Rota: a research project on algorithms for workforce scheduling and shift design optimization

Johannes Gärtner [a], Nysret Musliu [b] and Wolfgang Slany [b]

[a] *Ximes Corp. and Technische Universität Wien*
E-mail: gaertner@ximes.com
[b] *Technische Universität Wien*
E-mail: {musliu,slany}@dbai.tuwien.ac.at

The typical process of planning and scheduling a rotating workforce in an organization consists in designing shifts and then assigning employees to these shifts and to periods of rest (days-off). Successfully solving these problems has high practical relevance: Results from ergonomics indicate that shift schedules have a profound impact on the health and satisfaction of employees as well as on their performance at work. The solutions must also satisfy legal requirements and should meet the objectives of the employing organization. In the research project Rota, undertaken by the Database and Artificial Intelligence Group at the Vienna University of Technology in cooperation with Ximes Corp., systems for the design of shifts and assignment of employees to shifts and days-off were developed. We give an overview of these systems.

Keywords: workforce scheduling, shift design, local search, constraint satisfaction

1. Introduction

The typical process of planning and scheduling a rotating workforce in an organization consists of several stages [20]. The fist stage in this process is to determine the temporal requirements. At this stage the required number of employees with certain qualifications is determined 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, while at the same time designing the shifts and then assigning these shifts and days-off to employees. The literature provides different approaches for finding solutions on these stages. One of the approaches is to coordinate the design of shifts and the matching of shifts with employees, and to solve both like one problem [8]. Other approaches consider days-off scheduling and shift scheduling only after the shifts are designed [2,15,16]. One of the disadvantages of the second approach is that considering the design of shifts separately does not guarantee that a feasible solution for the assignment of these shifts can be found. However, solving the workforce scheduling problem in several separate stages makes the problem of general workforce scheduling easier to tackle.

Computerized workforce scheduling has been a subject of interest for researchers for more than 30 years. General definitions of workforce scheduling problems can be found in [8,18,12]. Tien and Kamiyama [20] give a good survey of algorithms used for workforce scheduling. Different approaches were used to solve problems of workforce scheduling. Examples for the use of exhaustive enumeration are [9] and [4]. Glover and McMillan [8] rely on integration of techniques from management sciences and artificial intelligence to solve general shift scheduling problems. Balakrishnan and Wong [2] solved a problem of rotating workforce scheduling by modeling it as a network flow problem. Smith and Bennett [19] combine constraint satisfaction and local improvement algorithms to develop schedules for anesthetists. Schaerf and Meisels [18] proposed general local search for employee timetabling problems. Critical features of the workforce scheduling algorithms are their computational behavior and flexibility for a wide range of problems appearing in practice. Recently Laporte [13] assessed rotating workforce scheduling algorithms proposed in the literature

AI Communications
ISSN 0921-7126, IOS Press.

saying that "[...] these are often too constraining and not sufficiently flexible for this type of problem".

Project Rota was undertaken by the Database and Artificial Intelligence Group[1] at the Vienna University of Technology in cooperation with Ximes Corp.[2] starting in 1998. The main objectives of the project were to develop efficient algorithms for shift design and for the assignment of days-off and shifts to the employees. In the following we give an overview of our results in this project, which consisted in developing two software packages, one for designing shifts and another one for assigning these shifts and days-off to employees.

2. The shift design problem

In the shift design problem we start from a given collection of possible shifts and workforce requirements for a certain cycle time (usually one week). We look for an optimal subset of the shifts together with an optimal assignment of workers to these shifts in such a way that the overall deviation from the requirements is small. A number of constraints are also considered:

- The possible shifts must satisfy all legal, ergonomic, and operating constraints.
- Excess and shortage of workforce can be acceptable to differing degrees.
- The number of selected shifts should be small.
- Workers should come to work less than 5 times per week on average, with shifts not longer than 9 hours; less often when shifts are longer than 9 hours (depending on circumstances there may be exceptions).
- Some other constraints may vary from case to case, e.g., there should be little variation in the selection of shifts between weekdays.

See Table 1 for an example of a set of possible shifts. With the typical resolution of 15 minutes, this example corresponds to a set of 324 possible shifts.

See Table 2 for an example for workforce requirements, i.e., how many persons should ideally be present at each time of the week. Staffing levels vary due to changing customer demands and internal organisational activities.

A typical solution for the problem in Table 2 is given in Table 3. Characteristics of this solution:

- Number of shifts used: 5 (out of the 324).
- Shortage: each day from 10h–11h there is a shortage of 2 workers.
- Excess: on Thursday from 9h–10h there is an excess of 2 workers.
- Average number of times workers have to commute per week: 5 (assuming 21 workers will work for 40h/week).

