Employee Scheduling in Service Industries with Flexible Employee Availability and Demand

Semra Ağralı*[1], Z. Caner Taşkın[2], and A. Tamer Ünal[2]

[1]Department of Industrial Engineering, Bahçeşehir University, 34353, Beşiktaş, İstanbul, Turkey {semra.agrali@eng.bahcesehir.edu.tr}

[2]Department of Industrial Engineering, Boğaziçi University, 34342 Bebek, İstanbul, Turkey {caner.taskin@boun.edu.tr}, {unaltam@boun.edu.tr}

Abstract

We consider an employee scheduling problem arising in service industries with flexible employee availability and flexible demand. In the system to be planned, there is a given set of service requirements and a set of employees at any time. Each employee belongs to one of various skill levels; each service requirement specifies the requested employee skill level, timing of the service delivery, and a weight that indicates its priority. Employees have individual flexible contracts with the organization, which are characterized by weekly/monthly contracted work hours, days the employee is available for work and availability of overtime. Furthermore, there are regulations on maximum work hours and minimum rest requirements of employees enforced by the government and the labor union. The problem that we investigate is to generate an assignment of employees to service requirements, which (i) ensures that the maximum weighted number of service requirements are met, (ii) satisfies government and labor union regulations, (iii) honors individual employee contracts with minimum deviation from the contracted work hours, and (iv) ensures a fair balance between employee schedules in terms of work assignments on holidays. We model the problem as a mixed-integer programming problem and discuss a reformulation strategy, which allows us to

*Corresponding author

solve practical problems in a reasonable amount of time. We also report our experience in a large health-care organization in Belgium.

Keywords: OR in service industries; Employee scheduling; Integer programming; Flexible contract; Flexible service demand

1 Introduction

Efficient management of workforce is one of the most important concerns in both manufacturing and service organizations, since it has a direct effect on the productivity of day-to-day operations and the quality of the service provided. Especially the employee scheduling problem has attracted considerable amount of research due to high costs associated with employee utilization. Employee scheduling problem is NP-Hard (Garey and Johnson, 1979), and this, by itself, indicates a major challenge. However, in addition to this major challenge, concerns like large number of strict rules and regulations set by governments and labor unions, which impose hard constraints over the feasible utilization of employees, and non-numerical objectives such as well-being and happiness of the employees, and fairness between employees makes the problem even more interesting as a research topic.

In this study, we investigate an employee scheduling problem that we frequently face in a group of special health-care organizations, such as care centers for children with disabilities and nursing homes for elderly. Operating environment of these special organizations involves two basic forms of flexibility. The first flexibility is in the demand for services rendered in these organizations. The types of patients served require special personalized care. Timing, content and length of services vary significantly, and it is virtually impossible to generate a balanced pattern of services for a prolonged period of time. Services such as emergency health-care, social events, and educational activities that need case-specific resources (personnel and equipment) in an irregular manner must be provided regularly. The second flexibility is in the availability of human resources that is utilized in the delivery of these highly flexible services. The span of skills required is very large, and consequently, it is almost impossible to have full time contracts with all the required human personnel. Hence, these health-care organizations need to provide a set of flexible services using employees with flexible contracts in addition to regular full time employees.

The necessity of providing service with flexible contracts is in-line with the general trend of increasing flexible working conditions in industrialized countries. In recent years, most European countries have installed new legislations to enable employees to request flexible working conditions, including contractual, spatial and temporal flexibility (Joyce et al., 2010). In terms of employee scheduling, flexible working contracts make it unnecessary to define a relatively fixed work management structure like a *shift*. In a flexible organization, the main management concern is not to manage the existence of the employee in a fixed location, but to dynamically manage the match between the availability of the employees with the service requirements.

In the operating environment that we consider, optimal management of employees is crucial from two perspectives. First of all, a major portion of the operational cost is employee-related costs. Although some of the organizations providing aforementioned services receive some form of government support, efficient operation is important to guarantee sustainability of the services. The second major concern is to keep the employees happy by assigning them tasks with a good match for their skills and by conforming to their preferences in terms of the timing of the task assignments. For these special service organizations, employee satisfaction is, again, a significant determinant of sustainability and high service quality.

In this study, we propose a mixed-integer programming model to solve the employee scheduling problem for service organizations with the following features:

- Employees have flexible contracts, which do not impose strict shift structures. Instead each employee has an individual contract with the organization that specifies his/her working availability per day, week, month, etc. This constrains the employee's availability throughout the planning horizon while allowing flexibility in his/her working pattern.

- Demand for services is given in the form of service requirements. Each requirement includes a service start and completion time, the minimum skill level needed to deliver the service, and a weight that indicates the priority level of the requirement.

- In addition to employee contracts, there are constraints that derive from the laws and regulations, such as minimum rest time between assignments, maximum total time that an employee can work within a week, etc.

- Fairness between employees needs to be defined and observed.

Based on the general constraints given above, we develop a model to assign employees to the service requirements. The objectives of the model are to:

- minimize the total number of (weighted) service demand that is not satisfied;

- maximize the utilization of total contracted hours;
- maximize employee satisfaction
  - by assigning tasks that are most appropriate to the skill level of the employee,
  - by minimizing overtime assignments;
- maximize fairness between employees by making balanced work assignments during holidays.

There are two major modeling choices that we made in this study to handle the flexible working environment:

- Contrary to the general approach in the literature, the model that we develop does not assume that time is discretized. Service requirements may have any time value on the continuous time scale as start and end times. Hence, the model generates flexible assignments to the employees in terms of working hours without increasing the number of decision variables.
- We do not generate a fixed shift structure (neither explicitly, nor implicitly). Our major concern is to satisfy the service requirements as much as possible without violating any conditions defined in the employee contracts, or legislative rules and regulations.

To the best of our knowledge, there is no prior study in the literature that models the employee scheduling problem with the flexibility of continuous time and without any reference to an underlying shift structure.

The rest of this paper is organized as follows. Section 2 provides background information and related literature review on the employee scheduling problem. We formally define the problem and provide a basic mathematical model in Section 3. In Section 4, we propose various re-formulations of the basic model to reduce its size and improve its strength. We used the model developed in this study to solve the employee scheduling problem imposed by a large health-care organization in Belgium. We performed a number of test runs with the proposed model using representative real data. We report our implementation experience in Section 5. We summarize our study and conclude the paper in Section 6 with general comments.

2 Background and Literature Review

Employee scheduling has been studied extensively in the literature. Ernst et al. (2004b) trace back the origin of staff scheduling to Edie (1954)'s work, in which toll booth operations are analyzed and the relief times for toll collectors are determined. Since then, employee scheduling has been one of the most important topics for various organizations, such as airlines (Burke et al., 2010; Chu, 2007; Felici and Gentile, 2004; Gopalakrishnan and Johnson, 2005), railways (Strupchanska et al., 2003), transit bus operators (Smith and Wren, 1988), retailers (Zolfaghari et al., 2007; Parisio and Jones, 2015), call centers (Bhulai et al., 2008; Caprara et al., 2003; Dietz, 2011) and health-care organizations (Li et al., 2012; Wright and Mahar, 2013; Meanhout and Vanhoucke, 2013). We refer the reader to Brunner (2010) and den Bergh et al. (2013) for recent surveys of the literature on employee scheduling. Since this paper addresses an employee scheduling problem that is basically applicable in the health-care industry, we will mainly concentrate on the evaluation of the literature specific to health-care.

Health-care is one of the major industries where better employee scheduling creates a significant added value. In health-care systems, nurse rostering problems have been extensively studied. Cheang et al. (2003) provide a bibliographic survey of the models and methodologies that are used to solve nurse rostering problems. Bard and Purnomo (2005) present a robust methodology for solving a nurse rostering problem that accommodates individual preferences while creating schedules. De Grano et al. (2009) also include nurse preferences in the scheduling problem. The authors develop a two-stage approach in which first nurses' preferences for specific days and shifts are obtained, and then a schedule that streamlines those preferences while satisfying important hospital constraints is built. Azaieza and Sharif (2005) develop a 0-1 linear goal program in which both hospital objectives and nurse preferences are accounted for. Another important scheduling problem in health-care is physician scheduling. Scheduling of emergency medicine residents is studied in Topaloglu (2006). The author develops a multi-objective programming problem and reports results of their tests in the emergency room of a major local university hospital.

