Article can be accessed online at http://www.publishingindia.com

# Staff Scheduling in a Product Support Centre

**Ajith Kumar J.***

**Abstract**

The present study addresses the problem of staff scheduling in the product support centre of a large multinational technology company in India. Essentially, the problem was to develop rosters that minimise the cost of staffing, while ensuring that service level constraints are satisfied. Broadly speaking, previous research on call centre scheduling has taken two approaches to address the problem, the split approach and the integrated approach. The former handles the full problem by sequentially solving two separate sub-problems – staffing and scheduling – taking the solution to the staffing problem as an input to the scheduling problem. In contrast, the integrated approach proceeds iteratively by solving both parts repeatedly and converging to an optimal solution. We describe the split approach that we took to solve the problem. Our contributions include the modification of a heuristic used earlier to schedule staff in an airport's immigration centre and applying the modified version in conjunction with discrete-event simulation to solve the staffing sub-problem in the product support centre. Using realistic data, we also demonstrate how the heuristic fits into the larger procedure of solving the full staff scheduling problem.

**Keywords:** Product Support Centre, Staff Scheduling, Discrete-Event Simulation, Heuristic

## Introduction

A product support centre is an organisational unit that a company operates to serve its customers. It consists of a team of service agents trained to respond to customer queries, problems, and other service requests. The motivation for this study came from a staff-scheduling problem faced by a real-life product support centre. In short, the problem was to determine the optimal number of staff that the support centre should employ, and the allotment of those staff into weekly shifts and rosters. Actually, managers needed a method which they could use to solve this problem anytime they needed. This paper presents details of the problem, it's positioning in academic literature, and the method that we took to address it.A useful contribution that emerged from the study was the development of a simulation-based heuristic that can help find the solution.

A product support centre resembles a telephone call centre in some ways. As in a call centre, it continuously receives enquiries and complaints from customers. Trained service agents respond to customers. Both the call centre and the product support centre work 24x7 through the year, and hence need staffing throughout. However, a product support centre also differs from a call centre in some ways. Customers make telephone calls to a call centre and interact with a service agent in real-time. Often, and particularly during busy hours, call centre customers wait in a queue to receive service. If the customer is made to wait too long, she may renege (abandon) the call and may or may not call again. On the other hand, customers of a product support centre either submit their enquiries online through a web-portal or send them through email, and wait for a response. As such, they do not interact with a service agent directly. They do not personally experience waiting in a queue and can attend to other work while their enquiry is processed by the product support centre. The situation of a product support centre customer leaving the system (e.g. withdrawing an enquiry), before the centre attends to her request, is rare. Thus, while customer time spent waiting in queue is an important performance parameter for a call centre, the total time elapsed (waiting time + processing time) in completing the response is more important in a product support centre.

## Background

This study addressed the issue of staff-scheduling in a product support centre belonging to a large multinational company, hereafter referred to as 'ERP Inc'. ERP Inc offered enterprise resource planning solutions to businesses worldwide. The product support centrewas located in Bangalore, India. It attended to the company's cloud-based solutions customers, which were mainly small and medium-sized businesses located across the globe.

* Professor, XLRI Xavier School of Management, Jamshedpur, Jharkhand, India. Email: akm@xlri.ac.in

ERP Inc's customers submitted online queries and complaints pertaining to the product through a dedicated web portal and awaited the company's responses. Each such customer incident generated a ticket, or in more technical terms, an 'arrival' to the support centre. In practically all cases, each arrival was processed by one service agent. Since arrivals happened around the clock, the support centre made service agents available 24x7 in three shifts of eight hours each.

