Automated Shift Design and Break Scheduling

Luca Di Gaspero, Johannes Gärtner, Nysret Musliu,
Andrea Schaerf, Werner Schafhauser, and Wolfgang Slany

Abstract Shift design and break scheduling are important employee scheduling problems that arise in many contexts, especially at airports, call centers, and service industries. The aim is to find a minimum number of legal shifts, the number of workers assigned to them, and a suitable number of breaks so that the deviation from predetermined workforce requirements is minimized. Such problems have been extensively investigated in Operations Research and recently have been also tackled with Artificial Intelligence techniques. In this chapter we outline major characteristics of these problems and provide a literature survey over solution techniques to solve them. We then describe in detail two state-of-the-art approaches based on local search techniques. Finally, we discuss our experiences with the application of one of these techniques in a real life case study.

1 Introduction

Designing shifts and breaks is one of the most important phases in the general employee scheduling problem. The typical employee scheduling process in an organization consists of several stages. Given a planning period (often called the temporal horizon), in a first phase the temporal requirements are collected which then determine the number of employees needed for each day and time interval during the

Luca Di Gaspero, Andrea Schaerf
DIEGM, Università degli Studi di Udine, Italy, e-mail: {l.digaspero,schaerf}@uniud.it

Johannes Gärtner, Werner Schafhauser
XIMES XIMES GmbH, Austria, e-mail: {gaertner,schafhauser}@ximes.com

Nysret Musliu
DBAI, Technische Universität Wien, Austria, e-mail: musliu@dbai.tuwien.ac.at

Wolfgang Slany
IST, Technische Universität Graz, Austria, e-mail: wolfgang.slany@tugraz.at

planning period. After these requirements have been established, the shifts can be designed. At this stage the number of employees for each shift on each day in the planning period has to be determined. In some cases, moreover, for each employee assigned to a shift, a set of breaks must also be scheduled.

Shift design and break assignment problems are known also under other names, for example as *shift design* [19, 10], *shift scheduling* [3, 2, 25, 28], and *break scheduling* [4, 5, 20, 31]. Such problems arise at airports, call centers, and in the service industry in general, and have been extensively investigated in Operations Research and Artificial Intelligence.

On the one hand, the quality of solutions for shift design and break scheduling is particularly relevant because of the legal issues concerning the working time of employees, the well-being of employees, and the importance of reducing costs while guaranteeing a high level of service. In some working environments the quality of solutions is even critical: For example, an inadequate break assignment to air traffic controllers can be the cause of safety problems due to diminished concentration. On the other hand, the search space of these problems is big and they are subject to conflicting constraints which makes it practically impossible to tackle them by means of exact methods.

In this chapter, after the formal introduction, we will describe solving methods for these problems. In particular two state-of-the-art approaches based on local search techniques will be described in detail. Moreover, we will provide a real life case study that includes shift design. We will describe the solving process and present our experiences with the application of these techniques in a real life case study. Furthermore, we provide a literature survey over solution techniques for these problems.

2 The shift design problem

The shift design problem is a variant of the shift scheduling problem and has been first introduced in [12], [21], and [19]. The definition of this problem is given below (this definition is taken almost verbatim from [21] and [19]).

The input of the problem:

- n consecutive *time slots* $[a_1, a_2)$, $[a_2, a_3)$, …, $[a_n, a_{n+1})$, all with the same length *slotlength* in minutes. Time point a_1 represents the begin of the planning period and time point a_{n+1} represents the end of the planning period. A value w_i, related with each slot $[a_i, a_{i+1})$, indicates the optimal number of employees that should be present during that slot.
- y *shift types* $v_1, \ldots, v_y$. Each shift type v_j has the related parameters v_j.min-start and v_j.max-start, which represent the earliest and latest start of the shift, and v_j.min-length and v_j.max-length, which represent the minimum and maximum length of the shift. An example of shift types is given in Table 1.
- Two scalar real-valued quantities, necessary to define the distance from the average number of duties.

AS: the upper limit for the average number of working shifts per week per employee

AH: the average number of working hours per week per employee

Table 1: Possible shift types

Shift type	min-start	max-start	min-length	max-length
M	06:00	08:00	07:00	09:00
D	10:00	11:00	07:00	09:00
A	13:00	15:00	07:00	09:00
N	21:00	23:00	07:00	09:00

The aim is to generate a set of k shifts $s_1, \ldots, s_k$. Each shift s_l is completely determined by the parameters s_l.start and s_l.length and must belong to one of the shift types. Additionally, each shift s_p has a set of parameters $s_p.w_i, \forall i \in \{1, \ldots, C\}$ (C is the number of days in the planning period) indicating the number of employees assigned to shift s_p during day i.

The objective of the problem is to minimize the following four components:

F_1 : sum of the *excesses* of workers in each time slot during the planning period.
F_2 : sum of the *shortages* of workers in each time slot during the planning period.
F_3 : *number of shifts*.
F_4 : *distance* to the average number of duties per week in case it is above a certain threshold. This component is meant to avoid an excessive fragmentation of workload in too many short shifts.

The problem is a multi criteria optimization problem, in which each criterion has different importance depending on the situation. The objective function is the weighted sum of the four components.

Usually the designing shifts for a week is considered (less days are also possible) and the schedule is cyclic (the element following the last time slot of a planning period is equal to the first time slot of the planning period).