Employee scheduling problem can be decomposed into various sub-problems. Classification given in Ernst et al. (2004b) identifies the following major sub-problems: demand modeling, shift construction, shift scheduling and employee assignment to shifts. In this study, we keep demand modeling out of our scope and discuss only scheduling-related work. Depending on the implementation case, some of the sub-problems regarding scheduling may not exist or some of the sub-problems may be jointly handled within a unified model. For example, in most employee scheduling problems in manufacturing organizations, shift construction is not a major concern since a restricted set of shifts are already defined by the organization, and the major issue is to assign employees to these fixed shifts (see Berman et al., 1997). In more flexible working environments, such as health-care organizations, shift scheduling and employee assignment may be performed in a single integrated model (see Ernst et al., 2004b).

In most of the studies involving employee scheduling, *shift* concept plays a crucial role. Shift defines a period of time that the employee is scheduled to be available in the expectation that there will be demand for his/her service. Hence, utilization of the employee within a shift is a related but separate concern for the organization. Mostly, employee scheduling aims to generate shifts or assign employees to shifts such that, based on a given demand pattern, expected utilization of the employee is maximized. Generally, shift is modeled explicitly, and it constitutes a basis for the set-covering formulation of Dantzig (1954). There is a number of studies that propose an implicit model of shifts to reduce the number of variables in the formulation (see Bechtold and Jacobs, 1990, 1996; Cote et al., 2011; Thompson, 1995).

Modeling of time is another major issue in employee scheduling problems. A common approach in the literature is to discretize time into small time buckets, and include time in the models as an additional index. Especially for problems with long scheduling periods or very small time buckets, the number of variables increase significantly due to the time index, and it becomes more difficult to obtain high quality solutions in a reasonable amount of time. On the other hand, if the bucket sizes are kept large, the accuracy of the model deteriorates.

The problem that we address in this study involves employees with different skill levels, where employees with higher skill levels may be assigned to services that require a lower skill level. In the literature, employee scheduling problem with this property is called the hierarchical employee scheduling problem. Billionnet (1999) proposes an integer programming model to solve a hierarchical employee scheduling problem with variable demand, in which each worker is assigned to one shift type. Based on this model, Seckiner Ulusam et al. (2007) propose a new model in which workers can be assigned to alternative shifts in a day during the course of a week. Narasimhan (1997) provides an optimal algorithm for shift scheduling of employees whose capabilities are arranged hierarchically in seven-day-a-week industries. Topaloglu (2009) considers scheduling of medical residents that arises in different clinical settings of a hospital. Residents are grouped according to different seniority levels that are specified by the number of years spent in residency training, and a schedule is formed by enforcing seniority-based workload rules.

Another aspect of our problem is scheduling a workforce that includes different types of workers including full-time and part-time. Mohan (2008) study a workforce scheduling problem consisting of part-time workers who have availabilities, preferences for the shifts, maximum and minimum allowable working hours in a day and a week, and seniority levels. They model the problem as an integer program in which the objective is to maximize employee satisfaction and to meet the demand requirements for each shift. Based on Mohan (2008)'s work, Akbari et al. (2013) study the scheduling problem of part-time and mixed-skilled workers in which workers productivity is variable during a day. Their model maximizes workers satisfaction while considering workers availability, productivity, priority preference, seniority level, and number of workers required.

Employee scheduling problems with multiple objectives is also very common in the literature. Besides the standard attempt to include objectives of different stakeholders (managers, employees, unions, regulators, etc.) into consideration, soft constraints that commonly exist in employee scheduling problems are also included into the model as additional objectives (see Burke et al., 2010; Kwak and Lee, 1997; Parr and Thompson, 2007; Topaloglu, 2006). Castillo et al. (2009) consider cost minimization and service level maximization simultaneously and propose a multidimensional paradigm. Multi-objective nurse scheduling problems are studied in Topaloglu (2006, 2009). These multi-objective problems are solved using goal programming.

3 Problem Definition and Mathematical Model

Before describing our mathematical formulation, we will discuss how some important aspects of our problem can be modeled in terms of the set definitions, variables and parameters given in Table 1.

We denote the set of employees by E. Each employee is characterized by his/her skill level $L_e \in L$, where L denotes the set of skill levels. We denote the set of service requirements by R. A requirement $r \in R$ represents the need for a *single* employee at a certain skill level for a specified duration of time within the planning horizon. Each requirement is characterized by its start time S_r, end time F_{er}, which depends on the employee that is assigned to perform the requirement, its minimum required skill level L_r^{min}, and its priority p_r. Note that S_r and F_{er} represent arbitrary time points on the continuous time scale and we do not make any assumptions regarding timings or durations of requirements other than requiring that

Table 1: Symbols used in our mathematical model

Set	**Description**
E	set of employees, indexed by e
R	set of employee requirements, indexed by r
$R(e) \subseteq R$	set of requirements to which employee $e \in E$ can be assigned
L	set of employee skill levels
W	set of constrained time intervals (CTI) within planning horizon, indexed by w
$W(e) \subseteq W$	set of CTIs within planning horizon for which employee $e \in E$ has a contract
H	set of holidays within planning horizon (including weekends), indexed by h
$H(e) \subseteq H$	set of holidays within planning horizon for which employee $e \in E$ has a contract
$R(h) \subseteq R$	set of requirements that coincide with holiday $h \in H$
Parameter	**Description**
L_e	skill level of employee $e \in E$
S_r	start time of requirement $r \in R$
F_{er}	end time of requirement $r \in R$ when employee e is assigned to perform the requirement
L_r^{min}	minimum skill level of requirement $r \in R$
p_r	priority of requirement $r \in R$, where a larger value indicates a higher priority
b_{er}	amount of time that employee e needs to rest after performing requirement r, if assigned to
S^w	start time of CTI $w \in W$
F^w	end time of CTI $w \in W$
d_{er}^w	the portion of time that requirement $r \in R(e)$ coincides with CTI $w \in W(e)$ when employee e is assigned to perform the requirement
c_e^w	the amount of time that employee $e \in E$ is contracted to work within CTI $w \in W(e)$
d_{ew}^{max}	the maximum amount of time that employee e is allowed to work within CTI $w \in W(e)$
Variable	**Description**
x_{er}	indicator variable that represents whether employee $e \in E$ is assigned to requirement $r \in R(e)$
f_r	indicator variable for unfulfilled demand of requirement $r \in R$
t_e^w	total amount of time employee $e \in E$ is scheduled to work within CTI $w \in W(e)$
o_e^w	the amount of time that employee $e \in E$ is scheduled to work over contracted hours within CTI $w \in W(e)$
u_e^w	the amount of time that employee $e \in E$ is scheduled to work under contracted hours within CTI $w \in W(e)$
y_e^h	indicator variable that represents whether employee $e \in E$ is scheduled to work during holiday $h \in H$
y_e	total number of holidays that employee $e \in E$ is scheduled to work
z_e	total number of holiday requirements that employee $e \in E$ is scheduled to work
y_l^{max}	maximum number of holidays that an employee having skill level $l \in L$ is scheduled to work
y_l^{min}	minimum number of holidays that an employee having skill level $l \in L$ is scheduled to work
z_l^{max}	maximum number of holiday requirements that an employee having skill level $l \in L$ is scheduled to work
z_l^{min}	minimum number of holiday requirements that an employee having skill level $l \in L$ is scheduled to work

$S_r < F_{er}$ for each $r \in R$. Also note that regular shift structures can be handled in our model by assigning the starting time of the shift as S_r and the end time of the shift as F_{er}. Moreover, having an employee-dependent finish time for each requirement makes our model more general, and if the duration of a requirement is not employee-dependent, we can simply assign the same time to F_{er} for all $e \in E$. Similarly, if the duration of a requirement depends on the skill level of the employee, then F_{er}-values for all employees having the same skill level can be set equal. Our definition of requirement assumes that each requirement is associated with only one employee. While this assumption may appear to be restrictive, simultaneous need for multiple employees can easily be handled by creating separate requirements that have common start and end times. For instance, assume that two nurses (skill level 1) and a doctor (skill level 2) are needed between February 25, 2015 21:00 and February 26, 2015 01:30. This case can be handled by defining three requirements r_1, r_2, r_3 such that $S_{r_1} = S_{r_2} = S_{r_3} =$ February 25, 2015 21:00, $F_{er_1} = F_{er_2} = F_{er_3} =$ February 26, 2015 01:30, $L_{r_1}^{min} = L_{r_2}^{min} = 1$ and $L_{r_3}^{min} = 2$.

