Demand Driven Employee Scheduling for the Swiss Market

C.N. Jones
Automatic Control Laboratory,
EPFL, Lausanne,
Switzerland,
colin.jones@epfl.ch

K. Nolde
Apex Optimization GmbH,
c/o Automatic Control Laboratory,
ETH Zurich, 8092 Zurich,
Switzerland,
nolde@control.ee.ethz.ch

June 24, 2013

Abstract

This paper studies the problem of demand driven employee scheduling for direct-sales retail outlets. The demand for sales personnel during each half hour is predicted a week in advance based on previous recorded data and the problem of computing a near-optimal schedule to meet this demand is posed as a mixed-integer linear programming model. This model is incorporated into a commercially deployed online scheduling package, which has been in use at over 38 retail outlets in Switzerland over the last year. The key difference in the proposed modeling approach is that no complex heuristics or decompositions are used; the full model is formulated and solved to near-optimality. This is possible due to the relatively small scale of the target stores and leads to significant improvements in schedule quality.

1 Introduction

Standard practice for Swiss retail chains is to schedule employees so that the total number of workers present in the store is approximately constant during open hours. The number of shoppers, however, fluctuates throughout the day, which results in periods of under- and/or over-staffing that in turn reduces the effectiveness of the workforce. This paper reports on a new scheduling system that has been developed specifically for the Swiss market by Apex Optimization GmbH. The tool seeks to match expected customer demand to the number of sales staff by optimizing the shifts of the work force. The system has been successfully used by 38 small to mid-sized retail stores of the Migros chain of Switzerland over the past year, and the results of this initial implementation are reported here.

Schedules are computed on a weekly basis, one or more weeks in advance. Each week, the employees and/or store managers specify a wide range of store and employee-specific constraints through a web-based interface. The system then formulates a mixed-integer optimization problem in order to select a shift schedule that minimizes over- and under-staffing against a predicted customer demand profile, which has been estimated from past sales records.

While such scheduling schemes have been proposed in other markets in the past (Burke et al., 2004; Cheang et al., 2003; Mirrazavi and Beringer, 2007), Switzerland poses several unique challenges due to the local corporate culture and specifically to the culture of the Migros chain of stores. Migros is a cooperative chain, which means that each store has a high level of independence. Coupled with the Swiss focus on employee well-being, this requires schedules that are able to satisfy all employee-level concerns. As a result, buy-in at all levels of the corporation was required, which was achieved through regular communication and feedback from front-line managers and regular and on-going customization of the scheduling solution for the needs and desires of the employees. While these challenges make scheduling more difficult in the Swiss market, significant benefits are also expected due to very high labor costs.

The scheduling problem is known to be NP-hard and difficult to solve in general (Lau, 1996). The reader is referred to the surveys (Ernst et al., 2004; Alfares, 2004; Howick and Pidd, 1990) for a general technical overview and to (Burke et al., 2004; Cheang et al., 2003; Eveborn and Ronnqvist, 2004) for specific surveys of the application of demand-driven scheduling systems to nurse rostering, scheduling of call-center personnel and zoo employees. It is standard practice to decompose the problem into two (Caprara et al., 2003) or more (Tien and Kamiyama, 1982) phases and in (Ernst et al., 2004), six common steps, or modules are identified:

1. Demand modelling: determining how many staff members are required at any given time
2. Days off scheduling: choosing patterns of working and rest days
3. Shift scheduling: choosing shifts to be worked and number of employees per shift
4. Line of work construction: designing sequences of shifts valid for an employee
5. Task assignment: assigning activities to shifts
6. Staff assignment: assigning employees to shifts

For example, (Mirrazavi and Beringer, 2007) reports on an implementation of automated employee scheduling in the Sainsbury's grocery chain of the UK using a method with similarities to that introduced in this paper in that it computes schedules by solving mixed-integer optimization problems. It differs, however, in the scale of problem that it focuses on since the average Sainsbury's store has approximately 800 employees. As a result, the method is based on decomposition, which is necessary to handle this large scale. Specifically, the optimization is broken into three phases: First is a weekly optimization, during which employee working and off days are allocated as well as department assignments (modules 2, 3 and 5), second a daily optimization that assigns specific shifts to employees (modules 4 and 6) and finally an interactive reallocation process that deals with any infeasibilities that arose due to this decomposition (module 6).

In this paper we focus on the scheduling of small to medium size retail stores with employee numbers in the range of 10–100. We demonstrate that for these relatively small stores, it is possible to compute near-optimal schedules without the need to resort to decomposition (i.e., all modules 2– 6 are solved in a single optimization problem). This is both possible and necessary due to the much smaller store sizes, which reduce problem size, but also reduce flexibility,

making constraints very difficult to meet if an additional hierarchy is imposed on the solution method. The result is a reduction in implementation effort on the part of the developer, who is saved from creating and testing appropriate heuristics and manual steps (e.g., the third phase of the (Mirrazavi and Beringer, 2007) approach). An improvement in the quality of the resulting schedules is also expected due to the lack of the artificially imposed constraints that arise from the decomposition process.

The structure of the paper is as follows: Section 2 provides an outline of the problem and describes the requirements of the relevant employee shift scheduling problem. A generic mathematical formulation is given in Section 3, before Section 4 provides a performance analysis for a customized version that has been implemented in 38 stores of Migros in the Aare region of Switzerland.

2 Problem description

The goal is to create a weekly employee roster satisfying all store and employee-level constraints. Customer frequency is estimated for the coming week from previous data and the schedules are optimized against the estimated demand for sales personnel. In this section we outline the basic requirements and regulations for such a system operating in the Swiss market, before introducing more mathematical particulars in the next.

2.1 Store-level Constraints

Store managers specify various constraints in order to ensure basic operation of the store such as:

- The minimum and maximum number of employees working at any given time.
- Specific skills or personnel required during each time of the day such as managerial responsibilities, specific training or knowledge of certain product groups.
- Store operation parameters, such as opening and closing times, allowed break periods and durations.