Formal definitions

The generated shifts belong to at least one of the shift types if:

$$\forall l \in \{1, \ldots, k\} \; \exists j \in \{1, \ldots, y\} \; /$$

$$v_j.\text{min-start} \leq s_l.\text{start} \leq v_j.\text{max-start}$$

$$v_j.\text{min-length} \leq s_l.\text{length} \leq v_j.\text{max-length}$$

In order to define the shortage and excess of workers, we first define the load l_d for a time slot d as

$$l_d = \sum_{p=1}^{k} x_{p,d}$$

where

$$x_{p,d} := \begin{cases} s_p.w_i & \text{if the time slot } d \text{ belongs to the interval of shift } s_p \text{ on day } i \\ 0 & \text{otherwise.} \end{cases}$$

The total excess F_1 and total shortage F_2 (in minutes) of workers in all time slots during the planning period is then defined as

$$F_1 = \sum_{d=1}^{n} (\max(l_d - w_d, 0) * slotlength)$$

$$F_2 = \sum_{d=1}^{n} (\max(w_d - l_d, 0) * slotlength)$$

Regarding F_3, this is simply equal to the number of shifts k.

The average number of working shifts per week per employee (*AvD*) is defined below:

$$AvD = \frac{(\sum_{i=1}^{k} \sum_{j=1}^{C} s_i.w_j) * AH}{\sum_{i=1}^{k} \sum_{j=1}^{C} s_i.w_j * s_i.length}$$

The penalty associated with the average number of duties is defined as.

$$F_4 = \max(AvD - AS, 0)$$

It is worth noticing that the criterion F_4 is not always taken into consideration in the literature.

A problem similar to this problem, called the *shift scheduling problem*, has been considered by several authors previously in the literature. We will describe later in the literature section the main differences between the shift design and shift scheduling problem.

3 Break scheduling

In this section we present a break scheduling problem which has been first mentioned in [5] and [26]. This problem appeared in the area of supervisory personnel, whereas a slightly different variant of the problem, in the case of call centers, has been solved in [4].

The definition of the break scheduling problem for supervisory personnel is as follows (this definition is taken almost verbatim from [5] and [26]):

In the same setting of the shift design problem, we are additionally given:

- Fixed shifts $(s_1, s_2, ..., s_n)$ representing employees working within the planning period. Each shift, s_i, has an adjoined parameter, $s_i.breaktime$, that specifies the required amount of break time for s_i in time slots.
- An employee is considered to be working during time slot $[a_t, a_{t+1})$ if that employee neither has a break during time slot $[a_t, a_{t+1})$ nor he/she has stopped working at time point. After a break, an employee needs a full time slot, usually 5 minutes, to become reacquainted with the altered situation. Thus, during the first time slot following a break, an employee is not considered to be working.

Similarly to shifts, also breaks b_j are characterized by two parameters, b_j.start and b_j.end, representing the time slots in which a shift or break starts and ends. Subtracting the value for start from the value for end gives the duration of shifts and breaks in time slots. The durations of shifts and breaks are stored in an additional parameter, duration. Moreover, each break is associated with a certain shift in which it is scheduled. We distinguish between two different types of breaks: lunch breaks and monitor breaks.

Formal definitions

Given a planning period, a set of shifts, the associated total break times, and the staffing requirements, a feasible solution to the break scheduling problem is a set of breaks with the following characteristics:

- Each break, b_j, lies entirely within its associated shift, s_i. That is,

$$s_i.\text{start} \leq b_j.\text{start} \leq b_j.\text{end} = s_i.\text{end}$$

- Two distinct breaks (b_j, b_k) associated with the same shift, s_i, do not overlap in time:

$$b_j.\text{start} \leq b_j.\text{end} \leq b_k.\text{start} \leq b_k.\text{end} \vee b_k.\text{start} \leq b_k.\text{end} \leq b_j.\text{start} \leq b_j.\text{end}$$

- In each shift, s_i, the sum of durations of its associated breaks equals the required amount of break time:

$$\sum_{b_j \in s_i} b_j.\text{duration} = s_i.\text{breaktime}$$

Criteria for finding an optimal solution

Among all feasible solutions for the break scheduling problem, we try to find an optimal one according to seven criteria, which we model as soft constraints:

C_1: Break Positions. Each break, b_j, may start, at the earliest, a certain number of time slots after the beginning of its associated shift s_i, and may end, at the latest, a given number of time slots before the end of its shift:

$$b_j.\text{start} \geq s_i.\text{start} + \text{distance to shift start}$$
$$b_j.\text{end} \leq s_i.\text{end} - \text{distance to shift end}$$

C_2: Lunch Breaks. A shift s_i can have several lunch breaks, each required to last a specified number of time slots (min lunch break duration), and should be located within a certain time window after the shift start. Let b_{lb} be a lunch break. Then,

$$b_{lb}.\text{start} \geq s_i.\text{start} + \text{distance to shift start}\, lb$$
$$b_{lb}.\text{end} \leq s_i.\text{end} - \text{distance to shift end}\, lb$$

C_3: Duration of Work Periods. Breaks divide a shift into work and rest periods. The duration of work periods within a shift must range between a required minimum and maximum duration:

$$\text{min work duration} \leq b_1.\text{start} - s_i.\text{start} \leq \text{max work duration}$$
$$\text{min work duration} \leq b_{j+1}.\text{start} - b_j.\text{end} \leq \text{max work duration}$$
$$\text{min work duration} \leq s_i.\text{end} - b_m.\text{end} \leq \text{max work duration}$$

where $(b_1, ..., b_j, b_{j+1}, ..., b_m)$ are the breaks of s_i in temporal order.

