

# A Mixed-integer Linear Program to create the shifts in a supermarket

Nicolò Gusmeroli\*      Andrea Bettinelli\*

March 28, 2024

## Abstract

The shift design and the personnel scheduling problem is known to be a difficult problem. It is a real-world problem which has lots of applications in the organization of companies. Solutions are usually found by dividing the problem in two steps: first the shifts are created, then the employees are assigned to them by respecting a bunch of constraints. The assignment of different tasks increases the complexity, since we have to consider the skills of the single employee necessary to perform any activity. In this paper we present a mixed-integer linear programming formulation which models together the shift creation and the construction of rosters for employees, with the objective of minimizing the amount of uncovered demand. Finally we provide the results for three real-world instances, confirming that this approach is promising.

## 1 Introduction

Building employee rosters while respecting legal and organizational constraints to satisfy personnel requirements is an NP-complete problem, see [1]. In the past, companies mainly adopted two techniques to simplify the problem: one was to fix the shift hours beforehand and then the employees were assigned in these shifts, in the other they repeated the schedules in a cyclic way over the weeks by shifting the employees, so the planning was done only once. This second approach left the possibility that some worker was not available, which was solved manually at any occurrence. Nowadays the situation has changed since the employees want to have more freedom in the assignment of the shifts, hence the companies started to let people decide the rest day in the week and to give preferences on the working hours. This flexibility made the mathematical modelling of this problem even more difficult and, most likely, less studied.

The problem studied consists in designing individual schedules for a group of heterogeneous workers, which amounts to fixing days-off, creating shifts, and

---

\*OPTIT srl, Bologna, Italy. Emails: {nicolo.gusmeroli, andrea.bettinelli@optit.net}

assigning available fixed tasks within these shifts, by respecting both the governor and the company rules, and it is called the Shift-Design Personnel Task Scheduling Problem (SDPTSP). Due to the difficulty of the problem the related literature work is scarce: to the best of our knowledge only few approaches have been proposed and no paper ever presented an explicit mathematical formulation. We aim to mind this gap by formulating a mixed-integer linear programming formulation (MILP) for the SDPTSP. The motivation for this work comes from a real-world application, namely the creation and the assignment of the shifts to the employees of a large Italian retail company, hence we consider additional constraints for our specific use case. The aim of this collaboration is threefold: automatize the rostering process, reduce the work of the managers, and improve the current schedules. Together with our customer we developed a model which was solved with standard algorithms, i.e., not specific for this problem, so there is big room for improvements. We were able to achieve the current demand satisfaction in few minutes, while, today, managers need some hours. Moreover, by setting a time limit of one hour, we obtained solutions that outperformed by far the ones of our customer, both in the missing demand and in the constraints violation (currently managers can violate some of them to have better schedules).

The paper is organized as follows. In the rest of this section we mention the related work. Section 2 presents the mixed-integer linear programming formulation for the SDPTSP. In Section 3 we show some experimental results. Finally, Section 4 concludes the paper and introduces the future research directions.

## 1.1 Literature review

The literature work related to personnel, staff, roster, or crew scheduling, is huge: there are thousands of papers studying several different variants of this problem. Due to the vast literature, we focus only on the few works with big importance for our study case, by mentioning the related problems and suggesting some references.

The problem of creating shifts is called Minimum Shift Design problem (MSD) and various solution methods have been proposed, for more details see [2]. The assignment of employees to fixed shifts, called personnel scheduling, has been introduced in the 1950s by Dantzig [3]. Solution methods have been classified into several categories, we refer to Alfares [4] for a comprehensive survey. The problems concerned with assigning a set of tasks with fixed times to a heterogeneous workforce having predetermined working times fall under the name of Personnel Task Scheduling problems (PTSP), see [5]. Considering also the design of the shifts makes the situation harder and defines the Shift-Design Personnel Task Scheduling Problem (SDPTSP). As far as we know, only two papers dealt with this problem, they used as objective function an Equity criterion, for which they proposed two different approaches: one based on constraint programming [6] and a two-step metaheuristic [7].

