Are fast food chains really that efficient? A case study on crew optimization.

Jan Zouhar[1], Irena Havlová[2]

Abstract.

Fast food chains have long been setting an example in operations management efficiency; however, as we show in the paper, there is still room for improvement. Our case study deals with crew optimization in a particular fast food restaurant. First, we describe the current practice in crew scheduling—which is mostly done manually by the employees. Next, we provide two alternative MILP formulations that can be used to find the optimal schedule under the given criteria; these formulations take into account the specific crew structure of the fast food restaurant (full-time vs. part-time employees, underage temporary jobs etc.), as well as its specific criteria. We discuss and compare the computational tractability of the proposed models for small-scale applications and comment on solution concepts for large-scale ones. Finally, we evaluate the restaurant's crew scheduling efficiency by comparing the quality of the actual (empiric) schedules to those obtained by MILP.

Keywords: supply chain efficiency, crew scheduling, mixed integer linear programming.

JEL classification: M54, L66
AMS classification: 90B70

1 Introduction

The huge success of fast food giants (such as McDonald's, Burger King or Kentucky Fried Chicken) has usually been attributed to their exceptional operations efficiency. Indeed, fast food chains were among the first companies outside the manufacturing industry that utilized modern production management principles, such as lean production and Just-in-Time (JIT) strategy. Undoubtedly, the sheer idea of a fast food restaurant logically encompasses some of JIT practices, e.g. short lead times, quick setups and small-lot production—but many fast food chains did not stop there. For instance, a couple years ago, McDonald's started using Kanban cards in order to establish a pull production system, and developed a new type of bun toaster to cut down on production time.

Another source of fast food restaurants' cost efficiency is the high degree of work standardization. Among other things, it allows them to hire staff with minimal training, making temporary jobs at the restaurant suitable for underage workers (especially students), who have few alternative job opportunities. As noted in [3], fast food industry exhibits a very high proportion of minimum-wage workers. Moreover, the presence of many part-time employees and/or temporary jobs increases the flexibility of personnel schedules; in other words, it creates the opportunity to adjust the amount of manpower at a particular time to predicted staff requirements (which reduces potential costs of staff shortage/overage).

However, as we demonstrate using a case study of a Czech fast food restaurant[3], these potential benefits from schedule flexibility are not exploited efficiently by the fast food restaurants' executives. The main reason is that, to our knowledge, crew scheduling is mostly done manually[4]—which both takes

[1]Univ. of Economics, Prague, Dept. of Econometrics, nám. W. Churchilla 4, 130 67 Praha 3, e-mail: zouharj@vse.cz
[2]Univ. of Economics, Prague, Dept. of Econometrics, nám. W. Churchilla 4, 130 67 Praha 3, e-mail: xhavil0@vse.cz
[3]Neither the name of the fast food chain nor the location of the restaurant can be disclosed due to confidentiality reasons.
[4]This was the case with the particular fast food restaurant we studied; the restaurant's executive reported that it was a common practice within the whole fast food chain. Obviously, there are exceptions to this rule, though probably not in the Czech Republic; [7] reports on a U.S. example of a fast food restaurant that uses an automated crew scheduling system.

a long time even for an experienced schedule manager and results in a sub-optimal schedule, since with more than only a handful of employees it is virtually impossible to find the optimal schedule by hand.

In the rest of the paper, we aim to support this claim using the data from our case study. In §2, we briefly review existing optimization approaches to crew scheduling. In §3, we present a thorough description of the crew scheduling problem we dealt with in our case study. As our case study was of a relatively small scale—the restaurant had 40 employees (full-time, part-time and temporary)—we could use mixed integer linear programming (MILP) as the optimization vehicle. In §4, we propose two quite different MILP formulations of the scheduling problem. Both have their pros and cons: one is easier to implement, the other is more computationally efficient; we provide a brief discussion of both models' computational tractability at the end of §4. Finally, in §5, we evaulate the restaurant's crew scheduling efficiency by comparing the quality of the actual (empiric) schedules to those obtained by MILP.