C_4: Minimum Break Times after Work Periods. If the duration of a work period exceeds a certain limit, the break following that period must last a given minimum number of time slots (min ts count):

$$b_1.\text{start} - s_i.\text{start} \geq \text{work limit} \Rightarrow b_1.\text{duration} \geq \text{min ts count}$$
$$b_{j+1}.\text{start} - b_j.\text{end} \geq \text{work limit} \Rightarrow b_{j+1}.\text{duration} \geq \text{min ts count}$$

where, once again, $(b_1, ..., b_j, b_{j+1}, ..., b_m)$ are the breaks of s_i in temporal order.

C_5: Break Durations. The duration of each break, b_j, must lie within a specified minimum and maximum value:

$$\text{min duration} \leq b_j.\text{duration} \leq \text{max duration}$$

C_6: Shortage of Employees. In each time slot, $[a_t, a_{t+1})$, at least r_t employees should be working.

C_7: Excess of Employees. In each time slot, $[a_t, a_{t+1})$, a_t most r_t employees should be working.

Objective function

For each constraint, we define a violation degree, $violation(C_k)$, specifying the deviation (in time slots or employees) from the requirements stated by the respective constraint. The importance of each criterion and its corresponding constraint varies from task to task. Consequently, the objective function for the break scheduling problem is the weighted sum of the violation degrees of all the constraints:

$$F(solution) = \sum_{k=1}^{7} W_k \times violation(C_k)$$

where W_k is a weight indicating the importance assigned to constraint C_k. Given an instance of the break scheduling problem, our goal is to find a feasible solution that minimizes this objective function.

4 Literature review

The shift design problem described in this chapter has been first introduced in [12], [21], and [19]. In these works a tabu search based algorithm to solve this problem is proposed. Additionally, their method orders moves and applies these first to regions with larger conflicts (larger over/under-staffing). The proposed solution methods have been used since several years in the commercial software package OPA of XIMES Inc. Di Gaspero et al. [14, 10] proposed the application of hybrid approaches based on local search and min-cost max-flow techniques. The hybrid algorithm improved results reported in [19] on benchmark examples.

A similar problem called shift scheduling problem has been extensively investigated in the literature. Although shift design problem and shift scheduling show some similarities there are some further specifics that characterize the shift design problem. In the shift design problem the number of employees for each shift over a whole week should be determined and the problem is cyclic (the shift starting on the last day that will last overnight will finish in the first day of the schedule). Furthermore, a different objective function is used for the shift design problem that also allows undercover and tries to minimize the number of used shifts.

Regarding the shift scheduling problem the first approaches have been proposed many years ago. Dantzig developed the original set-covering formulation [8] for the shift scheduling problem, in which feasible shifts are enumerated based on possible shift starts, shift durations, breaks, and time windows for breaks. Integer programming formulations for shift scheduling include [3], [29], and [1]. Aykin [1] introduced an implicit integer programming model for the shift scheduling problem in

1996 and later he compared an extended version [2] of his previous model with a similarly extended formulation introduced by Bechtold and Jacobs [3]. He observed that Bechtold and Jacobs' approach needed fewer variables whereas his approach needed fewer constraints. Several problems were solved using both models with the integer programming software LINDO. The model proposed by Aykin was shown to be superior.

Rekik et al. [25] developed two other implicit models and managed to improve upon previous approaches among them Aykin's original model. Tellier and White [28] developed a tabu search algorithm to solve a shift scheduling problem originating in contact centers which is integrated into the workforce scheduling system Contact Center Scheduling 5.5 from PrairieFyre Soft Inc. (`http://www.prairiefyre.com`). The solution of a shift scheduling problem with a planning period of one day, and at most three breaks (two 15 minutes breaks and a lunch break of 1 hour) has been considered in [24] and [6]. In [6] the authors make use of automata and context-free grammars to formulate constraints on sequences of decision variables. Quimper and Rousseau [24] investigate modeling of the regulations for the shift scheduling problem by using regular and context-free languages and solved the overall problem with Large Neighborhood Search. In addition to the previous model, the authors applied their methods in single and multiple activity shift scheduling problems. A new implicit formulation for multi-activity shift scheduling problems using context-free grammars has been proposed recently by Côté et al. [7].

Previously, break scheduling has been addressed mainly as part of the shift scheduling problem. Several approaches have been proposed for problem formulations that include a small number of breaks. These approaches schedule the breaks within a shift scheduling process. Such approaches include the previous mentioned methods for shift scheduling [8], [3], [29], [1], [2], [25], [28], [24], [6], [7]. Generation of three to four breaks per shift after the design of shifts with a greedy approach has been investigated in [13].

Some important break scheduling problems arising in call centers, airports, and other areas include a much higher number of breaks compared to the problem formulations in previous works on shift scheduling. Also, additional requirements like time windows for lunch breaks or restrictions on the length of breaks and work time emerged. These new constraints significantly enlarge the search space. Therefore, researchers recently started to consider a new approach which regards shift scheduling and break scheduling as two different problems and tries to solve them in separate phases.

The break scheduling problem that imposes same constraints for breaks defined in this chapter has been investigated in [5, 26, 20, 31, 30]. Beer et al. [5] applied a min-conflict based heuristic to solve this problem. This method has been applied in a real-life application for the supervision personnel. Beer et al. [5] also introduce real life benchmark instances containing shifts that include more than 10 breaks. The results presented in [5] have been further improved by memetic algorithms proposed in [20, 31, 30]. A similar break scheduling problem, which origins in call centers and includes also meetings and some slightly changed constraints, has been

described in [4] and [26]. Note that a simplified break scheduling problem can be formulated as temporal constraint satisfaction problem (STP) [9] therefore it can be solved in polynomial time. This algorithm can be applied to find legal position of breaks, but without taking into consideration over-staffing and under-staffing that are very important criteria when solving shift design and break scheduling problems.

