**Submitted to *Management Science***
**manuscript**

# Employee Call Timing Problem for On-Call Personnel Scheduling

**(Authors' names blinded for peer review)**

About 1.5% of the total working population in North America is involved in on-call employment. Such employment negatively affects employee lives due to irregular hours, no guaranteed work, and lack of flexibility. Research on optimizing traditional on-call scheduling systems to alleviate some issues is scant. We present a novel on-call personnel scheduling system and aim to optimize its operation with a dynamic, data-driven approach. Like traditional systems, it contacts on-call personnel in order of seniority to inform them about the availability of on-call work. However, flexibility in the system allows employees time to think and freely choose shifts available to them. It also allows senior employees to replace or bump junior employees from the current schedule if no other shift of their choice is available, provided certain conditions are met. The replacements, though allowed by the system, may cause employee dissatisfaction. The management seeks to quickly assign all available shifts to employees while ensuring minimum bumps. However, there is uncertainty associated with the time at which an employee would select a shift. The decision faced by the management is exactly when to make a call to an employee to avoid bumps. We call this problem the Employee Call Timing Problem and show that it is $\mathcal{NP}-complete$ even in the case of perfect information. Then, we design easy-to-implement policy approximations based on a threshold structure for the dynamic problem. These policies are tuned using offline solutions where all uncertainty is known beforehand. The offline solution-based policies are tested on real-world data and outperform the current policy used by our industrial partner.

*Key words*: on-call scheduling, personnel scheduling, computational complexity, operations research

## 1. Introduction

In the last few decades, Personnel Scheduling Problems (PSP) have received a lot of attention in operations research (Van den Bergh et al. (2013), Ernst et al. (2004), Van den Bergh et al. (2013), and Özder et al. (2020)). These problems were first introduced in Dantzig (1954) and Edie (1954) in the 1950s. PSPs today are diverse and frequently occur in retail, hospitality, healthcare, security guards, airlines, etc. Companies in these domains have the option of part-time contracts or flexible work hours and also want to consider employee preferences when creating work schedules. The relative importance of satisfying employee needs in staffing and scheduling decisions has grown as labor cost has become one of the major contributors to overall operating costs for companies.

Personnel scheduling for regular shifts considering a few weeks of planning horizon has been extensively studied. These problems aim to find optimal employee schedules subject to different operating constraints and objectives. However, scheduling environments are often subject to uncertainty, mainly originating from a demand point of view. Seasonal or weather-dependent fluctuations are examples that cause such demand uncertainty. In addition, resource availability is another source of uncertainty. Regular employees that have already been assigned to shifts may apply for a day off or become sick and thus be unable to perform the assigned shift. Due to the inherent uncertainty in the process, it is difficult to determine the right number of employees to serve customers beforehand. This can easily lead to under-staffing or over-staffing and thus may cause a drop in customer service levels, employee dissatisfaction, and an increase in operating costs.

Companies generally develop regular shift schedules a few weeks in advance to only cover a base demand. They rely on cost-effective options such as overtime or part-time employment to counter uncertainty. This paper focuses on *On-call* employment, a sub-class part-time work. Previous studies BLS (2017), Williams (2008), show that about 1.5% of total workers in North America are on-call workers. On-call work refers to a situation in which an employee is not assigned any specific hours of work in advance but is assigned specific times during which the employee commits to be accessible. In case of excess demand, outages, or absences during these times, the management can call and ask the employee to report for work. However, such simple and traditional on-call systems also suffer from many shortcomings. Work flexibility, irregular working hours, inadequate salaries, etc., are just some of the issues the employees face Golden (2015), Nicol et al. (2004). The irregular schedules also lead to many health issues relating to sleep, mental health, stress, etc. Most of these problems arise due to a lack of flexibility where employees do not have any say in forming on-call schedules since the management decides shifts at the last minute. Several suggestions have been made for developing strategies for on-call staffing in Olmstead et al. (2014). Our work explicitly focuses on a dynamic self-scheduling system that tackles this issue by allowing on-call employees to choose on-call shifts directly. Hence, employee preferences are directly considered in forming last-minute work schedules.

Personnel scheduling for employees under uncertainty is usually done in two steps. Since regular employees need to know their shifts before the demand is known (e.g. two weeks in advance), the shifts are scheduled using a forecast of the demand. Depending on the company legislation, recourse can be used in the second stage. We list a few papers applying this approach here. In Kim et al. (2015) the authors formulate a two-stage stochastic integer program with mixed-integer recourse. The here-and-now decision is to find initial staffing levels and schedules. The wait-and-see decision is to adjust these schedules at a time closer to the actual date of demand realization. Weekly decisions are made over the 12 weeks to adjust the planned schedules at the beginning of each week for that whole week. These decisions involve adding or canceling shifts. The added shifts can be scheduled among full-time, part-time, or casual labor. The authors of Bard et al. (2003) develop a two-stage integer program with recourse for the staff planning and scheduling problem. The first stage determines the number of full-time and part-time employees before demand is known. In the second stage, workers are assigned to specific shifts during the week when demand is revealed. The second stage also involves using overtime and casual/on-call labor. The contribution of Asgeirsson et al. (2005) addresses the problem of adjusting individual work assignments in an existing nurse schedule to account for daily demand fluctuations. Possible actions include using overtime, calling in nurses on their day off (on-call), using external resources, and pooling nurses. The problem is formulated as an integer program and solved within a rolling horizon framework that spans 24 hours. However, no previous studies consider employee preferences in the recourse actions involving part-time/on-call labor.

This paper proposes a new flexible and dynamic on-call shift scheduling system developed in collaboration with an industrial partner. Let us consider an employer with a large pool of on-call employees registered to do on-call jobs and that each employee in this pool possesses certain skills which make them eligible for a set of shifts. The management first decides on the number of shifts needed to be scheduled on a given day in the near future. This information is fed into an electronic call system, which then compiles the headcount of all eligible employees for the list of available shifts. An operating manager then activates the employee communication process in the system. The electronic call system then sends notifications/calls to employees about available shifts, tracks employee responses, and updates the shift schedule when an employee chooses a shift.

The electronic call system sends notifications to the employees on their phones in the order of seniority. The number of notifications sent is decided according to a preset policy. Once notifications are received, the employees can take their own time to respond to the system with a decision to accept an available shift. This delay in communication is allowed as the employee may be busy during the system call or may need some time to think or communicate with his/her family. It may also happen that the employee never responds. In this case, we can assume the employee does not want to work that day. The time between the point of communication from the system and employee response is referred to as *response delay*. When an employee responds and selects a shift, that shift is said to be occupied. As more employees respond and select shifts, shift occupancy increases. The system also allows a senior employee to select an eligible shift occupied by a junior employee provided the senior employee responds within a time interval called the *cutoff*. This leads to a schedule change wherein a previously scheduled employee is said to be *bumped*. The cutoff time is the maximum response delay allowed so that an employee can select any shift from the set of eligible shifts unless a more senior employee already occupies it. The cutoff time starts from the point when the first communication is made. If any employee calls after the cutoff; only unoccupied shifts can be selected, and the advantage of being senior is lost. Thus the system incentivizes the selection of shifts by offering employees who regularly accept shifts a higher seniority level which in return allows them a larger set of shifts to choose from. Bumped employees are informed and may choose from the remaining available shifts at the time of the bump. The shifts are scheduled among employees that occupy them at the point in time when all employees have responded or when the planning horizon is over.

The uncertainty involved in the system is that the response delay of an employee is not known in advance. If many employees are called too early in the horizon, this may cause several bumps in case some senior employees take more time to make their decision. A large number of bumps means continuous schedule changes for the more junior employees and disrupt their daily life. The employees may get frustrated with the system and leave it altogether. Calling too few to avoid bumps would mean not allowing the employees enough time to respond within the planning horizon, possibly yielding unassigned shifts.

The system thus needs to decide how many notifications/calls to send at any point in time, given the system's current state. From a management point of view, it is essential to have a calling policy that minimizes bumps and ensures complete shift occupancy. However, the presence of these two conflicting objectives makes the problem difficult. If one only wants to minimize total bumps, then an optimal policy is to call no one. Alternatively, if the management only wants to schedule all shifts, then calling everyone at once at the start is optimal. We refer to this problem where the decision is to find the best calling times as the Employee Call Timing Problem (ECTP).

The advantage of such a system is that it allows much more flexibility for the employer and the employees. On the one hand, the employer with a large employee pool can wait longer to schedule shifts. The electronic call system oversees the most tedious work on their behalf. On the other hand, the employees are free to choose shifts as per their availability or even reject all offered shifts. This increased employee involvement should improve employee satisfaction and reduce staff turnover. At the moment, our industrial partner schedules about 15000 employees for about 1.4 million shifts every year.

Our contributions are presented as follows: In Section 2, we describe this new and important problem faced in Personnel Scheduling that we call ECTP. We then show that even a simple and offline version of the ECTP with complete information is $\mathcal{NP}-complete$. In Section 3, we consider the stochastic and dynamic ECTP and develop policies based on deterministic offline solutions obtained with simulated data. In Section 4, we compare their performance with heuristic policy functions and demonstrate that offline solution-based policies outperform when uncertainty follows a Weibull distribution. We also compare all policies using real-world data from our industrial partner. Section 5 provides a summary and conclusion.

## 2. Problem Description and Computational Complexity

This section describes a simpler version of the ECTP and demonstrates that the problem is $\mathcal{NP}-complete$, even if in the case of perfect information. We first start with some notation in Section 2.1, while Section 2.2 describes the decision version of this problem and establishes certain preliminaries. Next, in Section 2.3, the reduction is defined from the Subset-Sum problem, which is already known to be $\mathcal{NP}-complete$. We develop an intuition as to why the reduction works in Section 2.4. Section 2.5 gives the structure of the optimal schedule for the ECTP obtained through the reduction and gives a proof that ECTP is also $\mathcal{NP}-complete$.

## 2.1. Notation

We define an offline version of the ECTP with the following assumptions and simplifications. Response delays for each employee are known at the start of the planning. All employee response times are shorter than the horizon. In reality, employee response time can be greater than the horizon, but since we are offline we can simply filter such employees. Hence, there is the same number of shifts to fill as employees. Employees are eligible for all available shifts and have the same preferences known at the start of planning. This also means that when responding, an employee $i$ will always get to select a shift that is at least ranked $i$ in the preferences. There is no cutoff time, senior employees can bump junior employees at any point in the planning. Table 1 defines the notations used for the problem. A call schedule is denoted by $S = [s_i, e_i]_{i \in \mathcal{E}}$. $S$ is a feasible call schedule for the

| **Sets** | | | |
|---|---|---|---|
| $\mathcal{E}$ | | | set of employees in the order of seniority with 1 being most senior |
| **Parameters** | | | |
| $M$ | | | number of employees |
| $R = [r_i]_{i \in \mathcal{E}}$ | | | set of response delays for employee $i$ |
| $H$ | | | planning horizon |
| **Variables** | | | |
| $s_i$ | $i \in \mathcal{E}$ | $s_i \geq 0$ | time of start of communication for employee $i$ |
| $e_i$ | $i \in \mathcal{E}$ | $e_i \geq 0$ | time of response for employee $i$ |
| $b_i$ | $i \in \mathcal{E}$ | $b_i \geq 0$ | number bumps caused by employee $i$ |

**Table 1 Model, Parameters, and Variables of the ECTP**

instance if all the employees are called in the order of seniority ($s_i \leq s_{i+1}$) and all of them respond within the horizon ($e_i \leq H$).

## 2.2. Preliminaries

Our main result in this section establishes the computational complexity of the ECTP. The decision version of the ECTP is as follows:

**INSTANCE** : Finite set R with duration $r_i \in Z^+$, for each $i \in \mathcal{E} = \{1,..,M\}$, a target horizon $H$ and a target number of bumps $B^*$ .

