TECHNICAL REPORT

INSTITUT FÜR INFORMATIONSSYSTEME

ABTEILUNG DATENBANKEN UND ARTIFICIAL INTELLIGENCE

Local search for shift design

DBAI-TR-2001-45

Nysret Musliu **Andrea Schaerf** **Wolfgang Slany**

DBAI TECHNICAL REPORT

NOVEMBER 2001

Institut für Informationssysteme
Abteilung Datenbanken und
Artificial Intelligence
Technische Universität Wien
Favoritenstr. 9
A-1040 Vienna, Austria
Tel: +43-1-58801-18403
Fax: +43-1-58801-18492
sekret@dbai.tuwien.ac.at
http://www.dbai.tuwien.ac.at/

TECHNISCHE UNIVERSITÄT WIEN

DBAI TECHNICAL REPORT DBAI-TR-2001-45, NOVEMBER 2001

Local search for shift design

Nysret Musliu[1] **Andrea Schaerf** [2] **Wolfgang Slany**[3]

Abstract. Designing shifts is one of the important stages in the general workforce scheduling process. In this paper we consider solving the shift design problem by using local search methods. First we propose a set of move types that give rise to a composite neighborhood relation. In the move selection process, we make use of the basic prohibition mechanisms of tabu search. In addition, in order to avoid having to explorate the whole neighborhood which could be prohibitively large, we evaluate the moves in decreasing order of their promise to yield some improvement. Furthermore, we propose an algorithm for generating a good initial solution, which also exploits knowledge about requirements and shift structure. Experimental results on both real-life and randomly-generated problems show the advantages of these ingredients. The solver is part of a commercial product and has shown to work well in practical cases.

[1]Technische Universität Wien and Ximes Corp. mailto: musliu@dbai.tuwien.ac.at
[2]Università di Udine. mailto:schaerf@uniud.it
[3]Technische Universität Wien. mailto: wsi@dbai.tuwien.ac.at

Acknowledgements: This work was supported by FFF project No. **801160/5979** and the Austrian Science Fund Project No. **Z29-INF**.

1 Introduction

The typical process of planning and scheduling a workforce in an organization consists of several stages [19]. The first stage in this process is to determine the temporal requirements. In this stage the number of needed employees of each qualification is found for every time slot of the planning period (e.g., every Monday between 6:00-10:00 there should be 4 employees at work). After this stage one can proceed to determine the total number of employees needed, interrelated with designing the shifts and then assigning these shifts and days-off to employees.

There exist two main approaches in the literature to solve these stages. One of the approaches is to coordinate the design of the shifts and the assignment of the shifts to the employees, and to solve it like one problem [9, 10]. The other considers days-off scheduling and shift scheduling only after the shifts are designed [4, 13, 16]. The main disadvantage of the latter approach is that it does not guarantee that a feasible solution for the assignment of these shifts can be found. On the other hand, solving the overall problem in several separate stages makes it easier to tackle.

We follow the second approach, and in this paper we consider in isolation the problem of designing the shifts. For this problem, we are given the workforce requirements for a certain period of time, along with constraints about the possible start times and the length of shifts, and an upper limit for the average number of duties per week per employee. The aim is to generate solutions that contain shifts (and the number of employees per shift) that minimize the number of shifts, over- and under-staffing, and differences in the average number of duties per week. We will give a detailed description of the problem in Section 2.

Kortsarz and Slany [12] show that it is possible to model the shift design problem as a network flow problem, namely as the cyclic multi-commodity capacitated fixed-charge min-cost max-flow problem. Exploiting this result, if one drops the objective of minimizing the number of shifts, the problem can be solved very efficiently with a polynomial min-cost max-flow algorithm. Using a reduction to the Minimum Edge Cost Flow problem (no. [ND32] in Garey and Johnson [7]), Kortsarz and Slany show also that, on the contrary, the general shift design problem is NP complete. In addition, it is also hard to approximate: Kortsarz and Slany [12] also show that there is a constant $c < 1$ such that approximating the shift design problem with polynomial bounds on numbers appearing in the requirements within $c \ln n$ is NP hard. Additionally, they convincingly argue that it is very unlikely that the shift design problem admits an efficient algorithm with an approximation guarantee that is substantially better than an O($\sqrt{n \log M}$) ratio (where M is the largest number in the requirements), which in practice would anyway be unusable.

In conclusions, the problem that includes constraints for the minimization of the number of shifts is NP hard and also hard to approximate. This is the reason why we rely on local search techniques for solving this problem. The definition of the appropriate neighborhood structure is one of the most important features of local search techniques. In this paper we propose several move types (or repair steps) that will be used to define the neighborhood relation.

We use basic principles of tabu search [8] for the exploration of the neighborhood in order to make it possible to escape from local minima. That is, tabu moves will not be accepted for a number of iterations, except if the solution has a better cost than the current best one.