ERP Inc's contracts with customers invoked service level constraints (SLC) that dictated how long the support centre could take to process an arrival and give a satisfactory response to the customer. An SLC was expressed in terms of the proportion of arrivals that must be processed within a specified duration of time after entry to the system. Such an SLC is known as the telephone service factor in call centres (Robbins & Harrison, 2010). All arrivals, however, were not of the same type. ERP Inc. classified its arrivals into multiple types based on the perceived level of importance and associated urgency, and assigned a priority level to each type. Normally, arrival types that had faster turn around requirements were considered more urgent and given higher priorities. As a result, a separate set of SLCs was specified for each arrival type.

The SLCs in our problem were global SLCs that were 'horizon-based' and not 'period-based'. This means that they were assessed over the entire scheduling horizon, a duration that spans several periods (such as days or weeks), and not separately for each time period. A discussion on different types of SLCs can be referred in the paper by Robbins and Harrison (2010). These authors note that outsourcing contracts often specify global SLCs. Other studies that have taken the global service level perspective include Cezik and L'Ecuyer (2008) where the service levels are expressed as functions of the staffing for a fixed sequence of random numbers driving the simulation. An optimal solution of this sample problem is also an optimal solution to the original problem when the sample size is large enough. Several difficulties are encountered when solving the sample problem, especially for large problem instances, and we propose practical heuristics to deal with these difficulties. We report numerical experiments with examples of different sizes. The largest example corresponds to a real-life call center with 65 types of calls and 89 types of agents (skill groups and Kooleandvan der Sluis (2003).

The SLC was most crucial among the product support centre's performance criteria. A simplistic way that ERP Inc. could ensure meeting all SLCs was to employ, train, and assign a large number of service agents to each shift in its operations. However, while abundance in staffing can preempt violations of SLCs, it can also create redundancies in the form of excess manpower in some shifts (if not all) and in turn, high employment costs. Naturally then, the company wanted to have the fewest possible agents in a shift that would ensure meeting all SLCs. The problem, thus, is intrinsically one of cost minimisation.

At the time of this study, managers were allotting service agents to shifts in a somewhat ad-hoc manner, while taking into consideration their leave plans and other contingencies. Owing to this, there was a general feeling that their scheduling could have inherent inefficiencies and they sought a systematic procedure that could drive an optimal staff-to-shift assignment.

The problem had yet another aspect: arrivals did not happen at a constant rate over time; their rates varied across shifts. While some shifts in the week received a large number of arrivals, others had much lesser numbers, implying a non-stationary arrival phenomenon. In some cases, consecutive shifts had very different arrival volumes. Such an arrival phenomenon generated at the product support centre, what was termed flexible demand by Ernst, Jiang, Krishnamoorthy and Sier (2004).

We reviewed academic literature to gain insight into methods that previous researchers have developed to address this problem. We then gathered qualitative information as well as numerical data from the support centre before building a simulation model and developing the required scheduling procedure. In the following, we first present a brief overview of pertinent literature. We then describe the methodology used, the model built, the analysis and its findings. We conclude with a discussion on potential ways to extend and generalise the contribution of this study.

## Literature Review

We could not locate research that has modeled staff scheduling in product support centres and ours is perhaps amongst the first studies in this domain. However, the extensive work done on the same problem in call centres comes close. The call centre scheduling problem inherently has two parts that can be called the 'staffing

problem' and 'scheduling problem' respectively. Previous research has taken two broad approaches to address the full problem. We call them the split approach (e.g. Bhulai, Koole, & Pot, 2008; Dileepan & Ettkin, 2010; Pot, Bhulai, & Koole, 2008) and the integrated approach (Atlason, Epelman, & Henderson, 2004, 2008; Avramidis, Chan, & L'Ecuyer, 2009; Cezik & L'Ecuyer, 2008; Henderson & Mason 1998; Robbins & Harrison, 2010).