In our model each employee has an individual contract with the organization that is characterized by a set of "constrained time intervals" (CTIs) and a contracted number of working hours within each CTI. In our model a CTI represents a duration of time during which the employee's workload is constrained either by the employee's contract, or due to government or labor union legislations. For instance, a contract may specify that the employee is only available between Monday and Thursday for a maximum duration of 25 hours per week and a total of 80 hours per month. This contract can be modeled by defining

- a CTI for each Friday-Sunday interval having contracted duration of 0 hours,
- a CTI for each week having contracted duration of 25 hours, and
- a CTI spanning each month having contracted duration of 80 hours.

CTIs allow us to easily incorporate employees whose contracts do not span the entire planning horizon. Also, issues such as yearly holidays, relatively long leaves of absence, part-time employees having different full-time-equivalent (FTE) contracts, part-time student employees who can work full-time during the summer, etc. can be easily integrated. Note that we do not enforce any restrictions on the number, timings or durations of CTIs; any two CTIs can be disjoint, partially overlap or fully overlap. In addition to modeling contract details, our definition of CTI can also be used to enforce labor laws and legislations that limit the amount of time that an employee can work per day or week, etc. Moreover, we do not have any restriction on the duration of the planning horizon.

We denote the set of CTIs that apply to employee $e \in E$ by $W(e) \subseteq W$, where W represents the set of all CTIs within the planning horizon. We denote the start time of CTI $w \in W(e)$ by S^w, its end time by F^w and the amount of time that employee e is available during w by c_e^w. Let us denote the set of service requirements that employee e can be assigned to by $R(e) \subseteq R$. Since each requirement has a minimum skill level, $r \in R(e)$ implies that $L_e \geq L_r^{min}$. Furthermore, explicitly defining $R(e)$ for each employee allows us to model employee-specific issues such as weekly available days for working, which is often part of the contract for part-time employees, and employees' off days within the planning horizon.

Let us define a binary variable x_{er} for each employee $e \in E$ and requirement $r \in R(e)$, whose value is one if employee e is assigned to satisfy requirement r, and zero otherwise. Constraint (1) ensures that each requirement is assigned to at most one employee

$$\sum_{e:r \in R(e)} x_{er} + f_r = 1 \quad \forall r \in R. \tag{1}$$

Note that f_r-variable takes on value one if r is not assigned to any employees and zero otherwise. Since x-variables are binary variables, f-variables can be relaxed as continuous. f-variables ensure that our model always has a feasible solution having $x_{er} = 0$ for all e and r, which is important for practical applications in order to have a robust system.

Each employee can only be assigned to at most one service requirement at any time. Furthermore, government regulations and labor union rules require that employees need to rest for at least some amount of time between two consecutive assignments. This issue is commonly encountered in shift scheduling literature (see Aykin, 1996; Caprara et al., 2003; Felici and Gentile, 2004) and is quite straightforward to model if the planning horizon is discretized into disjoint time periods with the same duration (such as morning, afternoon and night shifts). In such models, it is often enough to enforce that each employee can be assigned to a single requirement in each time period and that employees cannot be assigned to requirements in successive time periods. However, start and end times of requirements in our model are given on a continuous time scale without any restrictions on the timing or duration of requirements. Let b_{er} denote the minimum amount of time that employee e needs to rest after performing requirement r. Note that we allow for the rest duration to be defined per employee and requirement instead of a single rest duration that is enforced for all employees and requirements. This allows us to enforce an overnight rest after a long/complicated task while also being able to assign an employee to several short requirements in a row.

The flexibility in start and end times of requirements as well as rest durations results in various possible relationships between requirements. In particular, it is possible that two requirements

1. start and end at the same time,
2. partially overlap,
3. one of them starts before the other one starts and ends after the other one ends,
4. are disjoint with enough rest duration in between,

where only the last case is allowed in a feasible solution. As an example consider five requirements to which an employee e can be assigned, i.e. $r_1, r_2, r_3, r_4, r_5 \in R(e)$. Let start and end times plus the rest periods for the requirements be given in Table 2, where we assume that $t_1 < t_2 < \cdots < t_8$. Figure 1 shows the five requirements on the continuous

Table 2: Requirement data for the illustrative example

Requirements	Starting time, S_r	Finishing time plus rest period, $F_{er} + b_{er}$
r_1	t_1	t_4
r_2	t_2	t_7
r_3	t_3	t_6
r_4	t_5	t_8
r_5	t_5	t_8

time scale. It can be seen that r_4 and r_5 start and end at the same time (Case 1), r_1 and r_2 partially overlap (Case 2), r_2 starts before r_3 and ends after r_3 (Case 3) and r_1 and r_4 are disjoint with enough rest duration in between (Case 4).

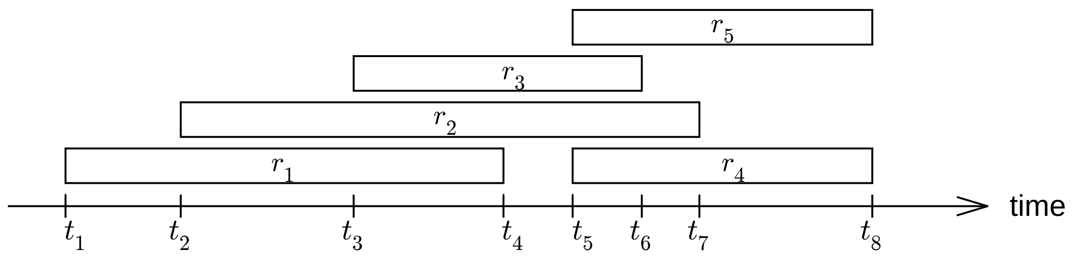

Figure 1: Requirement times for the illustrative example

We analyze pairs of requirements that each employee can potentially be assigned to, and generate constraints (2) to eliminate solutions that assign employees to requirements that do not satisfy the fourth case:

$$x_{e\bar{r}} + x_{e\hat{r}} \leq 1 \quad \forall e \in E, \hat{r}, \bar{r} \in R(e), \hat{r} \neq \bar{r}, S_{\bar{r}} \leq S_{\hat{r}} \leq F_{e\bar{r}} + b_{e\bar{r}}. \tag{2}$$

Constraints (2) enforce that if employee e is assigned to requirement $\bar{r}$, then e cannot be assigned to another requirement $\hat{r}$ that starts after $\bar{r}$ but before $\bar{r}$ ends and enough time for rest passes. Note that (2) are enough to handle the first three cases due to the symmetry of $\bar{r}$ and $\hat{r}$. On the example represented by Figure 1, the following set of constraints are generated:

$$\begin{array}{llll} x_{e1} + x_{e2} \leq 1, & x_{e1} + x_{e3} \leq 1, & x_{e2} + x_{e3} \leq 1, & x_{e2} + x_{e4} \leq 1, \\ x_{e2} + x_{e5} \leq 1, & x_{e3} + x_{e4} \leq 1, & x_{e3} + x_{e5} \leq 1, & x_{e4} + x_{e5} \leq 1. \end{array}$$

Note that constraints (2) are written for pairs of requirements, and therefore there is an $O(|E||R|^2)$ of such constraints. We will discuss a way of improving these constraints in Section 4.1.

