数理解析研究所講究録
1185 巻 2001 年 172-181

Theory and practice of shift scheduling

Wolfgang SLANY[1], Nysret MUSLIU[1,2],
Guy KORTSARZ[3] and Johannes GÄRTNER[2]

[1]Institut für Informationssysteme
Technische Universität Wien
A-1040 Wien, Austria
{wsi,muslija}@dbai.tuwien.ac.at

[2]Ximes Corp.
Schwedenplatz 2/26
A-1010 Wien, Austria
gaertner@ximes.com

[3]Open University of Israel
Ramat Aviv, Israel
guyk@shaked.openu.ac.il

Abstract: Generating high-quality schedules for a rotating workforce is a critical task in all situations where a certain staffing level must be guaranteed, such as in industrial plants, hospitals, or airline companies. 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. Moreover, shift schedules must satisfy legal requirements and should also meet the objectives of the employing organization. Shift scheduling comprises several sub-problems such as the allocation of workers to shifts or the design of the shifts themselves that present interesting challenges both from practical (our algorithms are sold in a successful commercial package) as well as theoretical (hardness, approximability) viewpoints. We describe some of our results in both areas. In particular, we studied in depth the (non-)approximability of the shift design problem, a network flow variant that has many applications beyond shift scheduling.

Keywords: shift scheduling, shift assignment problem, shift design problem, approximation guarantees, nonapproximability, Tabu search heuristics, implementation

1 The shift assignment problem

In the shift assignment problem we match employees with shifts or days-off for a given period of time under certain constraints.

Rotating workforce scheduling is the assignment of employees to shifts or days-off for a given period of time. For instance, a four week schedule for four work groups is given in Figure 1. A work group can consist in one or more employees. Group A works during the first week on Monday, Tuesday and Wednesday in Day shifts (D), Thursday it has free, 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.

	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			

Figure 1: A possible rotating schedule for 4 groups and 18 shifts per week

Generating high-quality schedules for a rotating workforce is a critical task in all situations where a certain regular staffing level must be guaranteed, such as in industrial plants or police departments. Results from ergonomics [3]

indicate that rotating workforce schedules have a profound impact on the health and satisfaction of employees as well as on their performance at work. Moreover, rotating workforce schedules must satisfy legal requirements and should also meet the objectives of the employing organization. Deciding whether a rotating workforce schedules that meets all constraints does exist is an NP-complete problem [10] and thus finding such schedules is itself hard in general. 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 construct acceptable schedules for most practical problems by hand. However, the time they need can be sometimes very long (one hour to several days), and, because of the large space of 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 appears in practice.

1.1 Previous work

Computerized workforce scheduling has been the subject of interest of researchers for more than 30 years. General definitions of workforce scheduling problems can be found in [5, 12, 8]. Tien and Kamiyama [14] 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 [6] and [4]. Glover and McMillan [5] rely on integration of techniques from management sciences and artificial intelligence to solve general shift scheduling problems. Balakrishnan and Wong [1] solved a problem of rotating workforce scheduling by modeling it as a network flow problem. Smith and Bennett [13] combine constraint satisfaction and local improvement algorithms to develop schedules for anesthetists. Schaerf and Meisels [12] 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 that appear in practice. Recently Laporte [9] assessed rotating workforce scheduling algorithms proposed in the literature and wrote that "[...] these are often too constraining and not sufficiently flexible for this type of problem".

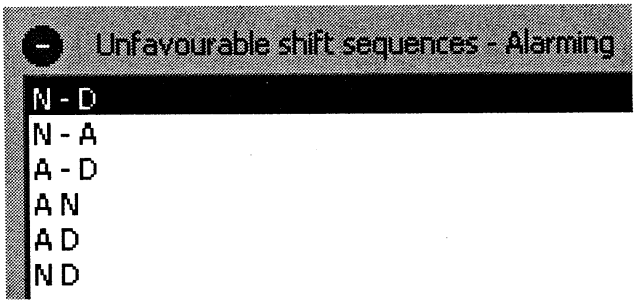

Figure 2: Forbidden sequence of shifts

Our main objectives were to develop a flexible system that shows good computational behavior and that can find high-quality solutions for most problems that appear in practice, especially in the case of tight problems for which it is very hard to find good solutions. To achieve these objectives we developed a new framework for solving the problem of rotating workforce schedules.

