

Title: Comparing mathematical, local search, and hybrid heuristics for solving the minimum shift design problem with breaks.

Wolfgang Slany¹
Institut für Softwaretechnologie
Technische Universität Graz
wsi@ist.tugraz.at

Abstract

The purpose of this paper is to compare a number of mathematical, local search, and hybrid approaches based on their practical efficiency and the accuracy of results to tackle the minimum shift scheduling problem with breaks keeping a number of constraints which occur in real life situations into consideration. This problem is of high practical importance as it appears in many areas of workforce scheduling like in big organizations, airports, hospitals, traffic control etc. We have used different approaches to solve the problem, which are:

1. Using the set covering model of Dantzig (1954) using several linear programming packages including CPLEX.
2. Using the implicit modeling approach of Bechtold and Jacobs (1990) also with CPLEX.
3. Using the minimum edge-cost flow approach of Di Gaspero et al. (2003) using Easy-Local++.
4. Using Operating Hours Assistant - A local Search Based System for Generation of Shifts with Breaks of Gaertner et al. (2004).
5. Using Constraint Satisfaction Programming (CSP) with the Mozart Programming System (<http://www.mozart-oz.org>).
6. Using a new hybrid approach combining Tabu and other local search heuristics with linear programming approaches.

The computational results are for a real life problem in a large European Airport.

1. Introduction

The *min*-SHIFT DESIGN problem (MSD) is of high practical importance as it appears in many areas of workforce scheduling like in big organizations, airports, hospitals, traffic control etc. The purpose of this paper is to present full scale models of the minimum shift scheduling problem with breaks, using the approaches suggested by Dantzig (1954) and Bechtold and Jacobs (1990), keeping a number of constraints, which occur in real life situations, into consideration, and comparing the various approaches for solving it based on their practical efficiency and accuracy of results. 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

¹ The research herein is partially conducted within the competence network Softnet Austria (www.soft-net.at) and funded by the Austrian Federal Ministry of Economics (bm:wa), the province of Styria, the Steirische Wirtschaftsförderungsgesellschaft mbH. (SFG), and the city of Vienna in terms of the center for innovation and technology (ZIT).

The min-shift design problem with breaks is a scheduling problem in many industrial contexts. Basically, the problem is to find an assignment schedule (shifts), with appropriate breaks during the shifts, and the number of employees to be assigned to these shifts so as to minimize the deviation from the organization's workforce requirements. The solution should take care of some practical considerations and meet certain constraints specific to each organization. First we describe problem of generation of shifts without breaks. Then the extension of this problem by including breaks is presented.

Instance:

1. n consecutive time intervals $[a_1, a_2), [a_2, a_3) \dots [a_n, a_{n+1})$, 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 beginning 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.
2. y shift types $v_1 \dots v_y$. 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.
3. An upper limit on the maximum number of employees that can be present at a particular time.

Problem:

Generate a set of k shifts $s_1 \dots s_k$. Each shift s_i has adjoined parameters s_i .start and s_i .length and must belong to one of the shift types. Additionally, each real shift s_p has adjoined parameters s_p . w_i for all i in $\{1 \dots C\}$ (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 weighted sum of the following components given below:

1. Sum of the excesses of workers in each time interval during the planning period.
2. Sum of the shortages of workers in each time interval during the planning period.
3. Sum of deviations of the shift lengths from the optimal shift length (input parameter).
4. Number of shifts k .
5. Number of shifts 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; the 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.

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 above 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).

Apart from the above constraints, there are certain practical considerations in real life situations that have to be taken care of. These are:

1. To make sure that the assignments are cyclic i.e. for example, for each week, the assignments at Sunday midnight should be same and consistent with the assignments at the beginning of the Monday morning next week.
2. While counting the number of assigned shifts, shifts on the same day, with same starting and ending times but with different break times/durations are to be considered as one shift.
3. Shifts on different days, with same starting and ending times are to be considered as one shift.

3. Approach using set covering model of Dantzig (1954)

3.1 Introduction

We have modeled the problem as a pure integer linear model in AMPL based on the set covering formulation introduced by Dantzig in 1954 which can be solved using CPLEX solver. We have taken into account the various above mentioned constraints one by one and formulated them mathematically. Consider the following situation/instance:

- a) There are four different types of shifts for employees namely the Morning Shift, Day Shift, Afternoon Shift and the Night Shift.
- b) Each shift has a minimum and maximum start time, which represent the earliest and latest start of the shift; and minimum, maximum length, which represent the minimum and maximum lengths of the shifts. Each shift also has an optimal shift length.
- c) Workforce requirements for the organization in every timeslot are given.

We give a formal description of the problem via an $n * m$ matrix as follows. We are given an $n * m$ matrix $TIMESLOT$ in which each row corresponds to a possible shift. Each entry $TIMESLOT_{st}$ in the matrix is either 1 if t is a valid timeslot for shift s or 0 otherwise. We are also given a vector requirements of length T of positive integers; each entry b_t corresponds to the workforce requirement for the timeslot t . With these settings, the problem can be stated as a system of equations we describe in the following sections.