In order to make the search more effective, we introduced a new method that limits the exploration of the neighborhood. A basic feature of the guided search procedure is that the search is concentrated in 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. In addition, we use an heuristic procedure for generating a good initial solution, which is based on the idea of starting (ending) a shift whenever the requirements increase (decrease).

In the experimental results, we show results concerning different kinds of tabu mechanisms, lengths of the tabu list, initial solutions etc. Computational results show that the effectiveness of the search and the quality of the solution is improved by including the knowledge about the problem during the search and also the initial solution procedure. These techniques were implemented in a commercial product called Operating Hours Assistant.

2 Shift Design problem

For the problem of shift design we are given:

- n consecutive time intervals $[a_1, a_2)$, $[a_2, a_3)$, $\ldots$, $[a_n, a_{n+1})$, all with the same length *slotlength* in minutes. Each interval $[a_i, a_{i+1})$ has an adjoined numbers w_i indicating the optimal number of employees that should be present during that interval. Time point a_1 represents the begin of the planning period and time point a_{n+1} represents the end of the planning period. The format of time points is: *day:hour:minute*.

- y shift types $v_1, \ldots, v_y$. Each shift type v_j has the following adjoined parameters: v_j.min-start, v_j.max-start which represent the earliest and latest start of the shift and v_j.min-length, v_j.max-length which represent the minimum and maximum lengths of the shift. In Table 1 an example of four shift types is given.

- 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 : Average number of working hours per week per employee

Table 1: Shift types in the shift design problem

Shift type	min-start	max-start	min-length	max-length
M	05:00	08:00	07:00	09:00
D	09:00	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 has adjoined parameters s_l.start and s_l.length and must belong to one of the shift types. Additionally, each real shift s_p has adjoined parameters $s_p.w_i, \forall i \in \{1, \ldots, C\}$ (C represents number of days in the planning period) indicating the number of employees in shift s_p during the day i. The aim is to minimize the four components given below:

- Sum of the excesses of workers in each time interval during the planning period.
- Sum of the shortages of workers in each time interval during the planning period.
- Number of shifts.
- Distance of 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 workers' load in too many short shifts.

This is a multi criteria optimization problem. The criteria have different importance depending on the situation. The objective function is a scalar function which combines the four weighted criteria. Let us note that we usually consider designing shifts for a week (less days are also possible) and assume that 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 at least to one of the shift types if:

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

v_j.min-start $\leq s_l$.start $\leq v_j$.max-start

v_j.min-length $\leq s_l$.length $\leq v_j$.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 follows.

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

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 shortage TS and total excess TE (in minutes) of workers in each time interval during the planning period is then defined as

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

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

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.

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

2.1 Previous work

The shift design problem is similar to a problem which has been addressed in the literature as the shift scheduling problem. Typically in the shift scheduling problem it is required to generate shifts and number of employees for each shift over as single day. The aim is to obtain the solutions without under-staffing and to minimize the number of employees. Let us note that the problem of shift design differs in several aspects from it. First, we consider the generation of shifts for a week. This entails the added difficulty of having to reuse shifts on all days of the week in order to minimize the overall number of shifts used. Also, the problem is cyclic, so special care must be taken to connect the end of the schedule to its begin. Additionally, we consider minimizing the average number of duties per week if it gets above some threshold. Moreover, in our problem under-staffing is allowed.

Shift scheduling problem has been solved mainly by using Integer Programming (IP). Dantzig [6] developed the original set-covering formulation for shift scheduling problem:

$$\text{minimize} \sum_{j=1}^{n} c_j x_j$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_j \geq r_i \text{ for } i = 1, 2, \ldots, m$$

$$x_j \geq 0 \text{ and integer for } n \in N$$

where

$n =$ index for shifts,
$x_j =$ number of employees assigned to shift j,
$r_i =$ number of employees required to work in the ith time period,
$c_j =$ cost of having an employee work in shift j,
$m =$number of time periods to be scheduled over a single day,

$$a_{ij} = \begin{cases} 1 & \text{if the time period } i \text{ is a work period of shift } j \\ 0 & \text{otherwise.} \end{cases}$$

In this formulation for each feasible shift there exists one variable. Feasible shifts are enumerated based on possible starting times, length, breaks, and time windows for breaks. When the number of shifts increases this formulation soon becomes impractical.

Bechtold and Jacobs [5] proposed a new integer programming formulation for shift scheduling. In this formulation the modeling of break placements is done implicitly. Authors reported superior results with their model compared to the set covering model. Their approach is limited to organizations which operate less than 24 hours per day.

Thompson [18] introduced a fully implicit formulation of the shift scheduling problem. The model combines work of Moondra [15] and Bechtold and Jacobs [5] in which respectively the length of shifts and the breaks are modeled implicitly. Authors reported better results compared to the model of Bechtold and Jacobs [5].