**QUESTION** : Does a feasible schedule $S$ exist with at most $B^*$ bumps for a given instance?

To prove that the ECTP is $\mathcal{NP}-complete$, we reduce from the Subset-Sum problem. Let us first define additional terminology that will be useful in the proof. Let ECTP($I$) denote the ECTP defined by instance $I$. Also, let $\mathcal{F}_{ECTP}(I)$ denote the set of all feasible solutions for ECTP($I$). Given any schedule $S$, the *makespan*, denoted by $C(S)$ is defined as the time required to schedule all shifts.

$$C(S) = \max_{i \in \mathcal{E}} \{e_i\}$$

The makespan is feasible if $C(S) \leq H$. For any two employees $i, i' \in \mathcal{E}$, $i \leq i'$ means employee $i$ is more senior than employee $i'$, and therefore the employee $i$ has to be compulsorily called before or simultaneously as employee $i'$. The difference between their response times $(e_i - e_{i'})$ will determine whether $i$ bumps $i'$. Also, let $B(S) = \sum_{i \in \mathcal{E}} b_i$ denote the total bumps suffered by the employees when following the call schedule $S$, where each employee $i$ induces $b_i$ bumps. A No Bump Schedule (NBS) is any schedule $S$ such that $B(S) = 0$, i.e., there cannot be two employees $i, i'$, and $i \leq i'$ such that $e_i > e_{i'}$. Any NBS if feasible for a given instance is also optimal. Let $C_0^*$ denote the minimum makespan of an NBS such that $B(S) = 0$. The following proposition gives an expression to calculate $C_0^*$ and its proof is given in Appendix A.

PROPOSITION 1. *The minimum makespan of a NBS is* $C_0^* = r_1 + \sum_{i=1}^{M-1} (r_{i+1} - r_i)^+$.

***Example:*** Consider an instance $I$ with $M = 6$, $H = 10$ and $R = [4, 1, 5, 3, 2, 5]$. Figure 1(a) gives an NBS for this instance. An NBS is feasible for ECTP($I$) if $H \geq C_0^*$. However, with $H = 10$, it is clear that an NBS is not feasible for this instance. Let $\mathcal{O}_{ECTP}(I) \subset \mathcal{F}_{ECTP}(I)$ denote the set of optimal schedules that minimize the bumps for instance $I$. There can be multiple optimal solutions for the same instance: Figure 1 (b) and (c) show two such schedules, $S_1^*, S_2^* \in \mathcal{O}_{ECTP}(I)$ for the instance. Introducing bumps in the schedule can save time to complete scheduling all shifts within the horizon. This is explained in Figure 1 (a) and (b). The time of the call to employee $E2$ in Figure 1 (a) is 3 and in (b) 2. Calling E2 early by one unit introduces a bump for employee E2, but now all the following employees can be called by the system one unit time earlier. Figure 1 (c) does the same but for employee E5.

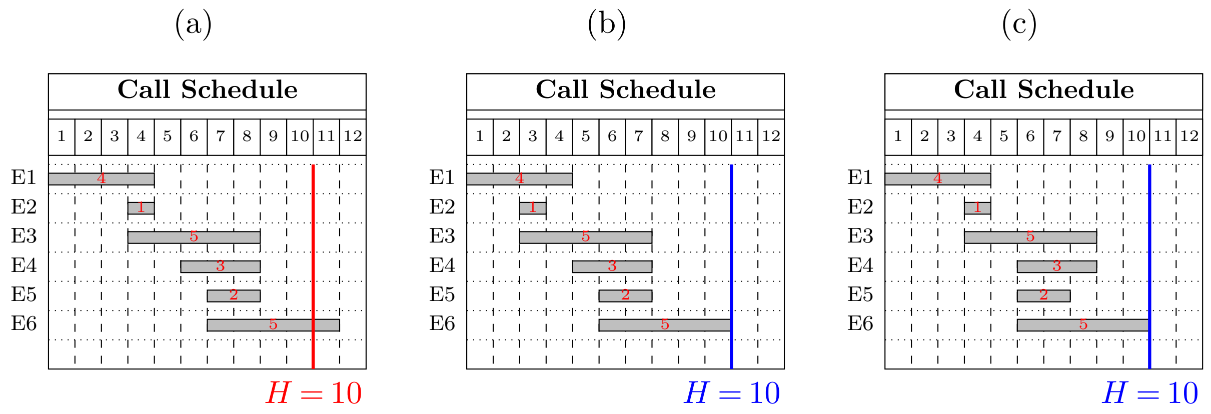


**Figure 1** (a) NBS : $C_0^* = 11$, (b) $S_1^*$ : $B(S_1^*) = 1$ and $C(S_1^*) = 10$, (c) $S_2^*$ : $B(S_2^*) = 1$ and $C(S_2^*) = 10$

## 2.3. Reduction

The SUBSET-SUM problem is defined as follows using Karp (1972), Garey et al. (1979).

**INSTANCE** : Finite set A, size $a_j \in Z^+$, for each $j \in A$ and a target positive integer $W$.

**QUESTION** : Is there a subset $A^* \subseteq A$ such that the sum of sizes in $A^*$ is exactly $W$ i.e. $\sum_{j\in A^*} a_j = W$?

Given $A = \{1, \ldots, N\}$, the sizes $[a_j]_{j\in A}$, and a target $W$, let $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$ denote the reduced instance. Let the total number of employees be $M = N + \sum_{j\in A} a_j + 1$. The letter $i$ is used to denote the seniority level of an employee. We define different sets of employees and their seniority as described in Table 2. There is one critical employee $i_k$ for each index of set $k \in A$. Also, any two critical employees $i_k, i_{k+1}$ are separated by a block of stable set employees $\mathcal{E}_k^S$ of size $a_k$. There are a total of $|A|$ blocks of stable employees. Finally, to complete we have a singleton set for the most junior employee.

| **Sets** | | |
|---|---|---|
| $\mathcal{E}$ | set of all employees in order of seniority, $= \{1, ...M\}$ | $\lvert\mathcal{E}\rvert = M$ |
| $\mathcal{E}^C$ | set of seniority indices of critical employees, $= \{i_k : i_k = k + \sum_{j=1}^{k-1} a_j, k \in A\}$ | $\lvert\mathcal{E}^C\rvert = N$ |
| $\mathcal{E}_k^S$ | $k^{th}$ block of stable employees, $= \{i : i_k < i \le i_k + a_k\}$ | $\lvert\mathcal{E}_k^S\rvert = a_k$ |
| $\mathcal{E}^S$ | set of seniority indices of stable employees, $= \cup_{k\in A}\mathcal{E}_k^S$ | $\lvert\mathcal{E}^S\rvert = \sum_{k\in A} a_k$ |
| $\mathcal{E}^L$ | singleton set for the last employee, $= \{M\}$ | $\lvert\mathcal{E}^L\rvert = 1$ |

**Table 2** Employee sets for the reduction

The response delay for each employee set is given by Equation (1). For any employee $i_k$ in the critical set, the response delay is simply the sum of all sizes till index $k$. For the stable employees in the $k^{th}$ block, the response delay is the same as the critical employee $i_{k-1}$. For the last employee, it is simply the sum of all sizes.

$$r_i = \begin{cases} \sum_{j=1}^{k} a_j & \text{if } i = i_k \in \mathcal{E}^C, k \in A, \\ r_{i_{k-1}} & \text{if } i \in \mathcal{E}^S, i_k < i < i_{k+1}, k \in A, \\ \sum_{j \in A} a_j & \text{if } i \in \mathcal{E}^L; \end{cases} \tag{1}$$

where $r_{i_0} = 0$.

The total planning horizon is set to $H = C_0^* - W$ where $C_0^* = 2\sum_{j\in A} a_j$ using Proposition 1. It is assumed that $W \leq \sum_{j\in A} a_j$, else it is polynomial to check that the subset sum is infeasible when $W > \sum_{j\in A} a_j$. One can easily verify that the above instance has size $r_i \in \mathbb{N}$, for each $i \in \mathcal{E}$ and a target positive integer $H \in \mathbb{N}$ and is achieved in polynomial operations. The following example illustrates the reduction.

***Example:*** Consider a Subset-Sum instance with $A = \{1,2,3\}$, sizes $a_1 = 1, a_2 = 4, a_3 = 7$ and with $W = 5$. As a consequence $|\mathcal{E}^C| = 3, |\mathcal{E}^S| = 12, |\mathcal{E}^L| = 1, M = 16$. The seniority indexes in the critical, stable, and last employee sets are defined below.

- $\mathcal{E}^C = \{1,3,8\}$
- $\mathcal{E}^S = \{2,4,5,6,7,9,10,11,12,13,14,15\}$
- $\mathcal{E}^L = \{16\}$

The response delay vector $R = [1,0,5,1,1,1,1,1,12,5,5,5,5,5,5,5,12]$. The horizon $H = C_0^* - W = 19$, where $C_0^* = 24$. Color codes are used to denote employees from their respective set on the $x$-axis using the following convention: $\mathcal{E}^C$, $\mathcal{E}^S$, $\mathcal{E}^L$ . The NBS and the optimal schedule $S^*$ are in Figure 2 (a) and (b). Let us define a special type of call schedule that will help us establish the proof.

Definition 1. Block Schedule $S(A')$: Given $A' \subseteq A$, a block schedule $S(A') = [s_i, e_i]_{i\in\mathcal{E}}$ for $ECTP([a_j]_{j\in\mathcal{A}}, W)$ is constructed as follows:

$$s_i = \begin{cases} 0 & \text{if } i = 1, \\ s_{i_k} + r_{i_k} - r_i & i = i_k + 1, \forall k \in A \backslash A', \\ s_{i-1} & \text{else ;} \end{cases} \tag{2}$$

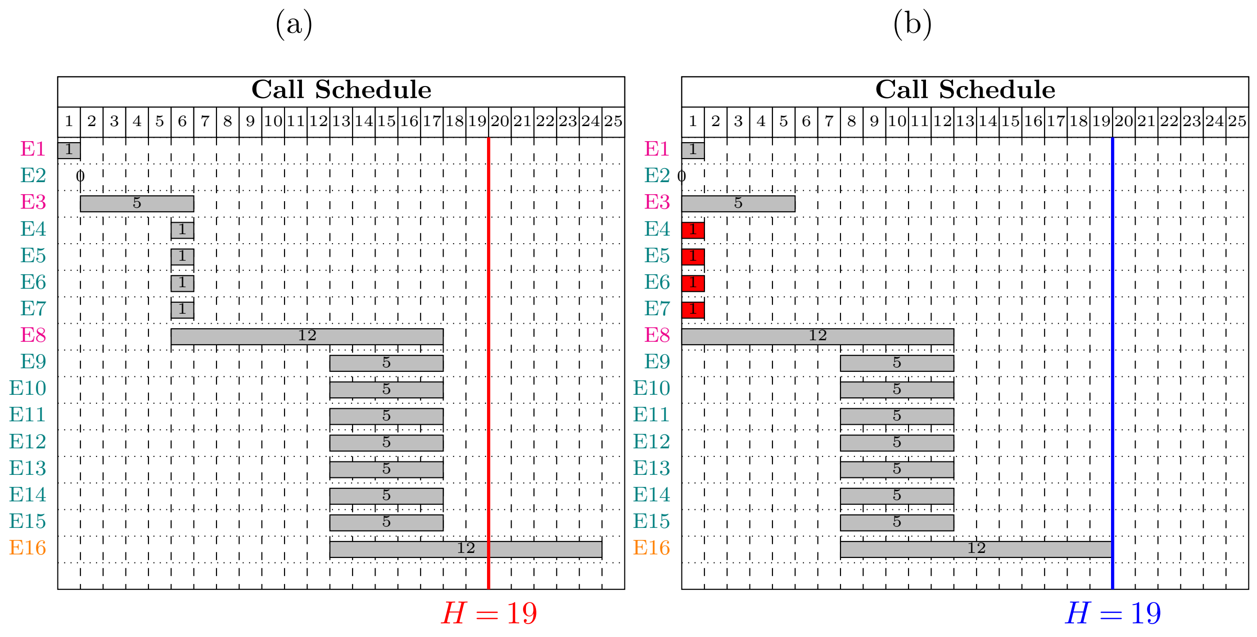


**Figure 2 (a) No Bump Schedule : $C_0^* = 24$ (b) S$^*$ : B(S$^*$) = 5 and $C(S^*) = 19$;**

One can easily verify that $\forall k \in A, r_{i_k} - r_i = r_{i_k} - r_{i_{k-1}} = a_k$ when $i = i_k + 1$ in Equation (2). Figure 2 (b) is an example of the block schedule $S(A')$ with $A' = \{1, 2\}$.