Auxiliary tasks that do not directly involve sales or customer-driven services are also incorporated into the model. For example, store managers specify an amount of additional time each day that must be reserved for activities such as back-office work, shelf or re-stocking that can be done outside of the busiest hours.

It was found during the implementation of the system that managers had to have a significant ability to directly influence the schedules created by the system if they were to accept the result. Initial commissioning phases involved very tight, somewhat restrictive constraints imposed by managers so that daily activities did not differ too much from current practice. Once trust in the system was developed, then relaxation of the constraints would be allowed and the full benefits of the system seen.

2.2 Employee-level Constraints

Governmental regulations restrict the type and duration of shifts that each employee works, as does their personal preferences. Regulations place restrictions on the possible schedule of each employee, such as:

- Maximum number of hours worked without a break, or total hours per shift. These constraints will differ from employee to employee due to, for example, medical reasons.
- The number and duration of breaks during a shift are specified as a function both of the duration of the shift and the number of contiguous hours worked.

Acceptance of the system required that each employee be able to restrict the selection of his or her shifts according to several preferential or contractual parameters:

- Minimum and maximum number of scheduled hours per week. Employees are hired to work a specific number of hours per week. While this absolute number is allowed to vary slightly from week to week, the total number over an extended period is fixed.
- Full-time employees will work five days per week, but constraints are chosen by part-time employees that specify how many days per week and which days that they might work.
- Pre-scheduled vacations or absences can make employees unavailable on certain days or during specific times of the day. For example, working parents may choose only morning shifts in order to be home when school lets out or employees may be unavailable due to staff training activities.

Work-life balance is critical to swiss employers and as a result several employee preferences are taken into account in the form of soft constraints that can be overridden in preference of schedule optimality. Examples of these constraints include such things as the fair distribution of late or evening shifts amongst the workers, consistent days off each week, number of long or short shifts, etc.

2.3 Demand specification

The primary goal of the rostering system is to match the number of sales staff present in the store at each point in time to the number of customers. The expected customer frequency is automatically estimated on a half-hourly basis by averaging the till data of the previous four weeks and normalizing. While the variance of this data can be high for some classes of store and certain locations, on average the majority of scheduled stores have clear demand trends that are readily estimated from past data.

3 Mathematical Model

The developed scheduling system implements all constraints discussed in the previous section, as well as a small number of client-specific customizations that were required for acceptance by

store-level managers. In this paper, we focus on the generic scheduling problem and hence cover the formulation of general constraints in the following section.

The work to be done in the store is broken into a set of activities and each activity is then undertaken by an employee during a specified activity shift. The optimizer is required to choose a specific shift every day for each employee for each activity in such a way as to satisfy all employee– and store–level constraints as described above. This is achieved by first enumerating all possible activity shifts that each employee can work during a pre–processing phase before posing a mixed-integer linear program (MILP) that selects feasible shifts for each employee in order to minimize a given cost-function.

3.1 Activities and Shifts

We use a discrete-time model and break the store hours for each day d into an ordered set of time intervals $\mathscr{T}_d$. Each activity in the set of possible activities $\mathscr{A}$ that an employee can undertake is then specified by a set of 'activity shifts', which define the times during which the employee is doing the activity. For each activity $a \in \mathscr{A}$, the set $\mathscr{S}_{a,d}$ is all shifts on day d during which an employee can do the activity. Each shift $s \in \mathscr{S}_{a,d}$ is defined by an ordered binary set $\sigma_s \in \{0,1\}^{|\mathscr{T}_d|}$, for which $\sigma_{s,t} = 1$ if an employee working shift s would perform activity a on day d at time t.

Each store defines a set of activities that an employee might undertake. These include the basic activities of any store, such as 'Being at work'[1], 'Lunch break' and 'Coffee break', which we denote as `work`, `lunch` and `break` respectively. There are also several store-specific activities that are not related to direct-selling, such as 'Back office work' or 'Shelf re-stocking', which are also included in the activity set $\mathscr{A}$. It is assumed that the primary activity of any store is the direct-selling of products to customers and so all employees who are at work, but are not doing a secondary activity are performing this task.

During the pre-processing phase, the set of possible shifts for each activity on each day are enumerated and the ordered sets σ_s for each shift s defined. We assume that all activities are contiguous and that the user has specified a minimum/maximum shift length as well as first and last time intervals during which the activity can be performed. Under these assumptions, it is straight forward to enumerate all possible vectors σ_s for a given activity. Alternatively, a subset of these possible shifts can be chosen, which would result in a reduced computational effort at the potential cost of tracking performance.

We include in each set of shifts $\mathscr{S}$ the null-shift $\sigma_\emptyset := \{0,\ldots,0\}$, which indicates that the employee is not performing the activity on this day. This allows the specification that an employee either must, or must not perform the activity (e.g. if they do not have the appropriate skill, or have taken the day off, etc).

For example, consider an activity a on day d with open hours from 9:00 to 20:00 with a minimum shift length of 6 hours and a maximum of 9, in which the discrete time intervals $\mathscr{T}_d$ break the day into one-hour increments. The set of shifts $\mathscr{S}_{a,d}$ would consist of five 6-hour contiguous shifts, four 7-hour, three 8-hour and two 9-hour shifts. We include also the null-shift

[1] We refer to the activity `work` as 'Being at work' rather than 'working' to signify that this is the time during which the employee is on the premises, e.g. an employee is at work during their lunch break.

$\sigma_\emptyset$ to indicate an employee who is not performing the activity, for a total of 15 possible shifts.

3.2 Feasible Shifts

Every day of the week, each employee e in the set of employees $\mathscr{E}$ is assigned one shift for each activity. Note that the assigned shift can be the null-shift, indicating that the employee is not doing the activity on the given day. For each employee e and possible shift $s \in \mathscr{S}_{a,d}$, we define a binary variable $\delta^e_{s,a,d} \in \{0,1\}$, which is 1 if employee e is working shift s. We add the constraint that $\sum_{a \in \mathscr{A}} \delta^e_{s,a,d} = 1$, which indicates that each employee works exactly one activity shift per day for each activity.