Another integer programming model for shift scheduling with multiple rest and lunch breaks and break windows was proposed by Aykin [2]. This formulation is applicable to the shift scheduling problem without the limitations in [5]. A comparison of different modeling approaches is given by Aykin [3].

3 Local search for shift design

The basic idea of local search techniques [1, 8, 11, 17] is to improve initial solutions iteratively during search. A neighbor of the current solution is selected from the neighborhood of the solution, which is constructed through changing the current solution using so called ‘moves’. Basically, the differences between individual local search techniques are found in the way they explore the neighborhood and the criteria on how to accept the descendant of the current solution. The appropriate neighborhood structure for these techniques is one of the most important features to reach ‘good’ solutions. In this section, we first give definitions of the moves used for exploring the neighborhood for the shift design problem. Afterward, we describe other features of the technique we used concerning the generation of neighborhood, acceptance of the descendant solutions and aspiration criteria.

3.1 Neighborhood relations

Before introducing the neighborhood relations, we have to define the search space. To this regard we define a state (an element of the search space) as consisting of a set of shifts and the duties of the shifts for each day during the planning period.

Shifts belonging to a state obviously must be *active*, which means that they must have at least one duty in some day, otherwise the do not contribute to the solution. However, we consider also a set of so-called *reserve* shifts, which have no duties. In particular, we constrain all states to contain exactly one reserve shift for each shift type. This is done to simplify the definition of the neighborhood relation, as shown below.

The neighborhood of the solution is obtained by introducing altogether new shifts, removing a shift, or changing one of the three main features of a shift, namely, start, length, and duties per day. The basic moves presented below change these features.

ChangeDuty: The duties of the shift for the particular day are decreased or increased by 1 employee. For example, suppose that shift F1 was assigned 5 employees during the first day. By applying this move to shift F1 (on the first day) two new shifts can be obtained that respectively are assigned 4 and 6 employees in the first day.

ChangeLength: The length of the shift is made longer or shorter for one time slot. For example, suppose that the length of shift F1 is 8:00 and a time slot of 30 minutes is used. By applying this move to shift F1, two new shifts with lengths 7:30 and 8:30 can be obtained.

ChangeStart: The starting time of a shift is moved forward or backward by one time slot (the length is left unchanged). For example, suppose that the shift F1 begins at 8:00 and the time slot is 15 minutes. By applying this move to shift F1 two new shifts can be obtained, starting at 8:15 and 7:45.

CreateShift: A new active shift is created. This move is implemented indirectly, by applying the move ChangeDuty to a reserve shift. A new reserve shift is inserted.

RemoveShift: One shift is removed from the solution. This move is implemented in an indirect way, such that a shift is removed when all its duties become equal to 0.

The moves we defined before are basic moves. Next, we define the composite moves, which are combinations of basic moves.

MoveBorders: The borders of two shifts are moved left or right. The starting time of the first shift (the one that begins earlier) and the end of the second shift remain fixed. For example, suppose that the solution contains shifts F1: (6:00-14:30) and S1: (14:30-23:00). By shifting the border of these two shifts for 30 minutes backwards, two new shifts will be created: F2: (6:00-14:00) and S2: (14:00-23:00)

ExchangeDuties: In a particular day, one duty is passed from one shift to the other. For example, suppose that shift F1 has the duties (2 2 3 3 3 3 0) and shift F2 the duties (1 1 1 1 1 1 1), where each element of the array represents duties of shifts for a particular day. Shifts F1 and F2 will have new duties by applying this move on these shifts and day 1: F1: (1 2 3 3 3 3 0); F2: (2 1 1 1 1 1 1).

JoinShifts: Whole duties of one shift (X) are added to the other one and the shift X is eliminated from the solution. For example, suppose that a solution contains shifts F1: (8:00-16:00) (3 3 3 3 3 3 3) and T1: (9:00-17:00) (2 1 1 1 1 1 1). By applying this move the shift T1 is deleted and shift F1 gets the summed-up duties (5 4 4 4 4 4 4).

SplitShift: A new shift is split off from an old one. The newly created shift differs from the original shift by one time unit in the start or length (four possibilities). When possible, one duty per day is transferred from the original shift to the newly created shift. For example, by applying this move to shift F1: (7-15) (2 3 3 3 3 3 3), one of the solutions may consist of shift F1: (7-15) with duties (1 2 2 2 2 2 2) and shift F2: (7-15:30) with duties (1 1 1 1 1 1 1), the time unit being 30 minutes.

ChangeStartKeepEndFix: The start of a shift is moved forwards or backwards, while its end is kept fixed. For example, if we use a time unit of 30 minutes, by applying this move to shift F1: (7:00-15:00), new shifts F2: (6:30-15:00) or F3: (7:30-15:00) can be obtained.