The split approach is modular in nature. It first solves the staffing problem, by dividing the scheduling horizon into small time periods, computing (or forecasting from past data) the average call arrival rate in each period, and then identifying the minimum staffing level in each period to ensure that specified SLCs are all met.It then solves the scheduling problem, by using these period-wise minimum staffing levels as right hand sides of the constraints of an integer linear program (ILP), and solving that ILP to minimise an objective cost function. The ILP's decision variables are the number of staff to be allotted to the different roster lines. The optimal solution to the ILP yields both the total staff size and the number of staff that must be assigned to each roster line.

A section of previous research has used analytical queuing models to address the staffing problem (e.g. Green & Kolesar, 1995; Green, Kolesar, & Soares, 2001). Analytic queueing models confer greater computational efficiency and are preferred when it can be reasonably assumed that the system reaches a steady state quickly, within each time period. However, the steady state assumption may not hold in many real-life situations (Henderson & Mason, 1998). Further, owing to the copious nature of arrivals, the optimal staffing in one period depends upon the staffing levels of other periods (Green, Kolesar, & Soares, 2003). Finally, the service level function, which is most commonly specified as the proportion of customers that must be served with a given time – as in our study – cannot be algebraically expressed (Atlason *et al*., 2004). Here, simulation is seen as a useful methodology that can provide both the flexibility to accommodate the dependencies between staffing in different periods and an ability to assess a service level function that cannot be algebraically expressed (Atlason *et al*., 2004, 2008).

The integrated approach, first inspired by Henderson & Mason (1998), iteratively shuttles between solving the scheduling and staffing problems. Each iteration first solves an ILP with staffing levels as constraint right hand sides, and then uses discrete-event simulation to test if the SLCs are met with those staffing levels. If the SLCs are not met, the procedure generates cutting planes– new constraints in the ILP that increase the lower bounds on the minimum – and solves the ILP again to find new (increased) staffing levels. The procedure terminates when some convergence criteria are satisfied with respect to the staffing vector.

Though the integrated approach overcomes some of the shortcomings of the split approach (such as the steady state assumption, the dependence between staffing levels and arrival rates across periods and the complex nature of the service level function), it faces a couple of distinct challenges. Henderson & Mason (1998) note that it is computationally intensive and requires the service level functions to be concave in nature. Though they have defended this assumption as being reasonable, and others (e.g. Atlason *et al*., 2004) have developed ways to test for concavity as well, there is an inherent lacuna in the procedure if this assumption is violated.

Both the split and integrated approaches have their strengths; yet research has not concluded on one 'best' approach to the problem. In our study, we followed a procedure that is based on the split approach. However, instead of analytical modeling, we used discrete-event simulation to handle the staffing problem, and then invoked the ILP to handle the scheduling problem. Our contributions include the development of a heuristic and its application in conjunction with discrete-event simulation to locate the minimum staffing vector. Before presenting the heuristic, we present a formal articulation of the problem.

## Formal Articulation of the Problem

Though our study's problem is very similar to the call centre scheduling problem, it differs in at least two distinct ways. First, in most call centre studies that we examined, all calls have been treated as having the same priority[1]. Second, there is typically only one service level constraint that applies either in each period separately, and sometimes over the entire scheduling horizon. In our product support centre, arrivals were of multiple priorities

1 Research has modeled different calls requiring different skills, however (e.g. Bhulai, Koole, & Pot, 2008; Avramidis, Chan, & L'Ecuyer, 2009). In such models, appropriately routing the calls would be a feature of the model.

and for each priority, multiple SLCs were applied. We incorporate these differences in our formulation.

As we go along, we define a few terms. A *shift* is an eighthour period that includes a half hour break in between. A *roster line* is a pre-defined set of shifts spread across several days, with some days off. Staff members are assigned to roster lines and not directly to the shifts. Possible examples of roster lines are:

a. shift 1 on each of five consecutive days beginning Monday and ending Friday, followed by off days on Saturday and Sunday.

b. shift 3 on five consecutive days beginning Thursday and ending Monday, with off days on Tuesday and Wednesday.