The mentioned problems have lots of real-world applications, Ernst et al. [8] presents some of the possible applications areas. For a very broad and detailed

survey on personnel scheduling and rostering problems we refer to [9], another interesting review combining managerial insights and technical knowledge is [10].

## 2 Definition of the MILP for the SDPTSP

We consider the Shift-Design Personnel Scheduling Task Problem: this means that given the two sets of the employees, with the individual skills and the time availability, and of the demands, with the associated task and times, we want to design the shifts, by fixing the times and the tasks, and to assign them to the different employees. The objective is to cover as much of the demand as possible while respecting legal obligations (max daily hours, max daily span, ...) and company constraints (min time on same activity, min work before break, ...).

We denote by  $R$ ,  $A$ ,  $D$ , and  $T$  the sets of employees, activities, days, and time slots, respectively.

The set of days is defined as  $D = \{1, \dots, |D|\}$ , where  $|D|$  is the planning horizon. The set of time slots, denoted by  $T$ , contains numbers representing the starting minute. For example, if the interval length is 30 minutes, so the time step is  $ts = 30$ , we represent the time slot 8-8:30 by the number  $16 = (8 \cdot 60)/ts$ , the time slot 8:30-9 by  $17 = (8 \cdot 60 + 30)/ts$ , ... , hence the set of time slots is  $T = \{16, 17, \dots\}$ .

To simplify the formulation, we denote by  $D_k$  the set of the first  $k$  days, i.e.,  $D_k = \{1, \dots, k\}$ , and its complementary, so the set of all days but the first  $k$ , by  $D_{\bar{k}} = D \setminus D_k$ . In the same way  $D^k$  is the set of the last  $k$  days, and the set of all the days but the last  $k$  is denoted by  $D^{\bar{k}}$ . Similarly we define  $T_k$  (resp.  $T^k$ ) as the set of the first (resp. last)  $k$  time slots and  $T_{\bar{k}}$  (resp.  $T^{\bar{k}}$ ) as the set of time slots but the first (resp. last)  $k$ .

Finally, we denote by  $tsD$  the number of time slots in a day, so  $tsD = 1440/ts$ .

### 2.1 Variables

The main variable that we use is the assignment of employee  $r$  to activity  $a$  at time slot  $t$  of day  $d$ , so  $x(r, a, t, d) = 1$  if  $r$  does  $a$  at  $t$  of  $d$  and 0 otherwise. Another important information is the *change of work status* of employee  $r$  at time slot  $t$  of day  $d$  on activity  $a$ , i.e.,  $y(r, a, t, d) = 1$  (resp. -1) if  $r$  starts (stops)  $a$  at  $t$  of  $d$ , while  $y(r, a, t, d) = 0$  means that  $r$  on day  $d$  either works both at  $t$  and  $t-1$  on  $a$  or that he does not perform  $a$  in any of these two intervals. We also need a variable indicating whether employee  $r$  works on day  $d$ , we have that  $z(r, d) = 1$  if  $r$  works on at least one time slot of  $d$  and  $z(r, d) = 0$  otherwise. Moreover for each employee  $r$  and for each day  $d$  we define the begin (resp. end) of work, which is represented by the first (resp. last) time slot with some activity assigned, which is defined  $b(r, d)$  (resp.  $e(r, d)$ ).

The demands are defined by exploiting the activity  $a$ , the day  $d$ , and the start and end time, namely  $t_1$  and  $t_2$  (such that  $t_1 < t_2$ ). It follows that

$dem(a, d, t_1, t_2) = k$  means that on day  $d$  between time slots  $t_1$  and  $t_2$  employees should work  $k$  minutes on activity  $a$ . In order to keep the problem feasible, we add a nonnegative slack on each demand, so  $\alpha(a, d, t_1, t_2)$  are the minutes of demand  $dem(a, d, t_1, t_2)$  which is not satisfied.

## 2.2 Constraints

In this section we present all the constraints of the model, by exploiting their mathematical formulation. They can be divided into three different groups: problem defining, i.e., giving the relationships between variables, legal, i.e., fixed by the law, and company, i.e., defined by our customer to have better schedules.