Recall that each employee has a contracted amount of time for each CTI $w \in W(e)$. While over or under utilization of the employee is possible, it is desired for each employee to work as close as possible to their contracted time. We define a continuous variable t_e^w to keep track of the total amount of time employee e is scheduled to work within CTI $w \in W(e)$. Similarly, we define continuous variables o_e^w and u_e^w that measure the amount of over and under utilizations associated with e in w, respectively. Constraints (3) relate t- and x-variables, where d_{er}^w is a parameter that represents the portion of time that requirement $r \in R(e)$ coincides with $w \in W(e)$. Formally, $d_{er}^w = \max(0, \min(F_{er}, F^w) - \max(S_r, S^w))$. Note that $d_{er}^w = F_{er} - S_r$ if r starts and ends within the same w. Recall that our definition of requirement assumes that each requirement is associated with only one employee. We use the d_{er}^w-parameter to model requirements overlapping with multiple CTIs. As an example, assume that a requirement is defined between Week 1 Sunday 10 pm and Week 2 Monday 3 am, and there is a CTI for each week that represents weekly working hours. In this case, d_{er}^w-parameters specify that two hours of the working time are associated with Week 1's CTI and three hours are associated with Week 2's CTI. Constraints (4) enforce the allocation of assigned working time among contracted time, and over and under utilizations. Finally, constraints (5) limit the total working time within CTI w by d_{ew}^{max}. d_{ew}^{max} can be a contract parameter to limit overtime, can represent the total length of w, or it can correspond to the maximum amount of time that an employee can work within a day or week as enforced by government regulations and labor union rules.

$$
\begin{aligned}
t_e^w &= \sum_{r \in R(e)} d_{er}^w x_{er} \quad \forall e \in E, w \in W(e) && (3)\\
t_e^w - o_e^w + u_e^w &= c_e^w \quad \forall e \in E, w \in W(e) && (4)\\
t_e^w &\leq d_{ew}^{max} \quad \forall e \in E, w \in W(e) && (5)
\end{aligned}
$$

An important issue in employee scheduling is to ensure fairness among employees with respect to working on weekends and other holidays. We denote the set of holidays within the planning horizon by H and the set of holidays that employee e can work on by $H(e) \subseteq H$. We define y_e^h for each $e \in E, h \in H(e)$ to indicate whether e is assigned to work during holiday h. Constraints (6) relate y_e^h-variables with x-variables. We also define variable y_e to count the total number of holidays that e is assigned to work and represent them in terms of y_e^h-variables in (7).

$$
\begin{aligned}
y_e^h &\geq x_{er} \quad \forall e \in E, h \in H(e), r \in R(e) \cap R(h) && (6)\\
y_e &= \sum_{h \in H(e)} y_e^h \quad \forall e \in E && (7)
\end{aligned}
$$

Note that y_e^h- and y_e-variables can be relaxed as continuous variables since the binary x-variables force y_e^h- and y_e-variables to binary and integer values, respectively. We also define variables y_l^{max} and y_l^{min}, respectively, to denote the maximum and minimum number of holidays that an employee belonging to skill level l is assigned to work. Note that we aim to ensure fairness separately within each skill level since availability of employees for some skill levels are highly constrained, while the average number of holidays that employees need to work is lower for other skill levels. Constraints (8) and (9) relate y_l^{max} and y_l^{min} variables to y_e-variables.

$$
\begin{aligned}
y_l^{max} &\geq y_e \quad \forall l \in L, e : L_e = l && (8)\\
y_l^{min} &\leq y_e \quad \forall l \in L, e : L_e = l && (9)
\end{aligned}
$$

Once again, y_l^{max} and y_l^{min} can be relaxed as continuous variables since objective function (17) ensures that $y_l^{max} = \max_{e:L_e=l}\{y_e\}$ and $y_l^{min} = \min_{e:L_e=l}\{y_e\}$ for all $l \in L$ in an optimal solution.

An additional concern regarding ensuring fairness is that the number of service requirements that coincides with a holiday that each employee is assigned to needs to be balanced. For this purpose we define z-variables that are analogous to the corresponding y-variables, and enforce the following constraints:

$$\begin{aligned} z_e &= \sum_{h \in H(e)} \sum_{r \in R(e) \cap R(h)} x_{er} \quad \forall e \in E && (10) \\ z_l^{max} &\geq z_e \quad \forall l \in L, e : L_e = l && (11) \\ z_l^{min} &\leq z_e \quad \forall l \in L, e : L_e = l. && (12) \end{aligned}$$

We next discuss our model's objective functions. Our first objective is to minimize the weighted total priority of unmet requirements. This can be modeled by the following objective:

$$\text{Minimize} \sum_{r \in R} p_r f_r. \tag{13}$$

Even though employees can be assigned to requirements needing a lower skill level than their own skill level, it is desired to avoid making such substitutions as much as possible. This can be modeled by objective (14):

$$\text{Minimize} \sum_{e \in E} \sum_{r \in R(e)} (L_e - L_r^{min}) x_{er}. \tag{14}$$

A significant goal of planning is to make assignments such that the contracted time durations of employees are utilized while also minimizing the amount of overtime usage. Objectives (15) and (16) minimize positive and negative deviations from the contracted time, respectively.

$$\text{Minimize} \sum_{e \in E} \sum_{w \in W(e)} o_e^w \tag{15}$$

$$\text{Minimize} \sum_{e \in E} \sum_{w \in W(e)} u_e^w \tag{16}$$

We also define the following two objectives to ensure a balanced assignment of employees to requirements coinciding with holidays. These objectives reflect the *fairness* concern of the management, and mathematically we formulate fairness as the difference between the largest and smallest load among employees within the same skill group.

$$\text{Minimize} \sum_{l \in L} y_l^{max} - y_l^{min} \tag{17}$$

$$\text{Minimize} \sum_{l \in L} z_l^{max} - z_l^{min} \tag{18}$$

In our implementation, we multiply each objective by a weight that is determined by the decision maker and minimize a weighted combination of these objectives. Each combination of weights is considered a scenario as described in detail in Section 5, which can be analyzed and compared to other scenarios in terms of key performance indicators.

4 Model Improvements

In this section we discuss how the model described in the previous section can be improved. We first observe that the rest constraints (2) dominate the size of the formulation. Recall that (2) are written for each employee and pairs of requirements that the employee cannot be assigned to without violating the rest constraint. There is an $O(|E||R|^2)$ number of such constraints and they can be generated in $O(|E||R|^2)$ time. Note that the number of such constraints grows rapidly as the number of employees $|E|$ and the number of requirements $|R|$ increase and can reach hundreds of thousands for a medium size company employing ≈ 50 employees and for a relatively short planning horizon, where the number of requirements is ≈ 500. As we will describe next, a reformulation allows the number of rest constraints to be reduced substantially while also significantly tightening the formulation. We also propose symmetry breaking constraints and valid inequalities to enhance solvability of our model.

4.1 Reformulation of the Rest Constraints

Consider an employee $e \in E$. Let us represent each requirement in $R(e)$ by a node in an undirected graph $G(e)$, where we add an edge $(\bar{r}, \hat{r})$ if and only if assigning both $\bar{r}$ and $\hat{r}$ to e violates the rest constraint. We observe that any feasible assignment of e to requirements in $R(e)$ corresponds to an *independent set* (also called *node packing* or *stable set*) on $G(e)$. The problem of finding an independent set of maximum cardinality is called the maximum independent set problem, and is known to be NP-Hard (Garey and Johnson, 1979). The maximum independent set problem has been studied extensively from graph theoretical and mixed-integer programming points of view (e.g., Beigel, 1999; Johnson et al., 1988; Padberg, 1973). Constraints (2) correspond to a natural, edge-based formulation of the independent set problem given in Padberg (1973). Let C denote a clique in $G(e)$, which is defined as a subset of nodes that have edges between all pairs of nodes. In our problem, a clique corresponds to a subset of requirements such that no two requirements in the clique can be assigned to the same employee without violating the rest constraints. The following constraint is valid for any clique C, and is facet-defining if C is maximal (see Padberg, 1973):

$$\sum_{r \in C} x_{er} \leq 1. \tag{19}$$

Since a pair of adjacent nodes forms a clique consisting of two nodes, constraints (2) are special cases of (19). Furthermore, a constraint (19) written for clique C is dominated by a constraint of the same type written for clique C' if $C \subset C'$. Therefore, it is enough to consider constraints (19) only for maximal cliques, which are not contained in any other clique. Let $\mathcal{C}(e)$ denote the set of all maximal cliques of $G(e)$. Then, the rest constraints (2) can be replaced by

$$\sum_{r \in C} x_{er} \leq 1 \quad \forall e \in E, C \in \mathcal{C}(e), \tag{20}$$

which yields a significantly tighter formulation (see also Butenko, 2003). A potential concern is that in general there is an exponential number of maximal cliques in a graph and cutting-plane algorithms are often needed to solve the resulting formulation. However, as we will show next, the structure of our problem allows us to efficiently generate all constraints (20).