Recently, an algorithm based on constraint programming and local search for solving of this break scheduling problem together with shift design has been investigated in [15]. This approach could not improve the results obtained by solving the break scheduling problem separately (after generating shifts).

5 Local search for shift design

Local search is a search paradigm which has evidenced to be very effective for a large number of AI problems [17]. This paradigm is based on the idea of navigating through the search space by iteratively stepping from one state to one of its "neighbors", which are obtained by applying a simple local change to the current solution.

The first local search algorithm for the shift design problem has been proposed in [12], [21], and [19]. These initial works proposed different move types that were used for the exploration of the neighborhood. Regarding the local search techniques, the basic principles of tabu search [16] were used to escape from local minima. In order to make the search more effective, a new method was introduced to explore only parts of the neighborhood. The search is focused on days in which a shortage or excess is present. Moreover, some of the moves are applied only to specific shift types in the region in which the shortage or excess appears. Additionally, an approach for generating a good initial solution, which is based on the idea of starting (ending) a shift whenever the requirements increase (decrease), was proposed. These techniques were implemented in a commercial product called Operating Hours Assistant, which has been used in different areas to solve shift design problems.

A hybrid local search approach for shift design has been presented in [10]. This solver comprises two stages, namely a greedy construction for the initial solution followed by a tabu search procedure [16] that iteratively improves it. The greedy construction method relies on the equivalence of the (non-cyclic) shift design problem to a variant of the Min-Cost Max-Flow network problem [23]. The greedy heuristic employs a polynomial subroutine which can easily compute the optimal staffing with minimum (weighted) deviation, but it is not able to simultaneously minimize the number of shifts used.

The second stage of the proposed heuristic is based on the local search paradigm and relies on multiple neighborhood relations. The local search model is defined by specifying three entities, namely the *search space*, the *neighborhood relation*, and the *cost function*. In details, the local search entities are described in the following.

5.1 Search space and initial solution

We consider as a state for shift design the set of shifts $Q = \{s_1, s_2, \ldots\}$ together with their associated parameters. The shifts of a state are split into two categories:

- *Active* shifts: at least one employee is assigned to a shift of this type on at least one day.
- *Inactive* shifts: no employees are assigned to a shift of this type on any day. These shifts do not contribute to the solution and to the objective function. Their role is explained later.

More formally, we say that a shift $s_i \in Q$ is active (resp. inactive) if and only if $\sum_{j=1}^{C} s_i.w_j \neq 0$ $(= 0)$.

5.2 Neighborhood relations

For tackling the shift design problem we consider three different neighborhood relations that are combined in the spirit of the multi-neighborhood search [11], and are a subset of those applied in [19] with some modifications.

Given a state Q of the search space the types of moves considered are the following:

ChangeStaff (CS): The staff of a shift is increased ($\uparrow$) or decreased ($\downarrow$) by one employee. This kind of move is described as a triple $\langle s_i, j, a\rangle$, where $s_i \in Q$ is a shift, $j \in \{1, \ldots, C\}$ is a day, $a \in \{\uparrow, \downarrow\}$.

ExchangeStaff (ES): One employee in a given day is moved from one shift to another one of the same type. The move is described by the triple $\langle s_{i_1}, s_{i_2}, j\rangle$, where $s_{i_1}, s_{i_2} \in Q$, and $j \in \{1, \ldots, C\}$.

ResizeShift (RS): The length of the shift is increased or decreased by 1 time-slot, either on the left-hand side or on the right-hand side. The move is described by the triple $\langle s_i, l, p\rangle$, where $s_i \in Q$, $l \in \{\uparrow, \downarrow\}$, and $p \in \{\leftarrow, \rightarrow\}$. In order to apply this move, the shift s_i', obtained from s_i must be feasible with respect to its original type.

When the effect of the move will transform an inactive shift in an active one (e.g., by increasing the staff of an inactive shift at a given day), a new inactive shift of the same type is randomly created.

Inactive shifts allow us to insert new shifts and to move staff between shifts in a uniform way. This approach limits the creation of new shifts only to the current inactive ones, rather than considering all possible shifts belonging to the shift types (which are many more). The possibility of creating any legal shift is rescued if we insert as many (distinct) inactive shifts as compatible with the shift type. Experimental results, though, show that there is a trade-off between computational cost and search quality which seems to have its best compromise in having 2 inactive shifts per type.

5.3 Search strategies

Our local search solver is driven by tabu search. A full description of this technique is out of the scope of this chapter and we refer to the book of Glover and Laguna [16] to for a general introduction. We describe the specialization of this technique to our problem.

Differently from [19], that uses tabu search as well, we employ the three neighborhood relations selectively in various phases of the search, rather than exploring the overall neighborhood at each iteration.

Our strategy is to combine the neighborhood relations CS, ES, and RS, according to the following scheme made of compositions and interleaving. In detail, our algorithm interleaves three different tabu search *runners* using the following neighborhoods:

- the ES alone
- the RS alone
- the set-union of the two neighborhoods CS and RS

The runners are invoked sequentially and each one starts from the best state obtained from the previous one. The overall process stops when a full round of all of them does not find an improvement. Each single runner stops when it does not improve the current best solution for a given number of iterations (called *idle iterations*).

This composite solver is further improved by performing a few changes on the final state of each runner, before handing it over as the initial state of the following runner. In details, the modifications at the performed are the merge of identical shifts and the recreation of a suitable number of inactive shifts.

For all three runners, the size of the tabu list is dynamic and each move remains in this list for a random number of iterations selected within a given range. Moreover, the tabu status of a move is dropped if it leads to a state better than the current best, in accordance with the standard *aspiration* criterion of tabu search.