2 Optimization approaches to crew scheduling

In general, crew scheduling is the process of creating work timetables for a firm's staff that (1) are feasible/satisfactory for individual employees and (2) efficiently cover the demand for manpower over the given time horizon. In all but the simplest cases, finding a good schedule is a very demanding task, and many automated decision-support systems have been developed for different application areas, which mostly come from the transportation, health care, and services industries; an extensive review is given in [5]. Solutions are typicaly obtained using one of the following techniques:

- *Mathematical programming.* Most crew scheduling problems can be described as a MILP problem; however, for large-scale problems and/or problems with complicated constraint structures, MILP models are not computationally tractable. Nevertheless, this is the approach we took in our case study, which is a rather small-scale one. As we show in §4, our solution admits some up-scaling, but for a large-scale version of the problem, heuristics have to be used instead.
- *Constraint programming* (CP). In general, CP is not a very efficient optimization technique; however, as noted in [5], CP is particularly useful for crew scheduling if the problem is highly constrained and/or when any feasible solution will suffice even if it is not optimal.
- *Metaheuristics.* Most metaheuristic approaches can be adapted to crew scheduling problems, e.g. genetic algorithms (see [8]) or simulated annealing (see [1]).
- *Other heuristics.* Apart from the applications of general metaheuristic approaches, a considerable effort has been put into designing tailor-made heuristics for specific types of staff scheduling problems (see e.g. [2]); several of these heuristics contain applications of AI techniques, as in [6].

3 Problem statement

Specifics of crew scheduling in fast food chains. Fast food crew schedules exhibits several features that make it difficult to optimize. Perhaps the most prominent one is crew heterogeneity. As mentioned above, fast food chains rely heavily on part-time and/or temporary employees, some of which are typically underage students. There are several implications of this. Firstly, there are multiple skill levels that have to be taken into account and scheduled in parallel; in our case study, there has to be at least one full-time manager present at each time to supervise the current shift. Secondly, there are no standard shifts, such as regular 8-hour blocks (which is the case in some other industries, and facilitates scheduling to a great extent). Different types of employees allow for and/or require different shift lengths. Finally, part-time employees can specify their *available times*, i.e. times when they can actually be on duty. These times are typically specified only a short time in advance: in our case study, non-full-time workers announce their available times only 14 days ahead. This effectively rules out long-term planning, and (more importantly) repetitive schedule plans; in other words, the schedule has to be re-optimized each week.

Scope of optimization. In our case study, we omit *night shifts* and *managers* from the model. According to the restaurant's executive, both of these are scheduled separately, on a monthly basis (unlike the remaining part of the schedule, which is prepared weekly for the reasons stated above).

Object types and sets. In our statement of the crew scheduling problem, we use the following objects: employees, days, time periods and shifts. These objects, together with the related sets and their notation,

are described in the following list:

E = the set of all **e**mployees. In the fast food chains, only a minor part of the staff is typically made up by *full-time employees*. The rest can be divided into *part-timers*, *temporary jobs*, and *minors* (underage temporary-job employees, mostly students). In the mathematical notation, we can write $E = E_{\text{full}} \cup E_{\text{part}} \cup E_{\text{temp}} \cup E_{\text{min}}$.

D = the set of (numbered) working **d**ays (typically, Sunday through Saturday, i.e. $D = \{1, \ldots, 7\}$).

T = the set of (numbered) **t**ime periods within a single day. (In our case study, these are half-hour periods from 6:00 to 22:00 each day, and therefore $T = \{1, \ldots, 32\}$.)

S = the set of acceptable shifts. In brief, shifts are collections of consecutive time periods; we will describe acceptable shifts in more detail later on.

Scheduling constraints. A feasible schedule must follow be the following conditions:

(*i*) An employee can only be assigned to work at her *available time*; available times are specified by the employee in advance.

(*ii*) On each day, an employee's time on duty has to be made up by consecutive time periods (together constituting a single *shift*).

(*iii*) An employee can be assigned to at most 5 shifts per week.

(*iv*) A shift has to be exactly 8 hours long for the full-timers, between 4 and 8 hours for the part-time and temporary-job employees and between 4 and 6 hours for the minors (underage employees).

(*v*) No employee can be assigned to more than 40 hours a week in total (note that this already comes from the conditions above). Moreover, full-time employees have to be given exactly 40 hours a week, and part-time employees at least 25.