1.2 The framework

The framework implemented in First Class Scheduler (FCS) to solve the problem of efficiently and flexibly generating rotating workforce schedules consist of the following stages:

1. Definition of hard constraints.

   In this step the user defines hard constraints. These constraints are about forbidden sequences of shifts (Figure 2), length of work and days off blocks and length of blocks of consecutive shifts (see screen shot in Figure 3).

2. Choosing a set of lengths of work blocks (a work block is a sequence of consecutive days of work shifts) (Figure 4).

3. Choosing a particular sequence of work and days-off blocks among those that have optimal weekend characteristics (Figure 5). For instance, sequence (7 7 4 7 5 7 5) means a block of 7 work days followed by a block of rest days of unspecified length subject to the

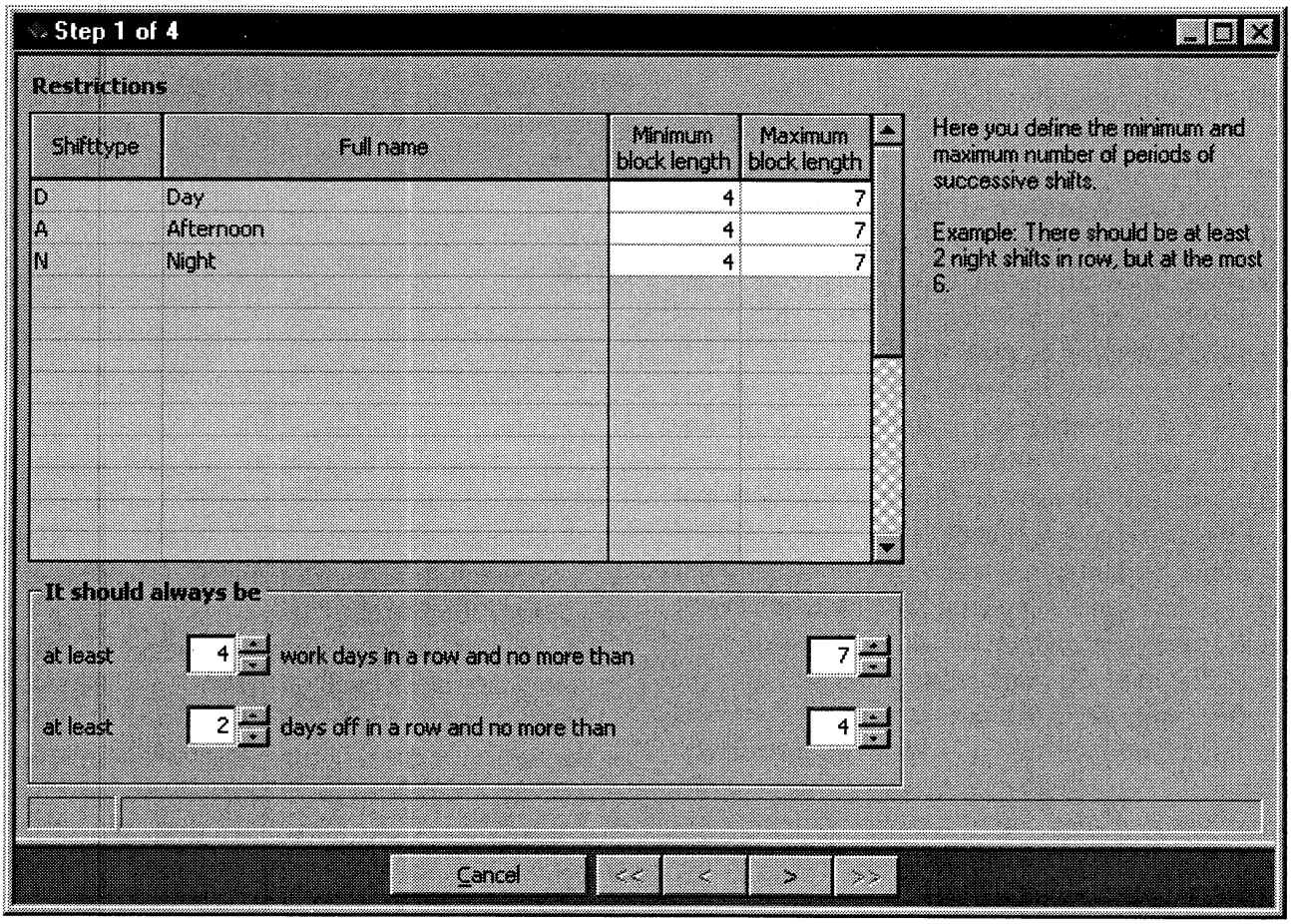

Figure 3: Definition of constraints

minimum and maximum lengths for days-off sequences chosen in step 1, followed again by a block of 7 work days, etc.