6 Local search for break scheduling

In this section we present in details a solution method based on local search for the break scheduling problem. The algorithm that will be described is based on the min-conflicts heuristic [18] and has been first proposed for break scheduling problem in [5], [4], and [26]. The following description is based on these references.

The solution of the break scheduling problem is represented as a set of breaks. For each shift, s_i, the breaks to be scheduled are instantiated at the beginning of a local search algorithm. At first, lunch breaks are generated, and then the remaining break time is distributed among monitor breaks. Hence, the duration of each lunch break is set to the exact number of time slots required by constraint C_2 (lunch

breaks), and the duration of each monitor break lies within the specified minimum and maximum limits imposed by constraint C_5 (break durations).

In the min-conflicts-based algorithm, the start of a break, b_j.start, is an integer variable that can be altered during the search process. In contrast, we require that the duration of a break, b_j.duration, remains unchanged and keeps its initially assigned value. However, we allow multiple breaks to be scheduled consecutively so that breaks of longer duration can be created.

6.1 Initial solution

Once the breaks are created, they must be placed in the given shift plan. We implemented two methods to schedule breaks within their associated shifts. The first simply schedules breaks randomly so that they do not overlap. The second schedules breaks so that the resulting break pattern completely satisfies constraints C_1 through C_5. This task is accomplished by formulating the problem as a simple temporal problem (STP) [9] and solving it by means of a randomized version of the Floyd-Warshall shortest-path algorithm [23].

A STP consists of a set of variables $X = X_1, \ldots, X_n$ and a set of constraints on those variables. The variables of an STP represent time points having continuous domains. Each constraint is represented as an interval that either restricts the domain values for a single variable X_i or restricts the difference $(X_j - X_i)$ of two distinct variables (X_i, X_j).

To schedule breaks correctly with respect to constraints C_1 through C_5, the start and end parameters of shifts and breaks are modeled as variables of an STP. The STP constraints will take care of the various limits imposed on break positions and on the duration of breaks and work periods. In detail the STP constraints are the following:

$$\begin{aligned} b_j.\text{start} &\in [(s_i.\text{start} + \text{distance to shift start}), s_i.\text{end}] \\ b_j.\text{end} &\in [s_i.\text{start}, (s_i.\text{end} - \text{distance to shift end})] \end{aligned} \qquad (C_1)$$

$$\begin{aligned} b_{lb}.\text{start} &\in [(s_i.\text{start} + \text{distance to shift start } lb), s_i.\text{end}] \\ b_{lb}.\text{end} &\in [s_i.\text{start}, (s_i.\text{end} - \text{distance to shift end } lb)] \end{aligned} \qquad (C_2)$$

$$\begin{aligned} b_1.\text{start} - s_i.\text{start} &\in [\text{min work duration}, \text{max work duration}] \\ b_{j+1}.\text{start} - b_j.\text{end} &\in [\text{min work duration}, \text{max work duration}] \\ s_i.\text{end} - b_m.\text{end} &\in [\text{min work duration}, \text{max work duration}] \end{aligned} \qquad (C_3)$$

$$b_1.\text{duration} \leq \text{min ts count} \iff b_1.\text{start} - s_i.\text{start} \in [\text{min duration}, \text{work limit}) \quad (C_4)$$

$$b_{j+1}.\text{duration} \leq \text{min ts count} \iff b_{j+1}.\text{start} - b_j.\text{end} \in [\text{min duration}, \text{work limit})$$

The two temporal constraints for C_4 are inserted if and only if $(b_1.\text{duration} \leq \text{min length})$ and $(b_{j+1}.\text{duration} \leq \text{min length})$, respectively. Constraint C_5 is satisfied by construction, since the solution creation process creates only breaks whose durations range between the required time limits.

6.2 Neighborhood relations

We developed two types of moves for the break scheduling problem. The first move (called assignment) assigns to a break a new start within its respective shift. The second move (denoted swap) exchanges the start times of two breaks associated with the same shift, meaning those breaks are actually swapped. Figure 1 illustrates these two moves. Given a feasible solution S to the break scheduling problem, the neighborhood $N(S)$ is the set of all solutions obtained by applying an assignment to a single break in S or by swapping two breaks within the same shift in S.

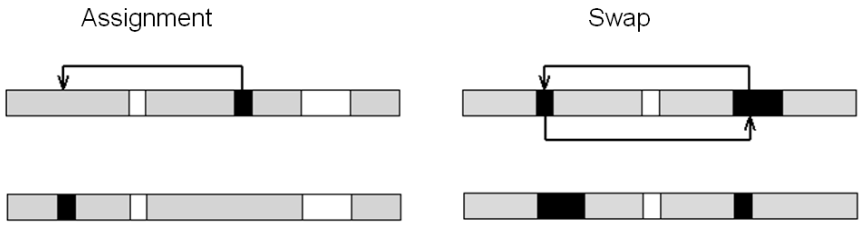


Fig. 1: The two moves used for the break scheduling problem. The assignment move assigns to a break a new start within its respective shift. The swap move exchanges the start times of two breaks associated with the same shift.

6.3 Min-conflicts heuristics

The minimum-conflicts heuristics aims at improving the current solution by concentrating only on the parts that cause constraint violations. During an iteration, the minimum-conflicts heuristic selects a break that violates a constraint and determines a move that decreases (or leave unchanged) the violation degree of the current solution. If such a move exists, it is applied to the current solution, and the search continues until some stopping condition is satisfied.