Scheduling objectives. Many different objectives of staff scheduling are mentioned in the literature; in our case study, we restricted ourselves to two objectives that reflect the goals pursued in practice in our case study: (1) minimize the difference between the number of workers on duty and the staff demand in each time period, and (2) minimize the number of shifts. (The intuition behind the latter is that employees prefer few long shifts to many short ones.) In our model, these two objectives are aggregated using weights (or cost parameters) corresponding to their relative importance:

$\mathsf{c}^{\mathsf{under}}$ = the **c**ost per unit of **under**coverage (staff shortage) = 10.

$\mathsf{c}^{\mathsf{over}}$ = the **c**ost per unit of **over**coverage (staff excess) = 5.

$\mathsf{c}^{\mathsf{shift}}$ = the **c**ost per **shift** = 1.

Data inputs. There are two types of data supplied for a week's schedule: staff requirements for each time period, and employees' available times; the latter are specified in the form of a "from–to" range. The mathematical notation used in the rest of the paper is as follows:

re_{dt} = the number of staff **re**quired at time period t on day d.

$\underline{\mathsf{av}}_{ed}$ = the beginning of **av**ailable time for employee e on day d (the number of the first available period).

$\overline{\mathsf{av}}_{ed}$ = the end of **av**ailable time for employee e on day d (the number of the last available period).

These data are typically supplied in a spreadsheet, such as the one shown in Figure 1.

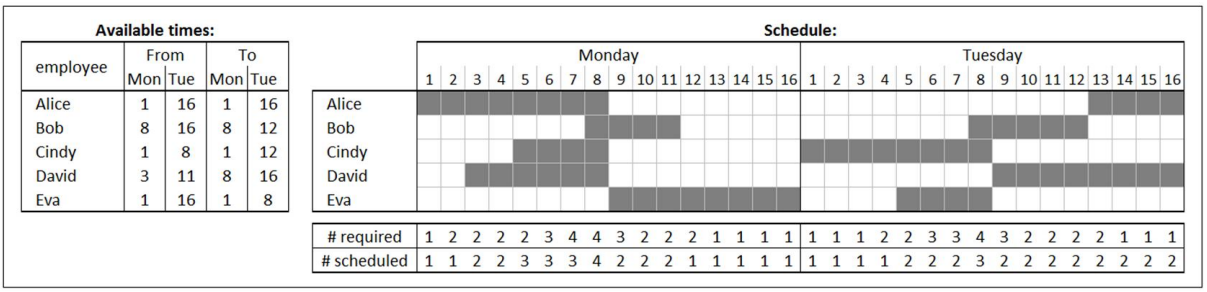

Figure 1: A minimalist example of a spreadsheet with data inputs and the resulting schedule. Data inputs are the numbers provided in the *Available times* table (the $\underline{\mathsf{av}}_{ed}$ and $\overline{\mathsf{av}}_{ed}$ values) and the *# required* row (re_{dt}). Using Model 1, the graphical schedule is created simply by applying suitable conditional formatting rules to the optimal 0–1 values of x_{edt} variables.

4 MILP formulations

In this section, we present two alternative MILP formulations of the problem, denoted as Model 1 and Model 2. The latter is analogous to the many variations of Dantzig's classical set-covering formulation [4]. Although this model is computationally quite efficient, it requires that large amounts of data are processed and prepared before the model is actually fed into an optimization software package, and the results are not easily translated into a readable schedule format, such as the one shown in Figure 1 (which is the format that was used by our study's restaurant in past). We explain these issues in detail in the discussion of Model 2; we nevertheless recognize that they might pose an obstacle for practical usage by small businesses. For this reason, we devised Model 1, which does not require any additional data processing—it works directly with the values of $\underline{\mathsf{av}}_{ed}, \overline{\mathsf{av}}_{ed}$, and re_{dt}—and, with proper conditional formatting in a *MS Excel* spreadsheet, the exported results look exactly the same as the schedule shown in Figure 1.

Model 1. Model 1 uses several decision variables, described in the list below. Binary variables are denoted by Latin letters, while Greek letters represent real non-negative variables.

