Scheduling agents using forecast call arrivals at Hydro-Québec's call centers

Marie Pelleau[1], Louis-Martin Rousseau[2], Pierre L'Ecuyer[1],
Walid Zegal[3], Louis Delorme[3]

[1] Université de Montréal, Montreal, Canada
me.pelleau@umontreal.ca, lecuyer@iro.umontreal.ca
[2] Polytechnique Montréal, Montreal, Canada, louis-martin.rousseau@polymtl.ca
[3] Institut de recherche d'Hydro-Québec, Montreal, Canada, {zegal.walid,
delorme.louis}@ireq.ca

Abstract. The call center managers at Hydro-Québec (HQ) need to deliver both low operating costs and high service quality. Their task is especially difficult because they need to handle a large workforce (more than 500 employees) while satisfying an incoming demand that is typically both time-varying and uncertain. The current techniques for determining the schedule of each employee according to the forecast call volumes at HQ are often unreliable, and there is a need for more accurate methods. In this paper, we address the concerns of the call center managers at HQ by modeling the problem of multi-activity shift scheduling. This model has been implemented and tested using real-life call center data provided by HQ. The main contribution of this paper is to demonstrate that a constraint programming (CP) model with regular language encoding can solve large problems in an industrial context. Furthermore, we show that our CP-based formulation has considerably better performance than a well-known commercial software package.

1 Introduction

The management of call center operations at Hydro-Québec (HQ) is a highly complicated task that often involves balancing contradictory objectives. Managers need to achieve simultaneously high levels of service quality and operational efficiency. The service quality is typically measured by key target performance metrics such as the average caller waiting time, i.e., the *délai moyen de réponse* (DMR). The daily target DMR is 120 seconds. The operational efficiency is typically measured by the proportion of time that agents are busy handling calls.

It can readily be seen that high levels of service quality are associated with low levels of operational efficiency, and vice versa. It is challenging to achieve the right balance. First, there is the problem of determining the appropriate *staffing level*, weeks or even months in advance, based on long-term forecasts of future incoming demand (i.e., future call volumes); this demand is typically both time-varying and random. Second, there is the problem of scheduling (and rescheduling) the available pool of agents based on updated forecasts, typically

made several days or weeks in advance, which is a problem of *resource deployment*. Finally, short-term decisions must be made, such as the routing of incoming calls in real-time to available agents or the mobilizing of agents at short notice because of unforeseen fluctuations in the incoming demand. At HQ, additional real-time control involves moving agents between the *front office* (where they answer calls) and the *back office* (where they perform other tasks such as paperwork). These decisions are based on short-term forecasts, updated one day or a few hours in advance.

In this paper, we focus on the *resource deployment* problem. Given a staffing level, we wish to specify which activity (such as taking calls, responding to emails, or working in the back office) each worker should perform in each period of the day. The creation of such a detailed schedule is generally referred to as the *multi-activity assignment problem* and is much more difficult than the traditional mono-activity version. The need to specify the occupation of an employee in each time period drastically increases the number of possible work shifts, which makes classical formulations (such as set covering [4]) intractable in this context.

In recent years, researchers have investigated using formal languages to solve complex scheduling problems where employees need to perform several tasks during a shift. These approaches can be combined with MIP (mixed integer programming) formulations and then solved directly, through column generation [5, 11, 2], via metaheuristics such as large neighborhood search [10] or tabu search [3], via a hybrid MIP/CP approach [12], or directly in CP through lazy clause generation [7].

As in [8, 6] we solve the activity assignment problem, which involves shift scheduling where the shift positions and breaks are fixed. The main contribution of this paper is to demonstrate that a constraint programming (CP) model with regular language encoding can solve large problems in an industrial context. In the data provided by HQ, there are 40 periods of 15 minutes covering the period from 8 a.m. to 6 p.m. There are between 129 and 142 activities, and from 369 to 544 employees to schedule. The problem is significantly larger than the test cases used in the existing literature on multi-activity scheduling. Furthermore, we show that our CP-based formulation has considerably better performance than a well-known commercial software package.

This paper is organized as follows. Section 2 introduces the problem and the main limitations of the current tool used by HQ. In Section 3, we present the model we used to address these limitations. Section 4 provides concluding remarks.

2 Problem Description

We consider the problem of designing work schedules for the agents. Given a day divided into periods, a set of employees, and a staffing requirement, we must select which activities should be performed by each employee during each period of the day, in order to satisfy the demand.

2.1 Problem characteristics

We now detail the different characteristics of the problem and give some of the notation.