In order to guarantee feasibility the satisfaction of the demand is modeled as a soft constraint. The compatibility between employees and activities depends on the individual skills of each worker and are defined by the compatibility matrix, which is created in a pre-processing step by matching the skills of the employees and the requirement for each activity. in this way we avoid to consider the skills in the model.

All the variables are derived from the assignment variable  $x(r, a, t, d)$ , the constraints defining these relationships are

(D1) definition of  $y$

$$\begin{aligned} y(r, a, t, d) &= x(r, a, t, d) & \forall r \in R, a \in A, t \in T_1, d \in D \\ y(r, a, t, d) &= x(r, a, t, d) - x(r, a, t-1, d) & \forall r \in R, a \in A, t \in T_{\bar{1}}, d \in D \end{aligned}$$

(D2) definition of  $z$

$$\begin{aligned} \sum_{a \in A} \sum_{t \in T} x(r, a, t, d) &\leq z(r, d) \cdot M & \forall r \in R, d \in D \\ \sum_{a \in A} \sum_{t \in T} x(r, a, t, d) &\geq z(r, d) & \forall r \in R, d \in D \end{aligned}$$

where  $M$  is a sufficiently big number, e.g.,  $M = |A| \cdot |T|$ ,

(D3) definition of  $b$  and  $e$

$$\begin{aligned} b(r, d) &\leq t + tsD \cdot \left(1 - \sum_{a \in A} x(r, a, t, d)\right) & \forall r \in R, t \in T, d \in D \\ e(r, d) &\geq (t + 1) \cdot \sum_{a \in A} x(r, a, t, d) & \forall r \in R, t \in T, d \in D \end{aligned}$$

(D4) each employee can perform simultaneously at most one activity

$$\sum_{a \in A} x(r, a, t, d) \leq 1 \quad \forall r \in R, t \in T, d \in D$$

(D5) correctness of start and end times, so

$$b(r, d) \leq e(r, d) \quad \forall r \in R, d \in D$$

The legal constraints depend on the work laws of the country, for us they are

(L1) maximum daily hours ( $k_{L1}$  time slots)

$$\sum_{a \in A} \sum_{t \in T} x(r, a, t, d) \leq k_{L1} \quad \forall r \in R, d \in D$$

(L2) maximum working hours in the planning horizon ( $k_{L2}$  time slots)

$$\sum_{a \in A} \sum_{t \in T} \sum_{d \in D} x(r, a, t, d) \leq k_{L2} \quad \forall r \in R$$

(L3) maximum  $k_{L3}$  consecutive working days

$$\sum_{d' \in [d, d+k_{L3}]} z(r, d') \leq k_{L3} \quad \forall r \in R, d \in \overline{D^{k_{L3}}}$$

(L4) maximum consecutive working time, namely every  $k_{L4} + j_{L4}$  time slots each employee must be off for at least  $j_{L4}$  slots, which is formulated

$$\sum_{t' \in [t, t+k_{L4}+j_{L4})} \sum_{a \in A} x(r, a, t', d) \leq k_{L4} \quad \forall r \in R, t \in \overline{T^{k_{L4}+j_{L4}-1}}$$

(L5) maximum daily span ( $k_{L5}$  time slots)

$$e(r, d) - b(r, d) \leq k_{L5} \quad \forall r \in R, d \in D$$

(L6) minimum rest between two working days ( $k_{L6}$  time slots)

$$tsD + b(r, d) - e(r, d-1) \geq k_{L6} \quad \forall r \in R, d \in D_{\overline{T}}$$

During several meetings with our customer we defined and implemented their internal constraints, i.e., the company constraints, which are

(G1) minimum working time after a break ( $k_{G1}$  time slots)

$$\sum_{t' \in [t, t+k_{G1})} \sum_{a \in A} x(r, a, t', d) \geq \sum_{a \in A} y(r, a, t, d) \cdot k_{G1} \quad \forall r \in R, t \in \overline{T^{k_{G1}-1}}, d \in D$$

(G2) demand satisfaction, formulated as

$$\sum_{t \in [t_1, t_2)} \sum_{r \in R} x(r, a, t, d) \cdot ts + \alpha(d, a, t_1, t_2) \geq dem(d, a, t_1, t_2) \quad \forall a \in A, d \in D, t_1, t_2 \in T$$