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. Thus, a solution in which we drop the objective to minimize the number of shifts can found with a very fast min cost max flow algorithm. For the problem in Table 2, the corresponding solution features less deviation by 2 worker-hours compared to the solution of Table 3, but needs a total of 18 shifts to achieve this provably optimal fitting of the requirements compared to 5 shifts in the solution of Table 3.

Kortsarz and Slany [11] have shown that the shift design problem is closely related to the NP-complete Minimum Edge Cost Flow problem (MECF = [ND32] in Garey & Johnson [5]). As a consequence, the shift design problem by itself is NP-hard. It is also difficult to approximate: Kortsarz and Slany [11] 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 be unusable. This strongly suggests that solving the problem to optimality is not feasible for anything beyond toy problems, and that heuristic methods must thus be used to solve realistically sized problems satisfactorily. This was indeed done in the project and is described in the present paper. The results generalize to the Minimum Edge Cost Flow problem and variants thereof (see [11]). This problem is one of the more fundamental flow variants with many applications. A sample of these appli-

[1] http://www.dbai.tuwien.ac.at/
[2] http://www.ximes.com/

Table 1
Typical set of possible shifts.

Abbreviation	Shift Type	Possible Start Times	Possible Durations
M	Morning	06:00 – 08:00	7h – 9h
D	Day	09:00 – 11:00	7h – 9h
A	Afternoon	13:00 – 15:00	7h – 9h
N	Night	22:00 – 24:00	7h – 9h

Table 2
Sample workforce requirement table for one week.

Start time	End time	Mon	Tue	Wen	Thu	Fri	Sat	Sun
06:00	08:00	2	2	2	6	2	0	0
08:00	09:00	5	5	5	9	5	3	3
09:00	10:00	7	7	7	11	7	5	5
10:00	11:00	9	9	9	15	9	7	7
11:00	14:00	7	7	7	13	7	5	5
14:00	16:00	10	9	7	9	10	5	5
16:00	17:00	7	6	4	6	7	2	2
17:00	22:00	5	4	2	2	5	0	0
22:00	06:00	5	5	5	5	5	5	5

Table 3
A typical solution for the problem in Table 2.

Shift	Start time	End time	Mon	Tue	Wen	Thu	Fri	Sat	Sun
M1	06:00	14:00	2	2	2	6	2		
M2	08:00	16:00	3	3	3	3	3	3	3
D1	09:00	17:00	2	2	2	4	2	2	2
A1	14:00	22:00	5	4	2	2	5		
N1	22:00	06:00	5	5	5	5	5	5	5

cations include optimization of synchronous networks, source-location, transportation, scheduling (for example, trucks or manpower), routing, and designing networks (for example, communication networks with fixed cost per link used, as in leased communication lines), see [11] for references.

In project Rota we implemented a min cost max flow algorithm to quickly get optimal values for all constraints besides the minimization of the number of shifts. These can then be used as reference values to evaluate solutions found with heuristic search methods.

2.1. Local search for shift design

As the problem that includes constraints for the minimization of the number of shifts is NP-hard and also hard to approximate, we rely on iterative methods for solving this problem. The basic idea of local search techniques (see [1,6,7,10,17]) is to improve initial solutions iteratively during the search. Descendants of the current solution are selected from the neighborhood of the solution which is constructed by 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 for accepting the descendants of the current solution. The appropriate neighborhood structure for these techniques is one of the most important features to reach 'good' solution.

A solution in the shift design problem consists of shifts and the duties of the shifts for each of the days during the planning period. The neighborhood of the solution is obtained by introducing altogether new shifts, removing a shift, or changing

