Demand-driven Delivery Staff Rostering: Preliminary Results

Andrea Rendl and Christina Burt

Satalia, UK
{andrea,christina}@satalia.com

Abstract. We present an optimisation problem that arises when planning rosters for drivers in a delivery company. The rosters, or so-called *shift patterns* are rota systems that determine, for each weekday over a fixed set of weeks, when and how many hours a driver (or vehicle) will work, i.e. perform deliveries. The shift pattern has to meet legal requirements, such as not exceeding the maximal working hours per week. Furthermore, shift patterns must have a simple structure to be accepted by the employees.
Typically, shift patterns are designed assuming the same delivery demand for every weekday. However, in reality, the demand is not stable, and delivery companies often observe peaks and lows on particular days. For instance, a delivery company's demand may increase over the week, with low demand on Monday and high demand towards the weekend. Dealing with varying demand is a big challenge for delivery companies. High demand days require hiring contractors that increase the operational costs and the contractors often do not provide the same quality of service as the company's own staff. Low demand days leave the drivers underemployed, often with the same pay. We therefore propose the design of a shift pattern that takes into account different levels of demand on each weekday.
In this paper, we define the Demand-based Delivery Staff Rostering Problem (DD-SRP), and present a mathematical model for the proposed problem, and solve the problem formulated in MiniZinc with a MIP solver and a CP solver and present results on different demand level scenarios.

1 Introduction

This paper both presents and addresses the problem of creating a cyclic staff roster for delivery drivers such that the staff availability matches demand. The use of rotational staff rosters is common in industry, and yet this problem is not well addressed in the academic literature. The lack of effort on this particular problem domain may be due to the complexity of the problem—it builds yet more requirements into an already challenging problem. In this paper, we show that the problem is tractable for realistic sized instances we base on our experience working on industrial problems.
Supply chain operations are performed by delivery companies that supply customers with goods. The goods are typically delivered with vans that are operated by drivers who work in a fixed *shift pattern*.

A shift pattern is a fixed rota that determines, over a fixed set of weeks, when and how many hours a driver will work on each weekday. For instance, a shift pattern may span over two weeks, and state that on the first Monday, the working hours are from 8am to 5pm, and the second Monday is a day off. Since the shift pattern is regularly repeated, this means that drivers on this pattern have every second Monday off.
A shift pattern is typically associated with a subset or *group* of drivers to simplify the operations and make up for days off of other drivers. For instance, if group-1 is on a two-week shift pattern where every first Monday is a day off, then it is common to have another group with another shift pattern, where the first Monday is a working day (see Fig. 1 as an example).
Shift patterns are typically designed with constant demand in mind. In other words, the shift patterns assume that the work load on each weekday is always the same. The reality is often very different. Delivery companies need to deal with a lot of variation in demand in many sectors. The variations can be seasonal, such as an increase of demand before seasonal holidays like Christmas. However, for many delivery companies, the variations are more constant and can be observed on a regular basis. For instance, for home deliveries, it is not uncommon to observe low demand on Mondays, but an increase of demand towards the weekend, when customers are more likely to be at home.
The variation in demand poses a huge problem for delivery companies. On one hand, there are high demand days, where the company is lacking staff and resources (vans) to perform all requested deliveries. These days require hiring contractors that perform the additional deliveries which increases the operational costs and often reduces customer satisfaction, since the contractors often do not provide the same quality of service as the company's own staff. On the other hand, there are low demand days where the drivers are under-employed and work shorter hours, often for the same pay as a full working day. These days also put a burden on the delivery company. We therefore propose to design shift patterns that optimally match the regular variation in demand.
We introduce the Demand-based Shift Design Problem (DSDP) that is concerned with finding a shift pattern that maximally aligns with the expected variations in demand.