(G3) minimum consecutive working time on the same activity ( $k_{G3,a}$  time slots)

$$\sum_{t' \in [t, t+k_{G3,a})} x(r, a, t', d) \geq y(r, a, t, d) \cdot k_{G3,a} \quad \forall r \in R, a \in A, t \in \overline{T^{k_{G3,a}-1}}, d \in D$$

(G4) compatibility between employee and activity

$$x(r, a, t, d) \leq cRA(r, a) \quad \forall r \in R, a \in A, t \in T, d \in D$$

where  $cRA$  is the compatibility matrix matching employees and activities such that  $cRA(r, a) = 1$  if  $r$  can do  $a$  and 0 otherwise

(G5) compatibility between employee and time slot

$$x(r, a, t, d) \leq cRTD(r, t, d) \quad \forall r \in R, a \in A, t \in T, d \in D$$

where  $cRTD$  is the availability matrix such that  $cRTD(r, t, d) = 1$  if  $r$  is available to work at  $t$  of  $d$  and 0 otherwise

(G6) the checkout management comprises two different activities, i.e., normal working *opCAS* and closure *clCAS*, and it has a specific daily rule: “each employee who works at the checkout has to do only one closure right after its last opening”: so, for each day, if an employee works on the checkout (do at least one slot *opCAS*), then he must do one slot *clCAS* right after its last *opCAS* slot; this is modelled by the following constraints

- an employee can do *clCAS* only if, the same day, he works also on *opCAS*

$$\sum_{t \in T} x(r, opCAS, t, d) \leq \sum_{t \in T} x(r, clCAS, t, d) \cdot M \quad \forall r \in R, d \in D$$

for  $M$  sufficiently big, e.g.,  $M = |T|$

- each employee does at most one *clCAS* per day

$$\sum_{t \in T} x(r, clCAS, t, d) \leq 1 \quad \forall r \in R, d \in D$$

- after doing *clCAS*, the employee cannot do *opCAS* until the next day

$$\sum_{t' \geq t} x(r, opCAS, t', d) \leq M(1 - x(r, clCAS, t, d)) \quad \forall r \in R, t \in T, d \in D$$

for  $M$  sufficiently big

- right before *clCAS* the employee must do a *opCAS* time slot

$$x(r, clCAS, t, d) \leq x(r, opCAS, t-1, d) \quad \forall r \in R, t \in T_1, d \in D$$

It is important to note that some constraints, e.g., (L6), have to consider the previous schedules; in the results of the next section we also add the historical information even though we did not report the exact formulation of these constraints throughout this section.

### 2.3 Objective function

In the optimization process several objective functions might be used, in general the aim is to minimize the costs (declined either as total working days or as uncovered demand).

The company we are working with asked us, as objective function, to cover the most demand possible in order to avoid the employees doing overwork. They also specified that the activities have different priorities, so we gave a penalty to the slack for each activity  $a$ , denoted  $p(a)$ . Moreover, their assignment of employees to tasks depends on the individual work experience, so an employee should be assigned to most suited activity even if he can perform several of them. In order to catch this, we added a multiplicative factor which depends on the matching between employee  $r$  and activity  $a$ , denoted  $c(r, a)$ . Thus, the objective function used in the model is

$$\min \sum_{a \in A} \sum_{d \in D} \sum_{t_1, t_2 \in T} p(a) \cdot \alpha(a, d, t_1, t_2) + \sum_{r \in R} \sum_{a \in A} \sum_{t \in T} \sum_{d \in D} x(r, a, t, d) \cdot c(r, a)$$

## 3 Instances and Computational Aspects

The main scope of this paper is to introduce the mixed-integer linear programming formulation for the Shift-Design Personnel Task Scheduling Problem, with the specific constraints for our case study. In order to validate this formulation and to provide relevance to this problem, we solve three benchmark instances corresponding to three different planning weeks of our customer with a granularity of 15 minutes (so  $|T| = 48$ , and  $|D| = 7$ ), and we compare the solution with the actual schedules. The instances have small-medium size, because  $|R| = 66$ , and  $|A| = 72$ .

