Local Search for Shift Design

Nysret Musliu * Andrea Schaerf † Wolfgang Slany *

* Technische Universität Wien.
Favoritenstrasse 9-11, 1040 Wien
Email: {musliu, slany}@dbai.tuwien.ac.at

† Università di Udine
Via delle Scienze 208, I-33100 Udine
Email: schaerf@uniud.it

1 Introduction

The typical process of planning and scheduling a rotating workforce in an organization consists of several stages [6]. The fist 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 different approaches in the literature how 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 [3]. Other approaches consider days-off scheduling and shift scheduling only after the shifts have been designed [1, 5]. 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.

2 The Shift Design Problem

In this paper we consider only the problem of designing the shifts. More formally, we have:

- A set of time slots composing each day and the number of days in the cycle. Each time slot corresponds to an interval of fixed length (e.g. 15 minutes).
- The workforce requirements for each slot for each day.
- A collection of *Shift Types*. A shift type determines the minimum and maximum length and the earliest and latest start of a shift. The shift types used in our experiments are shown in Table 1.
- An upper limit for the average number of working shifts per week per employee.

The aim is to generate a set of shifts and a number of workers for each day for each shift. The objective function of the problem (to be minimized) is the weighted sum of the following components:

1. Number of shifts.
2. Sum of the excesses/shortages of workers w.r.t. the requirements for all slots in all days.
3. Distance of the the average number of duties per week w.r.t. the upper limit.

Kortsarz and Slany [4] 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.

Table 1: Input shifts types

Shift type	Earliest begin	Latest begin	Shortest length	Longest length
Morning shift	05:00	08:00	7 hours	9 hours
Day shift	09:00	11:00	7 hours	9 hours
Afternoon shift	13:00	15:00	7 hours	9 hours
Night shift	21:00	23:00	7 hours	9 hours

Exploiting this modeling, if ones drop the objective to minimize 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 & Johnson [2]), Kortsarz and Slany show also that, on the contrary, the general problem is NP-complete. In addition, they show that the problem is hard to approximate: 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. Given that the problem is hard, we decided to rely on local search methods for solving it. In particular we use a variant of tabu search, and in the next section, we describe how we apply tabu search to the problem.

3 Local Search for Shift Design

In order to apply local search techniques, and tabu search in particular, we need to define the search space, the neighborhood relation, the tabu mechanisms, and the selection rule for the initial state for to shift design problem.

3.1 Search Space and Neighborhood Relations

An element of the search space consists of a set shifts and the duties of each shift for each of the days during the planning period.

According to this search space, we cosider several neighborhood relations obtained by introducing new shifts, removing a shift, or changing one of the three main features of a shift, namely, start, length and duties (workers) per day. In details, the neighborhood relations we consider are the following ones:

ChangeStart: the start point of a shift is moved forward or backward by one slot (the length is left unchanged).

ChangeLength: the length of the shift is increased or decreased by one slot (the start point is left unchanged).

ChangeDuty: the number of duties for a particular day of the shift is increased or decreased.

CreateShift: a new shift is inserted. This move is actually implemented indirectly: We always keep for each shift type a so-called *reserve* shift, which has all duties 0 and exists only as a virtual shift in the solution. When a duty is added to a reserve shift (by a ChangeDuty move), it becames a normal shift, and a new reserve shift is added.

RemoveShift: Remove a shift. This move is also implemented in an indirect way: when a shift has all duties 0, it is eliminated.

Besides the above mentioned basic moves, we consider also composite moves, which are also checked during the search. They are combination of the previous moves:

ExchangeDuties: pass one duty in a particular day from one shift to another one.

MergeShifts: two shifts are merged. One is eliminated, and all its duties are passed to the other one.

SplitShift: a shift is divided into two shifts. The newly created shift differs from the original shift by one time slot in length or begin. One duty per day (when possible) is passed to the newly created shift.

MoveBorders: Given two *touching* shifts, i.e. two shifts such that the end of the first one is the start of the second one, move the touching point forward or backward. That is, the first is enlarged and the second shrunk, or vice versa.

ChangeBegin: The start point of a shift is moved, while keeping its end fixed.

In the case of basic tabu search at each iteration, all neighborhood relations are considered, while in case of guided search some of the neighborhood relations may not be considered at each iteration. However in two cases we do not explore the whole neighborhood: based in the size of the neighborhood, for some of the move types, the exploration is stopped as soon as an improving solution is found, while for some others all neighbors are examinated.

3.2 Tabu Mechanism

We have experimented with two different prohibition mechanisms. In the first variant, for each of the moves described above we consider the *inverse move*. 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, the inverse is stored and will be tabu for a determined number of iterations. In the second variant, a solution itself is considered tabu (indeed in the memory are stored only the features of the solution: number of shifts, difference in the duties in each time slot and average number of duties per week).

The aspiration criterion is a very standard one: a tabu move is accepted only if it leads to the solution with a cost better than the best cost current one.

3.3 Initial Solution

We explore with two different ways of generating the initial solution. The first is a random selection of shifts (one shift per each shift type) with all duties 0, the second instead tries to find a "good" starting point.

The basic idea in constructing a good initial solution is that in every change of the number of duties in 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.