The minimum-conflicts search method applies only moves that do not decrease the current solution quality. Thus, if the search reaches a local optimum solution, the algorithm will not proceed any further, since it will not find solutions of better quality than the local optimum. To avoid this undesirable behavior, we apply an additional strategy known as random walk, which has been used successfully in algorithms for satisfiability problems [27], rotating workforce scheduling [22], etc.

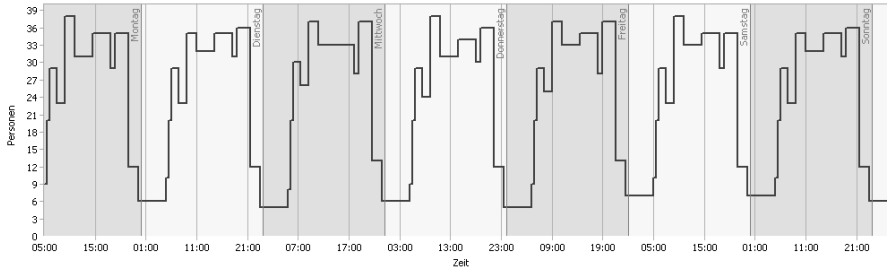
The random walk strategy also selects a break that violates a constraint. However, unlike the minimum-conflicts heuristic, it applies an arbitrary move to that break. On the one hand, the violation degree of the resulting solution could be worse than the previous one. However, on the other hand, performing such moves can help the algorithm to escape from local optima. We call the combination of both strategies min-conflicts random walk. The random walk strategy is carried out with a small probability p, whereas the ordinary minimum-conflicts search is carried out with a higher probability of $1 - p$. The specific value of p is determined experimentally.

7 A case study

In this section we consider a real life problem arising from the shift design domain. This example represents a particular sub-problem that was solved in an European airport. We selected this domain, because shift design problems in airports are of high practical relevance. To solve the particular shift design problem we apply the local search techniques proposed in [21, 19]. This method is included in a commercial scheduling system called Operating Hours Assistant(OPA) [12] owned by the XIMES Corp. OPA has been successfully applied by the consultants of the XIMES GmbH in many companies and institutions since more than ten years.

7.1 Temporal requirements

The temporal requirements for our case study are given for one week. The cycle should be considered (a night shift that begins on Sunday at 23:00 and is 8 hours long, impacts the first day of the week). Figure 2 shows the temporal requirements in tabular form and as a requirement curve. The first row of the table indicates that on Monday between 05:00-05:30 9 employees are required, on Tuesday 10 employees



Daten

Von	Bis	Mo	Di	Mi	Do	Fr	Sa	So
05:00	05:30	9,00	10,00	8,00	9,00	9,00	10,00	10,00
05:30	06:00	20,00	20,00	20,00	20,00	20,00	20,00	20,00
06:00	07:30	29,00	29,00	30,00	29,00	29,00	29,00	29,00
07:30	09:00	23,00	23,00	26,00	24,00	25,00	23,00	23,00
09:00	11:00	38,00	35,00	37,00	38,00	37,00	38,00	35,00
11:00	14:30	31,00	32,00	33,00	31,00	33,00	33,00	32,00
14:30	18:00	35,00	35,00	33,00	34,00	35,00	35,00	35,00
18:00	19:00	29,00	31,00	28,00	30,00	28,00	29,00	31,00
19:00	21:30	35,00	36,00	37,00	36,00	37,00	35,00	36,00
21:30	23:30	12,00	12,00	13,00	12,00	13,00	12,00	12,00
23:30	05:00	6,00	5,00	6,00	5,00	7,00	7,00	6,00

Fig. 2: Temporal requirements for the airport case study

are needed, etc. As we can see the number of employees varies during the planning period. The maximal number of employees in the planning period is 38, whereas the minimal number of employees is 5.

7.2 Shift types

In the next step we have to define the shift types that determine the possible start and length of the real shifts. Table 1 represents the four shift types that are used for this case study: M-morning shift, D-day shift, A-afternoon shift, and N-night shift. Shifts generated by the algorithm should fulfill the criteria given by the shift types. For example, morning shifts can start between 6:00 and 8:00 and their length should be between 7 to 9 hours.

7.3 Weights of the criteria

As explained in the problem definition the solution to the shift design problem is evaluated with an objective function, which combines four weighted criteria: excess in minutes, shortage in minutes, number of shifts, and distance from average number of duties per week. The weights of these criteria usually depend on the particular situation. In some cases it is very important to avoid completely the shortage of employees and in other cases the minimization of employees is more important. To

find a solution for our case study we will illustrate further two scenarios regarding the importance of criteria.

7.4 Generation of shifts

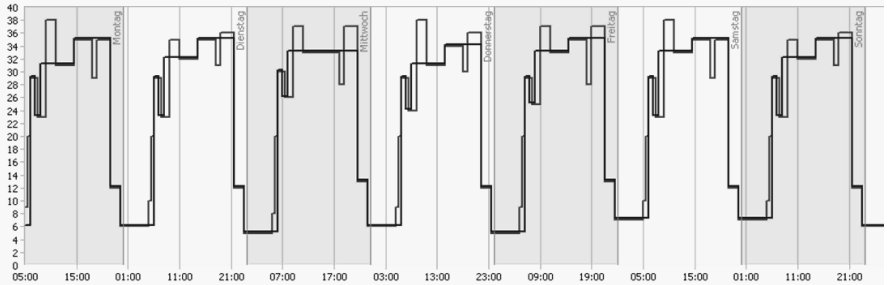
The local-search based algorithm for generating shifts iteratively improves the initial solution. OPA is an interactive software system that includes a graphical user interface that shows the improvements of the current solution and the relevant information regarding the overall shortage and excess, and the number of shifts. The decision maker can stop the algorithm at any time. If the decision maker is not satisfied with the current solution, he/she can change the weights and try to further improve the current solution.