The following results are obtained by using the commercial solver Xpress [11], with version 8.11 on a standard laptop Intel Core i7-8550 at 1.8 GHz with 16 GB RAM running Windows 10.

### 3.1 Resolution Steps

During the analysis phase and the study of the first results we were able to find some interesting considerations that led us to reduce the size of the problem. From a computational point of view, the most useful was the *identity* of some activities, i.e., there are some tasks which can be performed by the same set of employees, with nonoverlapping demand, and for which all the parameters are equal. Hence, we merged the identical tasks into macro-activities to reduce the size of the problem. By doing this as a pre-processing step we were able to simplify the problem and to obtain better results: we decreased the number of activities, on average, by 29%, and we improved the best solution by 37%; the detailed results can be seen in Table 1. After the optimization, the assignment to these macro-activities is redistributed on the original tasks by a simple backward assignment procedure.

As it is easy to imagine, the main difficulty of a commercial solver on these kind of problems is to give big improvements on the best solution: since there are many feasible solutions with similar values, the incumbent continuously decreases but very slowly. We solved this problem by implementing a base heuristic algorithm, it produced a feasible assignment in few seconds which was then passed to the solver as a warm-start. The heuristic we developed uses a greedy approach: it creates the shifts for all the employees satisfying the compatibilities of tasks and hours, then it checks for all the conflicts on the constraints and it fixes them by changing the tasks, in this phase the priority is given to activities with higher demand.

### 3.2 Experimental Results

In this section we present the results obtained on a set of three instances provided by the company we are working for, which correspond to three week of plannings of a medium-big size retail store.

Since the results we obtained were already rewarding, we agreed with our customer to set a time limit of one hour, which is far less the actual time needed (around 6 hours) by the store managers.

Instance	Merge	Heur	Best sol	Best bound	Gap
W20	N	N	701 384	23 073	97%
W20	Y	N	369 069	23 041	94%
W20	N	Y	87 691	23 073	74%
W20	Y	Y	95 517	23 041	76%
W21	N	N	556 049	4 780	99%
W21	Y	N	459 375	4 778	99%
W21	N	Y	103 632	4780	95%
W21	Y	Y	77 633	4 778	94%
W22	N	N	693 821	31 009	95%
W22	Y	N	297 904	30 976	89%
W22	N	Y	169 001	31 009	82%
W22	Y	Y	159 152	30 976	80%

Table 1: Computational results of Optit for the instances given by our customer with a time limit of one hour

In Table 1 we validated the computational approaches presented in Section 3.1 by exploiting the results for the different combinations of the preprocessing techniques explained. The second column indicates whether *identical* activities have been merged, while the third if the greedy heuristic has been used as a warm start. From these results it follows that using the heuristic decreases the best solution on average by 78%, while merging the activities gives an improvement of 37%. The total processing time is around 1 minute, so it is negligible compared to the time limit.

We claim that these results are far from being optimal, and it can be seen from the gaps in Table 1, this is mainly because our approach is not problem-dependent: we used a commercial solver on a general formulation without tuning the internal parameters and we did not implement specific heuristics. Anyway, these first results are very promising for us and rewarding for our customer.

Since these instances are provided by the company we are working with, there is no benchmark result to check the quality of the solution. The only comparison we can make is with the current plannings of our customer.

Instance	Case	$\sum \alpha$ [h]	Violations	Dept dem %
W20	C	374	579	86%
W20	O	59	0	96%
W21	C	366	544	86%
W21	O	62	0	95%
W22	C	356	744	86%
W22	O	110	0	95%

Table 2: Comparison of Optit results (case *O*) with actual planning (case *C*)

In Table 2 we compare our results (case *O*) with the real schedules of our customer (case *C*). In the third column we provide the total missing hours to cover the demand, i.e., the sum of the slacks without any multiplicative parameter, while the fourth indicates the number of violated constraints; since we use an exact model, we do not violate any constraint, but in the real life managers do it quite often. In the fifth column we give a percentage of saturation of the whole demand by considering the departments and not the single tasks, this is done because sometimes the managers assign the employees to only one task, but then they can freely shift activity within the same department if needed.

