*1 9–13	32	32		
14	wkr37		*d*2 *t*1 12–18	*d*3 *t*1 12–18	*d*4 *t*1 12–18	*d*5 *t*1 12–18	*d*6 *t*1 1–7		35	35		
15	wkr10		*d*2 *t*4 9–11		*d*4 *t*5 2–4	*d*5 *t*5 2–4	*d*6 *t*3 1–7		16	16		
16	wkr40	*d*1 *t*1 1–7	*d*2 *t*1 1–3	*d*3 *t*1 1–7	*d*4 *t*1 1–7		*d*6 *t*1 12–19		32	32		
17	wkr2			*d*3 *t*5 2–4		*d*5 *t*5 6–11		*d*7 *t*2 1–3	12	12		
18	wkr24			*d*3 *t*4 7–11	*d*4 *t*4 7–11	*d*5 *t*4 7–11	*d*6 *t*4 2–7	*d*7 *t*3 1–3	24	24		
19	wkr15	*d*1 *t*2 13–18		*d*3 *t*2 12–14			*d*6 *t*3 12–14	*d*7 *t*2 6–8	15	15		
20	wkr35	*d*1 *t*3 1–8	*d*2 *t*3 1–7			*d*5 *t*3 7–14	*d*6 *t*3 6–13	*d*7 *t*3 10–13	35	35		
21	wkr32	*d*1 *t*3 12–18	*d*2 *t*5 6–11		*d*4 *t*5 6–11	*d*5 *t*3 6–8		*d*7 *t*3 6–8	25	25		
22	wkr6	*d*1 *t*3 9–11		*d*3 *t*1 7–11			*d*6 *t*1 7–11	*d*7 *t*4 12–16	18	18		
23	wkr12	*d*1 *t*4 10–15	*d*2 *t*2 6–8	*d*3 *t*2 6–8		*d*5 *t*2 6–8			15	15		
24	wkr14		*d*2 *t*3 12–18		*d*4 *t*2 11–13			*d*7 *t*2 10–13	14	14		
25	wkr17			*d*3 *t*3 6–8	*d*4 *t*2 6–8	*d*5 *t*4 2–7	*d*6 *t*4 7–11	*d*7 *t*5 7–13	24	24		
26	wkr20		*d*2 *t*3 6–8	*d*3 *t*5 6–11	*d*4 *t*3 6–8		*d*6 *t*2 6–8	*d*7 *t*2 7–11	20	20		
27	wkr11		*d*2 *t*2 7–10	*d*3 *t*1 6–8	*d*4 *t*1 1–3	*d*5 *t*1 1–3	*d*6 *t*2 1–7		20	20		
28	wkr27		*d*2 *t*1 6–8	*d*3 *t*1 1–7	*d*4 *t*1 6–8	*d*5 *t*1 1–7		*d*7 *t*3 7–11	25	25		
29	wkr16		*d*2 *t*1 1–7	*d*3 *t*4 2–7		*d*5 *t*1 6–8	*d*6 *t*2 6–8	*d*7 *t*5 3–8	25	25		
30	wkr13	*d*1 *t*3 7–10				*d*5 *t*4 12–16	*d*6 *t*2 16–19	*d*7 *t*4 7–11	18	18		
31	wkr23	*d*1 *t*2 10–13	*d*2 *t*4 12–16	*d*3 *t*4 12–16	*d*4 *t*4 12–16			*d*7 *t*3 4–9	24	25	1	
32	wkr22				*d*4 *t*2 13–18	*d*5 *t*3 12–18	*d*6 *t*3 13–19		20	20		
33	wkr21	*d*1 *t*2 5–9		*d*3 *t*3 1–7	*d*4 *t*4 2–7			*d*7 *t*3 12–19	26	26		
34	wkr30	*d*1 *t*2 7–11	*d*2 *t*2 10–14	*d*3 *t*3 12–18	*d*4 *t*3 12–18	*d*5 *t*2 13–18			30	30		
35	wkr39	*d*1 *t*4 3–9		*d*3 *t*2 13–18		*d*5 *t*2 10–14	*d*6 *t*2 10–14	*d*7 *t*4 2–8	30	30		
36	wkr28		*d*2 *t*2 1–7		*d*4 *t*2 1–7		*d*6 *t*5 6–11	*d*7 *t*2 4–9	22	26	4	
37	wkr36	*d*1 *t*2 1–7		*d*3 *t*2 1–7	*d*4 *t*3 1–7	*d*5 *t*2 1–7	*d*6 *t*2 12–17		32	34	2	
38	wkr31		*d*2 *t*2 12–18	*d*3 *t*3 7–13	*d*4 *t*3 7–13		*d*6 *t*2 7–13	*d*7 *t*2 3–9	35	35		
39	wkr33	*d*1 *t*5 4–11	*d*2 *t*3 7–13			*d*5 *t*3 1–7	*d*6 *t*4 12–18	*d*7 *t*3 3–9	35	36	1	
40	wkr38		*d*2 *t*4 2–8	*d*3 *t*2 6–13	*d*4 *t*2 6–13	*d*5 *t*2 6–13		*d*7 *t*2 12–19	35	39	4	
											12	8

Table 8
Comparison of results of Loucks and our solution method for the Example.