1.1 Related Work

Staff rostering is a well studied topic. However, shift pattern scheduling, or line of work scheduling, as a holistic problem is less well studied (c.f in the literature review paper, [5] describe the problem but present only two examples), and incorporating demand is also not well studied [1]. Typically, the literature addresses the problem of allocating staff to existing shift patterns, which is typically formulated as a set covering problem (see, for example [2]). [8] described an application in airport staff scheduling where they first simulated demand in order to obtain the demand parameters for their integer programming model. They first generate staff number to shift allocations using a set covering model, and then the cyclic roster was constructed by hand. [3] combined the shift

pattern scheduling problem with the problem that generates demand in order to create better schedules for staff in a hospital nurse application. They formulate the staff to shift pattern problem as a set covering problem, and use branch-and-price to generate new feasible shift patterns to be considered. [1] allocate staff based on demand, but do not consider shift patterns. [6] study an SMT approach for building train crew rosters. They study three different techniques but only manage to solve small instances with SMT. The train crew rostering problem is similar to the problem we study in that it also involves finding rosters for drivers. However, the roster has to match the requirements of the train lines, and not a specific varying demand, as in the problem considered in this paper. To the best of our knowledge, solving the complete shift pattern problem as described in this paper has not been addressed.

2 Demand-based Delivery Staff Rostering

In this section we define the Demand-based Delivery Staff Rostering Problem (DD-SRP) in the context of a delivery company that has a fleet of vans and a set of drivers that deliver goods to customers. The aim of the DD-SRP is to find a shift pattern for the drivers that matches the varying estimated demand per weekday. More specifically, the shift pattern determines, for a fixed number of weeks, when and how many hours the drivers (who are assigned to the shift pattern) work.
Typically, there are several shift patterns, and an equally-sized set of staff are assigned to each shift pattern. Fig. 1 shows an example of two shift patterns (one in red and the other in blue) that span over two weeks. For instance, we see that for shift pattern 1, the first week has six working days (Mon-Sat), while shift pattern 2 has only four working days (Tue-Fri). The shift patterns are *s*ymmetric which means that the first week in shift pattern 1 is the same as the second week in shift pattern 2. This is an important feature that renders the overall patterns as simple and more acceptable to the staff. This also means that the full shift pattern is already given by the first week, since the following weeks are given by shifting the patterns from the first week.
The shift patterns are applied in a cyclic fashion, which means that they are repeated over and over again. This means that in the example above, the drivers/vehicles assigned to the first shift pattern will always have Monday off on every even week.
The shift patterns have to meet legal working requirements, such as maximal working hours, mandatory resting days after each shift, or lunch breaks.

Definition 1 (Demand-based Delivery Staff Rostering Problem). *Determine a shift pattern for delivery drivers that matches demand such that:*

- *shift patterns are cyclic, no matter how many are created;*
- *legal working requirements are met (e.g. maximum work hours).*

We define in detail all constraints in the next section, when we formulate the mathematical problem model.

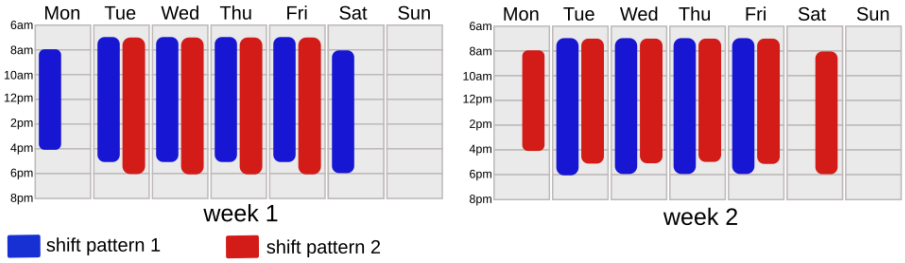

Fig. 1. Two shift patterns (in red and blue) that span over two weeks. For instance, in shift pattern 1, the Monday shift in week 1 starts at 8am and ends at 4pm. Note that the shift patterns are symmetric: for shift pattern 2, the Monday shift in week 2 is the same as the Monday shift in shift pattern 1 on week 1.

2.1 Mathematical Formulation

We have a fleet of v vans V that represent the workforce of drivers. In fact, in our mathematical model we build a shift pattern for the vans rather than the staff, assuming that a driver is assigned to each van. This is common practice in the delivery sector and makes it easier for the staff scheduler to account for holidays or sick leave. For instance, schedulers often assign an additional staff member to each shift to accommodate for sudden sick leave or holidays.