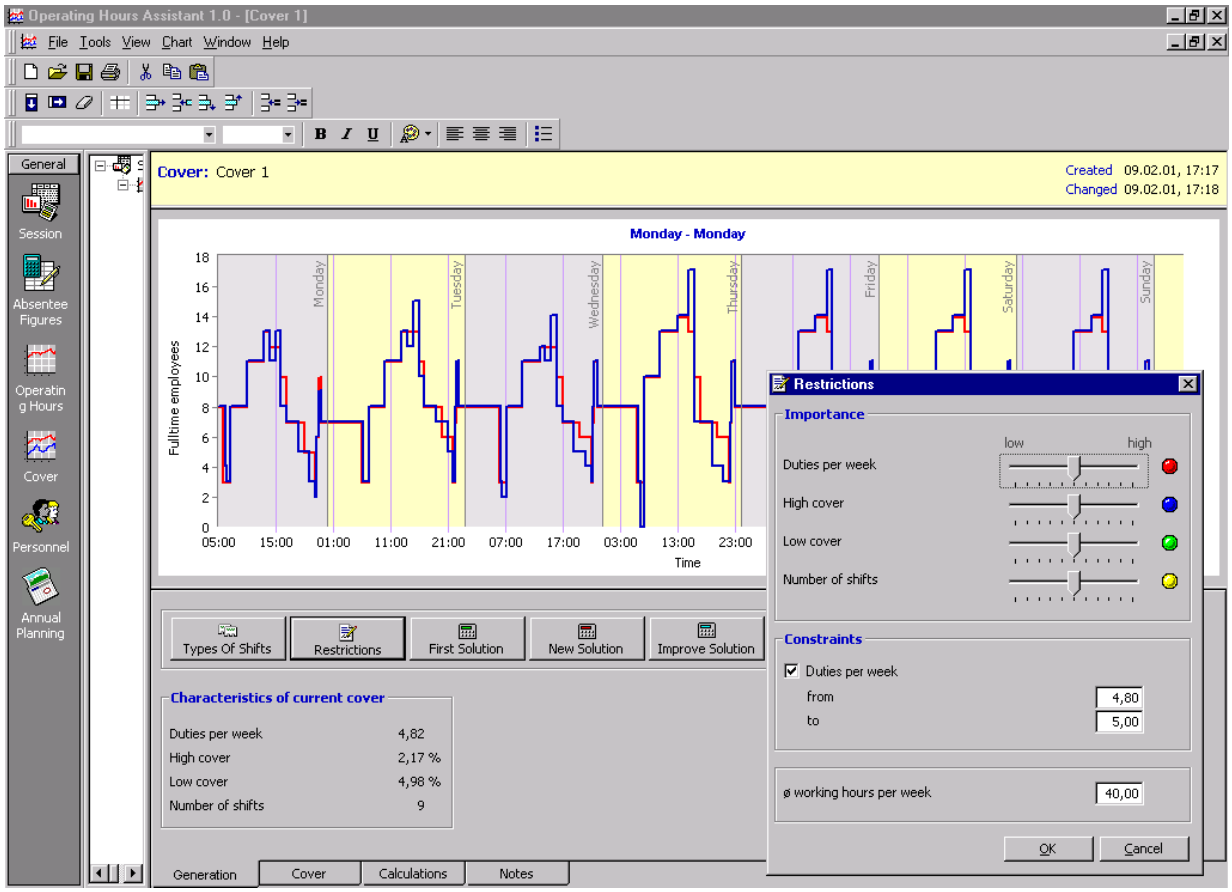

Fig. 1. Operating Hours Assistant.

one of the three main features of a shift, namely, start, length and duties per day. For the exploration of the neighborhood we use basic principles of the well-known Tabu search methods, where Tabu solutions which have been found are not accepted for a number of iterations to allow the escape from local optima. We first developed techniques that explore the whole neighborhood and accept the best solution from the neighborhood of the current solution which is not tabu, except in case the solution found is so far the best fitting. In problems taken from reality this technique has shown good results regarding the quality of the solution. However, to make the search more effective, we introduced new methods to guide the search during the exploration of the neighborhood. The basic idea here is to eliminate some of the moves during each iteration by considering the distance of the current solution from the ideal solution regarding the temporal requirements (minimization of shortage and excess is most important at this stage). Furthermore, a procedure for generating good initial solutions for this problem was developed. Computational results show that these ingredients added to the Tabu search technique significantly improve the effectiveness of the search. Together these techniques were implemented in a system called Operating Hours Assistant. A screenshot is given in Fig. 1.

Below we give an illustrative example and present results obtained by the guided search technique.

Problem 1: This problem is typical for many call-center problems. Temporal requirements are given in Table 4. Constraints about the shift types which can be used in the solution are given in Table 5. Fitness of the solution is a scalar function which is a linear combination of the four weighted criteria (excess and shortage in person-minutes, number of shifts, and average number of duties per week minus 5). Weights of the respective criteria are: $W1 = W2 = 1, W3 = 60, W4 = 1000$.

Table 4
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 5
Constraints about shifts in the call center problem.

Shift type	Earliest begin	Latest begin	Shortest length	Longest length
M (Morning shift)	05:00	08:00	07:00	09:00
D (Day shift)	09:00	12:00	07:00	09:00
A (Afternoon shift)	13:00	15:00	07:00	09:00

In Table 6 we show the solution produced by the guided search starting with a good initial solution.

The guided search has shown much better results compared to the Tabu Search by itself concerning fitness per number of evaluations. Of course, in a guided search the problem has to be processed additionally during each iteration to extract the rules. Nevertheless, time performances of guided search are much better than those of basic Tabu Search. A good initial solution can significantly increase the effectiveness of the search.

	1 Mo	1 Tu	1 We	1 Th	1 Fr	1 Sa	1 Su
A	D	D	D		A	A	N
B	N			D	D	D	
C		N	N	N	N		
D	A	A	A	A			

Fig. 2. A possible rotating schedule for 4 groups and 18 shifts per week.