## 2.4. Intuition

The crux of the reduction lies in the way the response delays are defined. They ensure the following.

- No bumps take place within any two stable set employees because for any $i, i' \in \mathcal{E}^S$ such that $i < i'$ we have $r_i \leq r_{i'}$.
- No bumps take place between any two critical set employees, as we have $r_{i_k} \leq r_{i_{k'}}$ for any $i_k, i_{k'} \in \mathcal{E}^C$ such that $i_k < i_{k'}$.
- Any critical employee $i_k$ can either bump only a whole block of stable set employees $\mathcal{E}_k^S$ to create time savings or bump none. Alternately this means that the critical employee $i_k$ can introduce a total of $a_k$ bumps in the schedule. As a consequence, the total bumps $B(S)$ will be a sum of these sizes. The aim is now to introduce bumps so that $B(S)$ equals to $W$. For example, in Figure 2 (a), one can see that E3 corresponding to index 2, can only bump E4, E5, E6, and E7 corresponding to the size $a_2 = 4$. E3 cannot bump any employee junior to E7. It is also ensured that no stable employee bumps any other employee from the entire set.
- A block schedule is always feasible and optimal. The block schedule $S(A)$ which calls everyone at the start is always feasible. Furthermore, when the total bumps $B(S) = \sum_{k \in A'} b_{i_k}$,

where $A' \subset A$, the block schedule $S(A')$ gives the minimum makespan and hence is also optimal.

- Any schedule $S$ such that $B(S) < W$ is infeasible. If a critical employee $i_k$ bumps a block of stable set employee $\mathcal{E}_k^S$, then it also creates maximum time savings of $a_k$. Note that a block schedule will create time savings of exactly $a_k$ units for a bump by critical employee $i_k$. Thus, starting from the NBS schedule that has a makespan of $C_0^*$, we set $H = C_0^* - W$ to make certain that any feasible solution has at least $W$ bumps.

The goal is to find a feasible block schedule $S$ such that $B(S) = W$. When this happens, all the indexes $\{k_1, k_2, \ldots, k_Z\}$ such that critical employee $i_{k_u}$ causing bumps are part of the subset and we would have a YES answer to Subset-Sum problem.

## 2.5. Proof

The proof of all propositions, corollaries, lemmas, and theorems is given in Appendix A. Let $\mathcal{F}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$ and $\mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$ denote the feasible and optimal set of solution for the $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$.

Proposition 2. *The block schedule $S(A')$ has the following properties:*

1. $\forall k \in A, s_{i_k} = s_{i_k - 1}$.
2. $\forall i \in \mathcal{E}_k^S, \forall k \in A$, $e_i = e_{i_k}$ *or* $s_i = s_{i_k}$.
3. $s_{i_{k+1}} = s_{i_k} + \mathbf{1}_{k \notin A'} a_k$, *where* $\mathbf{1}$ *denotes indicator variable.*
4. $B(S(A')) = \sum_{j\in A'} a_j$
5. $C(S(A')) = C_0^* - \sum_{j\in A'} a_j$.

Properties 1 and 2 mean that the block schedule respects the seniority constraint of the problem. Given the start of any critical employee $i_k$, property 3 allows us to find the start time of the next critical employee $i_{k+1}$. Property 4 gives an equation to compute the total bumps suffered by the block schedule. Property 5 establishes the total makespan of the block schedule.

Proposition 3. *For any $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, $C = e_M$.*

Proposition 4. *$\forall k \in A$, $\forall l \in A$ employee $i_k$ cannot bump employee $i_l$.*

Proposition 5. *$\forall i \in \mathcal{E}^{\mathcal{S}}$, $\forall i' \in \mathcal{E}$ employee $i$ cannot bump employee $i'$.*

Proposition 6. *$\forall k \in A$, $\forall i \in \mathcal{E} : i \geq i_{k+1}$, employee $i_k$ cannot bump employee $i$.*

Corollary 1. *$\forall k \in A$, $\forall i \in \mathcal{E}$ if employee $i_k$ bumps employee $i$, then $i \in \mathcal{E}_k^S$.*

Proposition 3 defines the makespan of the schedule. Since the last employee $i_M$ has the maximum response delay, it is simple to show that end of its interaction $e_M$, marks the end of the schedule for the $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$. Proposition 4, Proposition 5, and Proposition 6 establish that it is only the critical set employees who cause the bumps. Moreover, any critical employee $i_k$ can only bump the immediately following $a_k$ employees in the order of seniority, all of whom belong to the stable set from Corollary 1.

Lemma 1. *$\forall S^* \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, $\forall k \in A, b_{i_k} = a_k$ or 0.*

Corollary 2. *$\forall S \in \mathcal{F}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, the total bumps $B(S) = \sum\limits_{k\in A} b_{i_k}$.*

Lemma 2. *$S(A) \in \mathcal{F}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$.*

Lemma 1 proves that any critical employee $i_k$ will bump the immediately following $a_k$ junior employees or bump no one. As a consequence of Lemma 1, Corollary 2 establishes the total bumps in the schedule. Next, we show that the block schedule $S(A)$ is always feasible for the $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$ in Lemma 2. Hence the reduced problem is always feasible.

Lemma 3. *$\exists A' \subseteq A$ such that $S(A') \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$.*

Corollary 3. *$\forall S^* \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, the optimal objective value $B(S^*) \geq W$.*

The final piece of the proof comes from Lemma 3. It means that given any horizon $W$, there exists some subset $A'$ and a corresponding block schedule $S(A')$ that is optimal. This is a very important result as it allows only concentrating on a solution space that comprises block schedules.

Theorem 1. *For $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, $B(S^*) = W \Leftrightarrow \exists A^* \subseteq A$ such that $\sum_{j\in A^*} a_j = W$.*

Theorem 2. *$ECTP(I)$ is* NP-complete.

Theorem 1 proves that for any $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, the number of bumps in the optimal schedule $B(S^*)$ is always equal to the target $W$ defined by the Subset-Sum if and only if the Subset-Sum problem is feasible.

## 3. Dynamic Employee Call Time Problem

In the real world, employee callback times are stochastic. That is the exact time an employee will take to respond is not known beforehand. However, we assume that the distribution of the callback times is available. The uncertainty creates an additional layer of difficulty, as there is a chance that employees may not respond at all. This can happen if some employee has a response delay greater than the horizon. In such cases, waiting for employees to respond to avoid bumps may lead to vacant shifts at the end of the planning horizon. We describe the dynamic stochastic version of the ECTP called as DECTP in Section 3.1 and also give a MIP formulation for the offline problem. In Section 3.2, we design approximate policy functions for the DECTP.

## 3.1. Problem Description - DECTP

All data parameters from the ECTP are valid for the DECTP. Additionally, a shorter bump cutoff is considered in this problem. The general assumption is that employees know this cutoff time and would respond accordingly. Hence, if an employee is busy when a system call is made, the employee would call back within the cutoff to retain the ability to bump junior employees. Also, we consider that there are more employees than available shifts, which allows some buffer in case some employees don't respond. Employee response delays $r_i$ are independent and identically distributed (*i.i.d.*) random variables with a common probability distribution $\mathbb{P}$. Employees are eligible for all available shifts and have the same preferences known at the start of planning.

In the offline problem, checking the number of shifts that will go vacant is easy. All employees that have a response delay smaller than the horizon can occupy a shift. Therefore, there will be some vacant shifts only if the total number of employees with a response delay smaller than the horizon is less than the number of available shifts. Then we can optimize the call schedule to minimize the bumps subject to this shift vacancy. However, with stochastic response times, there is always a possibility of having more empty shifts than the deterministic offline solution. This motivates modeling the problem as a weighted multiobjective problem to minimize bumps and shift vacancies. Let DECTP-O denote the offline counterpart of the DECTP. Additional attributes are defined in table Table 3.

The MIP formulation for DECTP-O is given below. We set $G$ to be very high so as to ensure complete shift occupancy.

$$(MIP_{DECTP}) \quad \min \quad \sum_{i\in\mathcal{E}} \sum_{j\in\mathcal{E}:i<j} y_{ij} + GJ \tag{3}$$

| **Parameters** | | |
|---|---|---|
| $\delta_{ij} = r_i - r_j$ | $i,j \in \mathcal{E}$ | responds delay difference between two employees $i,j$ |
| $L$ | | total number of available shifts |
| $D$ | | bump cutoff |
| $G$ | | cost of vacancy |
| **Variables** | | |
| $y_{ij}$ | $i,j \in \mathcal{E}$ | binary variable to indicate if employee $i$ bumps $j$ |
| $z_i$ | $i \in \mathcal{E}$ | binary variable equal to 0 if employee $i$ responds within the horizon else 1 |
| $J$ | | total number of unassigned shifts |

Table 3 Model, Parameters, and Variables of the DECTP

## Constraints

$$s_i \leq s_j \qquad \forall i,j \in \mathcal{E}, i < j \tag{4}$$

$$s_i + r_i \geq (H+1)z_i \qquad \forall i \in \mathcal{E} \tag{5}$$

$$s_i + r_i \leq H + r_i z_i \qquad \forall i \in \mathcal{E} \tag{6}$$

$$s_i - s_j + \delta_{ij} \leq \delta_{ij} y_{ij} + (H + r_i) y_{ij} \qquad \forall i,j \in \mathcal{E}, i < j, r_i \leq D, r_i \geq r_j \tag{7}$$

$$L - (N - \sum_{i \in \mathcal{E}} z_i) \leq J \tag{8}$$

$$y_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{E}, i < j \tag{9}$$

$$s_i \in \mathbb{R}^+ \qquad \forall i, \in \mathcal{E} \tag{10}$$

$$J \geq 0 \tag{11}$$

The constraint in (4) ensures employee seniority is respected. (5) sets $z_i$ to 0 if an employee responds before the horizon. (6) sets $z_i$ to 1 if an employee responds after the horizon. (7) counts the bumps for any pair of senior and junior employees. Note for a senior employee $i$ to be eligible to bump junior employee $j$, the response delay for $i$, $r_i \leq D$ and $r_i \geq r_j$. Finally, (8) counts the number of vacant shifts. The following constraints are added to the formulation.

$$s_{i+1} - s_i \leq H z_i \qquad \forall i \in \mathcal{E}, r_{i+1} \geq r_i \tag{12}$$

$$y_{ij} \leq 1 - z_i \qquad \forall i,j \in \mathcal{E}, i < j \tag{13}$$

$$y_{ij} \leq 1 - z_j \qquad \forall i,j \in \mathcal{E}, i < j \tag{14}$$

(12) specifically refers to case when $r_{i+1} \geq r_i$. Since the next employee, $i+1$ has a greater response time; there is no incentive to wait to call $i+1$ unless that employee is to be pushed out of the horizon. Hence (12) will ensure that this employee is either called at the same time or responds after the end of the horizon. (13) and (14) ensures that employee $i$ can only bump employee $j$ if both of them respond within the time horizon.