3.2 Overcover and Undercover

We add the following constraint to calculate the workforce overcover and undercover during various timeslots:

Subject to $\forall t \in \{1..T\}$

$$\left(\sum_{i \in shifts} TIMESLOT[i][t] * assignment[i] \right) + undercover[t] - overcover[t] = requirements[t]$$

$\forall i \in shifts,$

$$undercover[i] \geq 0$$

$$overcover[i] \geq 0$$

where,

T is the number of timeslots in a cycle, for example, a week or a month depending on the planning period; and $undercover[t]$ and $overcover[t]$ are non-negative integers corresponding to the workforce excess and shortage during the corresponding time slot.

So, we add $weight_undercover * \sum_{i \in shifts} assignments[i]$ in the objective function (to be minimized) in order to take the workforce undercover into account.

Similarly, we add $weight_overcover * \sum_{i \in shifts} assignments[i]$ to the objective function to take the overcover into account.

3.3 Shift Length

Next, in the objective function, we need to consider the deviation of the shift lengths from the optimal shift length. The approach is similar to the one we used above to calculate the overcover and undercover. We add the following constraint:

Subject to $\forall i \in \text{shifts}$

$$(\text{ShiftUsed}[i] * \text{ShiftLength}[i]) + \text{ShiftShorter}[i] - \text{ShiftLonger}[i] = \text{ShiftUsed}[i] * \text{OptimalShiftLength}$$

where,

$\text{ShiftUsed}[i]$ is a binary variable for each shift which is 1 if there is at least one assignment in the shift or 0 otherwise; $\text{ShiftShorter}[i]$ and $\text{ShiftLonger}[i]$ are non-negative integers corresponding to the deviation of the shift length of shift i from the optimal shift length. For example, if the shift length for shift i is more than the optimal shift length, then $\text{ShiftLonger}[i]$ equals the number of timeslots $\text{ShiftLength}[i]$ exceeds $\text{OptimalShiftLength}$ and $\text{ShiftShorter}[i]$ is 0. Similarly, the vice versa follows if shift length is lesser than the $\text{OptimalShiftLength}$.

So, we add $\text{weight_shorter_shift_length} * \sum_{i \in \text{shifts}} \text{ShiftShorter}[i]$ in the objective function to take too short shifts into account.

Similarly, we add $\text{weight_longer_shift_length} * \sum_{i \in \text{shifts}} \text{ShiftLonger}[i]$ to the objective function to take long shifts into account.

To decide whether a shift is being used or not, i.e. to compute $\text{ShiftUsed}[i]$, we introduce a new constraint:

Subject to $\forall i \in \text{shifts}$

$$\text{assignment}[i] - (2 * \text{max_requirement} * \text{Shift_Used}[i]) \leq 0$$

where,

$$\text{max_requirement} = \forall t \in \{1..T\} \text{MAX}(\text{requirements}[t])$$

The idea is that if there is at least one assignment in a shift i , i.e. $\text{assignment}[i]$ is positive, then $\text{Shift_Used}[i]$ has to be 1 so that the left hand side (L.H.S) of the equation is negative.

3.4 Number of Shifts

Next, we need to consider the weightage for the number of shifts in the objective function. To calculate the number of shifts, we use the ShiftUsed variable computed above. But we also need to take into account the fact that while counting the number of assigned shifts, shifts on the same day(or different days) with same starting and ending times but with different break times/durations are to be considered as one shift.

To ensure this, we first figure out the number of actual different possible shifts in the complete cycle (say a week or a month), considering shifts on the same day (or different days) with same starting and ending times but with different break times/durations are to be considered as one shift. Let the set of such shifts be $\{\text{Actual shifts}\}$.

So, we get a mapping {All possible shifts} → {Actual shifts} where the set {All possible shifts} consists of all the possible shifts i.e. with zeroes in between corresponding to allowed breaks in each shift. For example, let's say that we have shifts $s_1, s_2 \dots s_n \in \{\text{All possible shifts}\}$ which map to a shift $s \in \{\text{Actual shifts}\}$. Now, for each $s \in \{\text{Actual shifts}\}$, we add constraints of the form:

$$assignment[s_1] + assignment[s_2] + \dots + assignment[s_n] - [(n+1) * max_requirement * Actual_Shift_Used[s]] \leq 0$$

where,

$Actual_Shift_Used[s]$ is a binary variable for each $s \in \{\text{Actual shifts}\}$ which is 1 if any of the $assignment[s_i]$ ($i \in \{1..n\}$) is non zero or 0 otherwise;

The idea is that if any of the $assignment[s_i]$ is non zero, then $Actual_Shift_Used[s]$ will have to be 1 so that the left hand side (L.H.S) of the equation is negative.

So, we add $weight_number_of_shifts * \sum_{i \in Actual_Shifts} Actual_Shift_Used[i]$ in the objective function to take the number of shifts into account.

3.5 Number of Shifts above a certain threshold

To consider the weightage for the number of shifts above a certain threshold, we add the following constraint:

$$\left(\sum_{i \in Actual_Shifts} Actual_Shift_Used[i] \right) + Less_Shifts + More_Shifts = Desired_Threshold$$

$$Less_Shifts \geq 0$$

$$More_Shifts \geq 0$$

Then we add $weight_more_shifts * More_Shifts$ in the objective function to take the number of shifts above a certain threshold into account.