4. Enumerating possible shift sequences for the chosen work blocks subject to shift change constraints and bounds on sequences of shifts (right side of screen shot in Figure 6).

5. Assignment of shift sequences to work blocks while fulfilling the staffing requirements (left side of screen shot in Figure 6).

6. Comparison between non-isomorphic rotating schedules that all satisfy the same requirements as specified in the previous steps. One or more of them can then be further inspected to check their detailed characteristics (see Figure 7 for one of several checks that can be performed) and further manipulated.

We give our motivation for using this framework. Our approach is focused on the interaction with the decision maker. Thus, the process of generating schedules is only half automatic. When our system generates possible candidate sets of lengths of work blocks in step 2 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, and on the other hand the search space for step 3 is significantly reduced. Thus we will be able to solve step 3 much more effectively. In step 3 our main concern is to find a best solution for weekends off. The user selection in step 2 can impact features of weekends off versus length of work blocks since these two constraints are the ones that in practice most often are in conflict. The decision maker can decide if he wishes optimal length of work blocks or better features for weekends off. With step 4 we satisfy two more goals. First, because of the shift change constraints and the bounds on the number of suc-

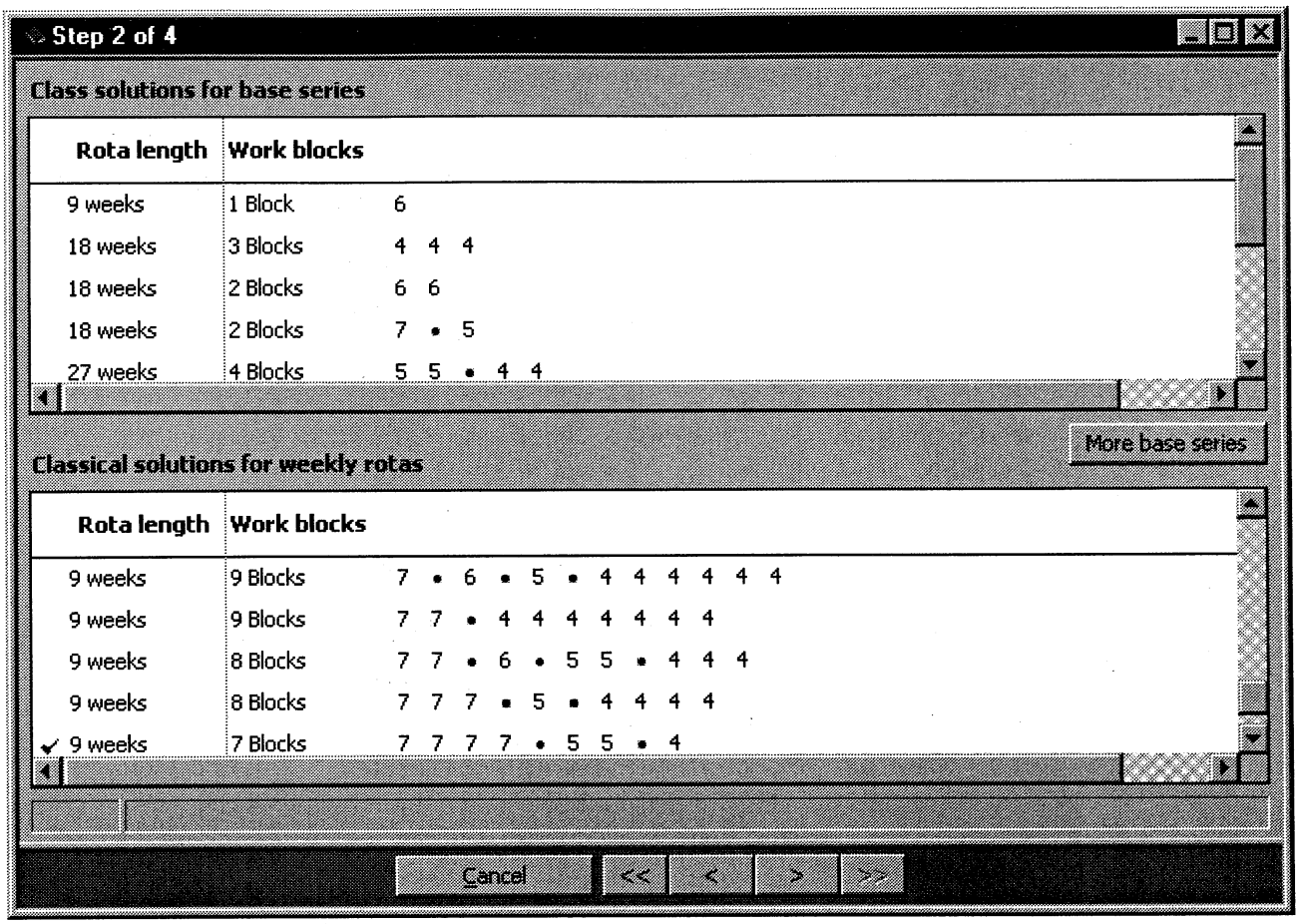

Figure 4: Class solutions

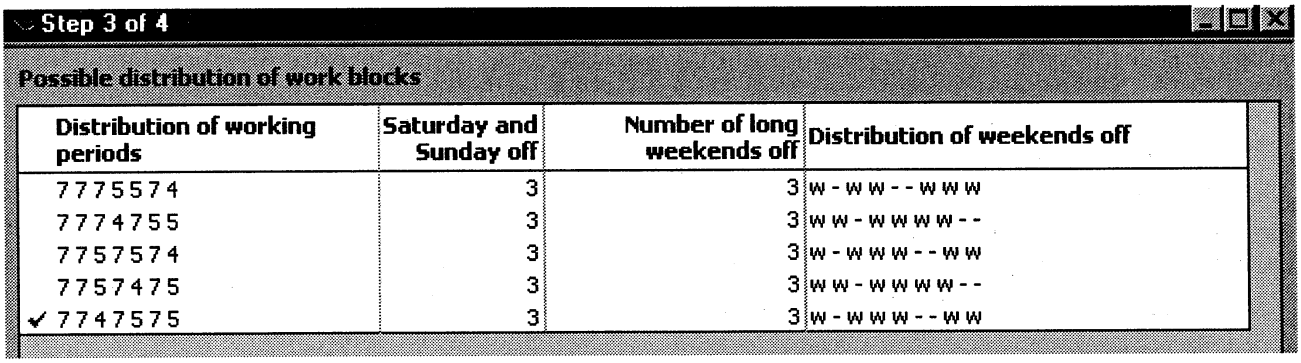

Figure 5: Weekends information

cessive shifts in a sequence, each work block has only few legal shift sequences (terms) and thus in step 5 backtracking algorithms will find very fast assignments of terms to the work blocks such 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. 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 (right side of Figure 6). Terms can have 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 to 5 backtracking algorithms. Their efficiency stems for one part from early pruning techniques based