### 3.2. Policy Function Approximations

We first formally define a model of the DECTP. Let $\mathcal{K} = \{0, 1, \ldots, K\}$ be the set containing all decision moments within the planning horizon $H$ for some list of shifts denoted by $L$. A decision epoch occurs at fixed time intervals $\tau$ where $\tau * K = H$. At any period, $X_k = (x_n, x_s, x_r)$ gives the system's state. Here $x_n$ is the number of calls made until epoch $k$, and $x_s$ is a list of indicators for each employee with a value of 1 if that employee has responded and 0 otherwise. $x_r = [x_{ri}]_{i \in \mathcal{E}}$ keeps track of the bump cutoff and is used to know if an employee can bump others or not. For every decision moment $k$ in the planning horizon, the policy decides the number of eligible employees $a_k \geq 0$ to call next given the state $X_k$.

The state space size is huge, so finding the exact optimal policy is difficult. However, it is often the case that one has a very good idea of making a decision or desires a policy with a simple and easy-to-implement approximate structure. The search for a good policy can then be restricted within a smaller space. These approximate policies are called policy function approximation (PFA)  Powell (2021). A PFA is an analytical function mapping a state to an action without solving an embedded optimization problem. The most common and easy type of policy function approximation is some parametric model that returns a decision that generally depends on the problem's state. Lookup tables, Linear Decision rules, Monotone Threshold policies, and Nonlinear approximations are some examples of PFAs. The art is developing the policy structure that best fits the problem.

For the DECTP, we approximate the optimal policies by a monotone threshold structure. A policy is said to be monotone threshold-type when it chooses an action $a_k$ only up to a threshold $\Omega(X_k)$ at epoch $k$. In our case, the best action $a_k$ would be directly proportional to the residual $(\Omega_f(X_k) - \omega_f(X_k))$, where $\omega_f(X_k))$ is the current value of some feature or basis $f$ in the current state and $\Omega_f(X_k)$ is the best threshold value for the same.

The problem now reduces to designing a good set of features and finding their optimal threshold parameters. However, it is still hard to find the optimal parameters. We rely on an approximate procedure to estimate suitable values for these thresholds. In the following subsections, we describe some static and dynamic monotone threshold policies for the DECTP for a set of features $f \in \mathcal{F}$ and also an algorithm to search the best estimates for the thresholds based on offline solutions of the problem.

**3.2.1. Static Policies** - These represent simple static policies that are easy to use. These policies are said to be static because the threshold parameters do not change over time.

- **Call $\eta$ and Wait $w$ epochs** (CW($\eta, w$)) - A simple two-parameter policy would be to makes $\eta$ calls every $w$ epochs. Both $\eta$ and $w$ can be tuned for the best performance. This is a simple state-independent policy that does not take any information from the state into account and will keep calling people at regular intervals.
- **Employee Cutoff Buffer Policy** - (ECBP($\Phi$)) - This policy is defined by a buffer of size $\Phi$ for employees who have a strictly positive bump cutoff, that is they still have time to call back and bump another employee. The current buffer status is given by $\omega_\phi = \sum_{i \in \mathcal{E}} \mathbb{1}_{\{x_{ri} > 0\}}$, where $\mathbb{1}$ is the indicator variable. At any point in the horizon, the number of calls to be made $a = (\Phi - \omega_\phi)$. The buffer size is considered to be constant throughout the horizon. Hence, as a result, the action taken here will always be positive, since the number of employees with a strictly positive cutoff will never exceed $\Phi$. Our industrial partner uses this form of threshold policy to make calling decisions.

Since the threshold parameters do not change over time, we can do a simple brute-force search to find the best estimates of the parameters while minimizing the average cost over all simulation runs.

**3.2.2. Dynamic Offline Solution-based Policies** This class represents calling policies in which the threshold parameters change with time. Algorithm 1 describes a pseudo-code for the function that defines the thresholds $\Omega_f$ for a given feature $f \in \mathcal{F}$. The thresholds are obtained using optimal offline schedules assuming all employee callback times are known in advance. The function takes as input the instance parameters and the number of offline instances to be solved defined by $Z$. Lines 3 to 9 in the pseudo-code generate and solve an instance $I_i$. The obtained solution is then used to calculate the feature value $\omega_f^{ik}$ for each decision epoch $k$ of the instance number $i$. Lines 10 to 16 apply an aggregator function $q \in \mathcal{Q}$

to the feature values to calculate an approximate $\Omega_f$ across all instances for each decision epoch $k$. The function outputs estimates of threshold values denoted by $\bar{\Omega}_f^q = [\bar{\Omega}_f^{kq}]_{k\in\mathcal{K}}$.

**Algorithm 1:** An algorithm to estimate the threshold for all features

```
1  Function Compile($M, L, H, \mathbb{P}, D, Z, Q$):
       Input  : The number of employees $M$; the number of available shifts $L$; the
                planning period $H$; the employee response delay probability
                distribution $\mathbb{P}$ the cutoff $D$; Number of offline instances to be solved
                $Z$; Set of aggregator functions $Q$
       Output: The estimated threshold vector $\bar{\Omega}_f^q$
2      for $i \in \{1, \ldots, Z\}$ do
3          Generate Instance $I_i$;
4          Solve Instance $I_i$ to get optimal schedule $S_i$ using $MIP_{DECTP}$;
5          for $f \in \mathcal{F}$ do
6              Compute the value of each feature $\omega_f^i = [\omega_f^{ik}]_{k\in\mathcal{K}}$ from the offline solution
                $S_i$;
7          end
8      end
9      for $f \in \mathcal{F}$ do
10         for $q \in \mathcal{Q}$ do
11             for $k \in \mathcal{K}$ do
12                 $\bar{\Omega}_f^{kq} = q([w_f^{ik}]_{i\in\{1,\ldots,Z\}})$;
13             end
14         end
15     end
16     return $\bar{\Omega}_f^{kq}$
17 end
```

At any decision epoch $k$, the decision $a = \lfloor \bar{\Omega}_f^{kq} - \omega_f^{kq} \rceil$ which is the residual between the computed value of the target feature from the offline solutions and its value from the current state $X_k$. $\lfloor . \rceil$ denotes the nearest integer after rounding, The following types of policies are defined for three different target features.

- **Offline Call Policy** (OCP($q$)) - The target feature $f = x_n$, is the cumulative calls made at each decision epoch. Let $a = \lfloor \bar{\Omega}^{kq}_{x_n} - \omega^k_{x_n} \rceil$ at every decision epoch. This is a myopic policy that only depends on the time elapsed since the start of the calling process.
- **Offline Employee Cutoff Buffer Policy** (OECBP($q$))- The target feature is the current buffer size ($f = \phi$) of employees who have a positive bump cutoff. Thus the action is to call $a = \lfloor \bar{\Omega}^{kq}_{\Phi} - \omega^k_{\phi} \rceil$, where $\omega^k_{\phi} = \sum_{i \in \mathcal{E}} \mathbb{1}_{\{x_{ri} > 0\}}$. This policy uses state information about the number of employees who can still cause bumps.
- **Offline Cumulative Cutoff Time Policy** (OCCTP($q$)) - The target variable is the cumulative bump cutoff time in the buffer ($f = \psi$). The action taken is given by

$$a = \lfloor \frac{\bar{\Omega}^{kq}_{\psi} - \omega^k_{\psi}}{D} \rceil$$

where $\omega^k_{\psi} = \sum_{i \in \mathcal{E}} x_{ri}$. This policy is similar to OECBP but weights each employee with respect to their residual bump cutoff time left.

The main aim of using this approach is to identify dynamic yet a simple policy with threshold parameters that vary across time. In terms of literature, there have been approaches that rely on offline solutions to develop good online policies. Recently, Pham et al. (2021), proposed a prediction-based approach for online dynamic radiotherapy scheduling to schedule incoming patients under the current allocation of resources. Their approach is based on a regression model trained to recognize the links between patients' arrival patterns, and their ideal waiting time in optimal offline solutions where all future arrivals are known in advance. De Filippo et al. (2021), considers multi-stage optimization problems under uncertainty that involve distinct offline and online phases. The authors combine a two-stage offline approach to make strategic decisions with an online greedy heuristic and then propose multiple methods to tighten the offline/online integration, leading to significant quality improvements.

## 4. Experiments

We perform two sets of experiments to test our approach. The first experiment establishes a proof-of-concept using Weibull distribution for response delays. We study the variation in target features described in Section 3.2.2 obtained from the offline solutions for such distributions. The second experiment aims to see if the proposed methodology can adapt well to real-world data from our industrial partner. All experiments and models are implemented in Python, and GUROBI 9.5 is used to solve all instances of the offline optimization problem. A maximum solving time of 4 minutes is set for each instance. GUROBI finds the optimal solution quickly, within seconds for all instances but it may take longer to prove optimality.

## 4.1. Scenarios and Evaluation

All experiments are conducted for a fixed horizon length of $H = 6$ hours. The total number of available shifts is also fixed at $L = 50$. A set of $M$ employees are available to schedule these shifts and have the same preferences. Experiments are carried out with two different cutoff times of 2 and 3 hours. Also, we set $Q^{\alpha} = D$, where $Q^{\alpha}$ is the time within which $\alpha\%$ of employees would respond to the system. Generally, it is assumed that employees are aware of the cutoff, and one expects the employees who want to choose a shift would respond before it expires. We conduct experiments with three different values of $\alpha \in \{40\%, 60\%, 80\%\}$. An instance is characterized by the tuple $(M, L, H, \mathbb{P}, D)$, which represents the number of employees, unoccupied shifts, the planning period, the employee response delay probability distribution, and the cutoff respectively. Given $D$ and $\alpha$, the task is to find the appropriate parameters for distribution $\mathbb{P}$ so that $Q^{\alpha} = D$. This tuple is referred to as the scenario for an experiment. We develop a simulation model of the electronic call system to mimic its operation. This simulation model inputs the parameters for a desired scenario and the calling policy. The model queries the current state information into the calling policy to find the action to be taken at each decision epoch.

We use the following approach to validate our approach for the dynamic policies. First, 1000 train instances are solved offline to estimate thresholds for all aggregator functions to form dynamic policies. Second, these policies are then simulated on 250 validation instances for their performance. And third, we choose the best aggregator functions and evaluate them on 250 test instances. Doing so ensures that our chosen policy parameters can generalize well for unseen data points. For the heuristic policies, all possible combinations of parameters are tested. For example, in a CW policy, all combinations of $\eta, w$ from a set of possible values are simulated, and then the best parameter is evaluated on the test set.

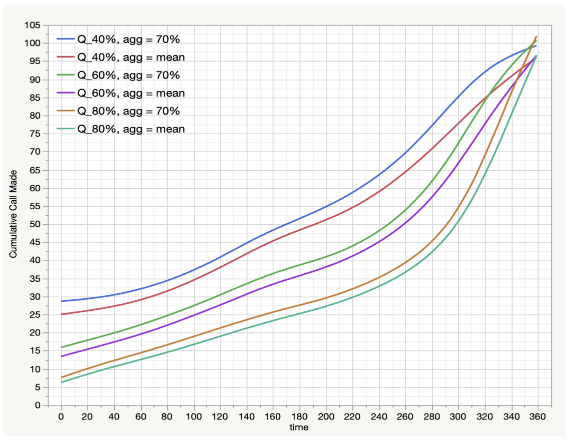
## 4.2. Offline Solution Analysis - Weibull Response Delays

The response delays are assumed to follow the Weibull distribution for each scenario. We hypothesize that most employees would generally respond quickly to system notifications in the real world. However, they still need a small delay to review the available shifts list and think about their decision. The Weibull distribution is flexible and can fit an extensive range of distribution shapes making for a good choice. The distribution parameters are set to meet the corresponding quantile for the response times.