3. Days-off and shift scheduling

Once the shifts are designed, the employees can be assigned to shifts and days-off for a given period of time. For instance, a schedule for four work groups is given in Fig. 2. A work group can consist of one or more employees. Group A works during the first week on Monday, Tuesday and Wednesday in Day shifts (D), Thursday it is off work, Friday and Saturday in Afternoon shifts (A) and on Sunday in Night shifts (N). In case of rotating schedules all groups have the same schedule during the whole cycle, and thus in the second week group A has the schedule of group B, group B the one of group C and so on. After 4 weeks (the length of a cycle) each group has its initial schedule again.

Typically rotating workforce schedules must fulfill some constraints like needed workforce per day, forbidden sequences of shifts, length of work and days-off blocks etc. Deciding whether a rotating workforce schedule that meets all constraints does exist is an NP-complete problem and thus finding such schedules is generally hard by itself [15]. This is consistent with the prohibitively large search spaces and conflicting constraints usually encountered in real-world problems in this area. Nevertheless, experienced professional planners can

Table 6
Solution for the call center problem found by the guided search method starting with a good inital solution.

Shift	Time	Mon	Tue	Wed	Thu	Fri	Sat	Sun
M1	07:00-15:00	5	5	5	5	5	1	1
M2	8:00-15:00	5	5	5	5	5	3	3
D1	10:00-19:00	3	3	4	5	5		
D2	12:00-19:00						4	4
D3	9:30-18:30	4	4	3	2	2		
A1	13:00-22:00	3	3	3	3	3	1	1
A1	15:00-22:00	10	10	10	10	10	4	4

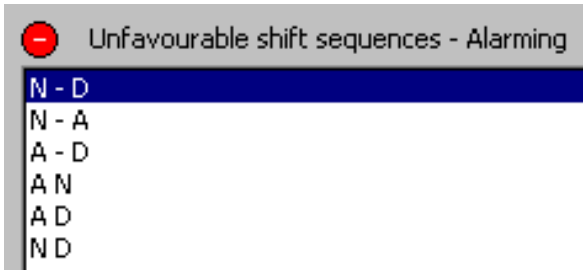

Fig. 3. Forbidden sequence of shifts.

construct acceptable schedules for most practical problems by hand. However, the time they need can sometimes be very long (one hour to several days), and, because of the large number of possible solutions, the human planners can never be sure how far their solution is from the best one. Therefore, the aim of automating the generation of schedules is to make possible the generation of high-quality schedules in a short time, thereby reducing costs and finding better solutions for the problems that appear in practice.

3.1. New framework for the generation of rotating workforce schedules

In our project we proposed a new framework for solving the problem of rotating workforce schedules. This framework consists of the following steps:

1. Definition of hard constraints. These constraints are about forbidden sequences of shifts (see Fig. 3), length of work blocks (a work block is a sequence of consecutive days of work shifts), days-off blocks and blocks of consecutive shifts (see Fig. 4).
2. Choosing a set of lengths of work blocks (see Fig. 5).
3. Choosing a particular sequence of work and days-off blocks among those that have optimal weekend characteristics (see Fig. 6).
4. Enumerating possible shift sequences for the chosen work blocks subject to shift change constraints and bounds on sequences of shifts.
5. Assignment of shift sequences to work blocks while fulfilling the staffing requirements (see Fig. 7).

First we outline our motivation for using this framework. Our approach is focused on the interaction with decision makers. Thus, the process of generating schedules is only half automatic. When our system generates possible candidate sets of lengths of work blocks in step 1, the decision maker will select one of the solutions that best reflects his preferences. This way we satisfy two goals: on the one hand, an additional soft constraint concerning the lengths of work blocks can be taken into account through this interaction, on the other hand the search space for step 2 is significantly reduced. Thus we will be able to solve step 2 much more effectively. In step 2 our main concern is to find the best solution for weekends off. The user selection in step 1 can impact features of weekends off versus length of work blocks since these two constraints are the ones that in practice are most often in conflict. The decision maker can decide if he wishes optimal length of work blocks or better features for weekends off. With step 3 we satisfy two more goals. First, because of the shift change constraints and the bounds on the number of successive shifts in a sequence, each work block has only few legal shift sequences (terms) and thus in step 4 backtracking algorithms will quickly find assignments of terms to the work blocks so that the requirements are fulfilled (if shift change constraints with days-off exist, their satisfaction is checked at this stage). Second, a new soft constraint is introduced.