	Loucks	Our method
Hours of overstaffing	6	4
Total deviation from target weekly hours	40	20[a]
Number of task changes within shifts	91	7[b]

[a] This happens to be the optimal total deviation.
[b] Due to adding some odd jobs such as Lobby Clean to regular tasks such as Bin Call.

days 1,...,7 are Sunday,...,Saturday, and Tasks 1,...,5 are Grill, Drive Thru, Counter, Bin Call, and French Fries, respectively. The 40 ILPs took approximately 18 minutes to find their optimal solutions.

Table 8 compares the results of our method with that of Loucks (1987) and Loucks and Jacobs (1991). Clearly, our method has found a better solution. Note that because Loucks (1987) and Loucks and Jacobs (1991) used task-hours (vs our task-shifts) as unit work interval, their method resulted in significantly more task changes within shifts.

Just out of interest, we entered our good shifts derived in the above example as fixed shifts in the ESP software (which uses Simulated Annealing for assigning the shifts to the employees), and ran it. Unfortunately, after a few minutes ESP terminated with 6 unassigned shifts.

6. Computational experiments

Based on the data from Loucks (1987) and Loucks and Jacobs (1991), we generated 100 random problems to test our solution method. The random problems were created "backwards" so that the optimal solution for each problem had zero overstaffing and zero total deviation from target weekly hours for each employee. For each of the 40 employees, the procedure was to randomly determine:

a. his/her Number of workdays: used frequency distribution

Days	1	2	3	4	5
Frequency	2	1	8	12	17

and days worked: used frequency distribution

Day	Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
Frequency	20	21	22	23	23	26	27

b. the start time for each shift (1 = 6 a.m., 2 = 7 a.m., etc.): used frequency distribution:

Start time	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Frequency	33	16	10	13	7	34	29	8	14	12	9	33	10	5	2	1

and shift length (which depends on the start time) for each shift: used frequency distributions

6 a.m. start time	Duration	3 hours	4	5	6	7	8
	Frequency	7	1	1	1	20	3
7 a.m. start time	Duration	3 hours	4	5	6	7	8
	Frequency	5	1	1	5	3	1
...							
8 p.m. start time	Duration	3 hours	4				
	Frequency	1	1				
9 p.m. start time	Duration	3 hours					
	Frequency	1					

c. the number of tasks he/she can perform: used frequency distribution

No. of tasks	1	2	3
Frequency	17	14	9

and the nature of tasks employee can perform: used frequency distributions:

if can perform 1 task:

No. of tasks	Grill	Counter
Frequency	13	6

if can perform 2 tasks:

No. of tasks	Grill & Drivethru	Grill & Counter	Drivethru & Counter
Frequency	1	1	12

d. the task performed on each work day: used frequency distributions:

If can perform Grill & Drivethru	Task	Grill	Drivethru
	Frequency	1	1

If can perform Grill & Counter	Task	Grill	Counter
	Frequency	1	1
If can perform Drivethru & Counter	Task	Drivethru	Counter
	Frequency	2	1

If can perform Grill, Counter, and Drivethru

Task	Grill	Counter	Drivethru
Frequency	10	1	10

e. whether available or not on a non-work day: used frequency distribution

Available	Yes	No
Frequency	1	1

f. the start time (1 = 6 a.m., 2 = 7 a.m., etc.) and availability length for each work day and available non-work days: used frequency distributions Available start time:

Start time	1	2	3	4	5	6	7	8	9
Frequency	155	44	22	14	12	11	14	15	8
Start time	10	11	12	13	14	15	16		
Frequency	7	19	30	10	3	2	1		

Availability length: If available start time = 6 a.m.

Duration	3 hours	4	5	6	7	8	9	10	11
Frequency	1	1	1	1	1	1	8	3	17
Duration	12	13	14	15	16	17	All day		
Frequency	8	17	1	1	2	1	91		

If available start time = 7 a.m.

Duration	3 hours	4	5	6	7	8	9	10	11
Frequency	1	1	1	1	18	5	2	2	1
Duration	12	13	14	15	16	17			
Frequency	7	1	1	1	1	1			

...

If available start time = 8 p.m.

Duration	3 hours	4
Frequency	1	1

If available start time = 9 p.m.

Duration	3 hours
Frequency	1

Note that the availability period should include the shift period during a work day determined in step b. We used 6 a.m. to midnight work hours every day.