Firstly, we assign weights to optimization criteria as follows: 1 - shortage, 1 - excess, 60 - number of shifts, 0 - distance from average number of duties per week. For the given requirements, constraints and the weights above, the local search algorithm generates the solution shown in Figure 3. This solution contains 8 shifts, the overall shortage is 4,2%, and the overall excess of employees is 2,32%. As we can see Figure 3 the shortage of employees appears in several time slots.

Note that the solution obtained by the Min-Cost Max-Flow algorithm has a total shortage of 3,63%, and a total excess of 2,45%. This polynomial algorithm computes the optimal staffing with minimum (weighted) deviation, however it is not able to simultaneously minimize the number of shifts used and the cycle is not taken into consideration. The number of shifts generated by this method is 24. Therefore, the solution obtained by the local search procedure is much better regarding the number of shifts, and regarding the covering of requirements it is only slightly worse than the solution obtained by the Min-Cost Max-Flow method.

In the second scenario we increased the weight of the shortage to 5. Other weights are not changed. Using these weights we can run the local search algorithm that uses the current solution as an initial solution. This algorithm finds a solution with 8 shifts that produce 1,49% shortage and 6,45% excess. The overall sum of the shortage and the excess is increased. However, the shortage which is now the most important criterion is significantly improved. The new generated solution is presented in Figure 4 (only the first day is shown). By increasing of the weight of the shortage, we could also find a solution with no shortage.

Such a result is quite typical for real life projects. It is not overall the minimization of shortage and excess but a multidimensional optimization problem with many factors to be considered. Typically users like to get an idea of consequences of weight changes (either by recalculation from scratch or by stepwise refinement of the solution at hand). This helps users to assess the potential and limits for further improvements and sometimes even causes a change in the requirements stated (. . . if there is no solution with . . . we cannot . . .). A quick optimization is crucial to facilitate such assessment process.



M1	Morning shift	06:00	14:30	23	23	25	23	24	22	22
M2	Morning shift	08:00	15:00	8	9	7	7	8	10	9
M3	Morning shift	07:00	14:30			1	1	1	1	1
A1	Afternoon shift	14:30	21:30	23	23	20	22	22	23	23
A2	Afternoon shift	15:00	23:30	6	7	7	7	6	5	6
A3	Afternoon shift	14:30	22:00	4	3	6	5	5	2	3
A4	Afternoon shift	15:00	22:00	2	2			2	5	3
N1	Night shift	22:00	07:00	6	5	6	5	7	7	6

Fig. 3: The shift design solution generated using local search

h

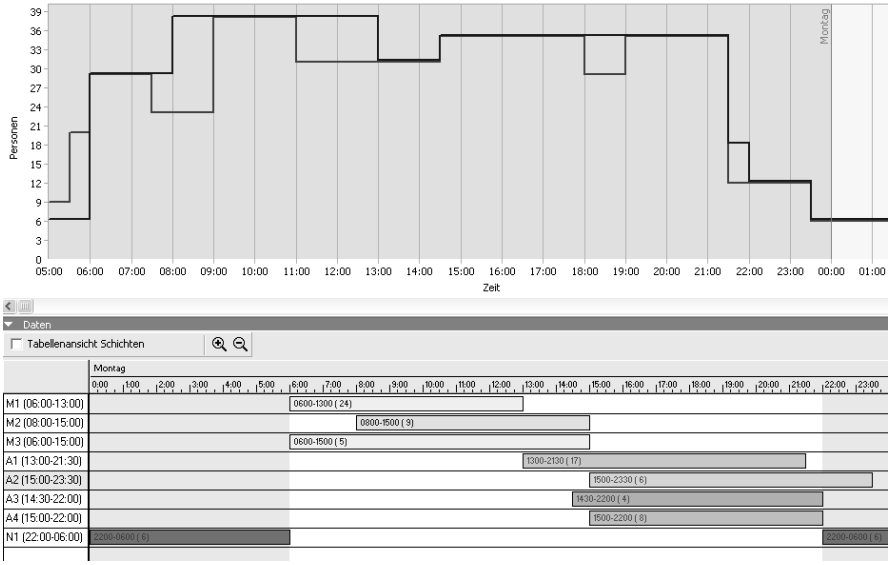


Fig. 4: The shift design solution generated by the local search. The weight of shortage is increased to 5

8 Conclusion

In this chapter we presented two real-life problems from the area of personnel scheduling. We gave a precise problem definition for shift design and break scheduling and then surveyed the existing literature for these two tasks and other related problems. The solution techniques based on local search that have been used successfully to solve different large real-life problems in these domains have been further introduced. Finally, we presented a case study and described the solution process in a commercial personnel scheduling system that exploits the local search techniques to solve the shift design problem.

The local search techniques presented in this chapter have been shown to be powerful tools for such problems. In our previous studies we have shown that the results of these techniques can be still improved by hybridization with other solution methods like network flow algorithms. Therefore, considering simpler problems that can be efficiently solved exactly can be very useful to get robust methods that combine the advantages of different solution paradigms.

Solving shift design problem and break scheduling (with a large number of breaks) simultaneously is still a challenging task. Although some work has been done in this direction, the results obtained by the approach that considers these two phases separately could still not be improved. Furthermore, it would be interesting to additionally consider the assignment of the shifts to the employees, include employee qualifications, and consider also the task assignment simultaneously.