It is easy to see that the missing demand in our solution is much smaller than the actual planning. If we consider the single activities we have a huge decrease of around 70%. With respect to the associated departments, we can see that on average our demand satisfaction is 10% more, which still gives a big improvement. Another important indicator of the quality of our solutions are the constraints violated in the current weekly schedules (around 600), which for us are 0 since we use an exact model.

It follows from the results that, in comparison with the actual planning, we are outperforming the current schedules both with respect to the covered demand and to the violation of the constraints.

## 4 Conclusion

The Shift-Design Personnel Task Scheduling Problem has been studied only by few authors in the literature and, to the best of our knowledge, no exact methods were proposed to solve this problem. The main work of this paper has been to

give the first MILP formulation for this problem, defining the generic constraints given by the law. Our study case is a big size Italian retail company, so we also implemented some constraints for this specific case, to provide them with better results. In order to validate our model we studied three different weeks of planning on a store, and we provided the results, by using the commercial solver Xpress. Moreover, we explained how to decrease the magnitude of the problem, in order to tackle more difficult, i.e., bigger, instances. Finally, we compared our results with the actual planning. We were able to decrease the uncovered demand by 70%, which corresponds to a huge saving. At the same time our solutions respected all the constraints, while, as of today, the schedules of our customer violate around 600 constraints every week. Considering the department and not the single tasks, the improvement is lower, but still very significant since we cover 10% more demand. It is also important to say that as of today managers need six hours every week for the planning, while we set a time limit of 60 minutes. Hence we obtained better results in a less amount of time.

As it can be imagined the company we are working with is very satisfied with these first results, but there is still ongoing work in different directions. As a first step, we are trying to catch some other needs, in order to add more specific constraints and to provide even better results. Another direction of future work is the development of specific techniques to speed up the computational times and, hence, to solve bigger instances. We saw that, by passing a feasible schedule to the solver as a warm-start, the quality of the solution improved drastically in a short amount of time, even with our simple greedy heuristic: we were able to reach the same results in few minutes instead of an hour. Thus, implementing a smarter heuristic is one of the tasks with higher priority to further improve the current solutions.

## References

- [1] Smet P., Wauters T., Mihaylov M., Berghe G. V.: The shift minimisation personnel task scheduling problem: A new hybrid approach and computational insights. *Omega* 46 (2014), pp. 64-73.
- [2] Di Gaspero L., Gärtner J., Kortsarz G., Musliu N., Schaerf A., Slany W.: The minimum shift design problem. *Annals of operations research* 155 (2007), pp. 79-105.
- [3] Dantzig G. B.: A comment on Edie’s “Traffic delays at toll booths”. *Journal of the Operations Research Society of America*, 2 (1954), pp. 339-341.
- [4] Alfares H. K.: Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research* 127 (2004), pp. 145-175.
- [5] Krishnamoorthy M., Ernst A. T.: The personnel task scheduling problem. *Optimization methods and applications* (2001), pp. 343-368.

- [6] Lapègue T., Bellenguez-Morineau O., Prot D.: A constraint-based approach for the shift design personnel task scheduling problem with equity. *Computers & Operations Research* 40 (2013), pp. 2450-2465.
- [7] Prot D., Lapègue T., Bellenguez-Morineau O.: A two-phase method for the shift design and personnel task scheduling problem with equity objective. *International Journal of Production Research* 53 (2015), pp. 7286-7298.
- [8] Ernst A. T., Jiang H., Krishnamoorthy M., Sier D.: Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research* 153 (2004), pp. 3-27.
- [9] Van den Bergh J., Beliën J., De Bruecker P., Demeulemeester E., De Boeck L.: Personnel scheduling: A literature review. *European journal of operational research*, 226 (2013), pp. 367-385.
- [10] De Bruecker P., Van den Bergh J., Beliën J., Demeulemeester E.: Workforce planning incorporating skills: State of the art. *European Journal of Operational Research*, 243 (2015), pp. 1-16.
- [11] FICO Xpress Optimization Suite, v. 8.11.