A Heuristic Based System for Generation of Shifts with Breaks

Johannes Gärtner
XIMES Corp.
Vienna, Austria

Nysret Musliu
Technische Universität Wien
Vienna, Austria

Wolfgang Slany
Technische Universität Graz
Graz, Austria

Abstract

In this paper a system for the automatic generation of shifts with breaks is presented. The problem of generating shifts with breaks appears in many areas of workforce scheduling, like in airline companies, airports, call centers etc. and is of high practical relevance. A heuristics algorithm for solving this problem is described. It is based on greedy assignment of breaks in shifts and repair steps for finding the best position of breaks inside of shifts. The commercial product in which the algorithms are included is used in practice. Computational results for a real life problem in a large European airport are given.

1 INTRODUCTION

Workforce scheduling is necessary in many industries like , e.g., industrial plants, hospitals, public transport, airlines companies. The typical process for planning and scheduling of a workforce in an organization consists of several stages. The first stage is to determine the temporal requirements for staffing levels. After these temporal requirements are defined, usually in the next phase the shifts are constructed. In this phase also the staffing level for each shift is determined. Then the total number of employees needed is calculated based on the average number of working hours for a certain period of time, usually one week.

In Table 1 an example of such temporal requirements is given. In this example the temporal requirements are given for one week. Based on these requirements and some constraints about the possible start and length of shifts, the aim in the shift design problem is to generate legal shifts that meet in the best way the temporal requirements. Additionally in some situation generation of breaks for each employee may be required. The detailed description of the problem we consider in this paper is given in the next section.

The shift design problem we consider in this paper is similar to a problem which has been addressed in literature as shift scheduling problem. Typically for this problem it is required to generate shifts and number of employees for each shift for a single day. The aim is to obtain solutions without under-staffing

Table 1: Possible temporal requirements for one week

Time interval/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
07:00-11:00	5	5	5	5	5	1	1
11:00-14:30	10	10	10	10	10	4	4
14:30-20:30	12	12	12	12	12	4	4
20:30-06:00	14	14	14	14	14	4	4

and to minimize number of employees. The shift scheduling problem has been addressed mainly by using Integer Programming (IP). Dantzig [3] developed the original set-covering formulation. In this formulation for each feasible shift exist one variable. Feasible shifts are enumerated based on possible start, length, breaks and time windows for breaks. When the number of shifts increases rapidly this formulation is not practical. Bechtold and Jacobs [2] proposed a new integer programming formulation. In their formulation, the modelling of break placements is done implicitly. Authors reported superior results with their model compared to the set covering model. Their approach however is limited to scheduling problems of less than 24 hours per day. Thompson [7] introduced a fully implicit formulation of shift scheduling problem. A comparison of different modelling approaches is given in by Aykin [1].

Note that the problem of shift design we consider in this paper differs in several aspects from the problem of shift scheduling addressed in these papers. First, we consider generation of shifts for a week. We consider also minimizing of number of shifts and in our problem under-staffing may be allowed to some degree. For solving this problem the local search techniques based in tabu search were proposed and implemented ([6]). Further improvement of these algorithms is presented in [5]. The local search algorithms have been included in the commercial product called Operating Hours Assistant(OPA).

In this paper we describe extension of this system considering generation of shifts together with breaks for each employee. Generation of breaks makes the problem of generation of shifts much more complex, correspondingly the automatic generation of shifts with breaks is a very important issue for schedulers as good solutions can reduce significantly the costs of organizations. In Section 4 we apply the system to a real problem of a large airport in Europe. Note that experienced professional planners can construct solutions for practical problems by hand. However, the time they need is sometimes very long (one hour to several days for very large instances), and, because of the large number of possible solutions, the human planners can never be sure how strong their solution differs from the best one. Therefore, the aim of automating the generation of shifts with breaks is to make possible the generation of good solutions in a short time, thereby reducing costs and finding better solutions for problems that appear in practice.