Let **x**:$x_j$, $j$=1,2,…,$n$ be the number of staff assigned to each of $n$ different roster lines, and let **c**: $c_j$ , $j$=1,2, …, $n$ be the corresponding unit staffing cost vector. Let the scheduling horizon be divided in to $m$ equal-sized time periods. The scheduling horizon is the time period for which a staff schedule is developed, for example, a day, a week, or a month. It is common in call centres to have cyclic schedules (or, tours), whereby staff assignments are repeated in each cycle of the scheduling horizon. Our product support centre too was following a cyclic scheduling system.

Let **A**: $a_{ij}$, $i$=1,2,…, $m$; $j$ = 1, 2, …, $n$ indicate whether roster line $j$ covers period $i$ or not, that is, $a_{ij}$ = 1, if roster line $j$ covers period $i$ and 0, otherwise. As such, the vector **Ax** denotes the actual number of staff realised in time period $i$.

Let $P$ be the number of different priorities that arrivals to the support centre can have, and let $p$ be their index. For example, if the arrivals are of five different priorities, then $P$ = 5, while $p$ = *2* indicates the second arrival category by priority.

Associated with arrivals of each priority there can be multiple SLCs, with each SLC specifying the proportion of arrivals of *that* priority, which must be served within a maximum time. Let $S$ denote the number of SLCs associated with each arrival category and let $s$ be their index[2]. For example, $S$ = 3 means that there are three SLCs associated with each priority in the problem, and for arrivals of priority 1 they might be as follows:($s$ = 1) at least 0.75 of the arrivals must be served within 20 minutes, ($s$ = 2)at least 0.90of the arrivals must be served within 25 minutes and ($s$ = 3) at least 0.99 of the arrivals must be served within 28 minutes of entering the system.

Let **r**: $r(p, s)$, $p$ = 1, …, $P$; $s$ = 1, …, $S$ denote the matrix of proportion values and **t**: $t(p, s)$, $p$ = 1, …, $P$; $s$ = 1, …, $S$ denote the matrix of time values associated with the SLCs. In the above example, $r_{11}$ = 0.75, $t_{11}$ = 20 and so on. In summary, **r** and **t** together help define the set of SLCs in the problem and are inputs to the problem. We note that the number of SLCs in the problem is $PS$.

While **r** and **t** together define the desired service levels, the *actual* service levels achieved depend upon three parameters: the actual number of staff available in each period of the scheduling horizon, the priority-wise arrival rates in each period and the rates at which the arrivals are serviced by the staff.

Let λ: $\lambda_{ip}$, $i$=1, 2, …, $m$; $p$ = 1, 2, …, P denote the mean arrival rate of tickets of priority $p$ in period $i$, μ: $\mu_p$, $p$ = 1, 2, …, P denote the mean service rate of tickets of priority $p$ and as noted earlier, **Ax** gives us the actual number of staff realised in time period $i$. Then, we can use **q**: $q(p, s,$ **Ax**, λ, μ), to denote the actual fraction of tickets of priority $p$ that complete their service within time $t(p, s)$. In other words, **q** is the matrix of actual service levels achieved.

Finally, let **d**: $d_i$, $i$ = 1, 2, … $m$ denote the staffing vector, where $d_i$ stands for the number of staff working in period $i$. Given these definitions and notations, the full problem and the sub-problems can be articulated as follows:

| Full problem | Staffing sub-problem | Scheduling sub-problem |
|---|---|---|
| Minimise Z = $\mathbf{c}^T\mathbf{x}$<br>Subject to:<br>$\mathbf{q}(p, s, \mathbf{Ax}, \lambda, \mu) \geq \mathbf{r}(p, s)$;<br>$\mathbf{x} \geq \mathbf{0}$ and integer | Minimised $_i$ for each $i$<br>Subject to:<br>$\mathbf{q}(p, s, \mathbf{d}, \lambda, \mu) \geq \mathbf{r}(p, s)$<br>$\mathbf{d} \geq \mathbf{0}$ and integer | Minimise Z = $\mathbf{c}^T\mathbf{x}$<br>Subject to:<br>$\mathbf{Ax} \geq \mathbf{d}$;<br>$\mathbf{x} \geq \mathbf{0}$ and integer |

2 Here, we assume that the number of SLCs is the same across priorities. This assumption, however, does not restrict the problem since we can take the maximum number of SLCs across priorities as a universal maximum. For priorities that have lesser SLCs than this maximum, we can create dummy SLCs to fill the gap, having the time limit as infinity (and use a very large number during execution).