Moves ChangeDuty and ExchangeDuties may be enhanced by applying these moves to a number of days larger than one.

3.2 Generation of neighborhood

We experimented with different variants for the generation of the neighborhood. In a first variant, only basic moves were applied in every iteration. In another variant, all moves (basic and composite ones) were applied to generate a neighborhood of solutions. However, in this case, for each move, not all neighborhood solutions were generated. Indeed, in all moves except the move ChangeDuty the exploration of neighborhood was interrupted as soon as a new better solution than the previous one was found.

Let us note that in all search methods the position and length of reserve shifts is changed if no improvement can be made to the solution.

3.3 Tabu mechanism

In order to avoid cycles during the search, a tabu list is used. The tabu list prohibits some of the solutions in the neighborhood to be accepted for the next iteration unless they fulfill certain criteria. We have experimented with two kinds of tabu lists. In the first variant, for each of the moves described above, we add the *inverse move* to the list. For example, the inverse of the move that increases by 1 the duty of shift X on day i, is the move that decreases by 1 the duty of shift X on day i. For each applied move in memory an inverse move is stored, including the information to which shifts the move was applied (one or two shifts) and also the day on which the move has been applied. The stored move in memory will be tabu for a certain number of iterations.

In the second variant, certain characteristics of a solution itself are considered tabu in combination with each other: the number of shifts, the difference of the duties in each time slot, and the average number of duties per week. Solutions with the same features are considered tabu for a specified number of iterations.

3.4 Selection criteria

The best solution from the neighborhood, if it is not tabu, becomes the current solution in the next iteration. The tabu solution will be accepted only if it has the best cost value found so far.

3.5 Cost function

The cost function is a scalar function which combines four weighted criteria.

$$\begin{aligned} Cost &= W1 \times ExcessInMinutes + W2 \times ShortageInMinutes + \\ & \quad W3 \times NumberOFShifts + W4 \times DifferenceOfDutiesPerWeek \end{aligned}$$

For a more efficient calculation of the excess and the shortage, a history of change of the solution is stored and these parameters are updated based on this history.

4 Exploting domain knowledge during the search

In this section, we describe a new approach that guides the search for the shift design problem, intending to make the search more effective. Furthermore, we present the algorithm for generating a good initial solution, which is also based on domain knowledge.

The share of the neighborhood that should be explored during every iteration can be reduced by using the knowledge about the problem during the search. The basic idea is to take moves which probably will most improve the solution. A similar technique for constraint satisfaction problems was used by Minton et al. [14]. This method is based on analyzing the 'distance' of the current solution to the optimal solution, with respect to the shortage and excess, which are the most important criteria in this problem. Just to give an idea, suppose that on day i between time 8:00-16:00 one employee is missing. A complete neighborhood search, among other moves, will perform also moves that decrease the duties of the shift for each day, which cannot contribute to any improvement in this situation. Furthermore, while this shortage appears at a particular day and particular shift, it is better to take moves that include only this day and shift types which cover this region: If, for example, the night shift ranges from 23:00-7:00, the moves in this shift probably will not bring any improvement in this situation. Our technique, based on the shortage/excess and shift types that could be used, determines the moves and other parameters that should be used for the neighborhood exploration of the current solution. Below we describe our technique in detail.

1. Find the shortage/excess (a zone of constant deviation) with the longest length.
2. Determine the contribution of each shift type in that shortage or excess: high, middle, small.
3. Begin with the shift type that has the highest contribution to the shortage/excess.
4. Determine the position of the shift types that contributes in the shortage/excess relative to the shortage/excess: left, middle, right.

5. Shortage/excess is classified in short, middle, or long relative to the shift type for which the neighborhood will be generated.

6. Based on the shortage/excess properties, position, and contribution of the shift type relative to the shortage/excess, determine moves, days, and a shift type with which the neighborhood will be explored.

7. If there exist improvements, repeat the steps from the beginning, or else if there exist no improvements, but still shift types that contribute in the shortage/excess then take the next shift type and continue with step 4. In case there exist no improvements and no more shift types that contribute to the shortage/excess, then, for the next iteration do a complete exploration of the neighborhood (like in basic tabu search).

Note that only the length of zones of constant deviation from the requirements is taken into account, but not the amount of the deviation. Indeed, by changing the duties of the corresponding shifts, any devation can easily be taken care off, as long as the whole zone of constant deviation is covered by some shifts. The challenge lies in finding the shifts best covering these zones.

The result of this technique is that the neighborhood will be generated using only some of the moves, and that these will be applied only to particular days and for particular shift types. Another important issue is the order of the moves. Since the procedure exits from the iteration for some of the move types as soon as a better solution than the current one is found, the moves that have more chances to improve the solution should be tried in the beginning. However, it is not always clear how to determine the order of the moves. For example, if the shortage/excess is long and the contribution of a certain shift type is high, then, logically, to improve the solution at this specific shortage/excess point, a move that increases/decreases duties of that particular shift type could be used.