All the steps of the algorithm, except the ILPs, were programmed in Excel's VBA. The ILPs were solved using Excel's standard Solver. Our method found a feasible solution to every problem, and created an average of only 1.01 hours of over staffing and an average of 17.31 hours of total deviation from the weekly targets, which is only approximately half-an-hour per employee. The average time to solve one problem was approximately 5 minutes.

7. Extensions of our method

The model used in this paper includes the basic characteristics of heterogeneous, part-time, limited-available employee scheduling problem. However, individual managers may have additional requirements. Some of these can be included in our method. Two of these and their treatment are as follows:

(a) Lunch breaks for long shifts. These can easily be included by inserting a break period in the long shifts of shift determination phase of Section 4.1: For example, the coefficients a_{hstd}, for all h, in Constraint (13) for a 6 a.m. till 3 p.m. shift with a lunch break during 10 a.m. to 11 a.m. hour can be represented by the column $(1,1,1,1,0,1,1,1,1,0,0,0,\ldots)$.

(b) At least two *consecutive* days off. This can only be done as a soft "constraint" by including a term in the objective function (24) which would make at least two-consecutive days offs attractive. This is done as follows. To the ILP (24)–(30), add six binary variables, $u_1,\ldots,u_6$, which represent pairs of consecutive days Sunday–Monday,…,Friday–Saturday, and six constraints $\sum_L x_{L,d=1} + \sum_L x_{L,d=2} \leqslant 2(1-u_1),\ldots,\sum_L x_{L,d=6} + \sum_L x_{L,d=7} \leqslant 2(1-u_6)$, and add $-(u_1+u_2+u_3+u_4+u_5+u_6)$ to (24). We have found through random experiments that approximately 80% of employee will have at least one two-consecutive days offs using this modification.

8. Conclusion

In this article, we have described a two-stage procedure for solving the employee scheduling problem for services employing heterogeneous, part-time employees with limited availability. First, good shifts are determined using a series of small integer linear programs for each day and each task. Then, an integer linear programming-based heuristic is used to assign the good shifts to the employees, one employee at a time. This heuristic assigns all the shifts of an employee to him/her at once, as opposed to assigning one shift at a time to an employee, done by most other methods given in the literature. Our method considers all the constraints for each employee and therefore does not need backtracking (i.e., exchanging shifts between two employees).

We have shown, through solving many random problems, that our method is robust. It provided a close-to-optimal (feasible) solution every time.

References

Alfares, H.K., 2004. Survey, categorization, and comparison of recent tour scheduling literature. Annals of Operations Research, 145–175.
Degnan Manning, C., Garf, R., Sweeney, J. 2006. Workforce Management Landscape: The Right People in the Right Place at the Right Time. AMR Research.
Ernst, A.T., Jiang, H., Krishnamoorthy, M., Owens, B., Sier, D., 2004. An annotated bibliography of personnel scheduling and rostering. Annals of Operations Research, 21–144.
Eveborn, P., Ronnqvist, M., 2004. Scheduler—A System for Staff Planning. Annals of Operations Research, 21–45 (12, Special Issue on Staff Scheduling and Rostering).
Glover, F., McMillan, C., 1986. The general employee scheduling problem: An integration of MS and AI. Computers and Operations Research 13 (5), 563–573.
Gopalakrishnan, M., Gopalakrishnan, S., Miller, D.M., 1993. A decision support system for scheduling personnel in a newspaper publishing environment. Interfaces 23 (4), 104–115.
Gordon, L., Erkut, E., 2004. Improving volunteer scheduling for the Edmonton folk festival. Interfaces 34 (5), 367–376.
Litchfield, J.A., Ingolfsson, A., Cheng, K.J., 2003. Rostering for a restaurant. INFOR 41 (3 Aug), 287–300.
Loucks, J.S. 1987. A Multi-objective Heuristic Procedure for the Tour Scheduling and Task Assignment of Fast Food Restaurant Workers. PhD Thesis, Indiana University.
Loucks, J.S., Jacobs, F.R., 1991. Tour scheduling and task assignment of a heterogeneous workforce: A heuristic approach. Decision Sciences 22 (4 Sep/Oct), 719–738.
Love Jr., R.R., Hoey, J.M., 1990. Management science improves fast-food operations. Interfaces 20 (2 Mar–Apr), 21–29.
Thompson, G.M. 1988. A Comparison of Techniques for Scheduling Non-homogeneous Employees in a Service Environment Subject to Non-cyclical Demand. PhD Thesis, Florida State University.
Zolfaghari, S., El-Bouri, A., Namiranian, B., Quan, V., 2007. Heuristics for large scale labour scheduling problems in retail sector. INFOR 45 (3 Aug), 111–122.