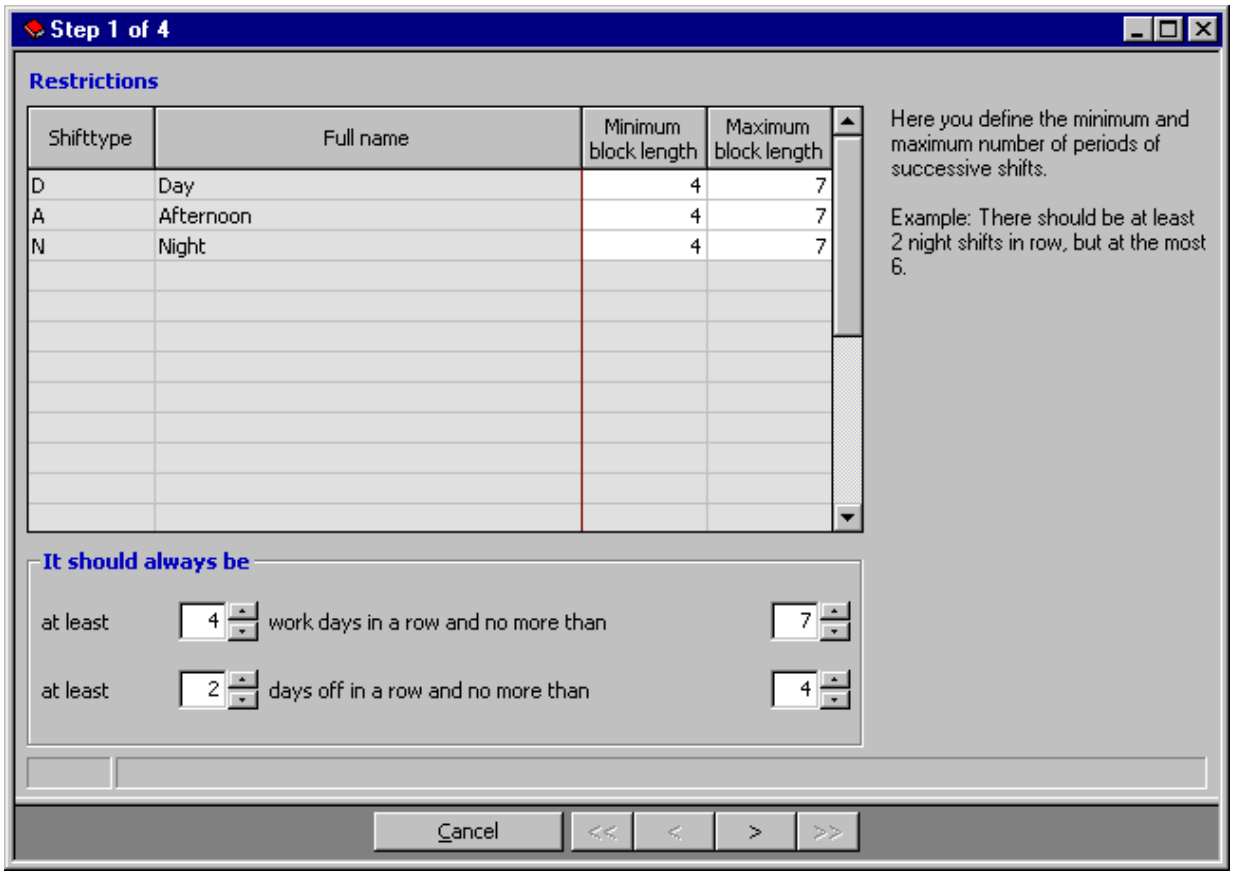

Fig. 4. Definition of hard constraints.

Indeed, as we generate a bunch of shift plans, they will contain different terms. The user has then the possibility to eliminate some undesired terms, thus eliminating solutions that contain these terms (see Fig. 7). Terms can have an impact on the fatigue and sleepiness of employees and as such are very important when high-quality plans are sought.

We implemented for each of steps 2-5 problem-oriented intelligent backtracking algorithms, which are very efficient because they make clever pruning based on problem constraints and because of reduced search space in each of the steps as a result of decomposition of constraint satisfaction in the steps.

Below we give an illustrative example for one benchmark example from the literature and the results produced from our package (called First Class Scheduler (FCS)). For more examples see [16].

Problem 2 (Laporte et al. [14]): There are three non-overlapping shifts D, A, and N, 9 employees, and the requirements are 2 employees for each shift and day. A weekly schedule that meets these constraints has to be constructed:

1. Rest periods should be at least two days-off.
2. Work periods must be between 2 and 7 days long if work is done in shift D or A and between 4 and 7 if work is done in shift N.
3. Shift changes can occur only after a day-off.
4. Schedules should contain as many weekends as possible.
5. Weekends off should be distributed as evenly as possible throughout the schedule.
6. Long (short periods) should be followed by long (short) rest periods.
7. Work periods of 7 days are preferred in shift N.

Balakrishnan and Wong [2] need 310.84 seconds to obtain the first optimal solution. The solution is given in Table 7. The authors also report about another solution with three weekends off, found with another structure of costs for weekends off.