δ^-_{dt} = staff shortage at time t, day d (negative part of the **d**eviation from the requirement re_{dt}).
δ^+_{dt} = staff overage at time t, day d (positive part of the **d**eviation from the requirement re_{dt}).
λ_{ed} = the **l**ength of the shift of employee e on day d ($= 0$ if e does not work on day d).
$x_{edt} = 1$ if employee e works in time period t on day d.
$y_{edt} = 1$ if employee e starts her shift in time period t on day d.
$z_{edt} = 1$ if employee e ends her shift in time period t on day d.

In order to make the MILP formulation easier to read, we use simplified notation for summation and iteration indices: unless stated otherwise, e, d, and t are indices that cycle through the entire sets E, D and T respectively, so that "$\sum_e$" effectively means "$\sum_{e \in E}$", "for all d" means "for all $d \in D$" etc. Moreover, we drop variable specification from the formulation, as we have already specified all variables above. The resulting formulation of Model 1 is as follows:

$$\text{minimize} \quad \mathsf{c}^{\mathsf{over}} \textstyle\sum_{d,t} \delta^+_{dt} + \mathsf{c}^{\mathsf{under}} \sum_{d,t} \delta^-_{dt} + \mathsf{c}^{\mathsf{shift}} \sum_{e,d,t} y_{edt}$$

$$\begin{aligned}
\text{subject to} \quad \mathsf{re}_{dt} - \textstyle\sum_e x_{edt} &= \delta^-_{dt} - \delta^+_{dt} && \text{for all } d, t, && (1a)\\
x_{edt} &= 0 && \text{for all } e, d, \text{ and } t < \underline{\mathsf{av}}_{ed} \text{ or } t > \overline{\mathsf{av}}_{ed}, && (1b)\\
x_{edt} - x_{ed,t-1} &\le y_{edt} && \text{for all } e, d, t, && (1c)\\
x_{edt} - x_{ed,t+1} &\le z_{edt} && \text{for all } e, d, t, && (1d)\\
\textstyle\sum_t y_{edt} &\le 1 && \text{for all } e, d, && (1e)\\
\textstyle\sum_t z_{edt} &\le 1 && \text{for all } e, d, && (1f)\\
\textstyle\sum_{d,t} y_{edt} &\le 5 && \text{for all } e, && (1g)\\
\textstyle\sum_t (t+1) z_{edt} - \sum_t t y_{edt} &= \lambda_{ed} && \text{for all } e, d, && (1h)\\
8 \le \lambda_{ed} &\le 16 && \text{for all } e, d, && (1i)\\
\lambda_{ed} &\le 6 && \text{for all } d \text{ and } e \in E_{\min}, && (1j)\\
\textstyle\sum_{d,t} x_{edt} &= 40 && \text{for all } d \text{ and } e \in E_{\text{full}}, && (1k)\\
\textstyle\sum_{d,t} x_{edt} &\ge 25 && \text{for all } d \text{ and } e \in E_{\text{part}}. && (1l)
\end{aligned}$$

The objective function is straightforward: the value of a schedule is a weighted sum of the total number of shifts and the deviations of scheduled staff numbers from the requirements; (1a) calculates these deviations from the schedule information contained in x_{edt}. Constraints (1b) through (1l) enforce the feasibility conditions (*i*) through (*v*) from section 2 in the following manner. (1b) requires that each person is assigned work only at their available times—which is (*i*). Constraints (1c) and (1d) provide the connection between x and y, z variables, saying that when x switches from 0 to 1 or vice versa in consecutive time periods, there has to be a beginning or an end of a shift, respectively; for the constraints to be correctly specified in the first and last time periods, we need to define constants $x_{ed0} = x_{ed,|T|+1} = 0$ for all d, e. (1e) and (1f) further require that each person starts and finishes a shift at most once per day—a one-shift-per-day requirement, which completes (*ii*). (1g) is a direct translation of (*iii*). (1h) calculates

the length of a shift (λ) from its beginning and end (y, z), and (1i), (1j) express the requirement regarding shift lengths in (iv). Finally, (1k) and (1l) together give (v).