The set of week days $D = \{1, \ldots, 7\}$ represents the seven week days Monday to Sunday. For each weekday, we know the expected customer demand in number of orders, which we denote, for each weekday, $O = \{O_{Mon}, \ldots, O_{Sun}\}$, where $O_i \in \mathbb{N}$. Furthermore, we denote $o \in \mathbb{R}^+$ as the number of orders that a van can deliver on average per hour. This parameter will be used to map the demand per orders to the number of vehicles necessary to match the demand.

We represent time in time units of $1/\tau$ hours, where τ is our time factor that determines the granularity of time. This means that for $\tau = 2$, we discretize time into half hours $\{1, \ldots, 48\}$, and the time unit 12 represents the time 6am (12 half hours). The set of available time units is $T = \{1, \ldots, 24 * \tau\}$.

Using the time units, we describe various features that arise in delivery. We have the lunch break duration t_{lunch}, the stem time t_{stem} which corresponds to the average time between the depot and the first/last delivery, the maximal working hours per day $\hat{t}_{day}$, and the maximal working hours per week $\hat{t}_{week}$, and finally, the paid working hours t_{paid} which are the number of hours that drivers are paid to work every week, on average.

The binary parameter vv_v^s states if vehicle v is assigned to shift pattern s. Since all vans (or drivers) are considered equal, we simply assign the first f drivers to shift pattern 1, the next f drivers to shift pattern 2, etc, where $f = |V|/|S|$ is the number of vans (drivers) divided by the number of shift patterns. All parameters are summarized in Table 1.

Parameter	description
$D = \{1, \ldots, 7\}$	the seven days in a week
$O = \{O_{Mon}, \ldots, O_{Sun}\}$	estimated average demand per day in orders
$o \in \mathbb{R}^+$	orders delivered per hour per van
S	the set of shift patterns with $\|S\|$ patterns
$\hat{t}_{day} \in T$	maximal working hours per day
$t_{lunch} \in T$	duration of the lunch break
$t_{paid} \in T$	paid working hours per week
$t_{stem} \in T$	average stem time
$\hat{t}_{week} \in T$	maximal working hours per week
τ	time factor for discretizing time, we use $\tau = 2$
$T = \{1, \ldots, 24 * \tau\}$	time units representing 24 hours in a day
$V = \{v_1, \ldots, v_v\}$	vans (equivalent to *drivers*)
$vv_v^s \in \{0, 1\}$	states if vehicle v is assigned to shift pattern s

Table 1. Problem parameters in alphabetical order

Decision Variables The integer variables s_d^s represent the start time of shift $s \in S$ on weekday $d \in D$, and similarly, variables e_d^s represent the shift end. The integer variables l_d^s represent the shift length as the number of working hours on weekday d in shift s. Binary variables w_d^s are 1 if weekday d in shift pattern s is a working day, and 0 otherwise. The integer variables vh_d represent the number of hours all vans in the fleet are working on weekday d, over all shift patterns. The float variables a_d represent the average number of serviced orders over all shift patterns for weekday d. Finally, the float variables u_d represent the unmet demand per weekday d. This includes both cases, where the demand is either too high or too low and therefore cannot be met by the van-hour capacity of the fleet. All decision variables are summarized in Table 2.

Decision variables	description
$a_d \in \mathbb{R}^+$	the average orders delivered on day d
$e_d^s \in T$	end of shift in shift pattern s on day d
$l_d^s \in T$	length of shift in shift pattern s on day d
$s_d^s \in T$	start of shift in shift pattern s on day d
$u_d \in \mathbb{R}^+$	the unmet demand on day d
$vh_d \in \{1..T_{Max}\}$	the number of hours all vans are working on weekday d
$w_d^s \in \{0, 1\}$	is 1 if day d is a working day in shift pattern s

Table 2. Decision variables in alphabetical order

Constraints In our formulation we have the following hard constraints. The first four constraints cover the shift constraints. Constraint (1) states that shift s on weekday d may not start before the earliest start time. Similarly, Constraint (2) states that shift s on weekday d may not end

later than the latest end time. Constraint (3) requires that the start of shift s is smaller or equal to the end time of the shift. Note that if day d in shift s is a day off, then s_d^s is equal to e_d^s. Constraint (4) restricts the length of shift pattern s.