Acknowledgments

This work was supported by the Austrian Science Fund (FWF): P24814-N23. Moreover, the research herein is partially conducted within the competence network Softnet Austria II (www.soft-net.at, COMET K-Projekt) and funded by the Austrian Federal Ministry of Economy, Family and Youth (bmwfj), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

References

1. T. Aykin. Optimal shift scheduling with multiple break windows. *Management Science*, 42:591–603, 1996.
2. T. Aykin. A comparative evaluation of modelling approaches to the labour shift scheduling problem. *European Journal of Operational Research*, 125:381–397, 2000.
3. S.E. Bechtold and L.W. Jacobs. Implicit modelling of flexible break assignments in optimal shift scheduling. *Management Science*, 36(11):1339–1351, 1990.
4. A. Beer, J. Gaertner, N. Musliu, W. Schafhauser, and W. Slany. Scheduling breaks in shift plans for call centers. In *Proceedings of The 7th International Conference on the Practice and Theory of Automated Timetabling*, Montreal, Canada, 2008.

5. Andreas Beer, Johannes Gärtner, Nysret Musliu, Werner Schafhauser, and Wolfgang Slany. An AI-based break-scheduling system for supervisory personnel. *IEEE Intelligent Systems*, 25(2):60–73, 2010.
6. Marie-Claude Côté, Bernard Gendron, Claude-Guy Quimper, and Louis-Martin Rousseau. Formal languages for integer programming modeling of shift scheduling problems. *Constraints*, 16(1):55–76, 2011.
7. Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based integer programming models for multiactivity shift scheduling. *Management Science*, 57(1):151–163, 2011.
8. G. B. Dantzig. A comment on Eddie's traffic delays at toll booths. *Operations Research*, 2:339–341, 1954.
9. Rina Dechter, Itay Meiri, and Judea Pearl. Temporal constraint networks. *Artificial Intelligence*, 49(1-3):61–95, 1991.
10. L. Di Gaspero, J. Gärtner, G. Kortsarz, N. Musliu, A. Schaerf, and W. Slany. The minimum shift design problem. *Annals of Operations Research*, 155:79–105, 2007.
11. L. Di Gaspero and A. Schaerf. Multi-neighbourhood local search with application to course timetabling. In E. Burke and P. De Causmaecker, editors, *Lecture Notes in Computer Science*, number 2740 in Lecture Notes in Computer Science, pages 263–278. Springer-Verlag, Berlin-Heidelberg (Germany), 2003.
12. Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. Rota: a research project on algorithms for workforce scheduling and shift design optimization. *AI Commun.*, 14(2):83–92, 2001.
13. Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. A heuristic based system for generation of shifts with breaks. In *Proceedings of the 24th SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge*, 2004.
14. Luca Di Gaspero, Johannes Gärtner, Guy Kortsarz, Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. The minimum shift design problem: Theory and practice. In *ESA*, pages 593–604, 2003.
15. Luca Di Gaspero, Johannes Gärtner, Nysret Musliu, Andrea Schaerf, Werner Schafhauser, and Wolfgang Slany. A hybrid LS-CP solver for the shifts and breaks design problem. In *Hybrid Metaheuristics*, pages 46–61, 2010.
16. F. Glover and M. Laguna. *Tabu search*. Kluwer Academic Publishers, Dordrecht (the Netherlands), July 1997.
17. Holger H. Hoos and Thomas Stützle. *Stochastic Local Search: Foundations & Applications*. Elsevier / Morgan Kaufmann, 2004.
18. Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: A heuristic repair method for constraint satisfaction and scheduling problems. *Artif. Intell.*, 58(1-3):161–205, 1992.
19. N. Musliu, A. Schaerf, and W. Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.
20. N. Musliu, W. Schafhauser, and M. Widl. A memetic algorithm for a break scheduling problem. In *8th Metaheuristic International Conference*, Hamburg, Germany, 2009.
21. Nysret Musliu. *Intelligent Search Methods for Workforce Scheduling: New Ideas and Practical Applications*. PhD thesis, Vienna University of Technology, 2001.
22. Nysret Musliu. Heuristic methods for automatic rotating workforce scheduling. *International Journal of Computational Intelligence Research*, 2(4):309–326, 2006.
23. C. H. Papadimitriou and K. Steiglitz. *Combinatorial Optimization: Algorithms and Complexity*. Prentice Hall, 1982.
24. Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–391, 2010.
25. M. Rekik, J.F. Cordeau, and F. Soumis. Implicit shift scheduling with multiple breaks and work stretch duration restrictions. *Journal of Scheduling*, 13:49–75, 2010.
26. Werner Schafhauser. *TEMPLE - A Domain Specific Language for Modeling and Solving Real-Life Staff Scheduling Problems*. PhD thesis, Vienna University of Technology, 2010.

27. Bart Selman, Henry A. Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In *Proceedings of the Second DIMACS Challange on Cliques, Coloring, and Satisfiability*, 1993.
28. P. Tellier and G. White. Generating personnel schedules in an industrial setting using a tabu search algorithm. In E. K. Burke and H. Rudova, editors, *The 5th International Conference on the Practice and Theory of Automated Timetabling*, pages 293–302, 2006.
29. G. Thompson. Improved implicit modeling of the labor shift scheduling problem. *Management Science*, 41(4):595–607, 1995.
30. M. Widl. Memetic algorithms for break scheduling. Master's thesis, Vienna University of Technology, Vienna, Austria, 2010. http://www.kr.tuwien.ac.at/staff/widl/thesis.pdf.
31. Magdalena Widl and Nysret Musliu. An improved memetic algorithm for break scheduling. In *Hybrid Metaheuristics*, pages 133–147, 2010.