Model 2. In Model 2, the main assignment variables do not assign employees to individual *time periods* as in Model 1, but to complete *shifts* instead. Obviously, the number of time periods per day and condition (iv) uniquely define the set of all possible shifts for each day, S. From the available times $\underline{\mathsf{av}}_{ed}, \overline{\mathsf{av}}_{ed}$, one can establish whether a particular shift can be assigned to a given employee on a given day—observing conditions (i) and (iv). Model 2 works with this sort of information, which has to be processed into the following parameters prior to solving the model:

$\mathsf{co}_{st} = 1$ if shift s **co**vers time period t.
$\mathsf{av}_{eds} = 1$ if employee e is **av**ailable for shift s on day d.
$\mathsf{le}_s =$ **le**ngth of shift s.

Note that the task of extracting these parameters from a data source such as the spreadsheet in Figure 1 is very tedious even for small-scale problems, if it is to be carried out manually. For instance, in our case study with 40 employees, 16 time periods and 7 working days, the array with co_{st} parameters has 6,336 entries and the array with av_{eds} has 52,920. Therefore, data preparation has to be automated somehow (for our case study, we programmed several VBA procedures in *MS Excel*).

Model 2 uses the same variables $\delta^-_{dt}, \delta^+_{dt}$ as Model 1; besides these, there is only one more type of binary variables: $x_{eds} = 1$ if employee e is assigned to shift s on day d. The formulation of Model 2 is given below; we use a similar convention regarding summation and iteration indices as with Model 1. The objective function is analogous to that of Model 1, as is the first constraint. Constraint (2b) enforces conditions (i) and (iv); (2c) and (2d) correspond directly to conditions (ii) and (iii) respectively, and constraints (2e), (2f) and (2g) together give (v):

$$\text{minimize} \quad \mathsf{c}^{\mathsf{under}} \textstyle\sum_{d,t} \delta^-_{dt} + \mathsf{c}^{\mathsf{over}} \sum_{d,t} \delta^+_{dt} + \mathsf{c}^{\mathsf{shift}} \sum_{p,s,d} x_{eds}$$

$$\begin{aligned}
\text{subject to} \quad \mathsf{re}_{dt} - \textstyle\sum_{s,e} \mathsf{co}_{st} x_{eds} &= \delta^-_{dt} - \delta^+_{dt} && \text{for all } d,t, && \text{(2a)}\\
x_{eds} &\le \mathsf{av}_{eds} && \text{for all } e,s,d, && \text{(2b)}\\
\textstyle\sum_{s} x_{eds} &\le 1 && \text{for all } e,d, && \text{(2c)}\\
\textstyle\sum_{s,d} x_{eds} &\le 5 && \text{for all } e, && \text{(2d)}\\
\textstyle\sum_{s,d} \mathsf{le}_s x_{eds} &\le 80 && \text{for all } e, && \text{(2e)}\\
\textstyle\sum_{s,d} \mathsf{le}_s x_{eds} &= 80 && \text{for all } e \in E_{\text{full}}, && \text{(2f)}\\
\textstyle\sum_{s,d} \mathsf{le}_s x_{eds} &\ge 50 && \text{for all } e \in E_{\text{part}}. && \text{(2g)}
\end{aligned}$$

Computational efficiency. Both models are of the MILP type with thousands of binary variables even for small-scale problems—which poses a potential threat to computational tractability; in Model 1, the number of binary variables is $3|E|\cdot|D|\cdot|T|$, and in Model 2 it is $|E|\cdot|D|\cdot|S|$. The latter number is typically greater, though the precise comparison depends on the number of time periods and the rules regarding acceptable shift lengths. In our case study, there were 26,880 and 52,920 binary variables in Model 1 and 2, respectively. On the other hand, Model 2 typically has fewer constraints, and more importantly, these constraints have a much simpler structure than those of Model 1. This simple structure seems to be effectively handled both in the pre-proccessing routines of today's MILP solvers and in the actual solving procedures. As a result, Model 2 outperformed Model 1 for all of our problem instances. To solve both models, we used the *Lingo 12* optimization software with its built-in solvers, installed on a PC with Intel dual dore 2.2 GHz processor and 2 GB RAM. Primarily, we worked with the empiric data from our case study (40 employees, 7 working days, 32 time periods); apart from that, we used several other test datasets with between 10 and 80 employees to analyze the effect of a varying problem size. In most cases, Model 2 was solved within 5 minutes; in a few isolated cases, the solver found a near-optimal solution within the first 5 minutes, and the optimal solution followed in another 5 to 10 minutes. With Model 1, the situation was hardly as optimistic. In terms of minutes, optimal solutions were found for none but the smallest problems (10 to 15 employees). For cca 15–25 employees, the optimal shedule was found in several hours' time, making it suitable for applications where the calculation can be carried out overnight. For larger problems, the solver was mostly unable to reach the optimum within first 10 hours; however, it typically did find a feasible schedule that was within 1–2% of the objective bound; in practical applications, these solutions can be used without too much loss of schedule quality.