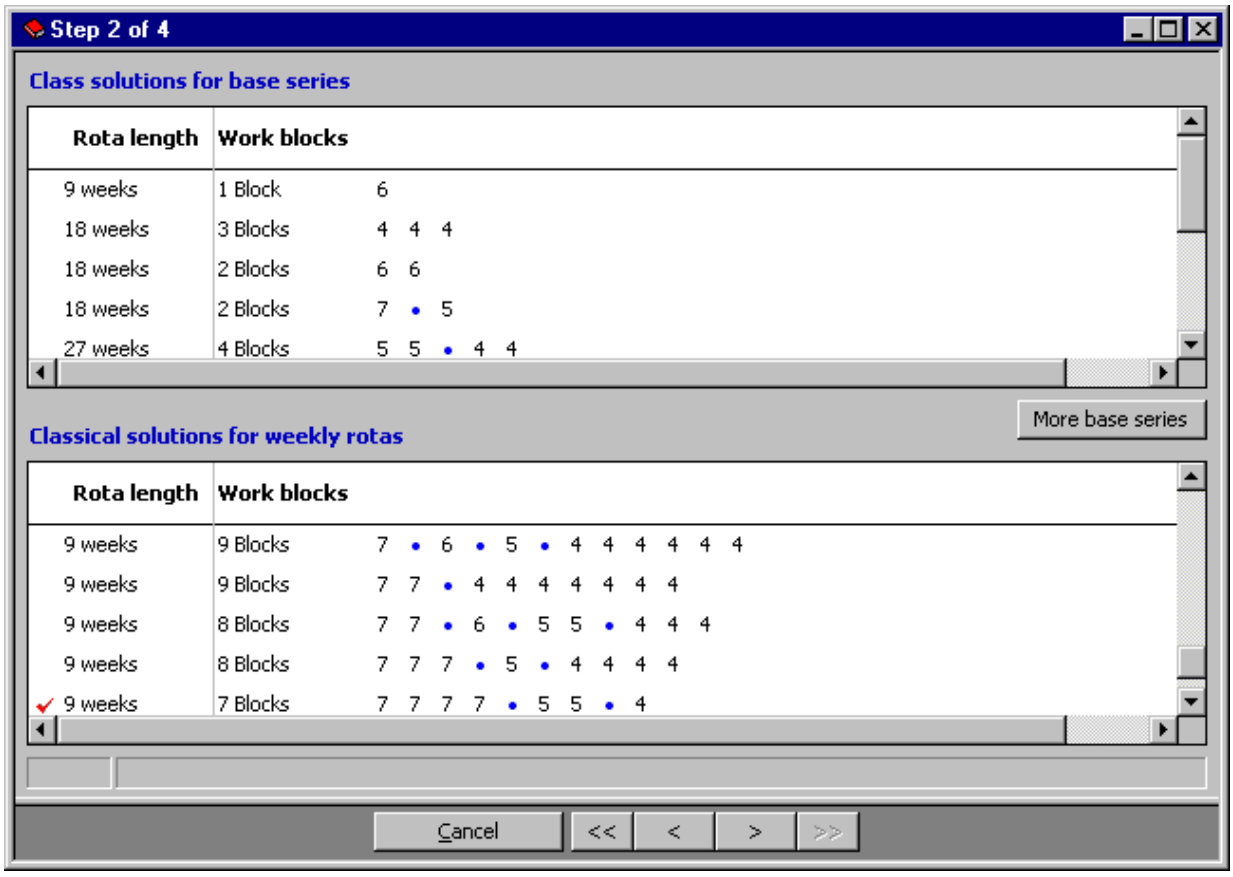

Fig. 5. Class solutions.

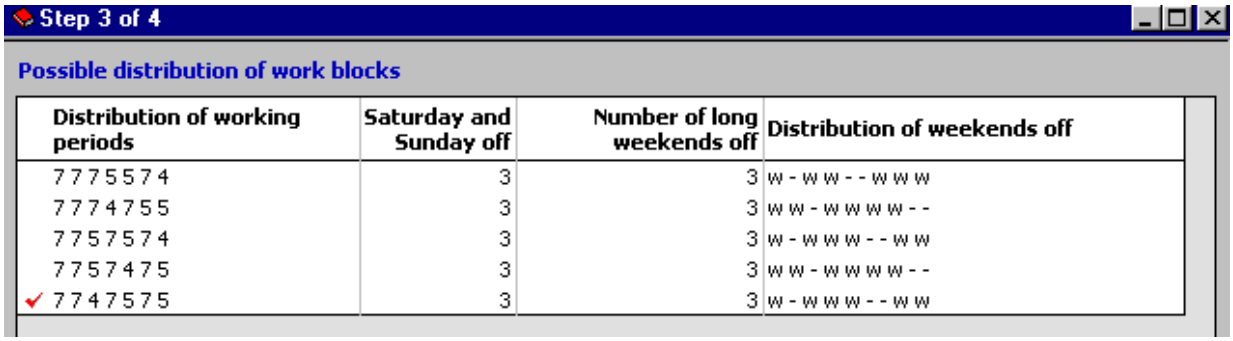

Fig. 6. Weekends information.