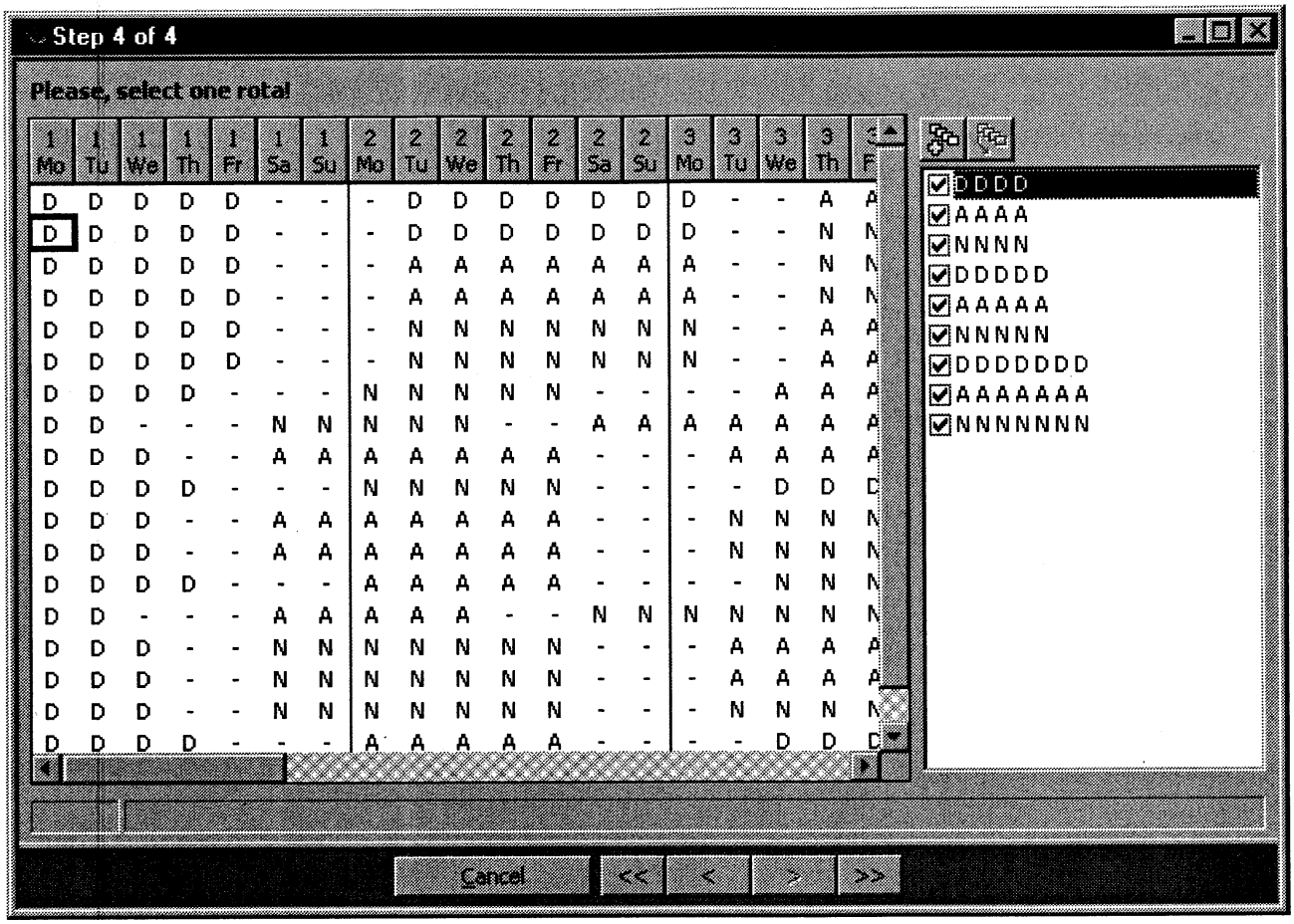

Figure 6: Selection of shift sequences and rotas

on problem constraints. Additionally, the search space could tremendously be reduced in each of the steps as a result of the decomposition of the constraints in the framework. A much more detailed description of the algorithms as well as benchmarks and comparisons with algorithms from the literature can be found in [11].

We implemented our optimization framework in a software package called First Class Scheduler (FCS) which is part of a general shift scheduling package Shift-Plan-Assistant (SPA) of XIMES[1] 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 through the interaction with the human decision-maker. Besides that the generated schedules fulfill all hard constraints, our framework also allows to incorporate preferences of the human decision maker regarding soft constraints that 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 but worse distribution of weekends. Moreover, this 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]) the FCS finds exactly these solutions. With FCS it is easy to model the most important constraints required in the central European context. The package has been very well received internationally, and German, English, Finnish and soon Dutch versions of it are in daily use throughout Europe.