Activities. In HQ, there are more than 100 different activities. Let the set of all the activities be A. There are two types of activities. The *nonproductive* activities are the breaks. We denote by A_{rest} the set of nonproductive activities. The *productive* activities are all the activities that are not in A_{rest}. The set of such activities is denoted by A_{prod}. And may be splited into three disjoints sets: the phone activities A_{phone} (taking the calls), the back-office activities A_{office} (doing paperwork), and the web activities A_{web} (responding to emails). We must distinguish between the different activities to formalize some of the concerns of the call center managers; see Section 2.2.

Periods. The call arrival process is uncertain and time-varying. In order to approximate it, the day is divided into m periods of 15 minutes. We retained this time division in the scheduling process. We define $T = \{1, \ldots, m\}$ to be the set of periods in a day.

Employees. Let E be the set of employees. Each employee has a set of skills $S_e \subseteq A$. For each employee $e \in E$ in each period $t \in T$, we must determine which activity should be performed. Let $x_{e,t} \in S_e$ be the activity performed by employee e in period t.

Staffing and Deviation. The staffing requirement indicates the desired number of staff for each activity in each period of the day; it is calculated beforehand. We denote by $d_{a,t}$ the staffing requirement for activity $a \in A$ in period $t \in T$. The goal is to ensure that the number of employees for each activity in each period of the day is greater than or equal to the requirement. However, it may not be possible to provide a schedule that fully satisfies the demand. In this context, the aim is to minimize the sum of the weighted staff shortages over all activities and periods, with weights that depend on both activities and time. This is referred to as *deviation* or *undercover* and denoted by $u_{a,t}$ with $a \in A$ and $t \in T$.

Thus, given a set of skills and a staffing requirement, building an optimal schedule corresponds to assigning activities to each employee for every period of the day. Each assigned employee must have the necessary skills, and the goal is to minimize the total deviation.

2.2 HQ's solution

HQ currently applies a widely used commercial scheduling software. This system is able to compute the staffing requirement, and given a staffing requirement it

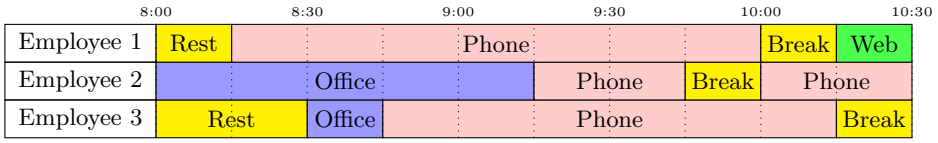


Fig. 1. Example of HQ schedule for three employees between 8 and 10:30

designs a schedule that satisfies the demand. Figure 1 shows an example of a schedule for three employees between 8:00 and 10:30. However, the current tool does not consider some of the following critical aspects of the management of HQ's call centers.

Priority management. Some activities have priority over others. For instance the activity of answering *failure* call type has priority over other call types. Moreover, any phone activity has priority over office and web activities. One may also want to prioritize certain times of the day, such as lunchtime when many employees are unavailable. The tool currently used at HQ does not allow the prioritization of activities, resources, and calls.

Activity transitions. The current tool is unable to efficiently manage sequences of different activities. It computes solutions in which employees must switch from one activity to another after only a short interval. For example, in Figure 1, Employee 3 has just 15 minutes of office work before switching to phone activity. However, HQ's managers wish to establish a minimum duration of one hour for each activity type. The curent tool cannot enforce this rule, so the schedules must be corrected manually.

Multi-skill management. Multi-skill management is necessary only for the phone activities. Given the stochastic context of call arrivals, it is desirable to assign an agent to a set of phone activities during the same period. However, the current tool is unable to select a subset of activities and assigns an employee to all the phone activities in his or her set of skills. In practice, the multi-skill assignment is performed using an internal simulator. This simulator is part of the software and is not documented. We do not know if it is stochastic or deterministic, or how the call arrival process is modeled. Furthermore, we do not have access to the results of the simulator. It generates a distribution of skills per agent per period. We define the *phone activity distribution ratio* to be the percentage of time allocated to a specific call type for an agent who is assigned to phone activities for a given period. This distribution is important because the good coverage of calls by agents relies on it. HQ thus uses the ratio below that is proportional to each activity's demand to evaluate the quality of a schedule.

Proportional ratio for multi-skill management. This ratio is designed to satisfy the conditions of the call center by taking into account the demand for a period. Let $e \in E$ be an employee, and $S_e \subseteq A$ his or her set of skills. Let $S_e^{phone} \subseteq S_e$ be the phone-related skills and $d_{a,t}$ the demand for activity $a \in A$ in period $t \in T$. If employee e is assigned to phone activities during period t, the ratio for each phone activity $a \in S_e^{phone}$ is

$$r_{e,a,t} = \frac{d_{a,t}}{\sum\limits_{a' \in S_e^{phone}} d_{a',t}}.$$

For all the other activities, this ratio is equal to 1. This ratio can be seen as a realistic assessment of the current conditions in HQ's call centers.