On the other hand, suppose we have an example where the shortage/excess is short. This kind of shortage/excess can be repaired for example by a move of type ChangeDuty, ExchangeDuties, ChangeLength, MoveBorders, etc. and here the order of the moves is not clear. According to the properties of the shortage/excess, the contribution of shift types and their relative position, there exist 27 possible combinations (each of the properties is classified in three classes). Some of these combinations are almost the same regarding the conclusion about the moves that should be taken. In our technique, the order of the moves is the logical order when possible. In cases when is not completely clear which order of the moves should be taken, it is taken like in the case of basic tabu search described previously. Note that if no improvement can be found through this procedure, a complete search of the neighborhood is done to allow the acceptance of solutions with worse fitness, and thus making possible an escape from a local minimum.

Example: In Figure 1 the requirements are given (only the first two days can be seen) for the first problem among the 30 randomly generated problems for which we will give computational results later. Given an initially empty solution and applying the above procedure, the longest shortage found would be the one that starts at the first day at 22:00, with length 8 hours and a shortage of one employee. The highest contribution in this shortage is from the night shift. The shortage is long and in the middle of the region of a night shift. The guided search procedure

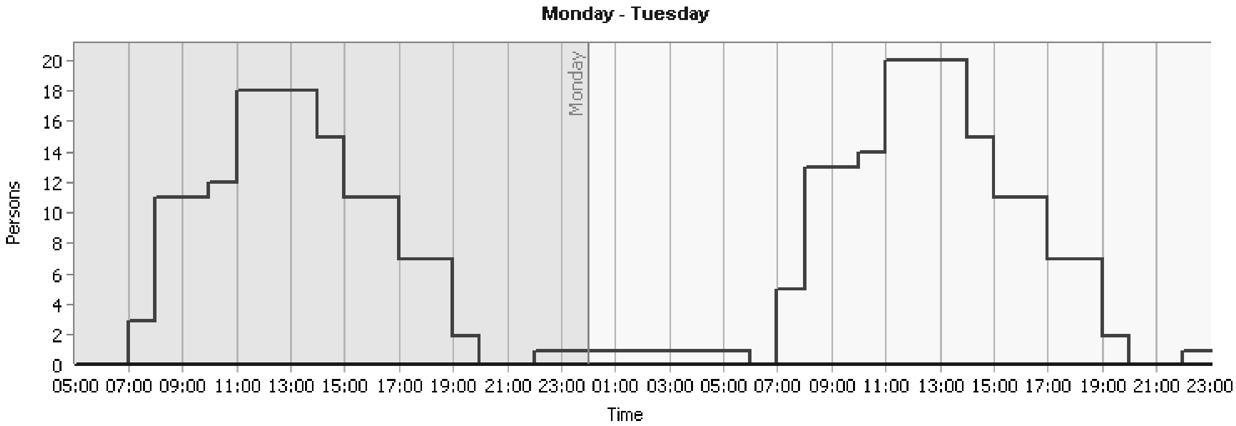

Figure 1: Requirements in example 1 for Monday and Tuesday

explores first the neighborhood that is constructed by applying the move ChangeDuty (duties are only increased) only for day one.

4.1 Initial Solution

Another ingredient added to the techniques described so far is a ‘good’ initial solution. The basic idea in constructing a ‘good’ initial solution is that in every change of the requirements there should begin or end a shift. Initial solutions constructed in this manner, of course, do not always contain all shifts needed to construct an optimal solution but contain at least some of those which could have either the correct start or end. Below, the procedure that generates the initial solution is described.

1. Detect all time points (day:hour:minute) in the planning period in which an increase of the requirements exists. The increases that begin at the same time (hour:minute) but on different days are considered like one increase even if they have different values (values of increases for each day are stored).

2. For each increase from the previous step, determine all shift types with start regions to which the increase belongs. For each such shift type construct one shift with starting time at the point of time of the increase (time where the increase begins) and some possible length based on the shift type. Additionally, mark the starting time of all shifts constructed in this step as constant during the search.

3. For each increase detected in step 1, select the first shift among the shifts constructed for that increase in the previous step. Give to the shift the duties which correspond to the values of the increase for each day.

4. Detect all time points (day:hour:minute) in the planning period in which a decrease of the requirements exists. The decreases that begin at the same time (hour:minute) but on the different days are considered like one decrease even if they have different values.

5. From the decreases from the previous step, eliminate decreases that can be reached with any of the shifts constructed before, based in the increase of duties.

6. For each remaining decrease determine all shift types, the end region (which is the time interval in which the shift type can have its end) of which contains the decrease and for each determined shift type construct one shift with end corresponding to the time of the decrease as well as having minimal legal length (determined from the constraints for the shift types). Additionally, mark the end of each shift constructed in this step as constant during the search.