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

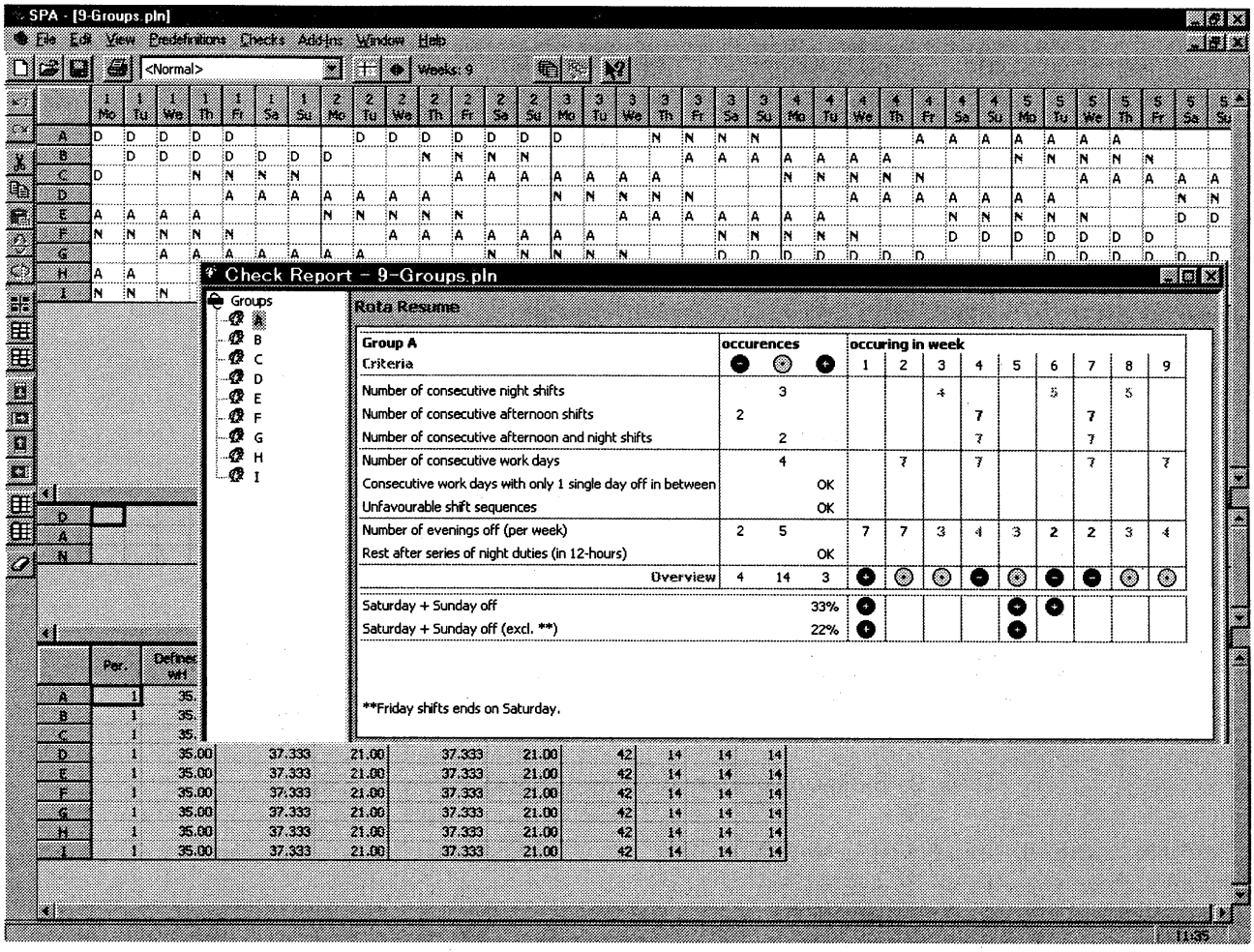

Figure 7: Full view of selected schedule, with ergonomics rota resumee

2 The shift design problem

In the shift design problem we are given a collection of shift templates and workforce requirements for a certain cycle time (usually one week). We look for an optimal selection of the shift templates together with an optimal assignment of workers to these shifts such that the overall deviation from the requirements is small, with a number of additional constraints:

- The shift templates are potential shifts under legal, ergonomic, and operating constraints.
- Over- and underhead can be differently weighted.
- The number of selected shifts should be small.
- Workers should come to work 4–5 times per week on average.
- Some other constraints varying from case to case, e.g. there should be little variation in the selection of shifts between weekdays.

Example: Shift templates = all possible shifts (typical temporal resolution: 15 minutes).

Example: Workforce requirements: How many persons should ideally be present at each time of the week.

Characteristics of the solution of Table 3:

- Number of shifts used: 5 (out of the 384)
- Underhead: each day from 10h–11h 2 workers
- Overhead: Wednesday 9h–10h 2 workers
- Average number of times workers have to commute per week: 4 (assuming 21 workers will work for 40h/week)

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 1: Typical set of shift templates. With a resolution of 15 minutes, this example corresponds to 384 possible shift templates.

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 2: Sample workforce requirement table for one week.

Another solution, found with a very fast min cost max flow algorithm: 2 worker-hours less, but the solutuion features a total of 18 shifts to achieve this better fitting of the requirements.

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 (following ideas from [2]). Basic idea:

- A matrix A with consecutive ones property (in the columns, corresponding to shift templates).
- A vector b of positive integers (corresponds to workforce requirements).

$$\text{E.g.,}\quad A = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix},$$

$b = (2, 3, 5, 4, 2, 1)$

- We look for a (positive) vector x such that
  1. $|Ax - b|_1$ is minimum (= minimum over-/underhead).
  2. Among all x minimizing $|Ax - b|_1$, the one that has a minimum number of non zero entries (= minimum number of shifts).
- We can also accommodate the other constraints and objectives, as we will hint at later.