With FCS constraint 1 is straightforward. Constraint 2 can be approximated if we take the minimum of work blocks to be 4. Constraint 3 can also be modeled if we take the minimum length of consecutive work-type shifts to be 4. We set the maximum length of consecutive shifts to 7 for each shift. Constraints 4 and 5 are incorporated in step 2, constraint 6 cannot be modeled directly and thus has to be taken care of manually after step 2, and constraint 7 could be modeled by selecting appropriate terms in step 3. Note, however, that seven consecutive night shifts are strongly depreciated according to standard shift scheduling guidelines [3], and thus we do not take preference constraint

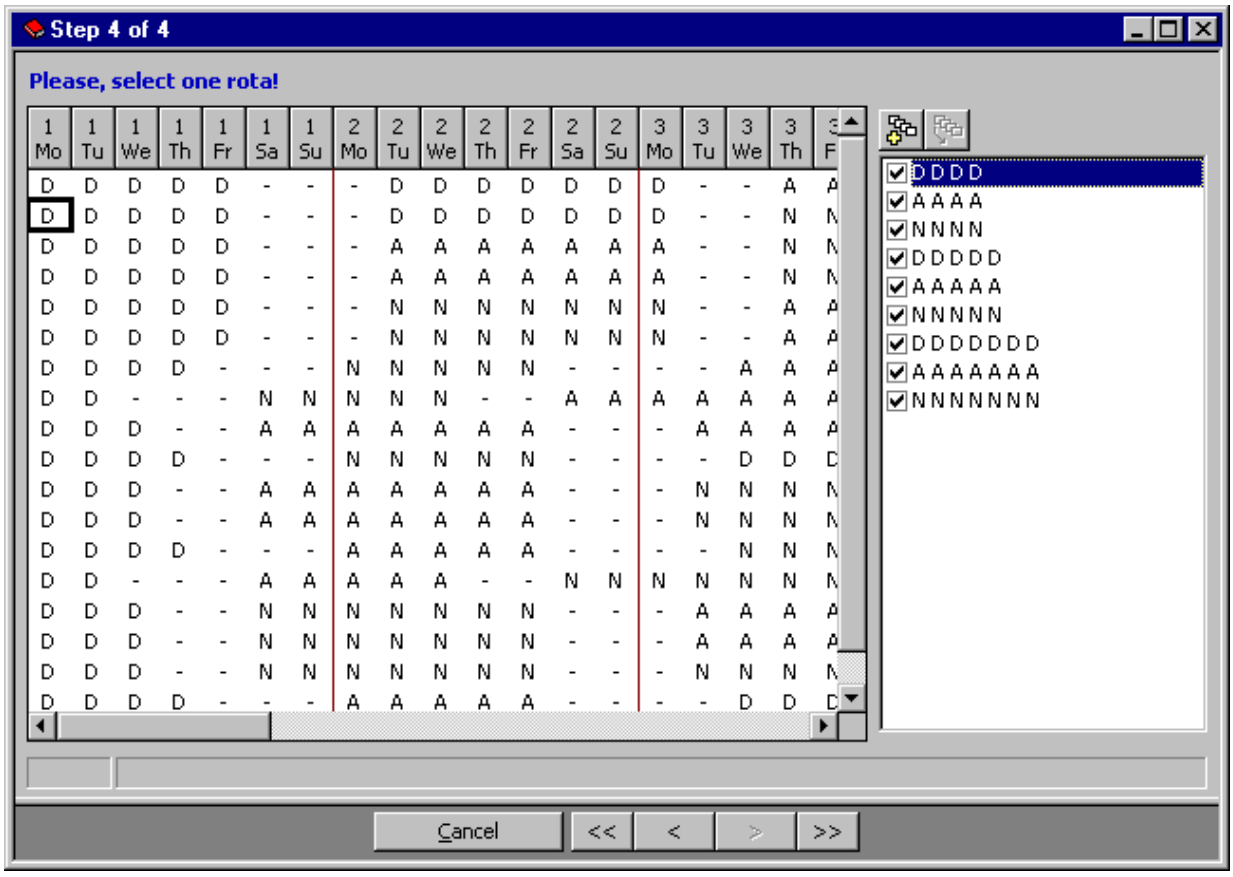

Fig. 7. Selection of shift sequences and rotas.

Table 7
Solution of Balakrishnan and Wong [2] of problem from [14].

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	A	A	A	A			D
2	D	D	D	D			
3		D	D	D	D	D	
4			A	A	A	A	A
5			N	N	N	N	N
6	N	N			A	A	A
7	A	A			D	D	D
8	D			N	N	N	N
9	N	N	N				

7 into account in the following.