Note that in some cases, employees can do more than one activity simultaneously. For example, the activities 'Having key to building', or 'Being a senior staff member on the sales floor' can be done in parallel to any other activity that does not involve the employee to leave the sales floor.

We can now write the schedule of a given employee by taking the product of the binary variables δ with the vectors σ

$$x^e_{a,t,d} := \sum_{s \in \mathscr{S}_{a,d}} \sigma_{s,t} \delta^e_{s,a,d} \ .$$

One can see from the above equation that employee e is doing activity a at time t on day d if and only if $x^e_{a,t,d}$ is equal to 1.

A feasible shift is one that satisfies all employee-level constraints given in the previous section. These are defined mathematically by placing linear constraints on the variables $\delta^e_{s,a,d}$, which we specify in the following sections. The constraints take the form of implications, such as

$$\delta^e_{s_i,a_1,d} = 1 \quad \Rightarrow \quad \delta^e_{s_j,a_2,d} = 0 \ , \tag{1}$$

which is read as 'Employee e can not work shift s_j performing activity a_2 if they are working shift s_i of activity a_1. A constraint of the form (1) can be written easily as a linear inequality

$$\delta^e_{s_j,a_2,d} \leq 1 - \delta^e_{s_i,a_1,d} \ .$$

3.2.1 Basic Constraints

The following set of constraints are required to create sensible shifts, i.e. those that don't have employees performing activities while not working, etc. During the pre-processing phase, each activity shift is considered in turn and the appropriate constraints added to the MILP as described below.

- All activities other than `work` can only be undertaken if the employee is currently working. Therefore, for each work shift $s_i \in \mathscr{S}_{\texttt{work},d}$ and activity shift $s_j \in \mathscr{S}_{a,d}$ such that

$$\sigma_{s_j} \not\subset \sigma_{s_i}$$

we add the constraint

$$\delta^e_{s_i,\texttt{work},d} = 1 \quad \Rightarrow \quad \delta^e_{s_j,a,d} = 0 \ ,$$

for each employee e, indicating that the work-shift has to be longer than, or equal to, the activity shift.

Further store-specific constraints can be implemented in a similar fashion, such as 'Shelf-filling cannot occur from 11:00 - 14:00', or 'Employees cannot take breaks during the first or last two hours of their shifts'.

- Some activities are mutually exclusive, such as being on break and performing back-office work. Constraints of the following form can be added to incorporate these limitations. Let $s_1 \in \mathscr{S}_{a_1,d}$ and $s_2 \in \mathscr{S}_{a_2,d}$:

$$\text{if } \sigma_{s_1} \cap \sigma_{s_2} \neq \emptyset \text{ then add constraints } \begin{array}{lll} \delta^e_{s_1,a_1,d} = 1 & \Rightarrow & \delta^e_{s_2,a_2,d} = 0 \\ \delta^e_{s_2,a_2,d} = 1 & \Rightarrow & \delta^e_{s_1,a_1,d} = 0 \end{array}$$

- Some work-shifts either require, or disallow other activities. For example, if an employee works more than a given number of hours, they must have a lunch break. These requirements can be added in a similar fashion as previous ones by adding constraints that require or disallow the null-shift for the break activity if the employee is working more than a specified amount of time `breakRequired`.

$$\text{if } \sum_{t \in \mathscr{T}_d} \sigma_{s,t} \geq \texttt{breakRequired}$$
$$\text{then add constraint } \delta^e_{s,\texttt{work},d} = 1 \quad \Rightarrow \quad \delta^e_{s_\emptyset,\texttt{lunch},d} = 0 \ . \quad (2)$$

Similar constraints can be added to account for more complex break rules or other such activities.

3.2.2 Employee-Specific Constraints

As discussed in Section 2.2, each employee defines a set of contractual, or preferential constraints that are added to the optimization problem.

- Depending on contractual arrangements, each employee is given a minimum number of days off each week. This constraint is easily encoded by requiring that the number of null-shifts each week is at least that required, where we recall that an employee working a null-shift is not doing that activity on the given day. For each employee, we add a constraint of the following form:

$$\sum_{d \in \mathscr{D}} \delta^e_{s_\emptyset,\texttt{work},d} \geq \texttt{requiredDaysOff}^{\texttt{e}} \ .$$

- A key requirement for acceptability of the scheduling system was that store managers could specify days that a specific employee is staffed, or has off. Such conditions can also be encoded by adding constraints on the null-shift. For example, if employee e is required to work on day d, we simply add the constraint

$$\delta^e_{s_\emptyset,\texttt{work},d} = 0 \ ,$$

which states that the employee must work a non-null shift on day d (i.e. is staffed).

Several store managers who were concerned that their part-time employees would have their days off in a block, rather than as singletons, used this constraint in order to specify the specific days off. This is done in a similar manner to the above constraint by forcing the shift on a given day d to be the null-shift, $\delta^e_{s_\emptyset,\texttt{work},d} = 1$.

- Many employees have specific hours of the week during which they cannot be staffed. Such constraints are easily added by simply setting the shift selection binary $\delta^e_{s,\texttt{work},d}$ to zero for all shifts that contain these hours. Similarly, the requirement that an employee must work given hours can be fulfilled by setting to zero all shifts that do not contain those hours for that employee.

- The condition that employee e is to work $\texttt{numHours}^e$, can be readily encoded in the following linear constraint:

$$\sum_{d\in\mathscr{D}}\sum_{t\in\mathscr{T}_d} x^e_{\texttt{work},t,d} - x^e_{\texttt{lunch},t,d} - x^e_{\texttt{break},t,d} = \texttt{numHours}^e \ .$$

The left hand side of the above equation computes the number of time slots that the employee is on the work-site, and is not on a break, or equivalently those times for which they are paid.