3.4 Exploiting Knowledge During Search

The share of the neighborhood that should be explored during each iteration can be reduced using knowledge about the problem. The basic idea is to take moves which probably will improve the solution. 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. The procedure of guided search is described below:

1. Find the shortage/excess with the longest length
2. Determine contribution of each shift type in the shortage or excess: high, middle, small
3. Begin with the shift type that has the highest contribution in the shortage/excess
4. Determine 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 used for exploring of the neighborhood
6. Based on the shortage/excess properties and position and contribution of the shift type relative to the shortage/excess determine moves, and days, and shift type with which the neighborhood will be explored

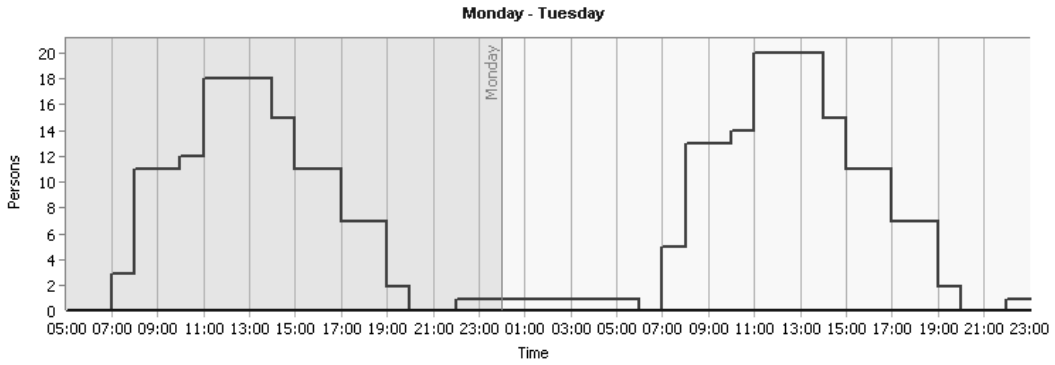

Figure 1: Requirements in example 1 for Monday and Tuesday

7. If there exist improvements repeat steps from the begin, else if there exist no improvements, but still shift types that contributes in the shortage/excess then take next shift type and go in step 4, else if there exist no improves and no more shift types that contributes to shortage/excess then apply for the next iteration complete exploring of the neighborhood

Basic feature of guided search procedure is that the search is concentrated only in the days where the shortage/excess appears. Moreover, some of the moves are applied only to the determined shift type, in region of which the shortage/excess appears.

Example: In the Figure 1 the requirements are given for the example 1 in the computational results (only the first two days can be seen). Applying the above procedure the longest shortage found will be the one which starts in the first day at 22:00 with length 8 hours and has 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 Night shift. The guided search procedure explores first the neighborhood that is constructed using the move ChangeDuty (only increase of duties, neighborhood solutions that would be obtained with decrease of duties are not generated), and does this only for day one (solutions for other days are not explored).

4 Computational results

In this section we report our preliminary computational results in a number of artificial examples generated randomly. For these examples we compare three techniques:

TS: The basic tabu search with random initial solution
TSaGS: TS in combination with guided search
TSaGSwIS: TSaGS with the good initial solution outlined above

All results are given for one run of the algorithm as the latter is deterministic. All three algorithms are stopped after 70000 iterations (for each instance).

The length of the Tabu list is taken to be 40 and the second prohibition mechanism (as described above) is used. The techniques are implemented in a software package called Operating Hours Assistant of Ximes[1] Corp. and the system works well on practical examples. All our results in this section have been obtained on an Intel P2 (333 Mhz).

In Table 2 results for 30 instances are given. For each instance, the table shows the minimum cost found and the number of iterations needed for finding this cost. In the second column the cost of best known solution is given (although this is most of the time an optimal solution, we cannot fully guarantee it). Each of the techniques generates solutions that are acceptable in practice. However,

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

Table 2: Experimental results for random examples

		Cost of solutions			Number of evaluations		
Instance	Best known cost	TS	TSaGS	TSaGSwIS	TS	TSaGS	TSaGSwIS
1	480	2040	480	480	22322	2658	887
2	300	750	540	1020	62455	24726	33978
3	600	1380	1800	600	70117	42736	5123
4	480	2250	4980	1950	69089	69950	25598
5	480	480	480	480	25102	17097	17876
6	420	480	720	1020	16954	31438	245
7	270	1080	510	600	52855	67167	26936
8	150	180	1935	2685	56736	55403	59171
9	150	345	150	225	38229	37633	14487
10	330	2160	1740	630	65144	69915	69624
11	30	30	30	30	2531	447	183
12	90	90	90	165	21530	2495	2009
13	105	105	510	105	24853	23430	2970
14	195	6120	2145	510	70213	38405	70529
15	180	180	180	180	2045	224	4
16	225	1380	1500	360	70380	71175	56705
17	540	7170	8340	1560	70209	67835	69698
18	720	3900	720	720	70341	52891	2278
19	180	2970	2685	180	70054	43754	889
20	540	540	540	1080	53946	17776	13969
21	120	120	120	120	42942	39001	186
22	75	75	90	225	9325	5652	1818
23	150	2505	1560	570	70204	65827	20258
24	480	480	480	480	30191	9382	5814
25	480	2580	1560	720	70632	67311	69052
26	600	720	600	600	46598	57192	3782
27	480	1320	1680	1620	69915	66653	1974
28	270	1410	1530	270	69338	19175	827
29	360	1620	1170	390	69937	62584	13491
30	75	75	75	75	6774	2186	566

Tabu Search in combination with Guided Search and Initial Solution (TSaGSwIS) achieves best results concerning number of found optimal solutions and difference on total cost between generated solutions and best known solutions.

References

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

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

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

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

[5] 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.

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