The objective of the full problem is to find an optimal **x**. However, the service level function **q**is nonlinear and is difficult to express algebraically. This makes the full problem an integer nonlinear program and difficult to solve. Following the split approach, we first solved the staffing sub-problem to obtain the minimum **d**. We used a heuristic in conjunction with a simulation model of the product support centre to find **d.** We took the **d** thus obtained as an input to the scheduling sub-problem– which is an ILP – where it served as the right hand side of the ILP's constraints. Solving the ILP yielded an optimal **x**. The heuristic to find **d** is a novel feature of our study, and we describe it in the following.

## The Heuristic

Our heuristic builds upon one used by Mason, Ryan, and Panton (1998) to optimise staffing in the immigration of an international airport. They termed the heuristic Algorithm 1. Algorithm 1 begins by finding an initial feasible solution ($\mathbf{d_0}$) of staffing levels; a feasible solution is any **d** that satisfies all SLCs. For convenience, the initial feasible solution is chosen with equal staffing in all periods. Algorithm 1then systematically examines each period in turn, reduces the staffing level in that period and checks whether the reduced staffing solution is also feasible. It rejects a reduction that leads to an infeasible solution and terminates upon reaching a dominant minimum. Termination happens when there is no time period left in which a feasible reduction is possible.

Our heuristic differs from Algorithm 1essentially on the rule for the next period to pick. In Algorithm 1, the next period to pick is the one with the greatest staffing level amongst all periods left in the candidate list. The candidate list initially consists of all time periods in the scheduling horizon. When there is a tie, the heuristic chooses the period with the lowest index number. In contrast, our heuristic picks the next period on the basis of the net arrival rate in the period (across all priorities). This rule is driven by the intuition that periods with lower net arrival rates would need lower staffing than those with higher arrival rates. When there is a tie, we break it on the basis of the net arrival rate of the immediately preceding period.

The formal articulation of our heuristic draws on the one presented by Mason *et al.* (1998, pp. 164), but incorporates the arrival rate vector (**λ**). We present the same here:

Let $Z = \{1, \ldots, m\}$ be the candidate list of all periods in the scheduling horizon.

Let $\mathbf{d_0}$ be the starting feasible solution.

Set $\mathbf{d} \leftarrow \mathbf{d_0}$

repeat

Let $z = f(\mathbf{d}, \lambda, Z)$ be the index of the next period to reduce.

Consider the solution **d'**:$(d_1, d_2, \ldots, d_{k-1}, d_k-1, d_{k+1}, \ldots, d_m)$

Check whether **d'** is feasible.

If yes then

Let **d** ← **d'**

else

Remove *z* from *Z*, i.e. let $Z \leftarrow Z \backslash z$

Until$Z = \{\}$

Staffing plan **d** is a dominant minimum

We experimented with two variations of this heuristic:

- Heuristic 1: begin with the period that has the lowest net arrival rate; then move in the order of increasing net arrival rate, and
- Heuristic 2: begin with the period that has the highest net arrival rate; then move in the order of decreasing net arrival rate.