- Each employee has a set of skills that define the activities that they can undertake. If an employee e does not have the appropriate training to perform a given activity a, we simply add a constraint to require that the employee must work the null-shift for activity a

$$\text{if employee } e \in \mathscr{E} \text{ cannot do activity } a \in \mathscr{A}, \text{ then } \delta^e_{s_\emptyset,a,d} = 1 \ .$$

- Some managers impose a constraint that employees are only allowed to work evenings (e.g. after 6pm) a given number of days per week in order to prevent fatigue. If $t_{\texttt{late}}$ is the first time that is considered 'late', then the following constraint limits the number of evening shifts per week:

$$\sum_{d\in\mathscr{D}} x^e_{\texttt{work},t_{\texttt{late}},d} \leq \texttt{maxLateDays}^e \ .$$

Note that no shift can start after the time slot $t_{\texttt{late}}$ due to store closing hours, and so an employee is working the evening shift if and only if they are working at the first 'late' time slot.

3.3 Store Constraints

Store managers specify which skills are required either on an hour-by-hour basis, or in terms of the total number of hours for a specific skill on a daily or weekly basis. The number of employees performing a given activity a at time t is then simply given by the following sum:

$$\alpha^a_{t,d} := \sum_{e \in \mathscr{E}} x^e_{a,t,d}$$

Since the number of employees performing a given activity is linear in the optimization variables δ, we can express the requirement that a minimum/maximum number of employees perform a given activity at a specific time as a linear constraint:

$$\texttt{minEmp}^a_{t,d} \leq \alpha^a_{t,d} \leq \texttt{maxEmp}^a_{t,d}$$

Similarly, a store manager can specify that a number of work hours must be put aside each day or week for a given activity in a linear fashion:

$$\sum_{t \in \mathscr{T}_d} \alpha^a_{t,d} = \texttt{hoursForActivity}^a_d \ .$$

3.4 Cost Function

The purpose of the scheduling system is to track customer frequency with the number of direct-selling employees. As discussed above in Section 3.1, an employee is assumed to be on the sales floor and executing the 'direct sales' activity if they are at work (i.e. $x^{\cdot}_{\texttt{work},\cdot,\cdot} = 1$) and not undertaking any other activity and so the number of salespeople working at a given time is:

$$\alpha^{\texttt{ds}}_{t,d} = \sum_{e \in \mathscr{E}} \left(x^e_{\texttt{work},t,d} - \sum_{a \in \mathscr{A} \setminus \{\texttt{work}\}} x^e_{a,t,d} \right) \ .$$

The error between the given demand curve, which is denoted $\eta_{t,d}$ and is assumed specified for each time interval, and the number of direct-sales employees at each time is then:

$$\varepsilon_{t,d} := \alpha^{\texttt{ds}}_{t,d} - \eta_{t,d} \ . \tag{3}$$

The goal is to minimize some norm of this error vector; the most obvious being quadratic, which would result in a mixed-integer quadratic program (MIQP):

$$J^{MIQP} := \sum_{d \in \mathscr{D}} \sum_{t \in \mathscr{T}_d} \varepsilon^2_{t,d} \ .$$

Experience has shown that a piecewise-affine norm, which results in a mixed-integer linear program, is more computationally efficient than a quadratic one. We therefore approximate the above cost function J^{MIQP} with a convex piecewise affine function, which provides very similar results, but only requires the solution of a mixed-integer linear program (MILP):

$$J^{MILP} := \sum_{d \in \mathscr{D}} \sum_{t \in \mathscr{T}_d} \| W \varepsilon_{t,d} + w \|_\infty \ ,$$

where the matrix W and w are linearizions of the quadratic function ε^2 at various points.

3.5 Computational Details

The objective and the constraints defined above result in a mixed integer linear programming problem (MILP), which frequently occur in scheduling formulations, see (Pinedo, 2007) and (Al-Yakoob and Sherali, 2007) for examples. However, MILPs are difficult to solve and their compute time depends both on the number of decision variables δ, the formulation or structure of the constraints and the various parameter values. While it is impossible to demonstrate that the above scheduling problem can be solved efficiently in general, we show below empirically that for the stores under consideration it is possible to solve it to an acceptable level of accuracy using standard MILP solvers such as (Cplex, 2006) or (Xpress, 2007).

Several computational and practical details are implemented in the Apex Optimization scheduling software, which are not covered in detail here. These include:

- The constraints are split into a hierarchy based on their criticality and the less-important ones are implemented as soft constraints. This allows a feasible schedule to be found, even if there is no schedule that strictly meets are desired restrictions. This is primarily an issue in smaller stores, in which managers and employees were being flexible in order to get the job done when scheduling manually.

  In particular, for the majority of stores the following constraints were softened:

  - Total employee hours per week
  - Number of days each employee works late per week
  - The requirement that n employees with a skill marked 'non-critical' are present in the store at a given time t
  - Number of additional hours set aside per day for unspecified activities (e.g., back office work)

- Solving the problem to optimality is in general not feasible or necessary. After experimentation with the optimization system, a time limit was imposed as a function of the store size and complexity, which generally results in a near-optimal schedule. See Section 4.3 for details.

- While the number of variables in the above model can become quite high, many of them will be removed during a pre-processing phase. During pre-processing, the system looks for binary variables that can be simply determined to be either one or zero, and converts these to constants. For example, if an employee does not have the skill to perform activity a, then all of the binary variables for choosing the skill for that employee can immediately be set to zero, except for that associated to the null-shift for the activity a, which is set to one. This pre-processing step can result in a large reduction in model size. For example, the pre-processing step removed 77% of the 56,160 binaries for the 16 employee store considered in Section 4.4.