$$s_d^s \geq earliestStart_d, \forall s \in S, d \in D \tag{1}$$

$$e_d^s \leq latestEnd_d, \forall s \in S, d \in D \tag{2}$$

$$e_d^s \geq s_d^s, \forall s \in S, d \in D \tag{3}$$

$$l_d^s = e_d^s - s_d^s, \forall s \in S, d \in D \tag{4}$$

$$l_d^s \leq M * w_d^s, \forall s \in S, d \in D, M > \hat{t}_{day} \tag{5}$$

$$\sum_{s \in S, d \in D} l_d^s - \sum_{s \in S, d \in D} w_d^s * t_{lunch} = t_{paid} * |S| \tag{6}$$

$$\sum_{d \in D} l_d^s - \sum_{d \in D} w_d^s * t_{lunch} \leq \hat{t}_{week}, \forall s \in S \tag{7}$$

$$(w_{Sat}^s + w_{Sun}^s = 0) + (w_{Sun}^s + w_{Mon}^{s+1} = 0) + (w_{Mon}^{s+1} + w_{Tue}^{s+1} = 0) = 1, \forall s \in S - 1 \tag{8}$$

$$(w_{Sat}^{|S|} + w_{Sun}^{|S|} = 0) + (w_{Sun}^{|S|} + w_{Mon}^1 = 0) + (w_{Mon}^1 + w_{Tue}^1 = 0) = 1 \tag{9}$$

$$vh_d = \sum_{s \in S, v \in V} vv_v^s * l_d^s - \sum_{s \in S} w_d^s * \sum_{v \in V} vv_v^s * t_{lunch}, \forall d \in D \tag{10}$$

$$a_d = o * (vh_d - 2 * t_{stem} * \sum_{s \in S} \{w_d^s * \sum_{v \in V} vv_v^s\}), \forall d \in D \tag{11}$$

$$u_d = |O_d - a_d|, \forall d \in D \tag{12}$$

Constraint (5) uses the big-M formulation to set w_d^s to 1 whenever the shift length is larger than zero, for shift pattern s on day d, and to 0 otherwise (when it is not a working day). Constraint (6) states that the average number of working hours over all shift patterns must be equal to the number of hours that are paid, minus the lunch break if the day is an actual working day. Note that this constraints considers only the *average* number of working hours since it is allowed that shift patterns differ in working hours for each week. In other words, a shift pattern that has 40 paid working hours may have one week with 35 working hours and another week with 45 working hours. Constraint (7) assures that for each shift pattern, the maximal number of working hours is not exceeded. Constraint (8) makes sure that there is a two day break between each shift (except the last shift). Similarly, Constraint (9) states the same constraint for the very last shift, linking it with the first week shift pattern. The van hours, which represent the number of hours worked by all vans, are determined by Constraint (10): the van hours consist of the sum of working hours (shift length l_d^s) over all vans v and shift patterns s, and subtracting the lunch breaks for all vans that were working on the respective shift.

Finally, we determine the average number of serviced orders a_d with Constraint (11), where we multiply the overall van hours vh_d with the

average number of serviced orders per hour, a_d, after subtracting the time during which no orders are performed: the average stem time to the first order and from the last order, for all vans that worked on the respective day d. Furthermore, in Constraint (12), the unmet demand u_d is set to the absolute value of the expected demand D_d for day d subtracted by the average serviced orders a_d through the shift pattern.

Objective We consider two objective variants. The first considers minimizing the maximal unmet demand. The second objective minimizes the sum over all unmet demands, and allows to use *weights* for each day, in case delivery companies prefer matching the demand on some weekdays more than others.

Minimising maximal unmet demand For the maximal unmet demand we introduce the float variable p which represents the maximal unmet demand.

$$p \geq 0.0 \tag{13}$$
$$p \leq \max(\{O_{Mon}, \ldots, O_{Sun}\}) \tag{14}$$
$$p > u_d, \forall d \in D \tag{15}$$
$$\min p \tag{16}$$

Constraints (13) and Constraints (14) set the lower and upper bound of the maximal unmet demand variable p. Constraint (15) restricts p to be greater than the unmet demand u_d of every day d. Finally, the objective in (16) is to minimise the maximal unmet demand p.