2 PROBLEM DESCRIPTION

First we describe problem of generation of shifts without breaks as considered in [6]. Then the extension of this problem by including breaks is presented.

Instance:

- 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 represents the end of the planning period. The format of time points is: *day:hour:minute*. For simplicity the temporal requirements are usually represented using longer time intervals. See one possible representation of temporal requirements for one week in Table 1.
- 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 2 an example of shift types is given.
- An upper limit for the average number of working shifts per week per employee.

Table 2: Possible constraints for shift types in the shift design problem

Shift type	Earliest begin	Latest begin	Shortest length	Longest 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

Problem:

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 k.
- Distance of the average number of duties per week in case it is above a certain threshold.

For the extended problem which includes also the generation of breaks, it is necessary to generate a predetermined number of breaks for each employee. In this case for each shift type the possible break types should be defined by the decision maker. The break type determines feasible time windows of breaks inside of the shift. In the system described in this paper we consider break types with following features: minimal length of break, maximal length of break, minimal and maximal distance of start of break from the shift begin, and minimal distance of end of break from the end of the shift. One or more breaks may be required to be generated for each shift and employee. The objective of the problem remains as described previously.

This is a multi criteria optimization problem. The criteria have different importance depending on the situation. We use for this problem the objective function, which is a scalar function which combines the four weighted criteria. Note that we consider the design of shifts for a week (less days are also possible) and assume that the schedule is cyclic (the consecutive element of the last time slot of the planning period is the first time slot of planning period).

3 APPLICATION DESCRIPTION

Generation of shifts with breaks is one of most important features of the latest version of Operating Hours Assistant (OPA) (version 2.0). This system is suitable for use in different kind of organizations, like call centers, airports, etc. It facilitates an appropriate planning of working hours, staffing requirements and generation of optimal shifts which fulfill the legal requirements and also minimize over and under staffing in the organization. Good solution in this stage of workforce scheduling can reduce significantly the cost of organization and also are important for later stage in workforce scheduling (generation of workforce schedules). In this paper we will concentrate on one of main features of OPA 2.0, which is the generation of shifts with breaks. The current version is based on the previous version which included local search algorithms for generation of shifts without breaks and also the tools for definition of temporal requirements, and constraint about the shifts, as well preferences of user considering different criteria. For a description of these algorithms see [6, 4]. Then we give a description of the heuristics for generating breaks.

3.1 Generation of breaks

We consider the generation of breaks as a separated stage from the generation of shifts. For each shift there should be generated a fixed number of breaks. However, each employee can have different breaks, i.e., if one shift has more employees the number of breaks which the shift will contain will be $NumberOfEmployees * NumberOfFixedBreaksPerShift$.

For generation of breaks we rely in the solution produced by the local search (included in version 1.0 of OPA ([6])). Intuitively one possibility would be to consider breaks in the phase of generating shifts with the local search techniques. However, this would imply a tremendous increase of neighborhood solutions that should be generated during each iteration in the local search. Indeed, if we would have applied only two simple moves in breaks (change length of break and shifting of break to the left or right), for each shift with n employee, where each employee should have m breaks, one needs to generate $m \times n * 2$ neighborhood solution for a particular shift. To avoid so large number of neighborhood solutions we consider generation of breaks only after the shifts are generated.

For generation of breaks after generation of shifts we propose a heuristic method which is based in combination of greedy search with the local improvement based technique. The algorithm for the generation of shifts with breaks consists of the following steps:

1. Generate shifts without breaks
2. Convert solutions to shifts with one employee per day
3. Generate the difference in temporal requirements (difference curve)
4. Assign fixed breaks to shifts based on greedy algorithm
5. Apply repair steps to breaks
6. Create solution with shifts and multiple employees

The first step in solving this problem is a procedure based on the local search algorithms. This procedure is not a subject of this paper and we describe it only briefly. This procedure is based on the iterative improvement of the initial solution based on repair steps. Repair steps were used to explore the neighborhood of solutions for this problem. In order to generate the neighborhood and accept the solution for the next iteration, basic principles of the tabu search technique are used. However, while tabu search can find acceptable solutions for this problem, the complete exploration of the neighborhood (using all defined moves) in each iteration is very time consuming. To make search more effective a new approach based on using of knowledge about the problem in combination with tabu search is used. For a detailed description of these algorithms see [6].