Let T denote the following matrix:

$$T = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 & & 0 \\ 0 & 1 & -1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & -1 & 0 & & 0 \\ & \vdots & & \ddots & & & \vdots \\ 0 & 0 & 0 & & 1 & -1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 1 & -1 \\ 0 & 0 & 0 & & 0 & 0 & 1 \end{bmatrix}$$

As T is regular the two sets of feasible x vectors for $Ax = b$ and for $TAx = Tb$ are equal. However, TA can (almost) be interpreted as a flow matrix, Tb specifying the finite capacities on source (s) and sink (t) edges, all other edges having infinite capacity.

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

Table 3: A typical solution for the problem from Table 2

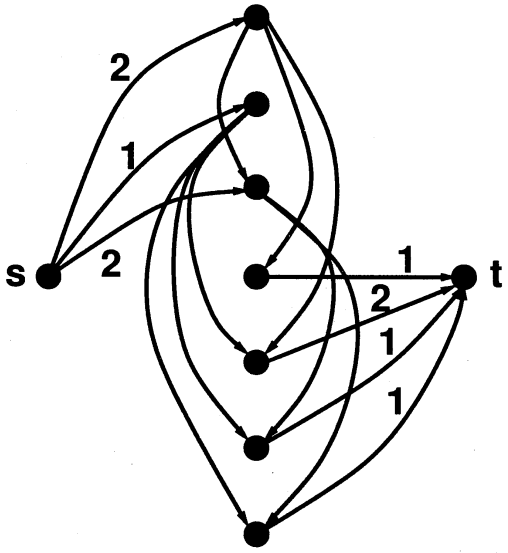

Figure 8: Flow graph corresponding to the sample shift problem.

$$TA = \begin{array}{c} \begin{matrix} -1 & -1 & -1 & & & & & \end{matrix} \\ \begin{bmatrix} 0 & 0 & 0 & -1 & -1 & -1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & -1 & -1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \end{bmatrix} \end{array},$$

$Tb = -2 \quad (-1,-2,1,2,1,1)$
$x^T = (0\ 1\ 1\ 1\ 0\ 0\ 1\ 1)$

Thus, shifts 2, 3, 4, 7, and 8 each are assigned one worker to perfectly satisfy this shift design problem.

- To minimize possible over- and underhead, we introduce slack variables (edges of infinite capacity and cost according to the relative weights of over- versus underhead). This part (min Σ cost $\times$ flow) can be solved by any min cost max flow algorithm (in polynomial time).
- To minimize the average number of times workers have to come in, assign uniform costs to all edges corresponding to shifts. Again solved through min cost max flow algorithm.
- To minimize number of selected shifts assign fixed charges to shift-edges. Closely related to the NP-complete Minimum Edge Cost Flow problem (MECF = [ND32] in Garey & Johnson). Remember also the second solution of the sample problem used many more shifts than the first one.

2.1 Theoretical results

Theorem 1 ([7]): 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.

The proof is based on a reduction from Set-Cover.

Theorem 2 ([7]): The shift design problem admits a $O(\sqrt{n \log M})$ ratio approximation algorithm (where M is the largest number in the requirements).

This means that we find a vector minimizing the overhead with $O(\sqrt{n \cdot \log M}) \cdot opt$ non-zero entries where *opt* is the minimum possible number of non-zero entries in a maximum flow.

The proofs of both results can be found in [7].

Similar results exist for the Minimum Edge Cost Flow problem and variants thereof (see [7]).

This problem is one of the more fundamental flow variants with many applications. A sample of these applications include optimization of synchronous networks, source-location, transportation, scheduling (for example, trucks or man-