With these given parameters for the problem we obtain 23 class solutions which are generated within 5 seconds. For each class solution we obtain at least one distribution of days-off, but it could be that no assignment of shifts to the work blocks exists because the range of blocks with consecutive shifts is too narrow in this case. As in this problem the lengths of blocks of consecutive shifts range from 4 to 7, no assignment of shifts can be found for many class solutions. Class solution {7 7 6 5 5 4 4 4} gives solutions with three free weekends, but they immediately follow each other. Class solution {7 7 7 7 5 5 4} shows a better distribution of week-

Table 8
First Class Scheduler solution of problem from [14].

Employee/day	Mon	Tue	Wed	Thu	Fri	Sat	Sun
1	D	D	D	D	D		
2		D	D	D	D	D	D
3	D			N	N	N	N
4					A	A	A
5	A	A	A	A			
6	N	N	N	N	N		
7			A	A	A	A	A
8	A	A				N	N
9	N	N	N			D	D

ends. If we select this particular class solution in step 1, our system will generate 5 solutions in step 2 in 1.69 seconds. We selected a solution with the order (7 7 4 7 5 7 5) of work blocks. Step 3 and 4 are solved simultaneously and the first solution came up after 0.08 seconds. 18 non-isomorphic solutions were found after 0.5 seconds. One of the solutions is shown in Table 8.

With class solution {7 7 7 7 7 7} the same distribution of weekends off can be found as in [14].

In general we can get to solutions much faster than Balakrishnan and Wong [2], even though they are reached in interaction with human decision makers. Because each step is so fast, the overall process of constructing an optimal solution does not take overly long either.

4. Conclusion and future work

We have implemented an optimization framework for the generation of rotating workforce schedules in a software package called First Class Scheduler (FCS), which is part of a general shift scheduling package Shift-Plan-Assistant (SPA) of Ximes Corp. Generation of rotating workforce schedules for most real cases can be done in a reasonably short time. FCS shows much better computational behavior for most previously published benchmarks compared to earlier developed algorithms. Another advantage of FCS is the possibility to generate high-quality schedules in interaction with human decision makers. Besides the fact that the generated schedules fulfill all hard constraints, our framework also allows to incorporate preferences of the human decision maker regarding soft constraints, which are otherwise more difficult to assess and to model. For instance, one may prefer longer blocks but better distribution of weekends to shorter work blocks yet worse distribution of weekends. Moreover, such interaction facilitates a better understanding of how requirements shape the solution space. The system thereby helps to relax requirements when the solution space is very tight. For some well-known scheduling problems with only a few good solutions (e.g., metropolitan and continental rota [4]) FCS finds exactly these solutions. With FCS it is easy to model the most important constraints occuring in the central European context. The package has been very well received internationally, and German, English, Finnish, and Dutch versions of it are in daily use throughout Europe.

We implemented algorithms for the shift design problem in a software package called Operating Hours Assistant. This product is in early stages of use and has seen first application in several organizations.

One interesting point for research in the future is to consider shift design and assignment of the shifts and days-off to employees as one problem, and solve both in coordination with each other. Also, we want to extend our algorithms to non cyclical schedules.

Acknowledgements

This research was partially supported by FFF project No. 801160/5979 and the Austrian Science Fund Project N Z29-INF.

References

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

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

[3] BEST. Guidelines for shiftworkers. Bulletin of European Time Studies No. 3, European Foundation for the Improvement of Living and Working Conditions, 1991.

[4] B. Butler. Computerized manpower scheduling. Master's thesis, University of Alberta, Canada, 1978.

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

[6] Fred Glover. Tabu search–part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.

[7] Fred Glover. Tabu search–part II. *ORSA Journal on Computing*, 2(1):4–32, 1989.

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

[9] N. Heller, J. McEwen, and W. Stenzel. Computerized scheduling of police manpower. *St. Louis Police Department, St. Louis, MO*, 1973.

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

[11] Guy Kortsarz and Wolfgang Slany. The minimum shift design problem and its relation to the minimum edge-cost flow problem. Unpublished manuscript, 2000.

[12] L. Kragelund and T. Kabel. Employee timetabling. Master's thesis, Department of Computer Science, University of Aarhus, Ny Munkegade, Building 540, DK-8000 Aarhus C, Denmark, 1998.

[13] G. Laporte. The art and science of designing rotating schedules. *Journal of the Operational Research Society*, 50:1011–1017, 1999.

[14] G. Laporte, Y. Nobert, and J. Biron. Rotating schedules. *Eur. J. Ops. Res.*, 4:24–30, 1980.

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

[16] Nysret Muslija, 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] Andrea Schaerf and Amnon Meisels. Solving employee timetabling problems by generalized local search. In *Paper presented at AI*IA '99*, 1999.

[19] Barbara M. Smith and Sean Bennett. Combining constraint satisfaction and local improvement algorithms to construct anaesthetist's rotas. In *Practical Approaches to Scheduling and Planning: Papers from the 1992 Spring Symposium*, pages 136–140, Menlo Park, California, 1992. AAAI Press.

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