4 Case Study: Migros Aare

In this section we report on the performance of the scheduling tool in practice by comparing the quality of manually created schedules to ones that are computed by solving a mixed-integer linear program. The scheduling model discussed in the previous section has been customized for the use of Migros, a large Swiss based retailer and has been implemented in over 38 stores in the Aare region of Switzerland. We further compare the Apex Optimization automatic scheduling system against that reported in (Mirrazavi and Beringer, 2007), which is currently being used to optimize the rosters of the supermarket chain Sainsbury's in the UK.

4.1 Quality Factor

To quantify the results of the comparison we make use of a quality factor as defined in the paper of (Mirrazavi and Beringer, 2007). This factor is the sum the absolute error $\varepsilon_{t,d}$ between the predicted demand for sales personnel and the scheduled number in each hour of the week (3), which is then normalized by the total number of sales-hours called for each week:

$$\Phi := 1 - \frac{\sum |\varepsilon_{t,d}|}{\sum \eta_{t,d}} \tag{4}$$

The quality factor Φ takes values in the range $[0,1]$, with 1 indicating perfect tracking.

The maximum achievable quality factor is bounded above by the constraints of the store and its employees. Investigations in cooperation with Migros have found that the vast majority of stores could be scheduled with a very high quality factor without modification of the employment structure that was in place, which consist for the most part of full-time employees. However, some smaller stores could benefit from more flexible part-time employees which would increase the maximum achievable quality factor.

4.2 Implementation

The automated scheduling system has been implemented by Apex Optimization as a web-based application. An AJAX-based browser interface allows store managers to access and enter all details of their employees through the Internet. When all store specifics have been recorded, the manager can request the system to compute a schedule. A screen-shot of the web-based system is shown as Figure 1 below.

4.3 Computational details

The analysis in this section is based on an initial commissioning phase during which 208 schedules were computed for a set of 45 departments in 25 stores. All schedules had a time resolution of 30 minutes, a duration of one week and ranged in size from four to 20 employees.

Computational difficulty varies depending on the number of employees in the store and the challenge of satisfying the constraints. It was found that a near-optimal schedule can be computed in 5–10 minutes for a store of 10–30 employees with a 1-hour discretization, but that 10–25 minutes were required for the desired 30 minute resolution using the Dash Optimization

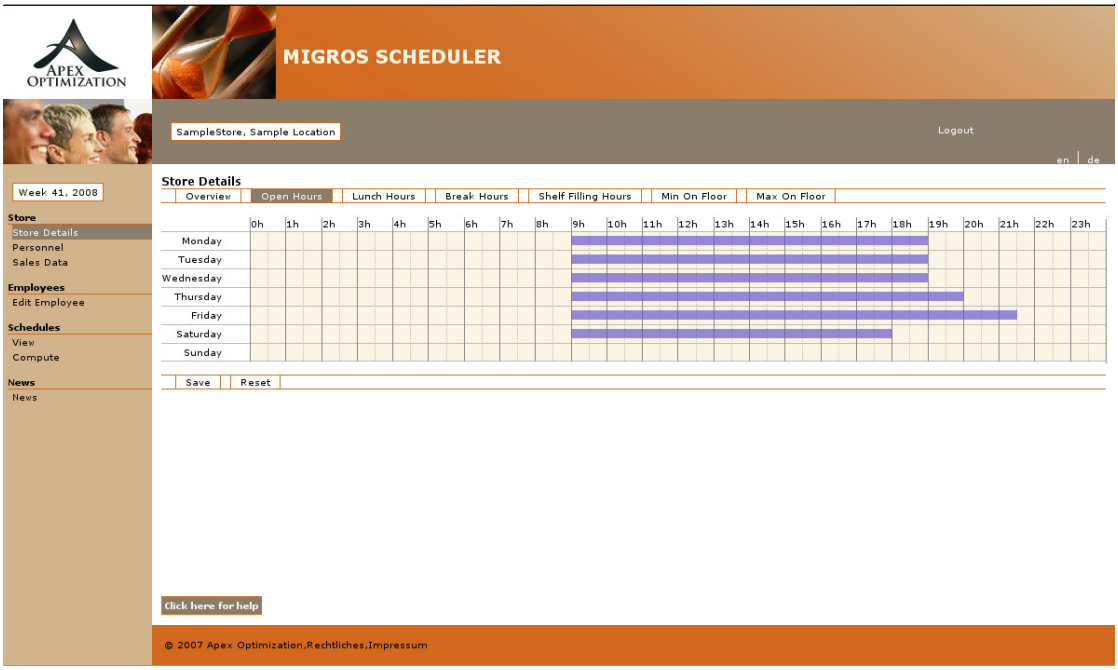


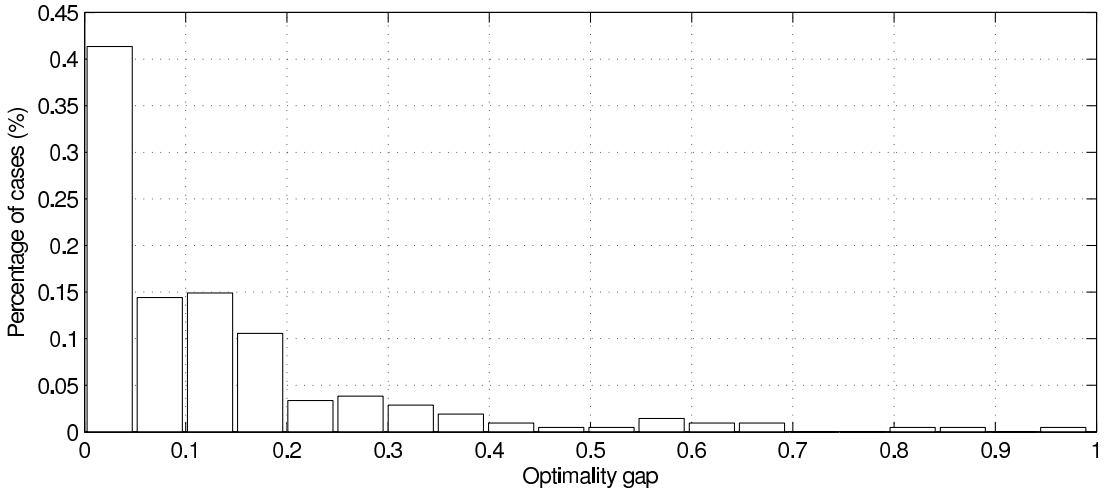
Figure 1: Apex Optimization web-based employee management and scheduling system.