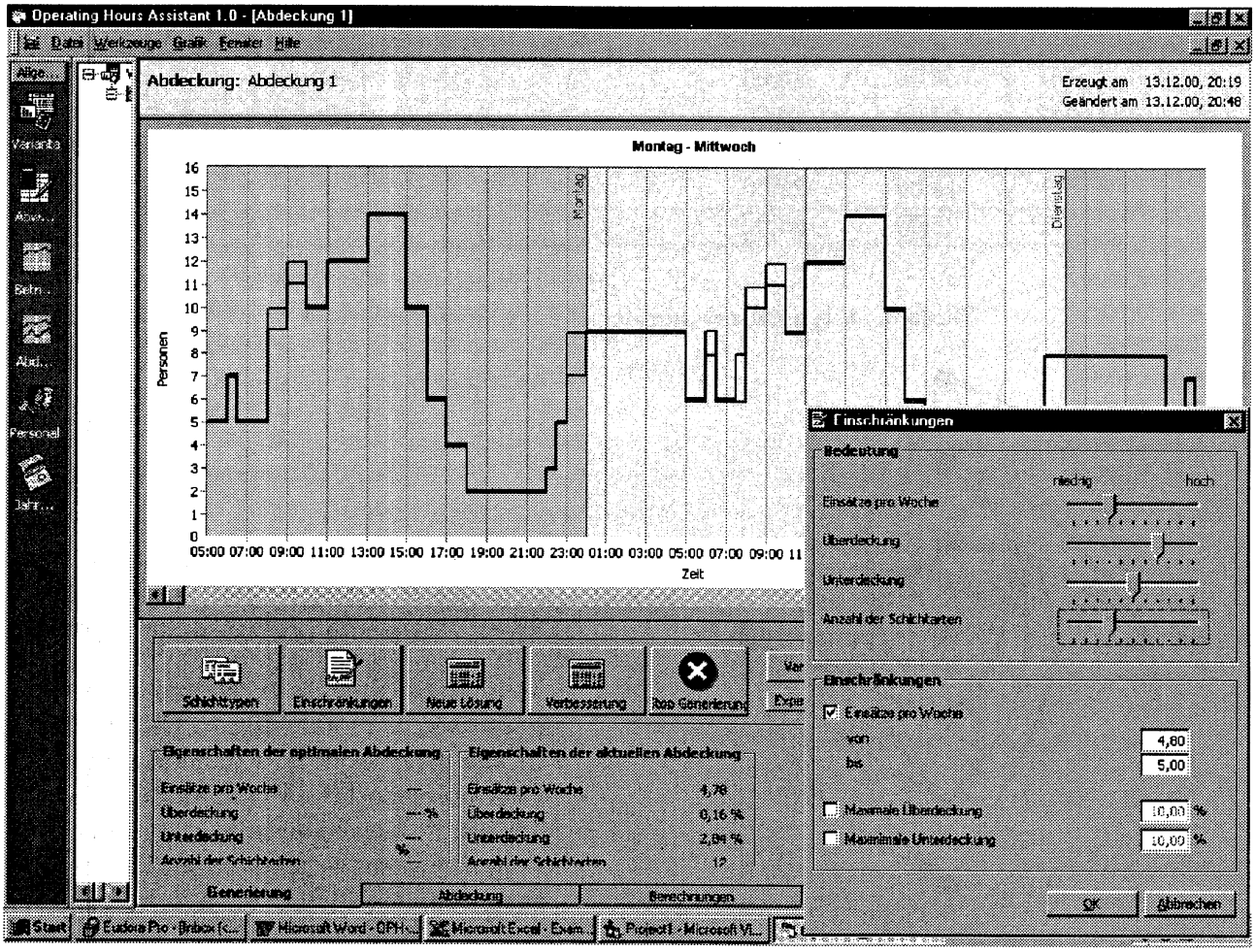

Figure 9: Operating Hours Assistant prototype screenshot (German language).

power), routing, and designing networks (for example, communication networks with fixed cost per link used, e.g., leased communication lines), see [7] for references.

2.2 Practical algorithms

The Operating Hours Assistant is work in progress. So far we implemented a min cost max flow algorithm to get reference values for all constraints besides the minimization of the number of shifts. Additionally, a Tabu search heuristic has been implemented that is working adequately for most problems, though there should be still room for improvement.

A screenshot of a preliminary German language version is given in Figure 9.

It is planned to compare our Tabu search heuristics with other heuristic methods (bundle-based relaxation methods, genetic algorithms, branch-and-bound methods, Lagrangian relaxation techniques, simulated-annealing heuristics, ...).

Acknowledgments

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

References

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

[2] J.J. Bartholdi, J.B. Orlin, and H.D. Ratliff. Cyclic scheduling via integer programs with circular ones. *Operations Research*, 28:110–118, 1980.

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

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

[7] Guy Kortsarz and Wolfgang Slany. The minimum shift scheduling problem. Unpublished manuscript, 2000.

[8] 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.

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

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

[11] Nysret Musliu, Johannes Gärtner, and Wolfgang Slany. Efficient generation of rotating workforce schedules. *Discrete Applied Mathematics*, 200? (to appear). http://www.arXiv.org/abs/cs.OH/0002018.

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

[13] 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.

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