3.6 Maximum number of allowed workers at a particular time

In real life situations, the hardware equipment available in an organization is often limited and thus it is necessary that the assignment plan does not exceed this limit. For example, we cannot assign a workforce of 30 software developers in a software company where only 25 computers are available. To ensure this, we add another constraint:

Subject to $\forall i \in shifts$

$$assignment[i] \leq Max_Possible_Employees$$

This would ensure that the assignment plan is implementable.

4. Approach using the implicit modeling approach of Bechtold and Jacobs (1990)

A detailed description of this approach can be found in [3].

5. Approach using hybrid heuristic composed of local search/tabu search heuristic combined with a fast min-cost-max-flow algorithm

A detailed description of this approach can be found in [1].

6. Results

Once the AMPL model was ready, we used the CPLEX and some other solvers to solve it. The following table gives the size of the problem and the times taken to solve it by CPLEX and other solvers.

Days	Objective Function	Granularity	No of shift types	Dantzig's approach (1954)	Bchtold and Jacobs approach (1990)
1 day	1 Break	1 hour	4	Time: 0.4 s Overcover: 7 Undercover:25 #Shifts: 2	Time: 0.4 s Overcover: 7 Undercover:25 #Shifts: 2
2 days	1 Break	1 hour	4	Time: 1.1 s Overcover: 16 Undercover:12 #Shifts: 4	Time: 1.1 s Overcover: 16 Undercover:12 #Shifts: 4
3 days	1 Break	1 hour	4	Time: 2 s Overcover: 24 Undercover:18 #Shifts: 4	Time: 2.8 s Overcover: 24 Undercover:18 #Shifts: 4
7 days	1 Break	1 hour	4	Time: 10 s Overcover: 28 Undercover:56 #Shifts: 5	Time: 27 s Overcover: 28 Undercover:56 #Shifts: 5
7 days	2 Breaks	1 hour	4	!	27 s
1 day	1 Break	30 mins	4	Time: 21 s Overcover: 16 Undercover:12 #Shifts: 4	Time: 17 s Overcover: 16 Undercover:12 #Shifts: 4
2 days	1 Break	30 mins	4	Time: 45 s Overcover: 32 Undercover:24 #Shifts: 4	Time: 50 s Overcover: 32 Undercover:24 #Shifts: 4
3 days	1 Break	30 mins	4	Time: 220 s Overcover: 48 Undercover:36 #Shifts: 4	Time: 283 s Overcover: 48 Undercover:36 #Shifts: 4

7 days	1 Break	30 mins	4		Time: >55 mins Overcover: 140 Undercover:98 #Shifts: 6
1 day	2 Breaks	30 mins	4	!	
2 days	2 Breaks	30 mins	4	!	
1 day	1 Break	15 mins	4	Time > 20 mins Overcover: 245 Undercover:571 #Shifts: 9	Time > 40 mins Overcover: 238 Undercover:597 #Shifts: 8

We compare our CPLEX experiments running times to the best algorithms described in [1].

Experiments have been run on different machines. The local search solvers are implemented in C++ using the EASYLOCAL++ framework (Di Gaspero and Schaerf, 2003b) and they were compiled using the GNU g++ compiler version 3.2.2 on a 1.5 GHz AMD Athlon PC running Linux Kernel 2.4.21. The greedy *min-COST max-FLOW* algorithm, instead, was coded in MS Visual Basic and run on a MS Windows NT 4.0 computer. CPLEX experiments have been run on 1.6 Ghz centrino machine with Windows XP platform.

The running times have been normalized according to the DIMACS net flow benchmark (<ftp://dimacs.rutgers.edu/pub/netflow/benchmarks/c>) to the times of the PC on which CPLEX experiments were conducted (calibration timings on that machine for above benchmark: t1.wm: user 0.030 sec t2.wm: user 0.180 sec). Because of the normalization the reported running times should be taken as indicatory only.

Set 1 of instances are available in self-describing text files from <http://www.dbai.tuwien.ac.at/proj/Rota/benchmarks.html>. A detailed description of the random instance generator used to construct them can be found in (Musliu et al., 2004).

We also used the MINOS solver instead of CPLEX from AMPL. However, results were suboptimal despite MINOS claiming them to be optimal.

References

- [1] Luca Di Gaspero, Johannes Gartner, Guy Kortsarz, Nysret Musliu, Andrea Schaerf, Wolfgang Slany. The minimum shift design problem: Theory and practice in 11th Annual European Symposium on Algorithms, Budapest, 2003.
- [2] Johannes Gartner, Nysret Musliu, Wolfgang Slany. A Heuristic Based System for Generation of Shifts with Breaks.
- [3] Stephen E. Bechtold and Larry W. Jacobs. Implicit modeling of flexible break assignments in optimal shift scheduling, 1990.
- [4] J. Herbers. (2005) Models and Algorithms for Ground Staff Scheduling on Airports, Doctoral Dissertation, March 2005.