5 Computational results

In this section, we first report computational results for 30 artificial examples generated by the random generator. Secondly, we report computational results for one real world example. The randomly generated examples can be found on http://www.dbai.tuwien.ac.at/proj/Rota/. We describe first how the random examples are generated. Later, we show results of experiments that were conducted with basic and composite moves and the different types of tabu lists. Further, we investigate the impact of the initial solution and guidance of the search by including the knowledge about the problem. The techniques we described in this chapter are implemented in a software package called Operating Hours Assistant of Ximes[1] Corp. and the system works well on real life examples. All our results in this section have been obtained on an Intel P2 333 Mhz.

5.1 Random instances generator

The basic idea for random examples is to generate for each shift type a specific number of time intervals which fulfill constraints about the start and length of the shifts. Each time interval corresponds to one legal shift. Randomly, 1-5 time intervals are generated for each shift type. The start of the time interval is generated randomly from the possible starts in the start region of a shift type and the length of time intervals is one random possible length (determined by the shift type). We use four shift types, and thus the optimal solutions to the problem can have from 1 to 20 shifts. In Table 2 constraints for shift types are given. The time interval is taken randomly to be 15, 30 or 60 minutes.

Duties for each time interval are generated such that initially, for each time interval, a random number between 1-5 is generated with equal probability. The time interval will have assigned (as number of employees) this random number for each day during the week with probability 0.9 (probability that the duties for each particular day change from the standard value is 0.1). If this value should change for a particular day, the new value (1-5) is generated randomly for that day. During the weekend the probability that the value changes is 0.6. If the value should change, a

[1]http://www.ximes.com/

Table 2: Shift types for the random generator

Shift type	min-start	max-start	min-length	max-length
M (Morning shift)	05:00	08:00	07:00	09:00
D (Day shift)	09:00	11:00	07:00	09:00
A (Afternoon shift)	13:00	15:00	07:00	09:00
N (Night shift)	21:00	23:00	07:00	09:00

new random number is generated (the probability that either Sunday or Saturday differs from this number is again 0.1).

Weights for shortage and excess are $W1 = W2 = 1$, for number of shifts the weight is equal to the length of the time unit in minutes ($W3 = 15$, 30, or 60) and for duties per week it is $W4 = 1000$. The maximum number of duties per week is set to 5. The average number of hours per week is 38.5. As we want to generate the problems for which we can calculate the optimal cost in advance, if the duties per week exceed 5 for an optimal solution regarding excess, shortage and number of shifts, we put the weights of the duties per week to 0. This means we consider indeed only minimizing the shortage, excess and number of shifts.

Like we described before, the random examples are generated such that each randomly generated time interval corresponds to one legal shift. Thus the optimal solution is likely to be the solution which will contain the correspondent shifts for each randomly generated time interval and its duties. Actually, there is a very small probabilitity that an even better solution may exist, but as this is the best solution easily known and the probability is very low for a better one, we will refer to it in this paper as the optimal solution and its cost as the cost of the optimal solution.

5.2 Computational results on random examples

In this section, we give results of four experiments over 30 randomly generated examples. The aim of these experiments was to determine the impact of composite moves, the length of the tabu list, and variants of the tabu mechanism. Techniques are compared regarding the dependence of the cost from the number of evaluations and quality of the solution found after a certain number of evaluations. All results are given for one run of the algorithm, as the latter is deterministic. For each technique the algorithm is allowed to begin the next iteration only if a maximum number of 100000 evaluations is not exceeded for each example and the search is interrupted each time if after 200 seconds no improvements could be found.

5.2.1 Basic moves

In this experiment the basic tabu search technique is used. During every iteration, by using only basic steps, a complete neighborhood of the solution is generated. For each of the tabu variants four different lengths of the tabu list were used, namely 5, 10, 20, and 40 memory slots.

Results of this experiment over 30 examples were not satisfactory. While in two tabu variants the results were improved with increasing the tabu length, the best results were poor. In order to improve the results, we included the composite moves. The results are described in the next section.

5.2.2 Impact of composite moves

In this experiment, tabu search is used and all moves are applied (basic and composite ones) for the composition of the neighborhood. However, we do not now explore the complete neighborhood for each move. Indeed, in all moves except the move ChangeDuty the exploration of the neighborhood is interrupted as soon as a new solution better than the previous one if found. The order of the moves in tabu search is the same as the order described in Section 3.1, except that move ChangeStartKeepEndFix is tried after move ChangeLength. The experiment was conducted for two variants of tabu search and four possible lengths of the tabu list (5, 20, 40, 70).