In previous work, HQ has implemented a MIP model using CPLEX. However, for large instances, this method sometimes returns an out-of-memory error. To address the limitations introduced above, and to reduce the large number of variables, we decided to use CP.

3 Constraint programming formulation

3.1 Proposed solutions to address the limitations

We address the issues of priority management and sequence limitations. For multi-skill management we currently use the method applied in HQ's existing tool: we assign an employee to all the phone activities in his or her set of skills, and we compute the deviation using the ratio introduced in the previous section.

Priority Management. Assigning a cost $C_{a,t}$ to each activity $a \in A$ and each period $t \in T$ allows us to prioritize some activities and periods over others.

Activity Transition. As part of the scheduling process, we propose to use a regular language to model the transition rule between the different types of activities. This rule states that an employee must perform productive activities of the same type (phone, office, or web) for a fixed duration (e.g., 1 hour) before switching to another activity. The rule is based not on activities but rather on families of activities. In addition, the rule must be validated for each employee $e \in E$. To model this sequence rule, we use an automaton and state that, for an employee, the word formed by the activities performed during the day must be recognized by the automaton.

Let $\Sigma = \{n, p, o, w\}$ be the alphabet of the automaton, where $n \in A_{\text{rest}}$, $p \in A_{\text{phone}}$, $o \in A_{\text{office}}$, and $w \in A_{\text{web}}$ belong respectively to the set of nonproductive, phone, office, and web activities. Let $Q = \{0, ..., 9\}$ be the set of states. The automaton defined on (Σ, Q) is given in Figure 2. Starting from the initial state 0, the automaton ensures that if an employee is assigned to a phone activity (state 1), an office activity (state 4), or a web activity (state 7) for a period of

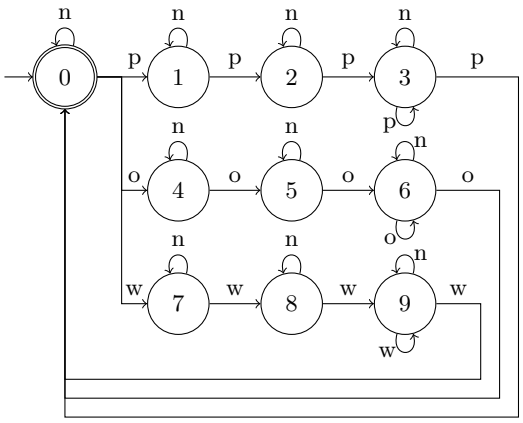


Fig. 2. Automaton for the activity transition rule

15 minutes, then it must perform an activity from that category for at least one hour. Breaks can occur during or after this block of activities. Moreover, the fact that state 0 is accepting ensures that even the last block of activities lasts at least one hour.

This automaton is represented by its transition table and modeled using a *Table* constraint. For each employee e and each period t, $q_{e,t} \in Q$ gives the state in the automaton.

3.2 Model

This model is a simplified version because all the breaks are fixed. We used the schedules designed by HQ's software and fixed the breaks at the same periods.

Variables

$x_{e,t} \in S_e$	Activity performed by employee e during period t
$u_{a,t} \in \mathbb{N}$	Shortage of employees performing activity a during period t
$q_{e,t} \in Q$	State in the automaton

Constraints

$$\min \sum_{a \in A_{\text{prod}}} \sum_{t \in T} C_{a,t} \times u_{a,t} \tag{1}$$

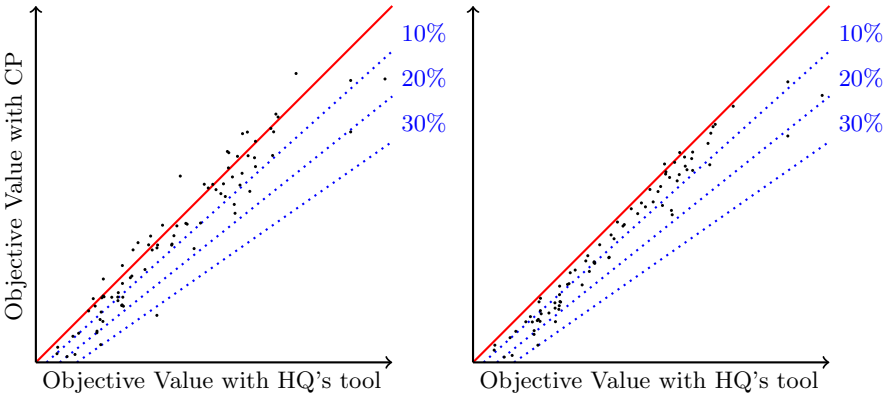
s.t.