In Heuristic 1, we pick $z = f(\mathbf{d}, \lambda, Z)$ using the rule, $z$: $\lambda_z = \min \lambda_i (i \in Z)$, whereas for Heuristic 2, it is $z$:$\lambda_z = \max \lambda_i (i \in Z)$. In both cases, when there is a tie for min$\lambda_i$, with say $\lambda_g = \lambda_h = \min \lambda_i$ we use the rule, $z$: $\lambda_z = \lambda_g$ if $\lambda_{g-1} < \lambda_{h-1}$ else $\lambda_z = \lambda_h$ (g, h ∈ *Z*).That is, to break the tie we take that period, whose immediately preceding period has a lower net arrival rate.

## Methodology

ERP Inc's product support centre makes use of an automated internet-based tool known as the Activity Management System (AMS) to manage the arrivals of customer tickets and their processing by service agents. We gathered records of all arrivals that occurred over a period of 91 days (13 weeks, or 3 months). Each record contained the following information about the arrival:

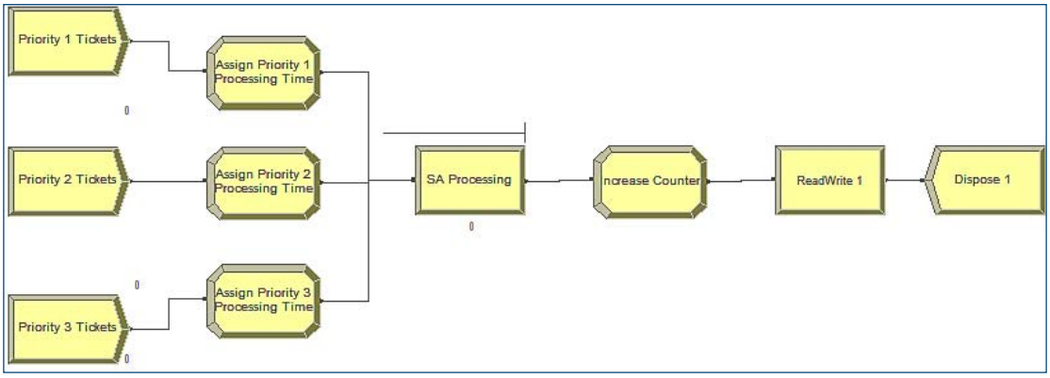


**Fig. 1: Arena Model Used in This Study**

**Table 1: Consolidated Arrival Pattern Used as an Input for Simulation**

| *Arrival Priority* | *Time period* | *AVERAGE NUMBER OF ARRIVALS PER HOUR* | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | *SUN* | *MON* | *TUE* | *WED* | *THU* | *FRI* | *SAT* |
| 1 | 06:00 to 14:00 | 0.79 | 0.19 | 0.21 | 1.00 | 0.67 | 0.55 | 4.50 |
| | 14:00 to 22:00 | 0.03 | 0.60 | 0.66 | 2.11 | 1.21 | 2.12 | 3.72 |
| | 22:00 to 06:00 | 0.89 | 0.21 | 0.47 | 0.48 | 0.70 | 0.08 | 2.23 |
| 2 | 06:00 to 14:00 | 2.11 | 6.19 | 7.15 | 8.26 | 8.26 | 8.27 | 2.20 |
| | 14:00 to 22:00 | 2.08 | 5.23 | 5.48 | 6.63 | 5.24 | 4.88 | 1.98 |
| | 22:00 to 06:00 | 0.88 | 1.54 | 3.09 | 3.79 | 3.10 | 2.89 | 1.63 |
| 3 | 06:00 to 14:00 | 0.12 | 0.27 | 1.05 | 0.90 | 0.56 | 0.45 | 0.01 |
| | 14:00 to 22:00 | 0.16 | 0.29 | 0.81 | 0.50 | 0.51 | 0.33 | 0.13 |
| | 22:00 to 06:00 | 0.14 | 0.20 | 0.82 | 0.71 | 0.71 | 0.48 | 0.08 |

its priority level, the date and time of its creation in the AMS, the date/time at which it was picked by a service agent, and the date/time at which the agent closed it and it left the system. In all, our data set had 13349 records. Some records had defective data such as missing values or extremely high and unrealistic numbers, suggesting erroneous entries. After cleaning, the dataset reduced to 13219 records, of which 2471, 9784 and 964 records belonged to ticket priorities 1, 2 and 3 respectively.