4.2 Enumerating All Maximal Cliques

Let t denote an arbitrary time instant within the planning horizon. Consider an employee e and let $C(e,t)$ denote the set of requirements whose start time is before t and finish time plus the rest duration is after t. Formally, we define

$$C(e,t) = \{r : r \in R(e), S_r \leq t < F_{er} + b_{er}\}. \tag{21}$$

Note that no two requirements in $C(e,t)$ can be assigned to e without violating the rest constraint. Hence, by definition, $C(e,t)$ is a clique in $G(e)$. In principle, we can identify all cliques of $G(e)$ by sweeping t through the planning horizon. Note that $C(e,t) = C(e,t')$

for any $t' > t$ if no requirement has a start time or end time plus rest duration between t and t'. In other words, $C(e,t) = C(e,t')$ if there exists no r such that $t < S_r \leq t'$ or $t \leq F_{er} + b_{er} < t'$. Therefore, it is sufficient to consider only time instants t corresponding to S_r and $F_{er} + b_{er}$ for some r to enumerate all cliques. Based on these observations, we propose an algorithm in the spirit of discrete event system simulation approach, in which each "arrival" event corresponds to the start time of a requirement and each "departure" event corresponds to the finish time of the requirement plus the rest period. Algorithm 1 formally defines our approach for generating all maximal cliques.

Table 3: Execution of Algorithm 1 on the illustrative example

Event	$\mathcal{C}$ (All maximal cliques)	C (Current clique)	$flag$
Initialization	$\emptyset$	$\emptyset$	0
$(r_1, t_1, Arrival)$	$\emptyset$	$\{r_1\}$	1
$(r_2, t_2, Arrival)$	$\emptyset$	$\{r_1, r_2\}$	1
$(r_3, t_3, Arrival)$	$\emptyset$	$\{r_1, r_2, r_3\}$	1
$(r_1, t_4, Departure)$	$\{(r_1, r_2, r_3)\}$	$\{r_2, r_3\}$	0
$(r_4, t_5, Arrival)$	$\{(r_1, r_2, r_3)\}$	$\{r_2, r_3, r_4\}$	1
$(r_5, t_5, Arrival)$	$\{(r_1, r_2, r_3)\}$	$\{r_2, r_3, r_4, r_5\}$	1
$(r_3, t_6, Departure)$	$\{(r_1, r_2, r_3), (r_2, r_3, r_4, r_5)\}$	$\{r_2, r_4, r_5\}$	0
$(r_2, t_7, Departure)$	$\{(r_1, r_2, r_3), (r_2, r_3, r_4, r_5)\}$	$\{r_4, r_5\}$	0
$(r_4, t_8, Departure)$	$\{(r_1, r_2, r_3), (r_2, r_3, r_4, r_5)\}$	$\{r_5\}$	0
$(r_5, t_8, Departure)$	$\{(r_1, r_2, r_3), (r_2, r_3, r_4, r_5)\}$	$\emptyset$	0

Table 3 shows the steps of Algorithm 1 on the illustrative example given in Figure 1. Algorithm 1 detects maximal cliques (r_1, r_2, r_3) and (r_2, r_3, r_4, r_5). Hence, our clique-based rest constraints (20) for employee e can be written as:

$$x_{e1} + x_{e2} + x_{e3} \leq 1$$
$$x_{e2} + x_{e3} + x_{e4} + x_{e5} \leq 1.$$

Recall that eight constraints are needed to enforce the original rest constraints (2) on the same problem instance.

Algorithm 1 EnumerateMaximalCliques(e)

```
Input: $e \in E$ {Current employee}
{This algorithm enumerates all maximal cliques associated with employee $e \in E$}
$\mathcal{C}(e) \leftarrow \emptyset$   {$\mathcal{C}(e)$ represents the set of all maximal cliques associated with $e$}
$EL \leftarrow \{(r, S_r, Arrival) : r \in R(e)\} \cup \{(r, F_{er} + b_{er}, Departure) : r \in R(e)\}$
{$EL$ represents the list of events to be considered, where each event is represented as a
triplet of form (associated requirement, event time, event type), and an $Arrival$ and a
$Departure$ event is associated with each requirement.}
sort $EL$ in nondecreasing order of event time, breaking ties by placing $Departure$ events
before $Arrival$ events.
$C \leftarrow \emptyset$   {$C$ represents the current clique}
$flag \leftarrow 0$   {$flag = 1$ indicates that the previously processed event was an $Arrival$ event}
for all $(r, time, type) \in EL$ do
  if $type = Arrival$ then
    $C \leftarrow C \cup \{r\}$
    $flag \leftarrow 1$
  else
    if $|C| > 1$ and $flag = 1$ then
      {$C$ represents a maximal clique}
      $\mathcal{C}(e) \leftarrow \mathcal{C}(e) \cup C$
    end if
    $C \leftarrow C - \{r\}$
    $flag \leftarrow 0$
  end if
end for
return $\mathcal{C}(e)$
```

The most expensive step in Algorithm 1 is the sorting step. Therefore, the algorithm executes in $O(|R|\log|R|)$ time for each employee. Since no cliques are generated while processing *Arrival* events and at most one clique is generated while processing each *Departure* event, $O(|R|)$ cliques are generated for each employee. Therefore, the total number of constraints (20) is $O(|E||R|)$ and they can be generated in $O(|E||R|\log|R|)$ time. Comparing our clique-based rest constraints (20) with the original rest constraints (2), we observe that

1. The number of constraints needed is reduced from $O(|E||R|^2)$ to $O(|E||R|)$,
2. The amount of time needed to generate the rest constraints is reduced from $O(|E||R|^2)$ to $O(|E||R|\log|R|)$,
3. The formulation is significantly tightened since (20) correspond to non-dominated, facet-defining inequalities,
4. Constraints (20) define specially ordered sets of type 1 (SOS1) on the corresponding x-variables, which can be exploited by integer programming solvers to apply specialized branching rules (see Wolsey, 1998).

4.3 Breaking Symmetry

It is well known that existence of symmetry in the formulation can make integer programming problems very difficult to solve. See Sherali and Smith (2001) for background information on symmetry breaking in integer programming and Punnakitikashem et al. (2008) for an application on a nurse scheduling problem. Symmetry often results from the existence of indistinguishable objects that are modeled explicitly in the formulation. If symmetry exists in the formulation, equivalent solutions can be obtained from each other by swapping values of decision variables corresponding to the indistinguishable objects. Recall that the class of problems that we consider have employee-specific contracts, which significantly reduces symmetry. However, we observe that there is still some symmetry in our formulation. In particular, consider two employees $\hat{e}$ and $\bar{e}$ belonging to the same skill level L_e. Assume that requirements $\hat{r}$ and $\bar{r}$ start and end at the same time, and that their minimum required skill levels are equal to a common skill set $L_r^{min} \leq L_e$. In this case, if $\hat{e}$ is assigned to $\hat{r}$ and $\bar{e}$ is assigned to $\bar{r}$, a symmetric solution can be obtained by simply swapping the assignments. In order to eliminate such symmetric solutions, let us define a unique ID for each requirement, which we denote by $ID_r, \forall r \in R$, and for each employee, which we denote by ID_e. Then, we can add the symmetry breaking constraints (22):

$$x_{\hat{e}\bar{r}} + x_{\bar{e}\hat{r}} \leq 1, \quad \forall \hat{e}, \bar{e} \in E, \hat{r}, \bar{r} \in R(\hat{e}) \cap R(\bar{e}), \\ S_{\hat{r}} = S_{\bar{r}}, F_{e\hat{r}} = F_{e\bar{r}}, L_{\hat{r}}^{min} = L_{\bar{r}}^{min}, ID_{\hat{e}} < ID_{\bar{e}}, ID_{\hat{r}} < ID_{\bar{r}}. \tag{22}$$