computational engine (Xpress, 2007). A fixed time-bound of 1,400 seconds (approx 23min) was chosen and a histogram of the resulting optimality gap can be seen in Figure 2(a) for the considered 208 schedules. Of note is that fact that the optimality gap is uncorrelated with the size of the store, possibly indicating that while larger numbers of employees require more constraints and binary variables (see Figure 2(b)), they also offer a higher level of flexibility, which simplifies finding integer feasible solutions. All reported schedules, even those with relatively high gaps, were accepted by the store managers as an improvement over their current manual scheduling practice in terms of demand tracking and constraint satisfaction. The system has also been applied to artificially generated larger stores of up to 200 employees, for which it can compute good schedules in under an hour.

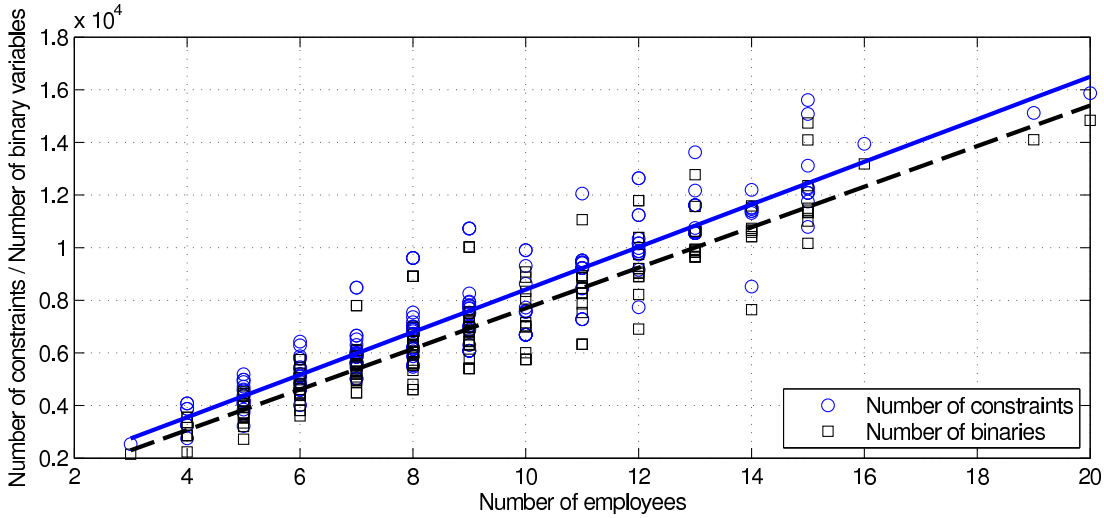
4.4 Case Study

Figure 4 shows the quality factors for manual and automatic scheduling of 38 stores of Migros Aare. Also shown for comparison are the factors for a similar study undertaken in the UK for the chain of Sainsbury's grocery stores (Mirrazavi and Beringer, 2007). The graph shows the percentage of stores (y-axis) that reached or exceeded the quality factor (x-axis). Thus, a curve lying further to the right corresponds to a better scheduling result.

The dashed curve shows the quality factor for manually created schedules taken from a sample of four of the most 'average' stores. The curve indicates that the quality of the manual schedules ranges from 65% to 76%, with an average of 69.6%. Examining the manual schedules we find that they generally follow a shift model where employees work the same length



(a) Histogram of optimality gap



(b) Relationship between number of employees and number of constraints / binary variables after pre-processing

Figure 2: Analysis of the 208 cases considered during initial commissioning phase discussed in Section 4.3

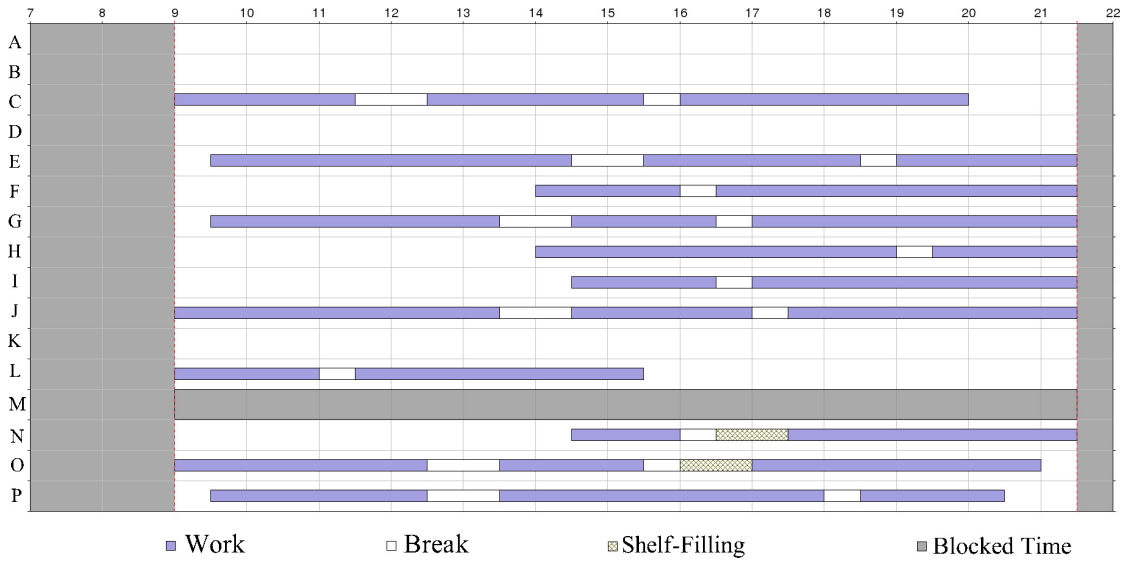


Figure 3: Example of an automatically generated employee schedule.

shift every day and many breaks often occur during the same period, reducing the match to the demand forecast.