The solution generated in the first phase contains shifts which can have more than one employee in more than one day. Next, the solution should be converted in shifts which can contain maximal one employee assigned to each particular day. Based on the solution without breaks and given temporal requirements from the problem the difference in temporal requirements (we call it difference curve, as in OPA the temporal requirements are presented graphically with some curve) is found. Next, the greedy algorithm for assigning of breaks to shifts is applied. The pseudo code of greedy algorithm is presented below:

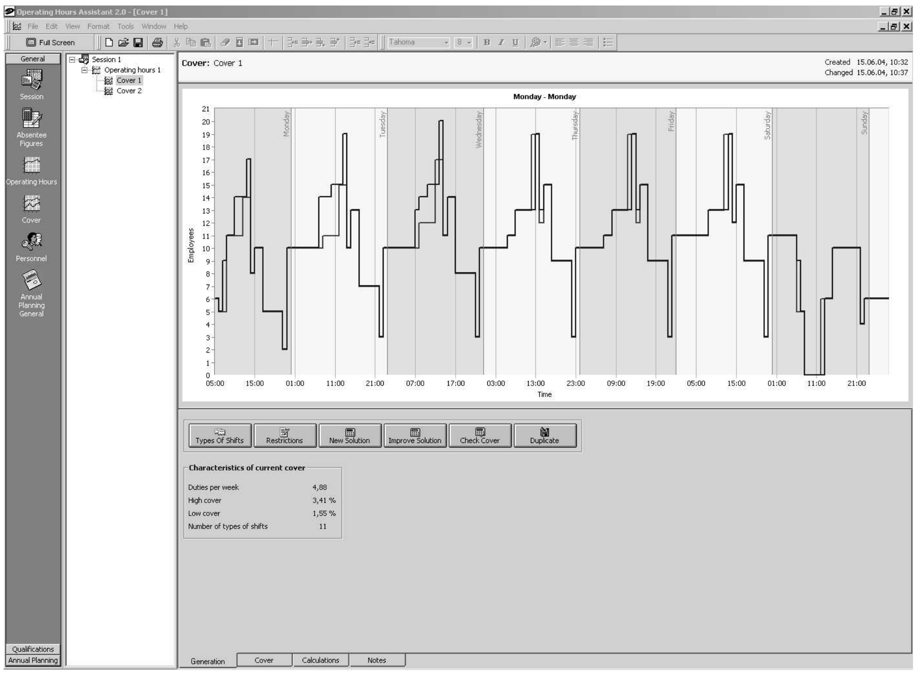
```
For each shift in Solution
    For i=1 to NumberofBreaksToBeAssigned
        FindLegalRegionOfBreak
        FindBestPositionOfBreak
        DeterminLengthOfBreak
    Next
Next
```

The legal region of each break is found based on the constraints about the length and possible begin of a break inside of shift. The best position of each break is found based on the difference curve. The main idea is to locate the break with a greedy procedure in the best position in which the weighted sum of undercover and overcover is minimized. For example, in case there is a region where the overcover is very high that is a good position for a break as in this way the over cover is decreased. The length of a break is set in this step to be as short as possible. The last step in the generation of breaks is to apply some repair steps to the breaks which are already assigned. The basic repair steps which are used are enlargement of length of breaks and changing the break position to the left or right (within the legal region of a break). After fixing positions of breaks inside of the shifts, the solution with shifts which have multiple employees is created, by joining the duties of shifts which have the same starting time and length. The shift which is created by joining several shifts will contain also all breaks of joined shifts. Representing a solution in this form eases the construction of workforce schedules (assigning of shifts to particular employee), which is usually the next phase in the workforce scheduling process.