To apply the heuristic and solve for the **d**, we built a discrete event simulation model of the system using Arena for Windows (version 10.0), as shown in Fig. 1. Using the data set, we prepared three essential inputs to the simulation model – the arrival pattern, the service time distributions and the staffing vector.

First, we will discuss the arrival pattern. The service agents work in 8 hour shifts that run 06:00–14:00, 14:00–22:00 and 22:00–06:00.We computed the net average arrival rates (priority-wise as well as overall) within each shift and found that shift-wise rates were somewhat repetitive from one week to the next. Based on this, we decided to adopt a *one-week* scheduling horizon. Treating each 8 hour shift as a distinct time period yields 21 time periods in the scheduling horizon. We consolidated the arrival rates into these time periods, and specified arrivals as the average number within each time period, for each priority category, on each week-day.

Table 1 presents this consolidated arrival pattern data. For example, on Sundays, 2.11 tickets of Priority 2 arrived on an average from 06:00 hours to14:00 hours, on Thursdays, it was 0.56 tickets of Priority 3 in the same period, and so on[3].

3 ERP Inc did not object to our publishing this study and its contributions, but requested us to conceal the original data. Accordingly, we have modified the key data. More specifically, the arrival rate, service time and SLC data presented here are not the original values. However, the numbers presented here are realistic and representative of those in real-life. It is important to note that the changed data affects neither the

**Table 2: Processing Time and the SLC Specifications**

| *Arrival Priority* | **Processing Time (*min*)** | *SLC* | |
|---|---|---|---|
| | | *85$^{th}$%ile (min)* | *99$^{th}$%ile (min)* |
| 1 | 1 + Expo(6.3) | 15 | 40 |
| 2 | 1 + Expo(12.1) | 30 | 60 |
| 3 | 1 + Expo(8.3) | 45 | 120 |

We assumed that arrival within each shift followed a Poisson process.

Second, we examined the time spent by service agents in processing arrivals of each priority category. Using the Input Analyzer feature of Arena, we identified the probability distributions followed by processing times, separately for each priority. Table 2 presents these distributions. For example, the time spent by the SAs in processing arrivals of priority 1 followed the distribution 1+Expo (6.3) minutes, where Expo (6.3) represents a negative exponential distribution with the mean of 6.3. We also gathered data on the SLCs associated with each priority (Table 2). For example, 0.85 of Priority 1 arrivals (85$^{th}$ percentile) had to be completed in 15 minutes, while 0.99(99$^{th}$ percentile) had to be completed within 40 minutes of the arrival, and so on.

Finally, we discuss the staffing vector. We noted that each 8 hour shift includes a half hour break in between. We learnt from discussions with managers that agents in a given shift do not all take the break at once. Rather, about half of them take the break first and the rest take it after the first group returns. This ensures that some arrivals are always being processed. We incorporated this in the simulation model as follows. If $d_i$ is the number of staff assigned to shift $i$ in the model, then the staffing level is $d_i$ for the first three and a half hours, $d_i/2$ for the next one hour and again $d_i$ for the remaining three and a half hours of that shift.

We validated the simulation model by running 30 replications, each for a period of 91 days (same as the data gathered), keeping the staffing levels similar to those currently practiced. We compared the number of arrivals generated and the average time durations[4] resulting from the simulation model, with those currently experienced in practice. While the number of arrivals matched very closely, we found acceptable similarities in the durations.