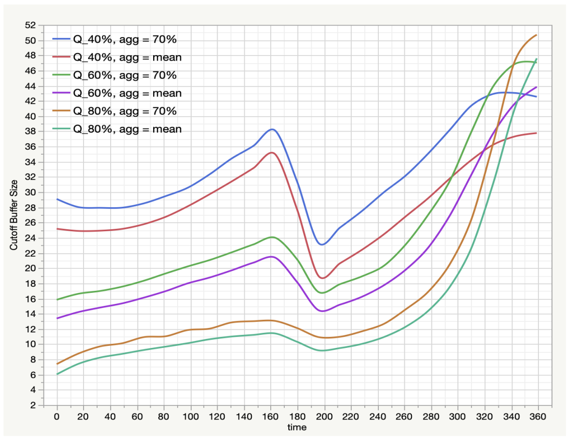
Algorithm 1 is used to estimate thresholds of all features for 1000 train instances with a total employee population of $M = 100$ for each of the six scenarios. We use descriptive statistics such as the "mean" and "quantiles" as aggregator functions for all features across all instances for each decision epoch. The quantile functions used were separated by 10%. Figure 3 shows the variation of each target feature with respect to time when $D = 3$ hours. Each smoothed curve refers to a value of the respective target feature for a set of $\alpha$ values and a descriptive statistic. Only the mean and the 70% quantile are plotted for clarity. Figure 8 given in Appendix B shows the same plots for $D = 2$ hours.

- In Figure 3a, the trend of how calls are made in the offline solutions is shown. Calls are made near-linearly for both aggregator functions from the start. However, the rate changes as the horizon nears and calls are made faster. This is expected as employees called in this period will not have sufficient time to bump others. In other words, employees taking a long time to respond are pushed out of the horizon to avoid bumps.
- In Figure 3b, the cutoff buffer feature sees an increasing trend in the target features from the start of the calling process. But at decision epoch $k = D$, there is a discontinuity in the curve as those employees called at the start who have not yet responded, exhaust their cutoff time. The drop at the discontinuity increases with a decrease in $\alpha$ since more employees are called at the start of the horizon. At this point, the buffer size stabilizes for a certain period. There is again a rise in the number of employees in the buffer near the end and is higher for higher values of $\alpha$.
- As for the Cumulative bump cutoff time in Figure 3c, we see a decreasing trend of target feature from those values seen at the start of the process for $\alpha = 40\%$ and 60%. The highest decrease is seen with $\alpha = 40\%$, which calls for more employees at the start. The behavior at the end is similar to curves in Figure 3b for the Cutoff Buffer Size.

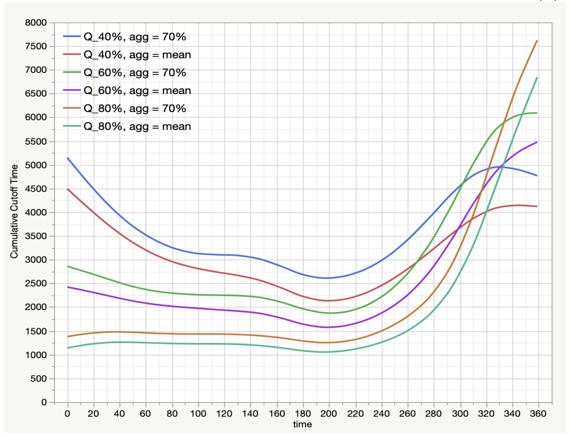
The following general observations are also made within all the plots. First, when $\alpha = 40\%$, the target features start higher as more employees have long delays than $\alpha = 80\%$ or 60%. This means more employees must be called from the start because employees generally take very long to respond. Second, the two descriptive statistics remain fairly parallel, indicating less feature variance. Third, curves from Figure 3 suggest two phases in the calling schedules. The first is a stable phase, where calls are made at a constant rate. The second phase is when calls are made rapidly when the horizon's end is near.



(a) Cumulative Calls Made



(b) Cutoff Buffer Size



(c) Cumulative Bump Cutoff Time

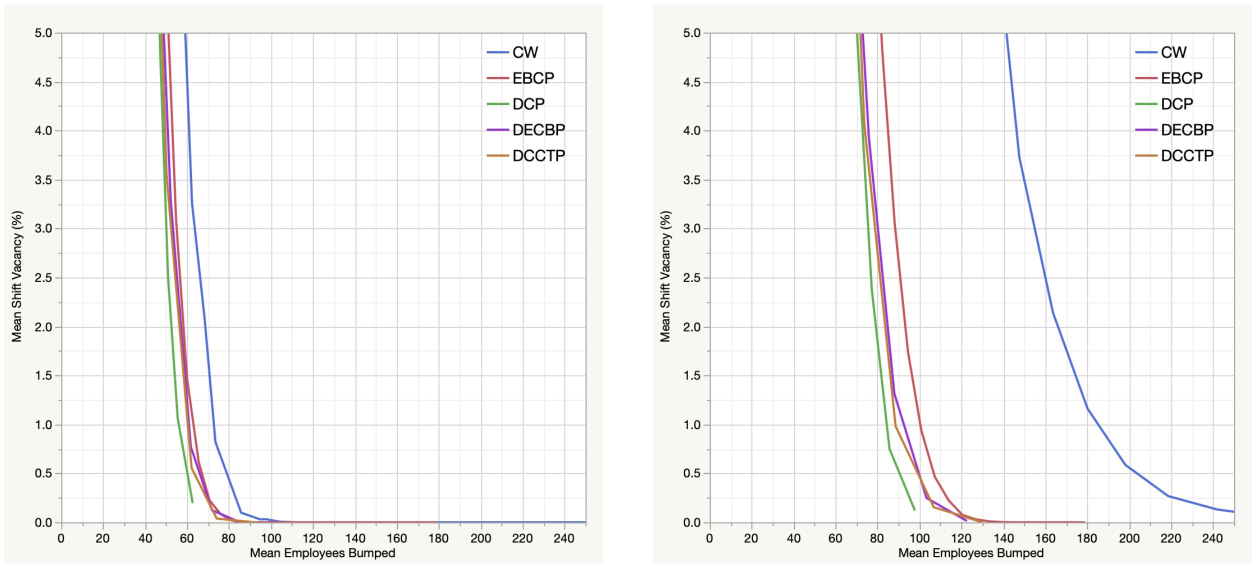
**Figure 3 Target Feature trend with time for Weibull Response Delay and Cutoff (D = 3) aggregated for all offline solutions**

## 4.3. Policy Evaluation

In this subsection, we analyze and compare the performance of various policies. Section 4.3.1 presents results that compare the threshold policies designed using the offline solutions with tuned heuristic policies for Weibull distribution of response delays. Section 4.3.2 describes their performance for real-world data of response delays obtained from our industrial partner.

**4.3.1. Evaluation with Weibull Response Delays:** Figure 4, Figure 5, Figure 6 shows a comparison between the five policies of the different values of $\alpha$ on the 1000 train instances. The figures plot the average number of bumps suffered by the employees against the average percentage of shifts vacant for a policy for response delays of 2 or 3 hours. The graphs contain one point for each unique parameter of the policy. In the case of the dynamic policies based on the offline solution, this parameter would be the descriptive statistic used. All policy points are connected by a line to form a Pareto frontier of that policy. Figure 4a, Figure 4b shows that for $\alpha = 40\%$, there is very little difference between all of the policies except for the CW policy. Since many employees respond later than the cutoff, the total bumps observed are less than those observed for higher values of $\alpha$. The CW policy is considerably worse as employees take longer times to respond. On the other hand, as $\alpha$ increases, the performance of the Offline Solution-based Policies is significantly better than heuristic policies. ECBP, the other heuristic policy, performs similarly to the CW for $\alpha = 80\%$. Also, in all cases, Offline Solution-based Policies show similar results.

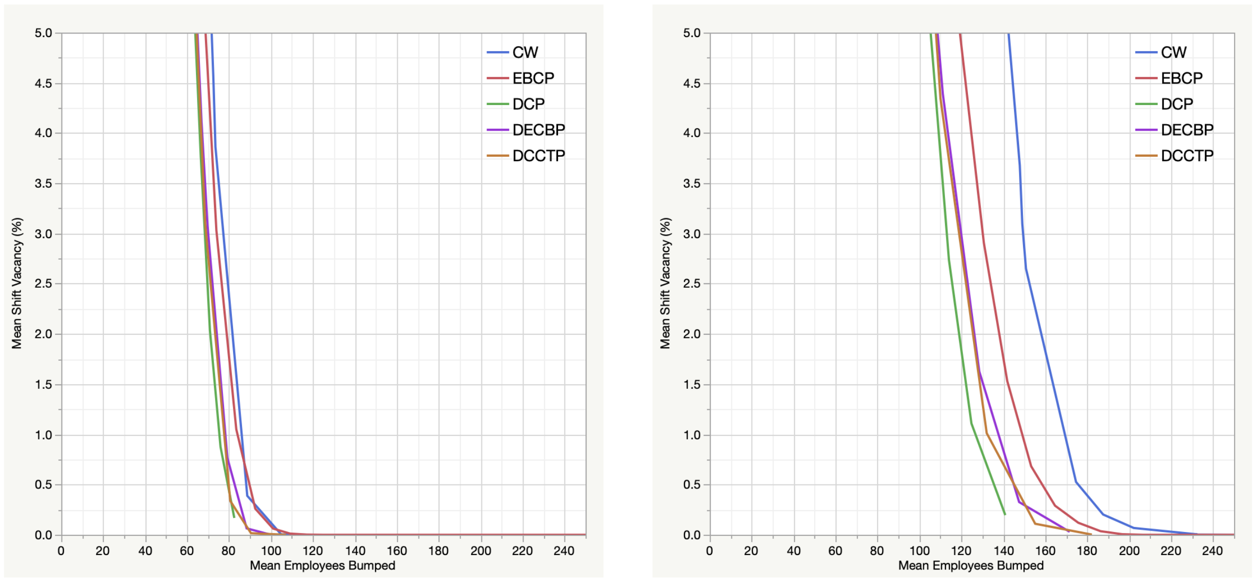
The decision-maker has two criteria to work with, reduction of total bumps and allocation of all shifts to employees. To select a policy to use with the electronic call system, the management can establish weights for each criterion and select a policy with minimum cost. Our industrial partner, however, would like a policy with a maximum shift vacancy of 0.5%. This vacancy percentage is tolerated because some employees respond after the horizon ends, and it is possible to use it over time. Table 4 summarizes the best policy parameters that respect this bound for each scenario regarding average bumps on validation instances. Higher quantile levels than the median perform well for offline solution-based policies. We then evaluate these policies on a set of new validation instances. Table 5 shows the policy performance with respect to average bumps of the best policies obtained using the validation instances. Table 6 shows the shift vacancies % for the test instances. The offline solution-based policies always outperform the heuristic policies. The difference in performance is stark in the case of a higher cutoff time of 3 hours. This is significant because our industrial partner uses a 3-hour bump cutoff time and ECBP to call employees. The parameters of this policy decided by the management were static and not tuned. We have tried to tune the parameters to optimize the policy. However, it is clear from the results table that this policy does not perform as well as some offline solution-based policies, even after tuning. OCP and OCCTP perform the best for the Weibull distribution.