4.4 Valid Inequalities

Valid inequalities are widely used to improve the solvability of integer programming formulations by eliminating some solutions while ensuring that at least an optimal solution remains within the feasible region. A class of valid inequalities can be derived for our problem by noting that substitution among employees having different skill levels needs to be avoided if possible. Consider two employees $\hat{e}$ and $\bar{e} \in E$ such that $L_{\hat{e}} > L_{\bar{e}}$. Similar to our symmetry breaking rule, let $\hat{r}$ and $\bar{r}$ be two requirements that start and end at the same time, with $L_{\hat{r}}^{min} > L_{\bar{r}}^{min}$. Then, employee $\hat{e}$ has to be assigned to requirement $\hat{r}$ in an optimal solution. We can write this observation as a valid inequality as follows:

$$x_{\hat{e}\bar{r}} + x_{\bar{e}\hat{r}} \leq 1, \quad \forall \hat{e}, \bar{e} \in E, L_{\hat{e}} > L_{\bar{e}} \\ \hat{r}, \bar{r} \in R(\hat{e}) \cap R(\bar{e}), S_{\hat{r}} = S_{\bar{r}}, F_{e\hat{r}} = F_{e\bar{r}}, L_{\hat{r}}^{min} > L_{\bar{r}}^{min}. \tag{23}$$

4.5 Improved Model

Our improved model, which uses the improved rest constraints (20) instead of (2) and adds symmetry breaking constraints (22) and valid inequalities (23) can be summarized as:

$$\begin{aligned}
\text{Minimize } & (13)\text{–}(18) \\
\text{subject to: } & (1), (3)\text{–}(12), (20)\text{–}(23) \\
& x_{er} \in \{0,1\} \quad \forall e \in E, r \in R(e) \\
& f_r \geq 0 \quad \forall r \in R \\
& t_e^w, o_e^w, u_e^w \geq 0 \quad \forall e \in E, w \in W(e) \\
& y_e, z_e \geq 0 \quad \forall e \in E \\
& y_e^h \geq 0 \quad \forall e \in E, h \in H(e) \\
& y_l^{max}, y_l^{min}, z_l^{max}, z_l^{min} \geq 0 \quad \forall l \in L.
\end{aligned}$$

5 Implementation

We implemented a decision support system based on the mixed-integer programming formulation described in the previous sections for a large health-care organization in Belgium. The organization has approximately 1500 employees in 17 locations, which are organized in three main regions. Figure 2 shows the organizational structure of the health-care organization. In order to assess the viability of our approach for a pilot study, we obtained data from the organization representing one of the larger locations. Our decision support system was implemented using ICRON Advanced Planning and Scheduling system (www.icrontech.com). ICRON provides a visual algorithm modeling environment that is based on the object-oriented design paradigm. It has extensive support for database and ERP systems integration as well as interfaces to various mixed-integer programming solvers including GLPK, Cbc, CPLEX and Gurobi.

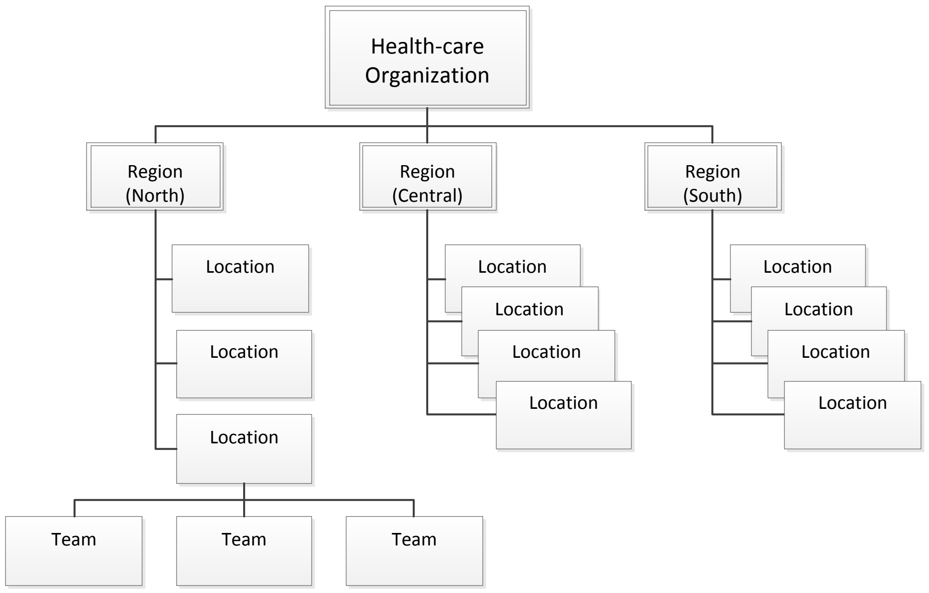

Figure 2: Organizational structure of the health-care organization

To obtain a realistic estimate of the effectiveness of our approach in a rolling horizon, we obtained 10 snapshots of the planning database over a time span of two months in 2014. While contents of these data sets are correlated, they provide sufficiently accurate information about the magnitude and dynamics of the planning problem. There are approximately

100 employees in the location that we considered for our pilot study. These employees are organized in five "teams," where each team consists of approximately 20 employees. Our analysis revealed that the planning problem can be separated over teams. Therefore, instead of solving a single optimization problem over 100 employees, a separate problem can be solved for each team. Hence, the size of the optimization problem to be solved is significantly reduced. Each data set contains approximately 3500 service requirements over a four-month planning horizon. Requirements have various lenghts ranging from 30 minutes to 8 hours. Employees are organized around ten skill levels and highly skilled employees are eligible for up to 800 requirements. Each employee has approximately 200 CTIs within the planning horizon representing daily, weekly, monthly working hour limits and employee unavailabilities. The employee assignment problem for each team consists of approximately 15,000 decision variables, out of which approximately 12,000 are binary assignment variables.

Our initial formulation based on our original rest constraints (2) takes several minutes to generate and contains approximately 55,000 constraints. During our computational tests that we performed on a PC with 4 GB memory and 2.13 GHz CPU using real problem data, we observed that the initial optimality gap with the original rest constraints (2) was 40%-50% on average, and commercial optimization solvers were able to reduce the optimality gap to 20%-25% after 30 minutes of computational time. On the other hand, our improved model based on our clique-based rest constraints (20) contains approximately only 10,000 constraints on the same data set. Our tests showed that it takes less than one minute to generate the model, and solutions within 1% of optimality can be found using default options of commercial solvers within one minute for most problem instances. Hence, the planning problems for all teams can be solved within approximately 15 minutes.

Following the pilot study, our decision support system has been extended into a full-fledged employee scheduling and dispatching system. Figure 3 represents the architecture of our decision support system in relation to the organization's existing information technology landscape. Unlike the initial implementation of our decision support system as a standalone application in the pilot phase, its final implementation consists of several components. The distinction of data and application logic is achieved by having separate environments for database and ICRON. The database environment contains a Planning Database, which is a Microsoft SQL Server database. Planning database stores planning-related information such as employee assignments, planning scenarios and key performance indicators (KPI) in addition to providing an integration point for existing backoffice systems that store global employee data such as employee contracts and availability, service requirements, holiday