With the automated scheduling procedure of this paper an average quality factor of 79% was achieved, based on schedule and forecast data for 38 stores of Migros Aare that currently use the system. When comparing the results for the manually computed schedules to the computer generated we find that the average quality increases from 69.6% to 79.0%. This means that by using the scheduling system the amount of over- and understaffing has been reduced by 9.4%, or 79 person hours being allocated more effectively for an average store of 20 full-time employees. Note that while this comparison was made for 'average' stores, the small sample size and high variability in manual schedule quality should be taken into consideration when interpreting this result.

Examining the schedules in detail, one sees that the scheduling system tracks the demand curve by using combinations of different length shifts and staggering break times to match less busy periods of the day. In many cases the forecast demand for Saturday will exceed the number of available personnel and therefore the desired staffing cannot be reached with the current personnel structure. An increase in the number of part time employees with the flexibility to work on Saturdays would help to improve the schedule quality further.

For comparison we also include in Figure 4 the data that are reported in (Mirrazavi and Beringer, 2007) on the quality of the schedules for 50 Sainsbury's stores in the UK, which average 74.6%. While it is tempting to conclude that the presented Apex system is superior in general due to the improved quality factors, this is a very difficult conclusion to draw due to the significant differences in data sets used. The variation in quality factor may be due to a number of reasons related to both the type of stores scheduled and to the optimization systems. The Sainsbury's study involved large grocery stores, which have 10 to 20 times more employees than the Migros stores and therefore made use significant of heuristics as well as hierachical and

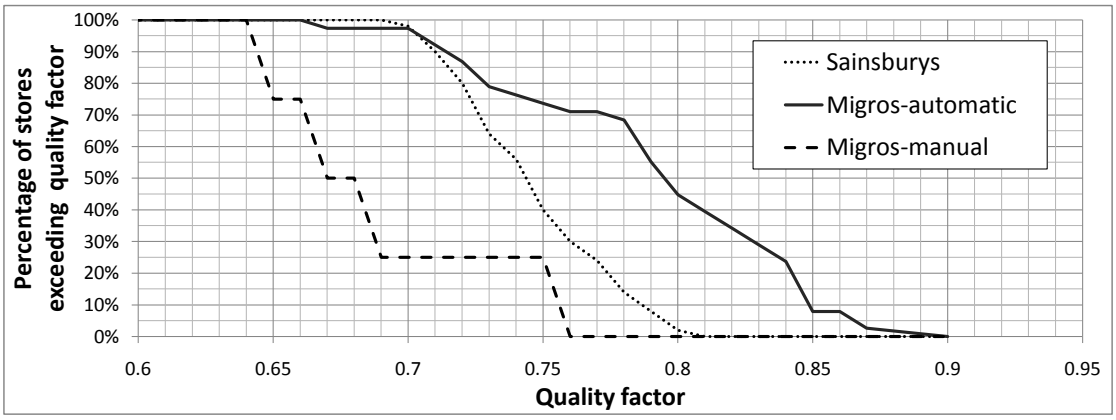


Figure 4: Comparison of manually created schedules to computer generated ones.

departmental decomposition schemes in order to reduce the search space. In comparison, the method presented in this paper is applied to stores that are small enough that no heuristics or complexity reduction techniques are required, which could be one factor that results in the larger quality factor observed.

To give a more detailed view on the effect of the changes we present an example for a store of Migros Aare where manually and automatically generated employee schedules are compared. A typical store of 16 employees has been selected; two part time, eleven full-time and three trainees, who have to attend pre-scheduled training sessions. Three skill-types were defined in this setup, each of which is held by a subset of the employees and the store-manager has specified how many employees with each type of skill are required at each point during the week. All legal and company requirements as well as individual restrictions on employee schedules have been enforced. On weekdays, the store is open from 9 a.m. until 7 p.m. except Thursday and Friday when it is open until 8 p.m. and 10 p.m. respectively. As is standard in Switzerland, it is closed on Sunday, and the Saturday hours are reduced to 8 a.m. – 5 p.m.

The forecast demand of the store is based on average historic sales data and the goal of the optimization routine is to design a roster that matches this demand as closely as possible. Figure 5 shows a comparison of the scheduled number of direct-sales employees against the forecast demand. The manual schedule is designed using a standard Migros approach, in which the employees come to work at the opening hour, take lunch breaks in a staggered fashion over a three hour period around noon, before going home when the store closes. The disadvantage of this traditional model is clearly seen when comparing the number of sales personnel with the forecast demand: The store is heavily overstaffed during the morning hours, when the demand is rather small, except on Saturday, when the demand by far exceeds the number of scheduled employees.

Using the automated scheduling system we see a very different picture. As an example a Gantt chart of the automatically generated schedule for Friday is shown in Figure 3. The automated system starts most shifts later in the day from Monday to Friday. In the Figure 3 this is the case for the employees F, H, I and N who only work during Friday afternoon. By

staggering the shift starts, work-hours are saved that can later be used on peak periods and Saturdays. While a significant improvement in tracking is seen, the number of sales personnel in the store on Saturdays still does not meet the demand. This suggests that a change in staffing structure by hiring additional part time employees that can work on Saturdays is called for.

Acknowledgments

The authors gratefully acknowledge the efforts of Sven Voser and of Migros Aare in conducting extensive case studies during the preparation of this work.

References

S. Al-Yakoob and H. Sherali. Mixed-integer programming models for an employee scheduling problem with multiple shifts and work locations. *Annals of Operations Research*, 155(1): 119–142, Nov. 2007.