(a) $Q^{\alpha} = D = 2$

(b) $Q^{\alpha} = D = 3$

**Figure 4 Pareto Front of all policies - Shift Vacancy % vs Number of Bumps,** $\alpha = 40\%$

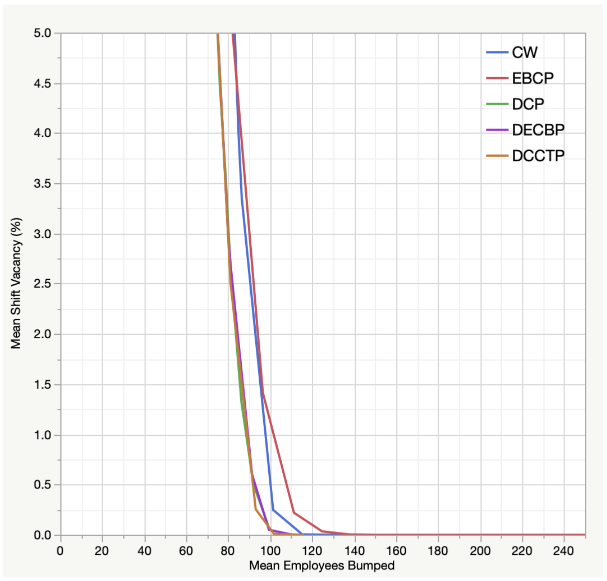


(a) $Q^{\alpha} = D = 2$

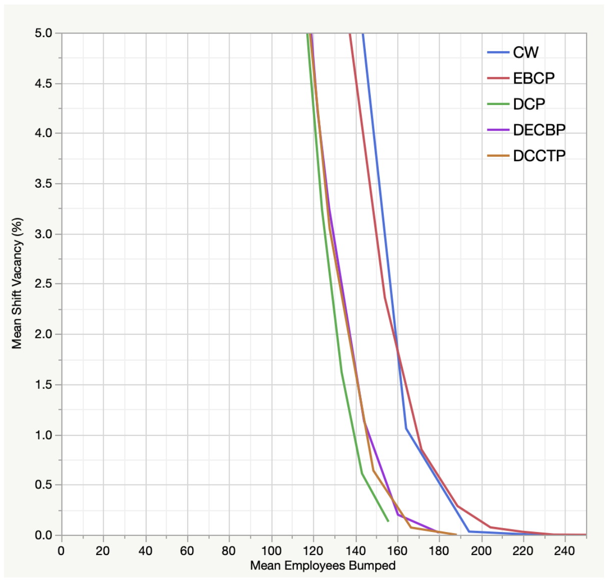
(b) $Q^{\alpha} = D = 3$

**Figure 5 Pareto Front of all policies - Shift Vacancy % vs Number of Bumps,** $\alpha = 60\%$

**4.3.2. Evaluation with Real World Data:** The results in the previous subsection help us develop a proof of concept with the response delays following a Weibull distribution. However, we also use real-world data from our industrial partner to evaluate our approach. The shift scheduling system has been in operation for more than five years, and we use data from 2018-2020 for further analysis. In general, on-call shift scheduling for a list of shifts can start 4 days before the start of shifts. However, our experiments only consider



(a) $Q^{\alpha} = D = 2$



(b) $Q^{\alpha} = D = 3$

**Figure 6 Pareto Front of all policies - Shift Vacancy % vs Number of Bumps, $\alpha = 80\%$**

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | **$\alpha$%** | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **40** | 3, 9 | 23 | 90% | 80% | 80 % |
| | **60** | 2, 9 | 16 | 90% | 80% | 70 % |
| | **80** | 2, 11 | 11 | 90% | 80% | 70 % |
| **3** | **40** | 4, 6 | 33 | 90% | 80% | 80 % |
| | **60** | 2, 7 | 25 | 90% | 80% | 80 % |
| | **80** | 2, 9 | 17 | 90% | 80% | 80 % |

**Table 4 Best Policy Parameters that minimize bumps in average subject to 0.5% shift vacancy on validation instances**

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | **$\alpha$%** | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **40** | 86.2 | 70.7 | **62.7** | 71.8 | 74.2 |
| | **60** | 88.7 | 92.5 | 82.5 | 88.4 | **80.6** |
| | **80** | 101.4 | 111.3 | 99.0 | 99.4 | **93.1** |
| **3** | **40** | 220.0 | 107.3 | **97.8** | 103.4 | 106.8 |
| | **60** | 187.6 | 164.6 | **140.9** | 147.5 | 155.2 |
| | **80** | 194.2 | 188.8 | **155.9** | 160.4 | 166.6 |

**Table 5 Average Bumps for the best policy of each PFA on test instances**

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | **$\alpha$%** | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **40** | 0.14 | 0.23 | 0.2 | 0.12 | 0.04 |
| | **60** | 0.39 | 0.26 | 0.17 | 0.06 | 0.34 |
| | **80** | 0.25 | 0.22 | 0.09 | 0.05 | 0.25 |
| **3** | **40** | 0.31 | 0.47 | 0.12 | 0.25 | 0.15 |
| | **60** | 0.2 | 0.29 | 0.2 | 0.33 | 0.11 |
| | **80** | 0.03 | 0.29 | 0.13 | 0.2 | 0.07 |

**Table 6 Shift vacancy % for the best policies on test instances**

scheduling personnel over a single day of operation with a six-hour planning period. As described earlier, the management uses a static ECBP policy with a fixed set of parameters every time the system starts making calls which we again tune for real-world response delays.

After cleaning the data, there were 23204 data points for the response delay. Figure 7, shows the cumulative distribution of the response delays from the cleaned data. Only 50% respond to the system to choose shifts, but a large proportion of them do so within one minute of the system calls. Those employees who do not respond within the horizon have their response delays artificially set to 1000, accounting for a 12-hour duration across the day. The assumption is that all such employees do not want to select a shift to work. One thing to note is that the system allows employees to choose among vacant shifts even after the end of the planning period. Ideally, the management would like to schedule all shifts within the planning period. This plot makes it clear that the number of employees needed to schedule all shifts is at least two times the number of shifts available since only half of the employees respond.

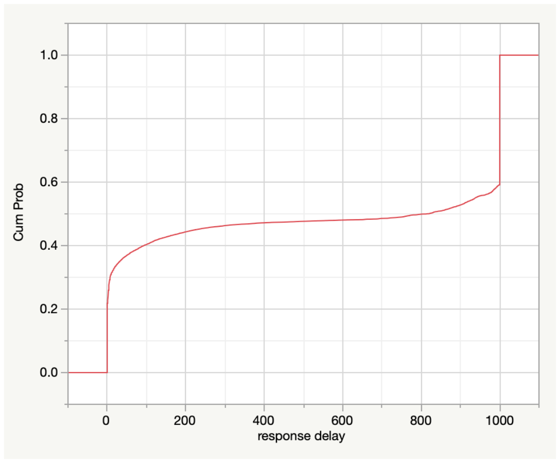


**Figure 7 CDF plot of Response Delay**

We keep the values for available shifts and planning period the same as before i.e., $|L| = 50, H = 6$ hours. A total of $M = 150$ employees are considered for all instances with two cutoff values, $D \in \{2, 3\}$ hours. The same training, validation, and testing procedure is used to evaluate our approach on real-world data. Response delays are sampled for each employee from the real-world data to generate instances. This data is divided into the train, validation, and testing datasets with the same split as the instances. Table 7 highlights the

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | $\alpha$% | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **42** | 3, 8 | 30 | 80% | 80% | 80% |
| **3** | **44** | 3, 8 | 42 | 90% | 80% | 80% |

**Table 7 Best Policy Parameters that minimize bumps in average subject to 0.5% shift vacancy on validation instances for real-world data**

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | $\alpha$% | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **42** | 70.1 | 121.1 | **55.7** | 64.4 | 65.8 |
| **3** | **44** | 97.4 | 198.2 | 75.3 | **74.2** | 76.5 |

**Table 8 Average Bumps for the best policy of each PFA on real data-based test instances**

| | | Policy | | | | |
|---|---|---|---|---|---|---|
| **D** | $\alpha$% | **CW** | **ECBP** | **OCP** | **OECBP** | **OCCTP** |
| **2** | **42** | 0.54 | 0.44 | 0.35 | 0.21 | 0.16 |
| **3** | **44** | 0.54 | 0.63 | 0.32 | 0.58 | 0.41 |

**Table 9 Shift vacancy % for the best policies on real data-based test instances**

| **D** | $\alpha$% | **Avg. Opt. Bumps** |
|---|---|---|
| **2** | **42** | 0.5 |
| **3** | **44** | 1.5 |

**Table 10 Average Optimal Bumps in the offline solution**

best parameters found using the validation instances. Table 8 gives the performance of these policies with the best-found parameters and Table 9 gives the corresponding shift vacancy percentage for the test instances. OCP and OECBP work best using real data regarding average bumps for a cutoff of 2 and 3 hours, respectively. The other offline solution-based policies for each cutoff also perform better than the heuristic PFA. Even a tuned ECBP is significantly worse than any of the other policies. The results show that there is room for improvement over the current ECBP. Our simple approach based on offline solutions seems to work very well. Table 10 also shows the average optimal bumps for the instances used in case of complete information. It suggests that given prior knowledge of employee response times, we can almost eliminate bumps from the process. However, in stochastic settings, bumps are an inherent system part.

## 5. Conclusion

Traditional on-call personnel scheduling systems have many issues, mainly relating to employee flexibility and ease of operation for the management. These systems have hardly received any attention in the operations research community to optimize their operations.

This paper presents a novel flexible, dynamic semi-automatic on-call personnel scheduling system to tackle these issues. One aspect of the flexibility in the system allows senior employees to replace/bump junior employees already scheduled to work a shift. These bumps are undesirable, and the management would like an operating policy to reduce them as much as possible while ensuring all shifts are filled. Our main contribution is to show that this problem of minimizing bumps while ensuring all shifts are scheduled is $\mathcal{NP}-complete$ even if all uncertainty is revealed beforehand by reducing from the SUBSET-SUM problem. Further, we propose three PFAs with a threshold structure for the dynamic version of the problem. Our approach uses an estimate of the threshold obtained by solving several instances offline and aggregating them using easy-to-compute descriptive statistics at each decision epoch. The threshold estimates are tested and validated on unseen instances. We build a proof of concept, using Weibull distributions to model the uncertain response delays and show that on real-world data, the proposed policies perform better than the tuned heuristic policy that is currently in practice.

Several future extensions for the problem are possible. First, we presented a simpler system version that only operates over one day. In reality, the scheduling can start four days before a list of shifts starts. Second, the cutoff time to bump someone could be a function of the time remaining and decreases as the system moves forward into the planning period. Finally, the most complex aspect of the system is that the employee will usually have different preferences for shifts, which will change from day to day. We plan to tackle these challenges in future research.

## References

Bard, Jonathan F, and Purnomo, Hadi W. 2005. Short-term nurse scheduling in response to daily fluctuations in supply and demand. *Health Care Management Science* **8** 315–324.

Bard, Jonathan F, and Binici, Canan, and desilva, Anura H. 2003. Staff scheduling at the United States postal service. *Computers & Operations Research* **30** 745–771.

Asgeirsson, Eyjólfur Ingi. 2014. Bridging the gap between self schedules and feasible schedules in staff scheduling. *Annals of Operations Research* **218** 51–69.

Dantzig, George B. 1954. A comment on Edie's "Traffic delays at toll booths". *Journal of the Operations Research Society of America* **2** 339–341.

De Filippo, Allegra, and Lombardi, Michele, and Milano, Michela. 2021. Integrated offline and online decision-making under uncertainty. *Journal of Artificial Intelligence Research* **70** 77–117.

Edie, Leslie C. 1954. Traffic delays at toll booths. *Journal of the Operations Research Society of America* **2** 107–138.

Ernst, Andreas T and Jiang, Houyuan and Krishnamoorthy, Mohan and Sier, David. 2004. Staff scheduling and rostering: A review of applications, methods and models. *European journal of operational research* **153** 3–27.