5 Empiric vs. optimal schedules

In order to evaluate the scheduling efficiency in our restaurant, we compared the empiric and MILP-obtained solutions for a ten-week period in the winter and spring of 2012. On average, the empiric objective function was 3–5% above the optimal one. There are three important notes on this comparison. Firstly, for the reasons stated in §3, we only focused on day shifts and non-managers. This simplification probably improves the relative performance of empiric schedules; we believe that if we included long-term scheduling of night shifts and managers, the gap between empiric and optimal schedules would grew wider. Secondly, due to the presence of a large number of temporary jobs with loose schedule restrictions, the problem was not overly constrained, especially on working days, which again allowed the shedule managers to find quite good solutions by hand. We expect that in other settings, the empiric schedules would fall futher below the MILP-obtained ones. Thirdly, it should be noted that the objective function in MILP models does not capture all the pros and cons of various alternative schedules. For instance, it does not account for the objectives listed under (*d*) in §3; in empiric schedules, these criteria are typically reflected, at least to a certain extent. Improving on this is a task for our future work.

6 Conclusions

The major aim of this paper was to show that OR methods are under-exploited by Czech businesses—even such businesses as the fast food chains, which are mentioned as those setting an example of operations efficiency. In order to show this, we focused on a particular fast food restaurant's crew scheduling; here, staff is scheduled manually, which both incurs unnecessary scheduling cost (in our case study, scheduling effort amounts to nearly one man-day a week) and results in a sub-optimal schedule. Optimal schedules for this restaurant were found using MILP. We presented two MILP formulations: while Model 2 was adapted from the "classical" scheduling MILP models, Model 1 was our invention. Although the latter is computationally less efficient, it requires no additional data processing, and thus is easily implemented; we believe that it might find its use in small businesses. Finally, as mentioned in §5, future research is needed to incorporate additional criteria in the MILP models (such as grouping of work days, fairness etc.) and test the computational tractability of these extended models.

Acknowledgements

This research was supported by the Grant Agency of the Czech Republic, project no. GA402/12/P241.

References

[1] Brusco, M. J., Jacobs, L. W.: A simulated annealing approach to the solution of flexible labour scheduling problems. *The Journal of the Operational Research Society*, **44**(12) (1993), 1191–1200.

[2] Burns R. N., Koop, G. J.: A Modular Approach to Optimal Multiple-Shift Manpower Scheduling, *Operations Research*, **35**(1) (1987), 100–110.

[3] Card, D., Krueger, A. B.: Minimum Wages and Employment: A Case Study of the Fast-Food Industry in New Jersey and Pennsylvania: Reply, *The American Economic Review*, **90**(5) (2000), 1397–1420.

[4] Dantzig, G.: A comment on Edie's traffic in toll booths, *Operations Research*, **2** (1954), 339–341.

[5] Ernst, A. T., Jiang, M., Krishnamoorthy, M., Sier, D.: Staff scheduling and rostering: A review, *European Journal of Operational Research*, **153** (2004), 3–27.

[6] Glover, F., McMillan, C.: The general employee scheduling problem: An integration of MS and AI. *Computers & Operations Research*, **13**(5) (1986), 563–573.

[7] Love R. R., Hoey J. M.: Management Science Improves Fast-Food Operations, *Interfaces*, **20**(2) (1990), 21–29.

[8] Tanomaru, J.: Staff scheduling by a genetic algorithm with heuristic operators. In: *Proceedings of the 1995 IEEE International Conference on Evolutionary Computation*, (1995), 456—461.