Note that the step of generating breaks does not have any impact on the number of shifts which were generated in the first phase. Considering undercover and overcover, in case there exist some overcover the quality of the solution can be increased during the generation of breaks.

4 APPLICATION USE

In this section we illustrate use of the system with the problem which appeared in one of largest airport in Europe. Although the extension of system for generation of shifts with breaks has been only recently implemented, the system has been already successfully used to solve some larger problems in airline companies and railway companies. Unfortunately we could not find similar benchmark problems from literature to compare our results and the results presented here are result of solving real life problems from consultants of XIMES Company.



Characteristics of current cover

Duties per week	4,88
High cover	3,41 %
Low cover	1,55 %
Number of types of shifts	11

Figure 1: Screenshot of Operating Hours Assistant

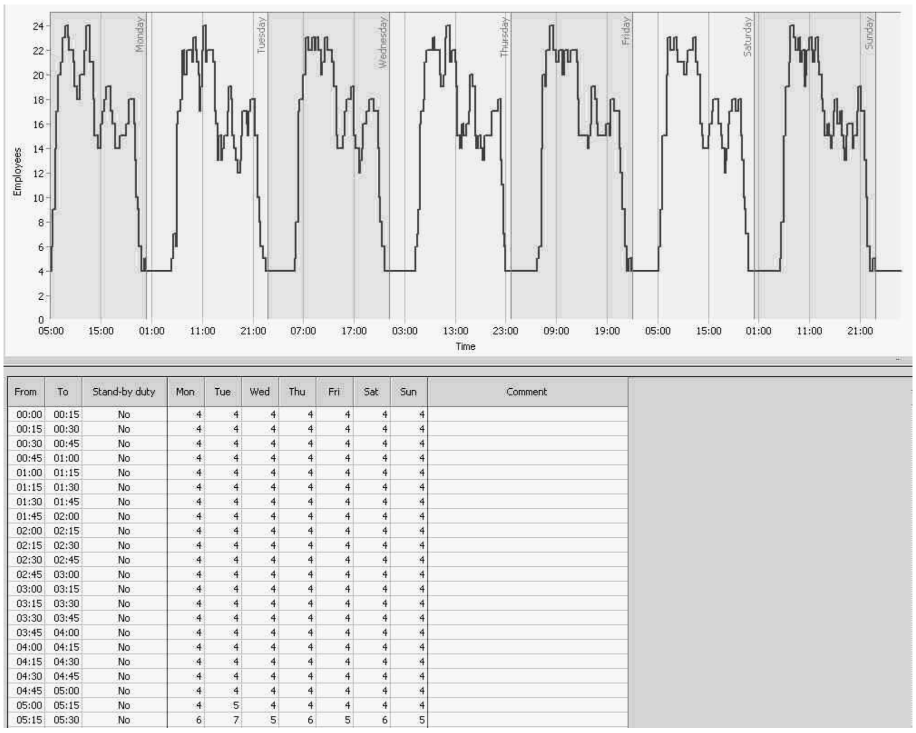
We now describe the process of the generation of shifts with breaks in OPA for a specific problem. In Figure 1 the screenshot of the software (OPA) is given. The generation of solution requires several steps to be defined from decision maker.

4.1 Definition of temporal requirements

The first step in designing shifts is to define the temporal requirements (see Figure 2). Typically, the temporal requirements would be given for a week, but they can also be given for one day or less then seven days. In our case, when the temporal requirements are given for a week, the cyclic structure has to be considered (e.g., the night shift that begins on Sunday at 23:00 is 8 hours long and impacts the first day of the next week). For the problem we consider here the temporal requirements defined for one week. Figure 2 shows part of the requirements input screen.