Garey, Michael R and Johnson, David S. 1979. Computers and intractability. *A Guide to the Theory of NP-Completeness.*

Golden, Lonnie. 2015. Irregular work scheduling and its consequences. Economic Policy Institute Briefing Paper.

Karp, Richard M. 1972. Reducibility among combinatorial problems. *Complexity of computer computations* 85-1003.

Kim, Kibaek and Mehrotra, Sanjay. 2015. A two-stage stochastic integer programming approach to integrated staffing and scheduling with application to nurse management. *Operations Research* **63** 1431–1451.

Nicol, Anne-Marie and Botterill, Jackie S. 2004. On-call work and health: a review. *Environmental Health* **3** 1–7.

Olmstead, John and Falcone, Deborah and Lopez, Jacy and Mislan, Linda and Murphy, Marialena and Acello, Toni. 2014. Developing strategies for on-call staffing: a working guideline for safe practices. *AORN journal* **100** 369–375.

Özder, Emir Hüseyin and Özcan, Evrencan and Eren, Tamer. 2020. A systematic literature review for personnel scheduling problems. *International Journal of Information Technology & Decision Making* **19** 1695–1735.

Pham, Tu-San and Legrain, Antoine and De Causmaecker, Patrick and Rousseau, Louis-Martin. 2023. Prediction-Based Approach for Online Dynamic Appointment Scheduling: A Case Study in Radiotherapy Treatment. *INFORMS Journal on Computing.*

Powell, Warren B. 2021. Reinforcement learning and stochastic optimization.

US Bureau of Labor Statistics. 2017. Contingent and Alternative Employment Arrangements. *USDL 18-0942.*

Van den Bergh, Jorne and Beliën, Jeroen and De Bruecker, Philippe and Demeulemeester, Erik and De Boeck, Liesje. 2013. Personnel scheduling: A literature review. *European journal of operational research* **226** 367–385.

Williams, Cara. 2008. Work-life balance of shift workers. *Statistics Canada* **9**.

## Appendix A: Proof

Proposition 1. *The minimum makespan of a NBS is* $C_0^* = r_1 + \sum_{i=1}^{n} (r_{i+1} - r_i)^+$.

***Proof of Proposition 1*** **Case 1**: $r_i \le r_{i+1}, \forall i \in \mathcal{E}$. Therefore, we can call employees $i$ and $i+1$ at the same time ($s_i = s_{i+1}$) without introducing a bump. The makespan of the schedule is increased by $r_{i+1} - r_i$. **Case 2**: $r_i > r_{i+1}, \forall i \in \mathcal{E}$. The only way to avoid a bump, in this case, is to make sure that both of them respond at the same time $e_i = e_{i+1}$. This is easily ensured by setting $s_{i+1} = s_i + r_i - r_{i+1}$. Hence, the makespan is not increased in this case.

Proposition 2. *The block schedule has the following properties:*

1. $\forall k \in A, s_{i_k} = s_{i_k - 1}$.
2. $\forall i \in \mathcal{E}_k^S, \forall k \in A,$ $e_i = e_{i_k}$ *or* $s_i = s_{i_k}$.
3. $s_{i_{k+1}} = s_{i_k} + \mathbf{1}_{k \notin A'} a_k$, *where* $\mathbf{1}$ *denotes indicator variable.*
4. $B(S(A')) = \sum_{j \in A'} a_j$
5. $C(S(A')) = C_0^* - \sum_{j \in A'} a_j$.

***Proof of Proposition 2*** Property 1, 2, and 3 are true by construction.

4. Consider some $k \in A'$, $r_{i_k} > r_i$ from Equation (1), and $s_{i_k} = s_i$ from Equation (2), $\forall i \in \mathcal{E}_k^S$. As a consequence

$$e_{i_k} = r_{i_k} + s_{i_k} > r_i + s_i = e_i, \forall i_k < i < i_{k+1}.$$

Since $a_k = i_{k+1} - i_k$ and using proposition Corollary 1, $b_{i_k} = a_k$.
Now consider some $k \in A \backslash A'$, and $e_{i_k} = s_{i_k} + r_{i_k} = s_i + r_i = e_i, \forall i_k < i < i_{k+1}$. As a consequence, no employee is bumped by $i_k$ and $b_{i_k} = 0$. Hence the total bumps are given by

$$B(S(A')) = \sum_{i \in \mathcal{E}} b_i = \sum_{k \in A} b_{i_k} = \sum_{k \in A'} a_k \tag{15}$$

5. $C(S(A')) = C_0^* - \sum_{j \in A'} a_j$.

Using property 3 and by recursion:

$$s_{i_{k+1}} = \sum_{j=1: j \notin A'}^{k} a_j \tag{16}$$

Now consider the employee with priority $M$.

$$s_M = s_{M-1} = s_{i_N} + \mathbf{1}_{N \notin A'} a_N. \tag{17}$$

Substitute equation Equation (16) in Equation (17),

$$s_M = \sum_{j=1: j \notin A'}^{N-1} a_j + \mathbf{1}_{N \notin A'} a_N$$
$$s_M = \sum_{j \notin A'} a_j$$

As a consequence one can now find $C(S(A')) = e_M$

$$\begin{aligned} C(S(A')) &= s_M + r_M \\ &= \sum_{j \notin A'} a_j + \sum_{j \in A} a_j \\ &= 2 * \sum_{j \in A} a_j - \sum_{j \in A'} a_j \\ &= C_0^* - \sum_{j \in A'} a_j \end{aligned}$$

Proposition 3. *For any $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, $C = e_M$.*

***Proof of Proposition 3*** It suffices to show that $r_M \geq r_i$, since $s_M \geq s_i$, due to seniority $\forall i \in \mathcal{E}$.
**Case 1**: $i = i_k \in \mathcal{E}^{\mathcal{C}}$.

$$r_{i_k} = \sum_{j=1}^{k} a_j \leq \sum_{j \in A} a_j = r_M$$

**Case 2**: $i \in \mathcal{E}_k^{S} : k \in A$.

$$r_i = r_{i_{k-1}} = \sum_{j=1}^{k-1} a_j < \sum_{j \in A} a_j = r_M$$

Hence $r_M \geq r_i$, $\forall i \in \mathcal{E}$.

Proposition 4. *$\forall k \in A$, $\forall l \in A$ employee $i_k$ cannot bump employee $i_l$.*

***Proof of Proposition 4***
**Case 1** $k \geq l$: The proposition is true due to seniority $\forall i_k, i_l \in \mathcal{E}$.
**Case 2** $k < l$ : To show that $e_{i_k} \leq e_{i_l}$. Consider $i_k \in \mathcal{E}^{\mathcal{C}}$ such that $r_{i_k} = \sum\limits_{j=1}^{k} a_j$ from Equation (1);

$$\begin{aligned} r_{i_l} - r_{i_k} &= \sum_{j=1}^{l} a_j - \sum_{j=1}^{k} a_j \\ r_{i_l} - r_{i_k} &= \sum_{j=k+1}^{l} a_j > 0 \end{aligned}$$

Hence $r_{i_l} > r_{i_k}$. Due to seniority $s_{i_l} \geq s_{i_k}$. Adding both inequalities

$$r_{i_l} + s_{i_l} > r_{i_k} + s_{i_k}$$

$$e_{i_l} > e_{i_k}$$

Proposition 5. *$\forall i \in \mathcal{E}^{\mathcal{S}}$, $\forall i' \in \mathcal{E}$ employee $i$ cannot bump employee $i'$.*

***Proof of Proposition 5***
**Case 1** $i \geq i'$: The proposition is true due to seniority $\forall i \in \mathcal{E}^s, i' \in \mathcal{E}$.
**Case 2** $i < i'$ : To show that $e_i \leq e_{i'}$.

Response delay for $i$, $r_i = r_{i_{k-1}} = \sum_{j=1}^{k-1} a_j$ where $i_k < i < i_{k+1}$ and $k \in A$ from Equation (1);

$$r_{i'} = r_{i_{k'-1}} : i_{k'} < i' < i_{k'+1} \tag{18}$$

Since $i < i'$, $k \leq k'$ .

$$r_{i'} - r_i = r_{i_{k'-1}} - r_{i_{k-1}}$$
$$r_{i'} - r_i = \sum_{j=1}^{k'-1} a_j - \sum_{j=1}^{k-1} a_j$$
$$r_{i'} - r_i \geq 0$$

Hence $r_{i'} \geq r_i$. Due to seniority $s_{i'} \geq s_i$. As a consequence, $e_{i'} \geq e_i$.

Proposition 6. *$\forall k \in A$, $\forall i \in \mathcal{E} : i \geq i_{k+1}$, employee $i_k$ cannot bump employee $i$.*

***Proof of Proposition 6*** $\forall k \in A$, $\forall i \in \mathcal{E} : i \geq i_{k+1}$, it suffices to show $r_{i_k} \leq r_i$.
**Case 1**: For $i \in \mathcal{E}^{\mathcal{C}} \cap \{i' : i' \geq i_{k+1}, i' \in \mathcal{E}\}$, the inequality holds due to proposition Proposition 3.
**Case 2**: For $i \in \mathcal{E}^{\mathcal{S}} \cap \{i' : i' \geq i_{k+1}, i' \in \mathcal{E}\}$, $r_i = r_{i_{l-1}}$ where $i \in \mathcal{E}_k^S$ and $l \in A$. Since $i > i_{k+1}$ and $i_{k+1} \in \mathcal{E}^{\mathcal{C}}$, $l \geq k+1$.

$$r_i - r_{i_k} = r_{i_{l-1}} - r_{i_k}$$

Using Proposition 4, where the response times of stable employees are monotonically non-decreasing, to get $r_{i_{l-1}} \geq r_{i_{k+1-1}} = r_{i_k}$,

$$r_{i_{l-1}} - r_{i_k} \geq r_{i_{k+1-1}} - r_{i_k}$$
$$r_{i_{l-1}} - r_{i_k} \geq 0$$

Hence $r_i \geq r_{i_k}$, and since $s_i \geq s_{i_k}$ due to seniority, $e_i \geq e_{i_k}$

Corollary 1. *$\forall k \in A$, $\forall i \in \mathcal{E}$ if employee $i_k$ bumps employee $i$, then $i \in \mathcal{E}_k^S$.*

***Proof of Corollary 1*** This is straight forward from Proposition 3 and Proposition 5 and by construction in Equation (1) where $r_{i_k} > r_i$, such that $k \in A$, $i \in \mathcal{E}_k^S$.

Lemma 1. *$\forall S^* \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, $\forall k \in A, b_{i_k} = a_k$ or $0$.*

***Proof of Lemma 1*** Let $S^* = [s_i, e_i]_{i\in\mathcal{E}} \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, and let $B^* = B(S^*)$ be the optimal number of bumps and $C(S^*)$ represent the makespan of the call schedule. Consider the employee block $\mathcal{E}_k^S$, along with critical employee $i_k$, for $k \in A$. The following two cases are possible:
**Case 1**: $\forall i$ such that $i \in \mathcal{E}_k^S$, $e_i < e_{i_k}$. This directly implies $b_{i_k} = a_k$.
**Case 2**: $\exists i'$ such that $e_{i'} \geq e_{i_k}$. A new of the schedule $S' = [s'_i, e'_i]_{i\in\mathcal{E}}$ is can be formed by redefining end times for employees of the $\mathcal{E}_k^S$ as follows

$$e'_i = \begin{cases} e_{i_k} & \forall i \in \{i : e_i < e_{i_k}, i \in \mathcal{E}_k^S\}, \\ e_i & \text{else} \end{cases}$$

This new schedule $S'$ is a feasible schedule since $\forall i \in \{i : e_i < e_{i_k}, i \in \mathcal{E}_k^S\}, r_i = r_{i_{k-1}}$ and $e'_i = e_{i_k}$ implies $s'_i = s_{i_k} + a_k$. Also $\forall i \in \{i : e_i \geq e_{i_k}, i \in \mathcal{E}_k^S\}$,

$$\begin{aligned} s'_i &= s_i \\ &\geq s_{i_k} + r_{i_k} - r_{i_{k-1}} \\ &\geq s_{i_k} + a_k \end{aligned}$$

In general, $\forall i \in \mathcal{E}_k^S$, $s'_i \geq \max\{s_{i_k} + a_k, s_{i-1}\}$. Hence it is easy to see that for the schedule $S'$, $b'_{i_k} = 0$. Hence $\forall k' \in A : k' \neq k$,

$$b'_{i'_k} = b_{i_k} \tag{19}$$

This equations follows from proposition Proposition 6 and Corollary 1. Also from Corollary 2, $b_i = b'_i = 0, \forall i \in \mathcal{E}^S$. Hence for schedule $S'$ $b'_{i_k} = 0$ since $\forall i \in \mathcal{E}_k^S, e'_i \geq e'_{i_k}$. Since $S^*$ is optimal, $B(S*) = B(S')$ and from equation Equation (19) $b_{i_k} = 0$.