Minimising weighted unmet demand The second objective is to meet the estimated demand, in other words, to minimise the sum of unmet demand u_d of all days d:

$$\min \sum_{d \in D} c_d * u_d \tag{17}$$

where each day is weighted by the weights $c_d \in [0.0..1.0]$. This allows the scheduler to choose which days have highest priority in meeting demand.

3 Preliminary experimental results

In this section we present preliminary experimental results to evaluate our problem model. We formulate our mathematical model in MiniZinc [9] and provide the model and input data online under the MIT license [1]. For solving, we use two solvers as backend solvers in MiniZinc: first, the CP solver Gecode [7] and the MIP solver COIN-OR cbc [4]. We choose these solvers since they are both open-source solvers that can deal with both integer and floating point variables.

[1] available at: https://github.com/angee/demand-shift-pattern

3.1 Problem instances

We assess our models on several different dimensions which we summarize in Table 3. First, we create instances of different sizes, covering different number of vans and different number of shift patterns. Second, we assess the models on different estimated demand scenarios. For simplicity, we only use the first, unweighted version of our objective. Each instance is named after the features discussed above, for instance, **v12_s4_peak-thu-fri.dzn** represents an instance with 12 vans, 4 shift patterns, the demand scenario that peaks on Thursdays and Fridays.

Feature	Range	Description
Vans/Drivers	12, 24, 60	the number of vans/drivers
Shift Patterns	2, 4, 6	the number of shift patterns
Demand Scenarios	linear peak-thu-fri	linearly increasing demand with low Sat demand peak on Thu-Fri

Table 3. Features of the problem instances

We set the problem constants as follows: the time factor $\tau = 2$, the lunch break is set to one hour, the maximal working hours per day are 12 hours, the maximal number of weekly working hours is 48 hours, the stem time of 30 minutes, and the average number of orders delivered per hour to 1.2 orders per hour. Note that these are very conservative settings that however reflect the standard in the industry.

3.2 Experiments

All experiments were conducted on the same machine, with 4 cores on a 2.30GHz Intel Core i5-6200U, and Ubuntu 16.04. We use the MiniZinc version 2.1.7 and the bundled versions of Gecode and cbc within it. The timeout was set to 300 seconds.

CP Search Strategy We have tested several search heuristics for the CP solver. However, none of the search strategies was able to outperform the default search setting for Gecode/fzn-gecode. The default search strategy of Gecode 6.0.1 selects the variable with largest accumulated failure count divided by domain size with decay factor d = 0.99, and a min-domain value selection. At the point of publication, we were not able to specify and test this exact type of search directly in MiniZinc. We therefore omit the search strategy in our MiniZinc model, and this way use the default-strategy of fzn-gecode in our experimental setup.

Results Our results are summarized in Table 4. Each row represents a problem instance, and the columns show the runtime and the solution quality. All objective results that are labelled with an asterisk * have been proven to be optimal by the solver. We see that the MIP solver,

Instance			Runtime (sec)		Objective	
vans	shift p.	demand	Gecode	cbc	Gecode	cbc
v12	s6	linear	300.000	300.000	—	3.81728
v24	s2	linear	0.027	0.353	*27.0	*27.03456
v24	s4	linear	300.000	10.006	—	*27.03456
v60	s6	linear	300.000	241.377	—	*17.0864
v60	s2	linear	0.026	0.328	*68.0	*68.0864
v60	s4	linear	300.000	11.444	—	*68.0864
v24	s6	linear	300.000	300.000	—	18.23456
v12	s2	linear	300.000	0.317	38.2	*13.01728
v12	s4	linear	300.000	17.668	—	*13.01728
v60	s4	peak-thu-fri	300.000	2.404	—	*42.0864
v12	s2	peak-thu-fri	0.013	0.292	*17.0	*17.01728
v60	s2	peak-thu-fri	0.014	0.327	*84.0	*84.0864
v12	s4	peak-thu-fri	300.000	2.161	—	8.01728
v24	s2	peak-thu-fri	0.047	0.284	*33.0	*33.03456
v60	s6	peak-thu-fri	300.000	0.728	—	*12.0864
v12	s6	peak-thu-fri	300.000	1.16	—	*2.41728
v24	s4	peak-thu-fri	300.000	5.765	—	*17.03456
v24	s6	peak-thu-fri	300.000	0.962	—	*5.43456

Table 4. Preliminary results with a timeout of 300 seconds. Solutions for which the solver proved optimality are marked with an asterisk *.