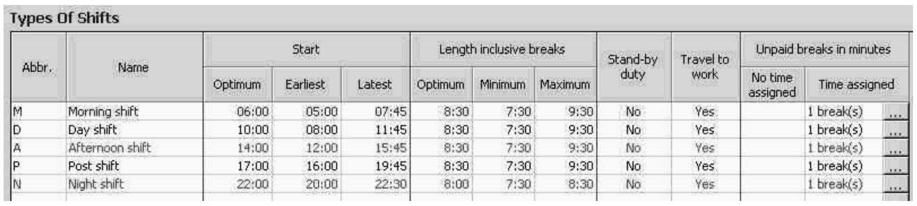
4.2 Constraints regarding shift types

Shift types determine the possible start and length of the shifts. In this case (see Figure 3), five shift types are defined, morning shift, day shift, evening shift, night shift, and post shift. Shifts generated by algorithms should fulfill



From	To	Stand-by duty	Mon	Tue	Wed	Thu	Fri	Sat	Sun	Comment
00:00	00:15	No	4	4	4	4	4	4	4	
00:15	00:30	No	4	4	4	4	4	4	4	
00:30	00:45	No	4	4	4	4	4	4	4	
00:45	01:00	No	4	4	4	4	4	4	4	
01:00	01:15	No	4	4	4	4	4	4	4	
01:15	01:30	No	4	4	4	4	4	4	4	
01:30	01:45	No	4	4	4	4	4	4	4	
01:45	02:00	No	4	4	4	4	4	4	4	
02:00	02:15	No	4	4	4	4	4	4	4	
02:15	02:30	No	4	4	4	4	4	4	4	
02:30	02:45	No	4	4	4	4	4	4	4	
02:45	03:00	No	4	4	4	4	4	4	4	
03:00	03:15	No	4	4	4	4	4	4	4	
03:15	03:30	No	4	4	4	4	4	4	4	
03:30	03:45	No	4	4	4	4	4	4	4	
03:45	04:00	No	4	4	4	4	4	4	4	
04:00	04:15	No	4	4	4	4	4	4	4	
04:15	04:30	No	4	4	4	4	4	4	4	
04:30	04:45	No	4	4	4	4	4	4	4	
04:45	05:00	No	4	4	4	4	4	4	4	
05:00	05:15	No	4	5	4	4	4	4	4	
05:15	05:30	No	6	7	5	6	5	6	5	

Figure 2: Definition of temporal requirements in Operating Hours Assistant



Types Of Shifts

Abbr.	Name	Start			Length inclusive breaks			Stand-by duty	Travel to work	Unpaid breaks in minutes	
		Optimum	Earliest	Latest	Optimum	Minimum	Maximum			No time assigned	Time assigned
M	Morning shift	06:00	05:00	07:45	8:30	7:30	9:30	No	Yes		1 break(s) ...
D	Day shift	10:00	08:00	11:45	8:30	7:30	9:30	No	Yes		1 break(s) ...
A	Afternoon shift	14:00	12:00	15:45	8:30	7:30	9:30	No	Yes		1 break(s) ...
P	Post shift	17:00	16:00	19:45	8:30	7:30	9:30	No	Yes		1 break(s) ...
N	Night shift	22:00	20:00	22:30	8:00	7:30	8:30	No	Yes		1 break(s) ...

Figure 3: Definition of shift types in Operating Hours Assistant

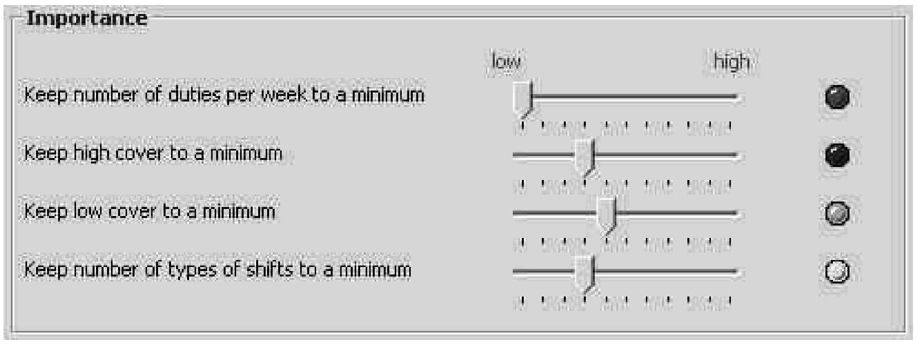

Importance
low
high
Keep number of duties per week to a minimum
Keep high cover to a minimum
Keep low cover to a minimum
Keep number of types of shifts to a minimum