The best results were obtained by applying the second variant of the tabu list (solution tabu) with a length of the tabu list of 20. These results are shown in Table 3. For each example, information for the best cost found, and number of evaluations needed for finding this cost is given. In the second column the cost of the optimal solution is given. The best results obtained by applying the first variant of the tabu list (moves tabu) were slightly weaker. When using all moves, the tabu list has not such a high impact as in the case when only the basic moves are used.

5.2.3 Impact of the initial solution

The aim of this experiment is to investigate the impact of the initial solution. In the previous experiment the initial solution was a very simple one: For each shift type it contained one empty shift template. In contrast, now the search starts from a good initial solution which is generated as described in Section 4.1. The length of the tabu list is taken to be 20 and the variant of making solutions tabu is applied.

In Table 5.2.3 the results of the tabu search with good initial solution are shown. These results are better compared to the results in Table 3 with respect to the cost of solutions the number of evaluations and the time needed to generate the best solutions.

5.2.4 Including the knowledge about the problem during the search

In this experiment, we show the results obtained by exploiting the knowledge about the problem (see Section 4) using tabu search with good initial solutions (TSwIS). We will call this technique tabu search and guided search with initial solutions (TSaGSwIS).

In Table 5 the results of TSaGSwIS are presented. This technique shows the best results with respect to all criteria: cost of solutions, number of found optimal solutions, number of evaluations per solution, and the time in which these solution were found. With this technique, 50 % of the optimal solutions are found in only one run.

Table 3: Results for 30 examples using TS with the variant of a solution-type tabu list

		Tabu Search with solution tabu	
Ex.	Cost of optimal solution	Cost	Number of evaluations
1	480	2040	22322
2	300	750	62191
3	600	600	79326
4	480	1590	100388
5	480	480	25045
6	420	480	19175
7	270	1080	41525
8	150	195	100597
9	150	300	54318
10	330	1620	93847
11	30	30	2531
12	90	90	21550
13	105	105	24745
14	195	4305	100193
15	180	180	2045
16	225	540	100779
17	540	3600	100019
18	720	720	97919
19	180	2970	69726
20	540	540	53840
21	120	195	34349
22	75	75	9301
23	150	2430	75001
24	480	480	28656
25	480	2160	94991
26	600	720	46458
27	480	540	80685
28	270	1380	72192
29	360	1620	69738
30	75	75	6774

5.3 Computational results on real-world problems

In this section we present the results for one real world problem taken from a call center.

Problem 1: This problem is a real problem from a call center. Temporal requirements for this problem are given in Table 6. Constraints about the shift types which can be used in the solution

Table 4: Results for 30 examples using TS with good initial solution

		Tabu Search with good initial solution	
Ex.	Cost of optimal solution	Cost	Number of evaluations
1	480	480	10012
2	300	390	36182
3	600	1020	80854
4	480	1590	101057
5	480	480	8613
6	420	420	5977
7	270	570	15497
8	150	615	15631
9	150	225	13034
10	330	450	80666
11	30	30	346
12	90	90	11976
13	105	105	2197
14	195	495	90481
15	180	180	148
16	225	540	73411
17	540	1170	75119
18	720	720	18979
19	180	195	34173
20	540	540	20790
21	120	120	2674
22	75	90	4158
23	150	570	39309
24	480	480	10120
25	480	1050	46090
26	600	660	31033
27	480	480	8537
28	270	270	7297
29	360	390	23856
30	75	75	986

are given in Table 7. Weights of the criteria are: $W1 = W2 = 1, W3 = 30, W4 = 1000$. Average number of working hours is 38.5 and an upper limit of the average number of duties per week is 5.

In Table 8 the solution produced by TsaGSwIS is shown. The second variant of the tabu mechanism (solution tabu) is used and the length of the tabu list is 20.

Table 5: Results for 30 examples using TSaGSwIS

		Tabu search and guided search with good initial solution	
Ex.	Cost of optimal solution	Cost	Number of evaluations
1	480	480	691
2	300	420	2623
3	600	900	85917
4	480	1170	67207
5	480	480	2299
6	420	420	887
7	270	630	17468
8	150	180	5722
9	150	225	10348
10	330	510	69811
11	30	30	68
12	90	90	1664
13	105	105	2494
14	195	390	46569
15	180	180	10
16	225	375	41182
17	540	1110	51679
18	720	720	1759
19	180	195	6985
20	540	540	27534
21	120	120	279
22	75	105	2414
23	150	540	3026
24	480	480	8674
25	480	690	29465
26	600	600	4151
27	480	480	7755
28	270	270	694
29	360	390	9461
30	75	75	379

The produced solution has no excess and a shortage of 3.99 %. The average number of duties per week is 4.82. The solution is generated after 3771 evaluations and in about 40 seconds.