COIN-OR cbc, can solve most problem instances to optimality using very little time. The CP solver, Gecode, can also solve some instances to optimality, and we notice that when it proves optimality, it outperforms the MIP solver in terms of runtime. However, the CP solver also struggles with several instances for which it cannot find a solution within the 300 seconds timeout.

We presume that there are several reasons for the MIP solver outperforming the CP solver. First, the current model is formulated rather in MIP-style and a formulation that involves e.g. global constraints might also provide a better performance of the CP solver. Second, we cannot exploit the full power of the CP solver, since we cannot specify customized search heuristics on MiniZinc level that are tailored to the problem and the instance. For instance, we cannot specify a value selection heuristic for the van hour variables vh_d that searches for values that try to match the demand for each day. At the moment it is only possible to formulate such specific heuristics on solver library level.

Optimal solutions. We also observe that none of the solutions can completely match the demand, even the optimal solutions have some amount of unmet demand, in some cases even as high as 84 unmet orders. A reason for this might be the very conservative constant settings that do not allow for much flexibility. For instance, the low number of allowed working hours per week (currently 48 hours) might be too low to achieve a shift pattern that has zero unmet demand.

4 Conclusions

In this paper we have presented a novel problem that arises in the supply chain industry when designing a shift pattern for drivers in a delivery company that matches the demand. We gave a formal problem description and an initial mathematical model that we implemented in MiniZinc and evaluated on two different types of solvers, a CP solver and a MIP solver. We see that the MIP solver outperforms the CP solver, however we note that the comparison between the two solvers is not completely fair, since we evaluate the CP solver on a MIP-style model and cannot exploit its full power on Minizinc level.
For future work, we want to improve and extend our model with optional constraints, formulate a CP-style variant, and thoroughly assess the model on a large benchmark set. This way we will be able to evaluate the impact of the conservative constant settings on the results, and test if there are scenarios in which we can achieve zero unmet demand.

Bibliography

[1] B. Addis, R. Aringhieri, G. Carello, A. Grosso, and F. Maffioli. Workforce management based on forecasted demand. In Advanced Decision Making Methods Applied to Health Care, pp. 1—11. Springer, Milano, 2012.

[2] Aickelin, Uwe, and Kathryn A. Dowsland. An indirect genetic algorithm for a nurse-scheduling problem. Computers & Operations Research 31, no. 5 (2004): 761—778.

[3] Beliën, Jeroen, and Erik Demeulemeester. A branch-and-price approach for integrating nurse and surgery scheduling. European journal of operational research 189.3 (2008): 652—668.

[4] COIN-OR cbc solver: https://projects.coin-or.org/Cbc

[5] A. Ernst, T., Houyuan-Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. European journal of operational research 153, no. 1 (2004): 3—27.

[6] Alan M Frisch and Miquel Palahi. Alan Feasibility of Building Better Traincrew Rosters with Complete Solvers. In Proc. of the Fourteenth International Workshop on Constraint Modelling and Reformulation, August 2015.

[7] Gecode constraint solver: http://www.gecode.org/

[8] Mason, Andrew J., David M. Ryan, and David M. Panton. Integrated simulation, heuristic and optimisation approaches to staff scheduling. Operations research 46.2 (1998): 161—175

[9] N. Nethercote, P.J. Stuckey, R. Becket, S. Brand, G.J. Duck, and G. Tack. MiniZinc: Towards a standard CP modelling language. In C. Bessiere, editor, Proceedings of the 13th International Conference on Principles and Practice of Constraint Programming, volume 4741 of LNCS, pages 529—543. Springer, 2007.