Figure 4: Definition of weights about the criteria in Operating Hours Assistant

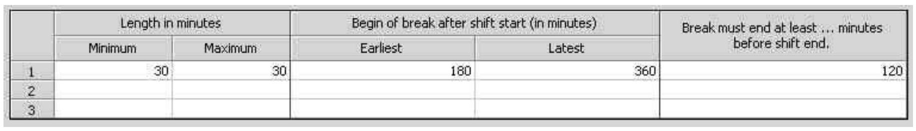
the criteria required by the shift types. For example, morning shift can start between 5:00-7:40 and their length should be from 7,5 - 9,5 hours.

4.3 Weights of criteria

The solution to the shift design problem is evaluated with a scalar function, which combines four weighted criteria: excess in minutes, shortage in minutes, number of shifts and distance from average number of duties per week. OPA offers the possibility to change the importance of these criteria (Figure 4). The decision maker can include his preferences easily. For this case the following weights are selected from decision maker for the criteria: $overcoverWieght = 0.6$, $undercoverWeight = 0.8$, $NumberOfShiftWeight = 36$, $distanceFromAverageNumberOfDutiesPerWeek = 0$

4.4 Definition of break types

The last phase in problem definition is to define break types for each shift. In Figure 3 each shift type has exactly one break type. Break types determine the possible legal region of the start of a break and its length. In this case for each shift type has the same break type (see Figure 5).



	Length in minutes		Begin of break after shift start (in minutes)		Break must end at least ... minutes before shift end.
	Minimum	Maximum	Earliest	Latest	
1	30	30	180	360	120
2					
3					

Figure 5: Definition of break types in Operating Hours Assistant

4.5 Generation of shifts

After the constraints have been defined, the algorithm for the generation of shifts can be called. As described in this paper the generation of shifts with breaks is done in two phases. In the first phase the algorithm for finding of a solution for shifts without breaks is executed. This algorithm iteratively improves the initial solution. User get a visual representation of the solution found so far and can thereby watch improvements. The algorithm can be stopped at any time and can be started all over again from any solution. In case the decision maker is not satisfied with a preliminary solution, the weights can be changed and attempts could be made to improve the current solution. After the shifts without breaks are generated the method proposed in this paper is applied to generate breaks for each shift and each employee.

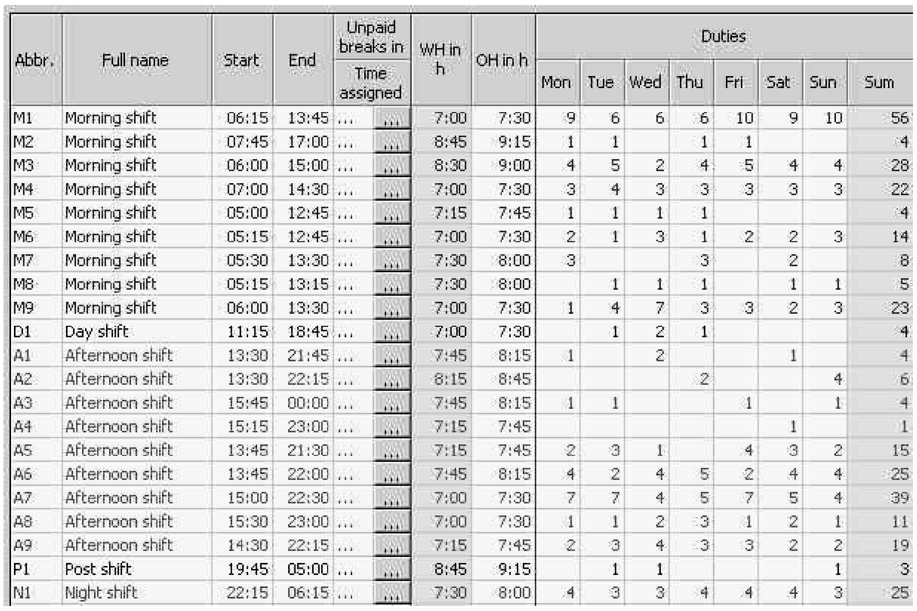
For the given requirements, constraints and weights, algorithm which combine basic tabu search and guided search and starts with a good initial generates the solution with shifts without breaks which has the following features: $overcover = 8.04\%$, $undercover = 0\%$, $NumberOfShift = 21$, $AverageNumberOfDutiesPerWeek = 4.9$.