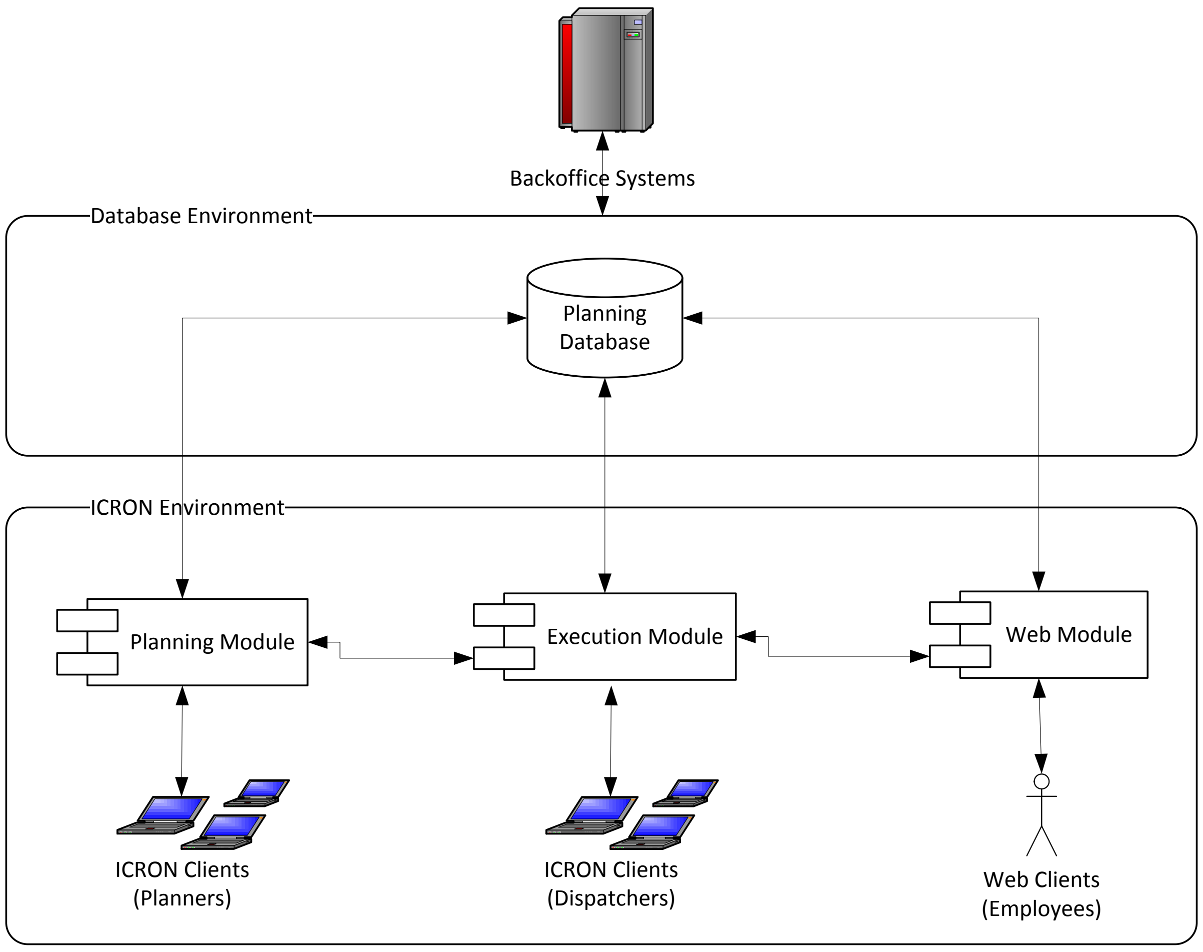

Figure 3: Architecture of our decision support system

calendars, etc. Planning Module, which has been implemented in ICRON, is based on our optimization model. It reads information about employees such as skill levels and contract details, service requirements and various business rules from the Planning Database. It then processes these pieces of data to generate an object model of the planning problem in computer's memory, in which each employee, service requirement, etc. is represented as an object. Next, various relationships between these objects (such as the set of requirements that can be performed by each employee) are constructed. Planning Module then generates the mathematical programming formulation given in Section 4.5, passes it to the selected solver and then reads the solution returned by the solver. In the organization there are approximately 100 planners. Each planner is responsible of managing a number of teams in a location. More than one planner may be assigned to a team depending on the size and the dynamics of the team. Planners interact with the Planning Module via a client application, also developed in ICRON, that visualizes the optimal solution in various formats including Gantt charts and tabular reports.

Our decision support system allows planners to generate various scenarios and analyze them in terms of KPIs. KPIs have been identified by the organization's management as: i) percentage of satisfied requirements, ii) percentage of requirements assigned to an employee whose skill level exactly matches its minimum required skill level, iii) total underutilized employee-hours below contracted time, iv) total overutilized employee-hours above contracted time, v) maximum percentage difference between the largest and smallest load among employees within the same skill group. Note that these KPIs are in alignment with our optimization model's objective functions (13)–(18). A set of objective function weights that generate good schedules in practice have been determined during the implementation project. During a planning session, planners can either use these default weights, or can specify the priority of each KPI as a number between 0 and 1. Our DSS then calculates normalized weights of the corresponding objective functions, runs the optimization model and stores its solution along with KPI values as a scenario. Planners can also manually modify a solution generated by our model, and store it as a scenario. Several scenarios generated by choosing different KPI priorities or via manual modifications can be compared within our DSS. This capability allows planners to rapidly generate several alternative plans and compare them. Note that evaluating and comparing different schedules by investigating individual assignments suggested by our DSS would be very difficult. Instead, planners evaluate schedules generated by our DSS and their manually generated schedules in terms of KPIs listed earlier. During their training, planners learn how KPIs react to changes in objective weights, and use this capability to guide the system towards preferred solutions if needed.

Once a consensus is reached between planners, the generated plan is written to the Planning Database and is transferred to the Execution Module. Execution Module is also implemented in ICRON, and its main responsibility is to facilitate operational level dispatching. The plan generated by the Planning Module may need to be modified at the operational level due to various factors such as last-minute changes in employee availability and service requirements, and tasks taking longer than expected to complete. Dispatchers track the progress of plan at the operational level and make minor adjustments to the plan as needed throughout the day. Similar to the Planning Module, Execution Module interacts with dispatchers via a client application.

Remaining employees interact with the decision support system via the Web Module, which is an ASP.NET application. Employees have access to their up-to-date schedules via interactive web pages. Furthermore, they can also submit requests for holidays and their preferences via the Web Module. Such requests and preferences are written to the Planning Database, transferred to Planning Module and Execution Module, and are taken into account in future planning sessions.

We have observed various advantages of the modular design of our decision support system throughout design, implementation and roll-out phases. In particular, the fact that each system component has distinct, well-defined roles and related user groups has enabled the project team to work in smaller focused groups. For instance, the Planning Module is based on our optimization model and is used by planners while the Web Module is based on web-based reports and is used by employees who are not involved in planning. This separation of responsibilities has allowed multiple sub-teams to work in parallel in implementation, testing and user training.

6 Summary and Conclusion

In this paper, we discussed an employee scheduling problem that is frequently faced in a group of health-care organizations, with flexible demand and employee availability. The driving factor for the operating environment for these organizations is the flexibility of service demands in terms of timing, content and length of service. In such a complex demand environment, managing the employees with vastly different skills, which constitutes a major operational cost item, under traditional shift structures generates an overwhelming burden on the budget. A resolution to this conundrum is obtained by following a recent trend in European countries regarding employee management: flexible contracts. Flexible employee contracts provide possibility to design the work hours of an employee around a set of work principles rather then around a relatively static shift structure.

We proposed an integer programming model to solve the employee scheduling problem that arises in this context: demand is flexible and flexible employee contracts define the availability of employees. In our model, demand for a service is defined as a service requirement that has a starting time, a finishing time, a priority and a minimum skill level required to perform that service. Employees are assigned to these service requirements based on their skill levels and the conditions set by their contracts. Our objectives are minimizing weighted unsatisfied service requirements, maximizing the utilization of contracted hours, maximizing employee satisfaction by assigning requirements that are consistent with their skill levels and maximizing fairness between employees by making balanced work assignments during holidays.

Our model diverges from the main line of research in the literature in two basic properties: firstly, we consider a continuous time scale in the temporal definitions regarding service requirements and employee contracts. The general approach in the literature is to discretize time into buckets. Having large time buckets reduces the expressiveness of the model in representing the temporal constraints and variables; whereas small buckets increase the size of the model and prohibits efficient solution procedures. Our approach in modeling the time eliminates such difficulties. The second novel property of our model is that we do not restrict the assigned working times of an employee based on a shift structure. Availability restrictions of the employee is determined by the employee contracts. Likewise, we make optimal work assignment decisions based on the work and rest conditions imposed by the contracts. Hence, even without an underlying shift structure, our model generates an applicable and satisfactory work environment for both the service organization and the flexible employee.

In addition to a basic integer programming model, we also proposed a number of reformulations to tighten and reduce the size of the formulation so that it can be used to efficiently solve real size problems. The proposed model is implemented and tested on a series of real size data sets provided by a large health-care organization in Belgium. Our experience shows that each instance of the integer programing model provides solutions within 1% optimality gap under one minute, and the full employee planning process of the health-care organization can be completed in around 15 minutes. While our model was initially implemented as a stand-alone application, it has later been extended into a full-fledged employee scheduling and dispatching system for organization-wide roll-out.

Our model can be extended in a number of ways. In particular, one extension could be to explicitly minimize irregularity between working hours of employees on their work days. This can be realized by adding variables that count the amount of working hours assigned to each employee for his/her working days, and adding an objective function that minimizes the difference between the minimum and maximum assigned working hour variables.