H. Alfares. Survey, categorization, and comparison of recent tour scheduling literature. *Annals of Operations Research*, 127(1–4):145–175, March 2004.

E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, Nov. 2004.

A. Caprara, M. Monaci, and P. Toth. Models and algorithms for a staff scheduling problem. *Mathematical Programming*, 98(1):445–476, Sept. 2003.

B. Cheang, H. Li, A. Lim, and B. Rodrigues. Nurse rostering problems–a bibliographic survey. *European Journal of Operational Research*, 151(3):447–460, Dec. 2003.

Cplex. *ILOG CPLEX 10.0 User's Maual*. ILOG, France, January 2006.

A. Ernst, H. Jiang, M. Krishnamoorthy, and D. Sier. Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1):3–27, Feb. 2004.

P. Eveborn and M. Ronnqvist. Scheduler - a system for staff planning. *Annals of Operations Research*, 128(1):21–45, Apr. 2004.

R. Howick and M. Pidd. Sales force deployment models. *European Journal of Operational Research*, 48(3):295–310, 1990.

H. C. Lau. On the complexity of manpower shift scheduling. *Computers & Operations Research*, 23(1):93 – 102, 1996.

S. Mirrazavi and H. Beringer. A web-based workforce management system for sainsburys supermarkets ltd. *Annals of Operations Research*, 155(1):437–457, Nov. 2007.

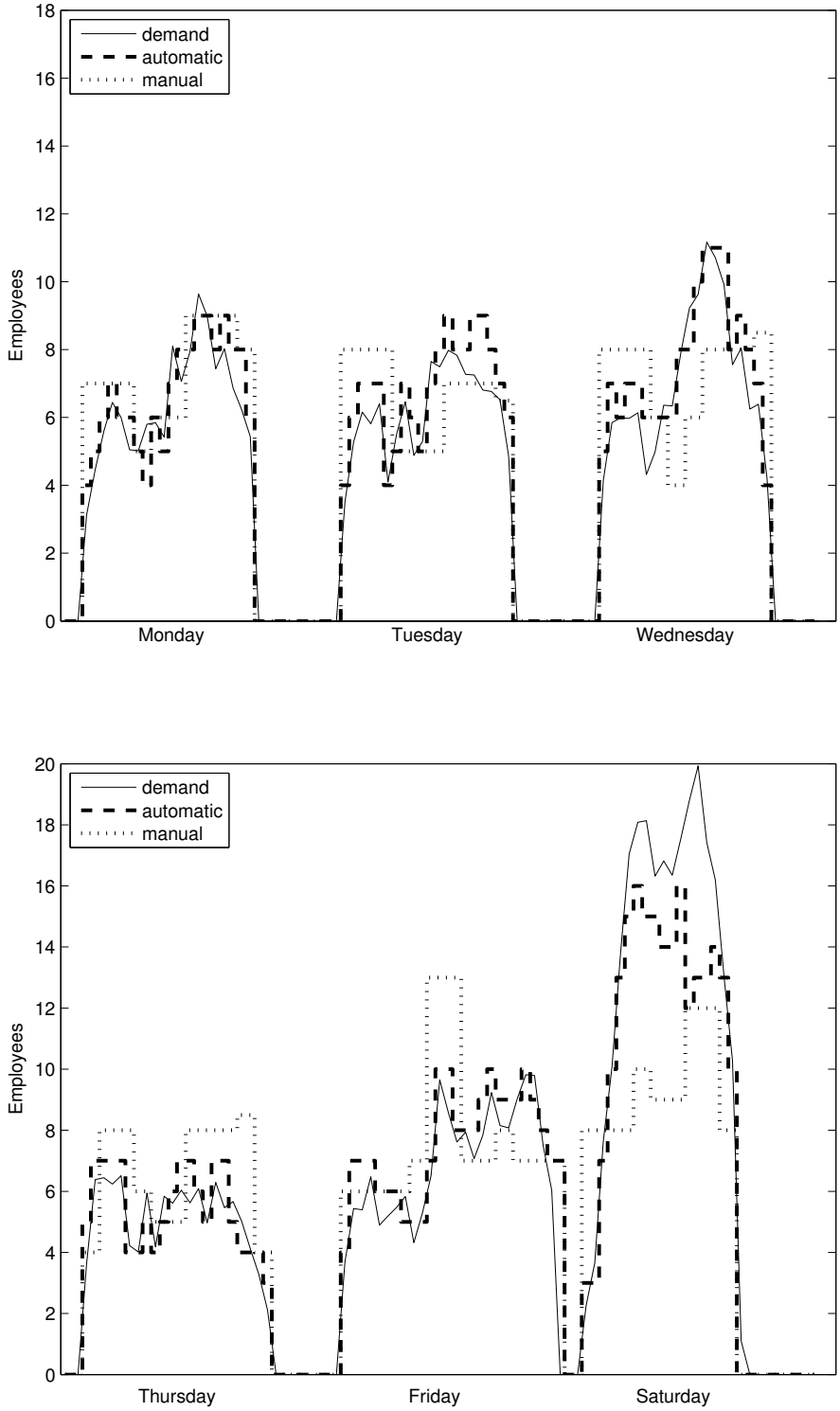


Figure 5: Comparison of the over- and understaffing, when using the automatically or the manually generated schedules. The quality-factor of the manually generated schedule is 68.1% compared to 90.2% for the automatically generated schedule.

M. L. Pinedo. *Planning and Scheduling in Manufacturing and Services (Springer Series in Operations Research and Financial Engineering)*. Springer, 2007. ISBN 0387221980.

J. M. Tien and A. Kamiyama. On manpower scheduling algorithms. *SIAM Review*, 24(3): 275–287, 1982. doi: 10.1137/1024063.

Xpress. *Xpress-Optimizer Reference Manual Release 18.10*. Dash Optimization, November 2007.