The algorithm for the generation of breaks which is applied after this phase generates all breaks for each employee and shift and could improve significantly the quality of the solution consideration weighted sum of overcover and undercover. The solution found after two phases has these features: $overcover = 3.53\%$, $undercover = 2.36\%$, $NumberOfShift = 21$, $AverageNumberOfDutiesPerWeek = 5.23$.

In in Figure 6 the solution which contains 21 shifts for this problem found by OPA is presented. The last column in this Figure indicates the number of breaks for each shift. For example the shift M6 has 14 breaks which are represented in Figure 7

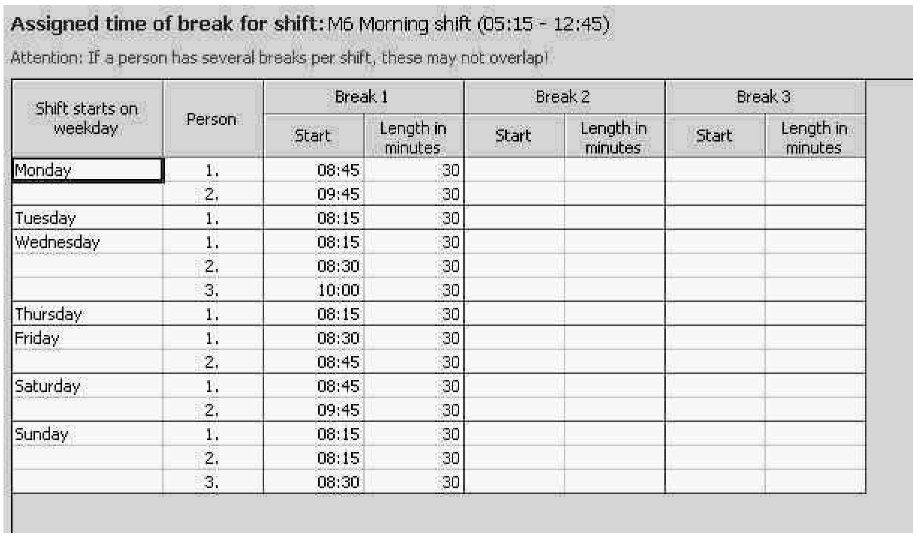
5 CONCLUSION

In this paper a system for the generation of shifts with breaks was presented. A method for the generation of breaks was discussed in detail. The presented method is based on the greedy assignment of breaks in legal positions in shifts and improvement of solution by local changes of length and position of breaks. The method was shown to give good results in practice. The use of this application at a large European airport was presented. The system can produce better solutions than experienced professional planners in shift scheduling although it