$$\sum_{e \in E} ((x_{e,t} == a) \times r_{e,t,a}) + u_{a,t} \geq d_{a,t}, \qquad \forall a \in A_{\text{prod}}, t \in T \tag{2}$$

$$\text{Table}((q_{e,t}, x_{e,t}, q_{e,t+1}), \text{automaton}) \qquad \forall e \in E, t \in \{1, \ldots, m-1\} \tag{3}$$

$$q_{e,1} = 0 \qquad \forall e \in E \tag{4}$$

$$q_{e,m} = 0 \qquad \forall e \in E \tag{5}$$



(a) Impact Based Search (b) Phone First

Fig. 3. Comparison of the CP results and the HQ schedules.

3.3 Implementation

We implemented this model using the Or-Tools[9] library for Java. The main reason for using Java is to be able in future work to communicate directly with the call-center simulator [1]. In this version we used pure CP, with a timeout of 5 minutes. We try several strategies to select the next branching variable during the search. We report here the results for the classical *Impact Based Search* and a dedicated search strategy we call *Phone First*, which consist of first assigning employees that can perform phone activities and focus on office, web, and rest activities afterwards.

We tested our implementation on daily data for February to May 2011. In these instances, there are 40 periods of 15 minutes corresponding to 8 a.m. to 6 p.m. There are between 129 and 142 activities, of which 40 are phone activities, and from 369 to 544 employees to schedule. The experiments were run on a 1.7-GHz Intel Core i7.

A comparison of our solution and the HQ solution on 82 instances shows that we reduced the understaffing by an average of 5% with the *Impact Based Search*, and 9% with the *Phone First* strategy. Figure 3 compares the HQ's currently used tool with CP for both search strategies. Points below the bisector are instances on which CP outperforms the current solution, we can thus see that both search strategies perform quite well. *Phone First*, however, seems to be more robust as it degrades the solution in only two cases.

4 Conclusion

In this paper, we have investigated a large industrial multi-activity assignment problem. We took into account the concerns of the call center managers and proposed a solution that considers issues not handled by their current tool. We

implemented this model with Or-Tools and tested it on a significant number of real instances. Future work will involve solving the full multi-activity shift scheduling problem, where the shift positions and breaks are not fixed.

References

1. Eric Buist and Pierre L'Ecuyer. A java library for simulating contact centers. In *Proceedings of the 37th Conference on Winter Simulation*, pages 556–565, 2005.
2. Marie-Claude Côté, Bernard Gendron, and Louis-Martin Rousseau. Grammar-based column generation for personalized multi-activity shift scheduling. *INFORMS Journal on Computing*, 25(4):461–474, 2013.
3. Sana Dahmen and Monia Rekik. Solving multi-activity personalized shift scheduling problems with a hybrid heuristic. Technical report, Faculté des sciences de l'administration, Université Laval, 2012.
4. George B. Dantzig. A comment on edie's "traffic delays at toll booths". *Journal of the Operations Research Society of America*, 2(3):339–341, 1954.
5. Sophie Demassey, Gilles Pesant, and Louis-Martin Rousseau. A cost-regular based hybrid column generation approach. *Constraints*, 11(41):315–333, 2006.
6. Mahsa Elahipanah, Guy Desaulniers, and Ève Lacasse-Guay. A two-phase mathematical-programming heuristic for flexible assignment of activities and tasks to work shifts. *Journal of Scheduling*, 16(5):443–460, 2013.
7. Graeme Gange, Peter J. Stuckey, and Pascal van Hentenryck. Explaining propagators for edge-valued decision diagrams. In *Proceedings of the 19th International Conference on Principles and Practice of Constraint ProgrammingPrinciples and Practice of Constraint Programming*, pages 340–355, 2013.
8. Quentin Lequy, Mathieu Bouchard, Guy Desaulniers, François Soumis, and Beyime Tachefine. Assigning multiple activities to work shifts. *Journal of Scheduling*, 15(2):239–251, 2012.
9. Google OR-Tools. https://code.google.com/p/or-tools/.
10. Claude-Guy Quimper and Louis-Martin Rousseau. A large neighbourhood search approach to the multi-activity shift scheduling problem. *Journal of Heuristics*, 16(3):373–392, 2010.
11. Marìa I. Restrepo, Leonardo Lozano, and Andrés L. Medaglia. Constrained network-based column generation for the multi-activity shift scheduling problem. *International Journal of Production Economics*, 140(1):466–472, 2012.
12. Domenico Salvagnin and Toby Walsh. A hybrid mip/cp approach for multi-activity shift scheduling. In *Proceedings of the 18th International Conference on Principles and Practice of Constraint Programming*, pages 633–646, 2012.