Table 6: Temporal requirements for the call center problem

Time interval/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
07:00-08:00	5	5	5	5	5	1	1
08:00-08:30	10	10	10	10	10	4	4
08:30-09:30	12	12	12	12	12	4	4
09:30-10:00	14	14	14	14	14	4	4
10:00-12:00	17	17	17	17	17	4	4
12:00-13:00	17	17	17	17	17	9	9
13:00-17:00	20	20	20	20	20	9	9
17:00-18:00	18	18	18	18	18	8	8
18:00-18:30	20	20	20	20	20	5	5
18:30-19:30	18	18	18	18	18	5	5
19:30-20:00	16	16	16	16	16	5	5
20:00-22:00	13	13	13	13	13	5	5

Table 7: Constraints about shifts in the call center problem

Shift type	min-start	max-start	min-length	max-length
M (Morning shift)	05:00	08:00	07:00	09:00
D (Day shift)	09:00	11:00	07:00	09:00
A (Afternoon shift)	13:00	15:00	07:00	09:00

6 Conclusions

In this paper we presented a local search approach for the shift design problem. We proposed basic and composite moves for the generation of the neighborhood during every iteration. Experiments

Table 8: Solution for the call center problem with TSaGSwIS

Shift	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
M1	07:00-15:00	5	5	5	5	5	1	1
M2	08:00-15:00	5	5	5	5	5	0	0
D1	09:00-17:00	2	2	2	2	2	0	0
D2	10:30-19:30	5	5	5	5	5	0	0
D3	09:00-18:00						3	3
E1	15:00-22:00	10	10	10	10	10		
E2	13:00-22:00	3	3	3	3	3	5	5

over randomly generated examples confirmed that composite moves improve the quality of the generated solutions. To avoid cycles during the search, we used basic principles of tabu search and experimented with different mechanisms for prohibiting the solutions to be descendant of the current solution. We also experimented with different lengths of tabu lists for two variants of the tabu list method. Experiments showed that the length of tabu list had much more impact in case where only basic moves were used. Furthermore, we proposed a procedure for the generation of good initial solution, which has shown to improve the results compared to the basic tabu search over the randomly generated examples. In order to make the search more effective, we proposed a method which exploits knowledge about the problem during the search. Using this method the neighborhood of a solution is explored only selectively during every iteration. Experimental results confirmed that this method improves the search effectiveness. Probably, local search can be refined even more for this problem, but the main objective here was to show that knowledge about the problem can significantly contribute to the search by using it in combination with classical techniques. The methods are part of a commercial product and have been already successfully used in many real world examples. In this paper we also described a set of random examples for the shift design problem which can be used by other researchers to compare their results with ours.

References

[1] Emile Aarts and Jan Karl Lenstra, editors. *Local Search in Combinatorial Optimization*. Wiley, 1997.

[2] Turgut Aykin. Optimal shift scheduling with multiple break windows. *Managament Science*, 42(4):591–602, 1996.

[3] Turgut Aykin. A comparative evaluation of modeling approaches to the labor shift scheduling problem. *European Journal of Operational Research*, 125:381–397, 2000.

[4] Nagraj Balakrishnan and Richard T. Wong. A network model for the rotating workforce scheduling problem. *Networks*, 20:25–42, 1990.

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

[6] G.B. Danzig. A comment on eddie's traffic delays at toll booths. *Operations Research*, 2:339–341, 1954.

[7] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman and Co., 1979.

[8] Fred Glover and Manuel Laguna. *Tabu search*. Kluwer Academic Publishers, 1997.

[9] Fred Glover and Claude McMillan. The general employee scheduling problem: An integration of MS and AI. *Comput. Ops. Res.*, 13(5):563–573, 1986.

[10] W. Ken Jackson, William S. Havens, and Harry Dollard. Staff scheduling: A simple approach that worked. Technical Report CMPT97-23, 1997.

[11] S. Kirkpatrick, C. D. Gelatt Jr., and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220:671–680, 1983.

[12] Guy Kortsarz and Wolfgang Slany. The minimum shift design problem and its relation to the minimum edge-cost flow problem. Unpublished manuscript.

[13] Hoong Chuin Lau. On the complexity of manpower scheduling. *Computers. Ops. Res.*, 23(1):93–102, 1996.

[14] Steven Minton, Mark D. Johnston, Andrew B. Philips, and Philip Laird. Minimizing conflicts: a heuristic repair method for constraint satisfaction and scheduling problems. *Artificial Intelligence*, 58:161–205, 1992.

[15] L. Moondra. An LP model for work force scheduling for bank. *Journal of Bank Research*, 7:299–301, 1976.

[16] Nysret Musliu, Johannes Gärtner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. Technical Report DBAI-TR-2000-35, Institut für Informationssysteme der Technischen Universität Wien, 2000. http://www.arXiv.org/abs/cs.OH/0002018.

[17] Colin R. Reeves, editor. *Modern Heuristic Techniques for Combinatorial Problems*. Halsted Pr., 1993.

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

[19] James M. Tien and Angelica Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3):275–287, 1982.