Corollary 2. *$\forall S \in \mathcal{F}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, the total bumps $B(S) = \sum_{k\in A} b_{i_k}$.*

This corollary directly follows from Lemma 1.

Lemma 2. *$S(A) \in \mathcal{F}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$.*

***Proof of Lemma 2*** For $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, any feasible schedule must satisfy the seniority constraint and the horizon constraint. Any block schedule $S(A')$ already satisfies the seniority constraint. For the horizon constraint, the following equation must hold:

$$H = C_0^* - W \geq C_0^* - \sum_{j\in A'} a_j = C(S(A))$$

Since $H \geq C(S(A))$, $S(A)$ is a feasible schedule.

Lemma 3. *$\exists A' \subseteq A$ such that $S(A') \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$.*

***Proof of Lemma 3*** Let $S^* = [s_i^*, e_i^*]_{(i\in\mathcal{E})}$ be an optimal schedule with total bumps $B(S^*)$ and makespan $C(S^*) \leq H$ for $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$. From Lemma 2 a feasible solution always exists. We are now going to show $\forall S^* \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$ there exists a block schedule $S(A') = [s_i, e_i]_{(i\in\mathcal{E})}$ where $A' \subseteq A$ such that $B(S^*) = B(S(A'))$ and $C(S(A')) \leq C(S^*) \leq H$.
From Lemma 1 $\forall k \in A, b_{i_k} = a_k$ or 0. One can simply construct $A' = \{k : b_{i_k} = a_k\}$. The block schedule $S(A')$ has total bumps $B(S(A')) = \sum_{j\in A'} a_j = B(S^*)$. Also $\forall k \in A$, the following two cases exists:
**Case 1** $k \in A'$, $b_{i_k} = a_k$: For this condition to be true $\forall i \in \mathcal{E}_k^S, e_i < e_{i_k}$. Hence $s_{i_k} \leq s_i \leq s_{i_{k+1}}$.
**Case 2** $k \notin A'$, $b_{i_k} = 0$: For this condition to be true $\forall i \in \mathcal{E}_k^S, e_i \geq e_{i_k}$. This implies $s_i + r_i \geq s_{i_k} + r_{i_k}$. Hence it follows that $s_{i_k} + a_k \leq s_i \leq s_{i_{k+1}}$.
Combining the two cases to get:

$$s_{i_N} \geq \sum_{j=1}^{N-1} \mathbf{1}_{j\notin A'} a_j$$

Therefore, for the last employee, $s_M \geq s_{i_N} + \mathbf{1}_{j\notin A'} a_N \geq \sum_{j\notin A'} a_j$. Hence $C(S(A')) = e_M + s_M + r_M \geq \sum_{j\notin A'} a_j + \sum_{j\in A} a_j$ and $S(A')$ is also an optimal schedule.

COROLLARY 3. *$\forall S^* \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$, the optimal objective value $B(S^*) \geq W$.*

***Proof of Corollary 3*** From Lemma 3, $\exists A' \subseteq A, S(A) \in \mathcal{O}_{ECTP}\big([a_j]_{j\in\mathcal{A}}, W\big)$ such that $B(S(A')) = \sum_{j\in A'} a_j$ and $C(S(A')) = C_0^* - \sum_{j\in A'} a_j$.

$$C(S(A')) = C_0^* - B^* \leq H = C_0^* - W$$

Simplifying, $B^* \geq W$

THEOREM 1. *For $ECTP\big([a_j]_{j\in\mathcal{A}}, W\big)$, $B(S^*) = W \Leftrightarrow \exists A^* \subseteq A$ such that $\sum_{j\in A^*} a_j = W$.*

***Proof of Theorem 1***

**Case 1** $\exists\ S^* = [s_i^*, e_i^*]$ such that $B(S^*) = W$: From Lemma 3, one can build an optimal block schedule $S(A^*)$ with $B(S(A^*)) = \sum_{j\in A^*} a_j$. Hence, $W = \sum_{j\in A^*} a_j$.

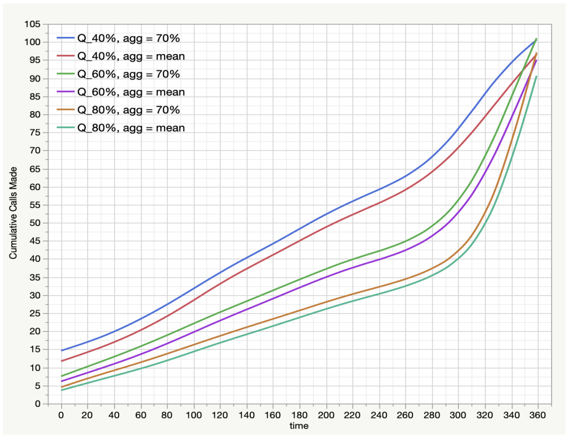
**Case 2** Suppose $\exists A^* \subseteq A$ such that $W = \sum_{j\in A^*} a_j$: Construct a feasible block schedule $S(A^*)$ with $B(S(A^*)) = \sum_{j\in A^*} a_j = W$ and makespan $C(S(A^*)) = C_0^* - \sum_{j\in A^*} a_j = H$. By Corollary 3, $S(A^*)$ is optimal.

THEOREM 2. *$ECTP(I)$ is* NP-complete.

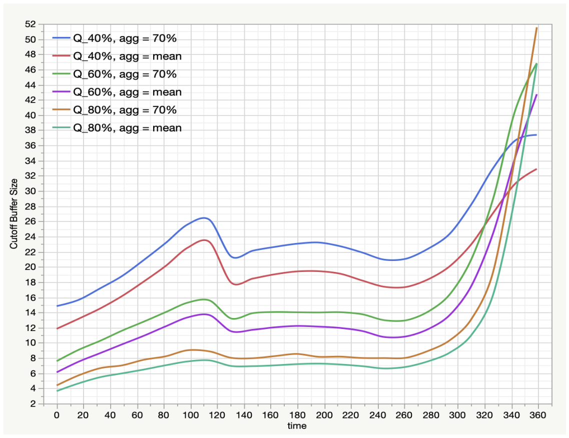
***Proof of Theorem 2*** This follows since Subset-Sum is *NP-complete.*

## Appendix B: Plots

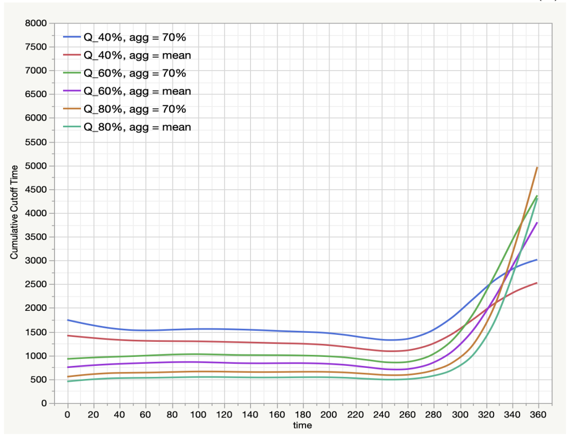
The plots in Figure 8, shows the mean and 70% quantile of all the target features time across the time with Weibull response delays and $D = 2$ hours. The trends seen are similar to those observed with $D = 3$ hours. However, the starting values of all features are lesser than those seen with $D = 3$ hours. This is obvious as employees call back faster and hence allowing more time to wait for their responses. Figure 9 shows the same trends but for real-world data with $D = 3$ hours. The trends of target features broadly remain the same as the ones with Weibull distribution.



(a) Cumulative Calls Made

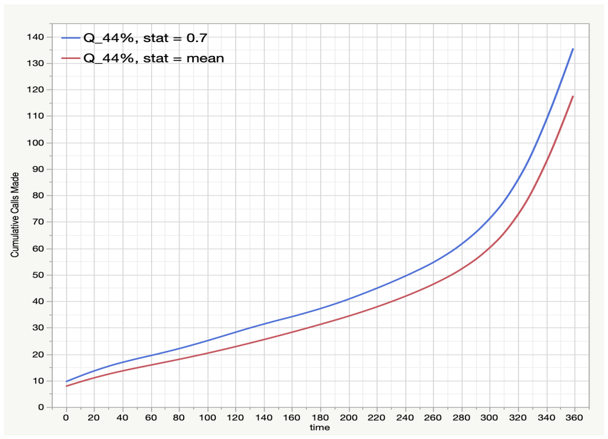


(b) Cutoff Buffer Size

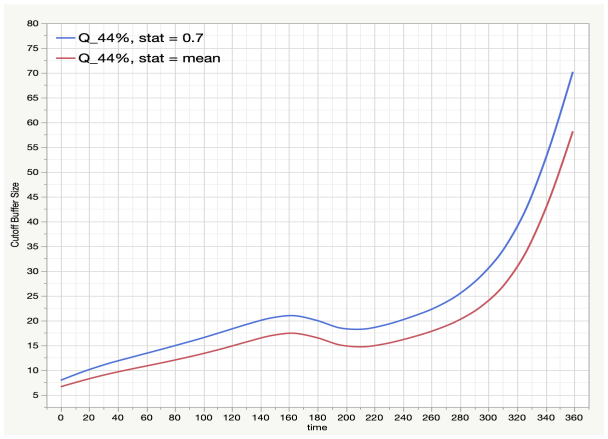


(c) Cumulative Bump Cutoff Time

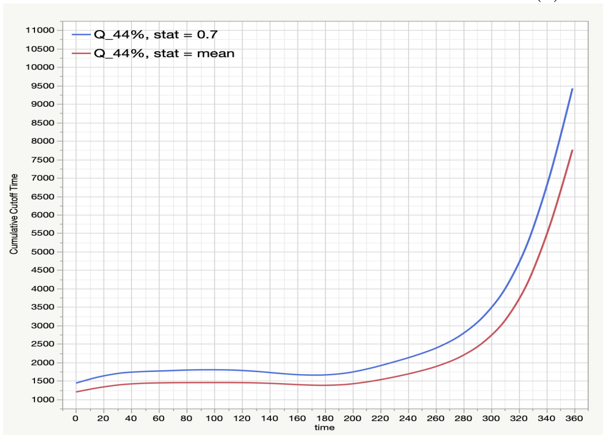
**Figure 8 Target Feature trend with time for Weibull Response Delay and Cutoff (D = 2) aggregated for all offline solutions**



(a) Cumulative Calls Made



(b) Cutoff Buffer Size



(c) Cumulative Bump Cutoff Time

**Figure 9 Target Feature trend with time for Response Delay from real data and Cutoff (D = 3) aggregated for all offline solutions**