Acknowledgments. This research was partially supported by Boğaziçi University Research Fund grant no 5028 and TÜBİTAK TEYDEB grant no 3120788. We would like to thank Leonard C. Piet, Bart Bonekamp, Patrick Weyn (Déhora Consultancy Group), Taner Sezgin, Mahir Yüksel, Seda Lafcı, Meltem Kunt and Mert Ertürer (ICRON Technologies) for their valuable contributions in the implementation project.

References

Akbari, Mohammad, M. Zandieh, Behrouz Dorri. 2013. Scheduling part-time and mixed-skilled workers to maximize employee satisfaction. *International Journal of Advanced Manufacturing Technology* **64** 1017–1027.

Aykin, T. 1996. Optimal shift scheduling with multiple break windows. *Management Science* **42**(4) 591–602.

Azaieza, M.N., S.S. Al Sharif. 2005. A 0-1 goal programming model for nurse scheduling. *Computers & Operations Research* **32** 491–507.

Bard, J.F., H.W. Purnomo. 2005. Preference scheduling for nurses using column generation. *European Journal of Operational Research* **164**(2) 510–534.

Bechtold, S. E., L. W. Jacobs. 1996. The equivalence of general set-covering and implicit integer programming formulations for shift scheduling. *Naval Research Logistics (NRL)* **43** 233–249.

Bechtold, S.E., L.W. Jacobs. 1990. Implicit modeling of flexible break assignments in optimal shift scheduling. *Management Science* **36**(11) 1339–1351.

Beigel, R. 1999. Finding maximum independent sets in sparse and general graphs. *Proceedings of the tenth annual ACM-SIAM symposium on discrete algorithms*.

Berman, O., R.C. Larson, E. Pinker. 1997. Scheduling workforce and workflow in a high volume factory. *Management Science* **43**(2) 158–172.

Bhulai, S., G. Koole, A. Pot. 2008. Simple methods for shift scheduling in multi-skill call centers. *Manufacturing & Service Operations* **10**(3) 411–420.

Billionnet, A. 1999. Integer programming to schedule a hierarchical workforce with variable demands. *European Journal of Operational Research* **114**(1) 105–114.

Brunner, J.O. 2010. Literature review on personnel scheduling. *Flexible Shift Planning in the Service Industry*, *Lecture Notes in Economics and Mathematical Systems*, vol. 640. Springer Berlin Heidelberg, 5–12.

Burke, E.K., P. De Causmaecker, G. De Maere, J. Mulder, M. Paelinck, G. Vanden Berghe. 2010. A multi-objective approach for robust airline scheduling. *Computers & Operations Research* **37** 822–832.

Butenko, S. 2003. Maximum independent set and related problems, with applications. Ph.D. thesis, University of Florida.

Caprara, A., M. Monaci, P. Toth. 2003. Models and algorithms for a staff scheduling problem. *Mathematical Programming* **98** 445–476.

Castillo, I., T. Joro, Y.Y. Li. 2009. Workforce scheduling with multiple objectives. *European Journal of Operational Research* **196**(1) 162–170.

Cheang, B., H. Li, A. Lim, B. Rodrigues. 2003. Nurse rostering problems – a bibliographic survey. *European Journal of Operational Research* **151** 447–460.

Chu, S.C.K. 2007. Generating, scheduling and rostering of shift crew-duties: Applications at the Hong Kong international airport. *European Journal of Operational Research* **177**(3) 1764–1778.

Cote, M-C., B. Gendron, L-M. Rousseau. 2011. Grammar-based integer programming models for multiactivity shift scheduling. *Management Science* **57**(1) 151–163.

Dantzig, G. 1954. A comment on Edie's "Traffic delay at toll booths". *Operations Research* **2** 339–341.

De Grano, M.L., D. Medeiros, D. Eitel. 2009. Accommodating individual preferences in nurse scheduling via auctions and optimization. *Health Care Management Science* **12** 228–242.

den Bergh, Jorne Van, Jeroen Belien, Philippe De Bruecker, Erik Demeulemeester, Liesje De Boeck. 2013. Personnel scheduling: A literature review. *European Journal of Operational Research* **226**(3) 367–385.

Dietz, D.C. 2011. Practical scheduling for call center operations. *Omega* **39** 550–557.

Edie, L. 1954. Trafic delays at toll booths. *Journal Operations Research Society of America* **2**(2) 107–138.

Ernst, A.T., H. Jiang, M. Krishnamoorthy, B. Owens, D. Sier. 2004a. An annotated bibliography of personnel scheduling and rostering. *Annals of Operations Research* **127** 21–144.

Ernst, A.T., H. Jiang, M. Krishnamoorthy, D. Sier. 2004b. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research* **153** 3–27.

Felici, G., C. Gentile. 2004. A polyhedral approach for the staff rostering problem. *Management Science* **50**(3) 381–393.

Garey, M. R., D. S. Johnson. 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, New York.

Gopalakrishnan, B., E.L. Johnson. 2005. Airline crew scheduling: State-of-the-art. *Annals of Operations Research* **140** 305–337.

Johnson, D.S., M. Yannakakis, C.H. Papadimitriou. 1988. On generating all maximal independent sets. *Information Processing Letters* **27**(3) 119–123.

Joyce, K., R. Pabayo, J.A. Critchley, C. Bambra. 2010. Flexible working conditions and their effects on employee health and wellbeing. *Cochrane Database of Systematic Reviews* **2**.

Kwak, N. K., C. Lee. 1997. A linear goal programming model for human resource allocation in a health-care organization. *Journal of Medical Systems* **21** 129–140.

Li, J., E.K. Burke, T. Curtois, S. Petrovic, R. Qu. 2012. The falling tide algorithm: a new multi-objective approach for complex workforce scheduling. *Omega* **40** 283–293.

Meanhout, B., M. Vanhoucke. 2013. An integrated nurse stastaff and scheduling analysis for longer-term nursing staff allocation problems. *Omega* **41** 485–499.

Mohan, Srimathy. 2008. Scheduling part-time personnel with availability restrictions and preferences to maximize employee satisfaction. *Mathematical and Computer Modelling* **48** 1806–1813.

Narasimhan, R. 1997. An algorithm for single shift scheduling of hierarchical workforce. *European Journal of Operational Research* **96**(1) 113–121.

Padberg, M. W. 1973. On the facial structure of set packing polyhedra. *Mathematical Programming* **5** 199–215.

Parisio, A., C.N. Jones. 2015. A two-stage stochastic programming approach to employee scheduling in retail outlets with uncertain demand. *Omega* **53** 97–103.

Parr, D., J. Thompson. 2007. Solving the multi-objective nurse scheduling problem with a weighted cost function. *Annals of Operations Research* **155** 279–288.

Punnakitikashem, P., J.M. Rosenberger, D.B. Behan. 2008. Stochastic programming for nurse assignment. *Computational Optimization and Applications* **40**(3) 321–349.

Seckiner Ulusam, S., H. Gokcen, M. Kurt. 2007. An integer programming model for hierarchical workforce scheduling problem. *European Journal of Operational Research* **183**(2) 694–699.

Sherali, H. D., J. C. Smith. 2001. Improving discrete model representations via symmetry considerations. *Management Science* **47** 1396–1407.

Smith, B.M., A. Wren. 1988. A bus crew scheduling system using a set covering formulation. *Transportation Research Part A: General* **22**(2) 97–108.

Strupchanska, A. Kirilova, M. Penicka, D. Bjorner. 2003. Railway staff rostering. *FORMS2003: Symposium on Formal Methods for Railway Operation and Control Systems*.

Thompson, G.M. 1995. Improved implicit optimal modeling of the labor shift scheduling problem. *Management Science* **41**(4) 595–607.

Topaloglu, S. 2006. A multi-objective programming model for scheduling emergency medicine residents. *Computers & Industrial Engineering* **51**(3) 375–388.

Topaloglu, S. 2009. A shift scheduling model for employees with different seniority levels and an application in healthcare. *European Journal of Operational Research* **198**(3) 943–957.

Wolsey, L A. 1998. *Integer Programming*. Wiley-Interscience, New York, NY.

Wright, P.D., S. Mahar. 2013. Centralized nurse scheduling to simultaneously improve schedule cost and nurse satisfaction. *Omega* **41** 1042–1052.

Zolfaghari, S., A. El-Bouri, B. Namiranian, V. Quan. 2007. Heuristics for large scale labour scheduling problems in retail sector. *INFOR* **45**(3) 111–122.