Abbr.	Full name	Start	End	Unpaid breaks in		WH in h	OH in h	Duties							
				Time assigned				Mon	Tue	Wed	Thu	Fri	Sat	Sun	Sum
M1	Morning shift	06:15	13:45	...	...	7:00	7:30	9	6	6	6	10	9	10	56
M2	Morning shift	07:45	17:00	...	...	8:45	9:15	1	1		1	1			4
M3	Morning shift	06:00	15:00	...	...	8:30	9:00	4	5	2	4	5	4	4	28
M4	Morning shift	07:00	14:30	...	...	7:00	7:30	3	4	3	3	3	3	3	22
M5	Morning shift	05:00	12:45	...	...	7:15	7:45	1	1	1	1				4
M6	Morning shift	05:15	12:45	...	...	7:00	7:30	2	1	3	1	2	2	3	14
M7	Morning shift	05:30	13:30	...	...	7:30	8:00	3			3		2		8
M8	Morning shift	05:15	13:15	...	...	7:30	8:00		1	1	1		1	1	5
M9	Morning shift	06:00	13:30	...	...	7:00	7:30	1	4	7	3	3	2	3	23
D1	Day shift	11:15	18:45	...	...	7:00	7:30		1	2	1				4
A1	Afternoon shift	13:30	21:45	...	...	7:45	8:15	1		2			1		4
A2	Afternoon shift	13:30	22:15	...	...	8:15	8:45				2			4	6
A3	Afternoon shift	15:45	00:00	...	...	7:45	8:15	1	1			1		1	4
A4	Afternoon shift	15:15	23:00	...	...	7:15	7:45						1		1
A5	Afternoon shift	13:45	21:30	...	...	7:15	7:45	2	3	1		4	3	2	15
A6	Afternoon shift	13:45	22:00	...	...	7:45	8:15	4	2	4	5	2	4	4	25
A7	Afternoon shift	15:00	22:30	...	...	7:00	7:30	7	7	4	5	7	5	4	39
A8	Afternoon shift	15:30	23:00	...	...	7:00	7:30	1	1	2	3	1	2	1	11
A9	Afternoon shift	14:30	22:15	...	...	7:15	7:45	2	3	4	3	3	2	2	19
P1	Post shift	19:45	05:00	...	...	8:45	9:15		1	1				1	3
N1	Night shift	22:15	06:15	...	...	7:30	8:00	4	3	3	4	4	4	3	25

Figure 6: Solution generated by OPA



Assigned time of break for shift: M6 Morning shift (05:15 - 12:45)

Attention: If a person has several breaks per shift, these may not overlap!

Shift starts on weekday	Person	Break 1		Break 2		Break 3	
		Start	Length in minutes	Start	Length in minutes	Start	Length in minutes
Monday	1.	08:45	30				
	2.	09:45	30				
Tuesday	1.	08:15	30				
Wednesday	1.	08:15	30				
	2.	08:30	30				
	3.	10:00	30				
Thursday	1.	08:15	30				
Friday	1.	08:30	30				
	2.	08:45	30				
Saturday	1.	08:45	30				
	2.	09:45	30				
Sunday	1.	08:15	30				
	2.	08:15	30				
	3.	08:30	30				

Figure 7: Breaks generated for shift M6

is based on a simple algorithm. It makes possible the generation of a good solutions in a relatively short time, thereby reducing costs and finding better solutions for problems that appear in practice. For the future work it would be interesting to further analyze whether generating shifts and breaks during the local search would give better results than solving this problem in two phases, although in our opinion this would increase the time to reach solutions because the number of neighborhood solutions during each iteration would be tremendously increased. Further, it would be interesting to analyze the behavior of other algorithms (e.g., tabu search) when the repair steps are applied in breaks in the last phase.

Acknowledgment

This work was supported by FWF (Austrian Science Fund) Project No. **Z29-N04**.

References

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

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

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

[4] Johannes Gärtner, Nysret Musliu, and Wolfgang Slany. Rota: A research project on algorithms for workforce scheduling and shift design optimisation. *Artificial Intelligence Communications*, 14(2):83–92, 2001.

[5] Luca Di Gaspero, Johannes Gärtner, Guy Kortsarz, Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. The minimum shift design problem: Theory and practice. In *11th Annual European Symposium on Algorithms, Budapest*, 2003.

[6] Nysret Musliu, Andrea Schaerf, and Wolfgang Slany. Local search for shift design. *European Journal of Operational Research*, 153(1):51–64, 2004.

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