We then performed what-if analyses on the staffing vector with each of the two heuristics described earlier. We first found a 'ceiling vector' $\mathbf{d_0}$, which, as explained earlier, is an initial feasible solution from which the heuristic begins its descent towards a dominant minimum. For this, we began by assigning exactly 1 service agent to each period, that is $\mathbf{d_0}$ = (1, 1, …, 1) and ran the simulation. When the solution turned out to be infeasible, we increased the staffing level by 1 unit in each period and ran the simulation again. We repeated this until we found the minimum feasible ceiling vector to be (5, 5, …, 5).

Each iteration of the heuristic involved making a unit reduction in the staffing level in one of the 21 shifts, running the simulation and examining if any constraint was violated. The heuristic examined each of the 21 shifts in turn at least once (but possibly several times), and terminated when a dominant minimum was reached, that is, when no further reduction in staff was possible without violating a constraint[5]. We conducted all our iterations by manually changing the staff levels in the Arena model. In each iteration, we ran 30 replications of the simulation for 91 days, same as the duration of the data. After reaching the dominant minimum, we ran the second-stage ILP, to compute the minimum team size needed if that staffing plan was followed. Since our problem was not large, we could do this using MS Excel's Solver. We took cost per employee as the same across all roster lines; having different costs will not affect the methodology.

methodology/heuristic that we present (which are the focus of the paper), nor the paper's conclusions.

4 Priority-wise:averaged over all tickets, the time spent in queue and the total time spent in the system.

5 Each iteration results in either an acceptable unit reduction in the number of staff, or a constraint violation.When a constraint violation happens, that shift will no longer be examined.This means that the number of iterations to reach the dominant minimum in a given trial of a heuristic, equals the total number of reductions made across all shifts (since one reduction happens in each iteration), plus the number of shifts.

**Table 3: Results**

| *Time Period (Shift)* | *Net Average Arrival Rate ($h^{-1}$)* | **Ceiling (*initial feasible solution*)** | **Optimal *d* reached by** | | |
|---|---|---|---|---|---|
| | | | ***Mason et al.'s* Algorithm 1** | *Heuristic 1* | *Heuristic 2* |
| Sun 1 | 1.72 | 5 | 4 | 2 | 3 |
| Sun 2 | 1.56 | 5 | 4 | 4 | 4 |
| Sun 3 | 0.83 | 5 | 4 | 3 | 3 |
| Mon 1 | 4.64 | 5 | 4 | 4 | 4 |
| Mon 2 | 4.00 | 5 | 4 | 4 | 4 |
| Mon 3 | 1.19 | 5 | 4 | 3 | 4 |
| Tue 1 | 5.41 | 5 | 4 | 4 | 5 |
| Tue 2 | 4.24 | 5 | 4 | 4 | 4 |
| Tue 3 | 2.43 | 5 | 4 | 4 | 5 |
| Wed 1 | 6.37 | 5 | 4 | 4 | 5 |
| Wed 2 | 5.34 | 5 | 4 | 5 | 4 |
| Wed 3 | 2.95 | 5 | 5 | 4 | 5 |
| Thu 1 | 6.28 | 5 | 5 | 4 | 4 |
| Thu 2 | 4.14 | 5 | 4 | 4 | 4 |
| Thu 3 | 2.48 | 5 | 4 | 4 | 5 |
| Fri 1 | 6.26 | 5 | 5 | 5 | 4 |
| Fri 2 | 4.03 | 5 | 5 | 4 | 4 |
| Fri 3 | 2.19 | 5 | 5 | 4 | 5 |
| Sat 1 | 2.47 | 5 | 4 | 4 | 4 |
| Sat 2 | 2.17 | 5 | 4 | 4 | 4 |
| Sat 3 | 1.63 | 5 | 4 | 3 | 4 |
| **Minimum manhour/week ($\Sigma_i d_i$)*8** | | 840 | 712 | 648 | 704 |
| Variance | | | 0.190 | 0.429 | 0.362 |
| **Minimum team size needed ($\Sigma_j x_j$)** | | | 18 | 17 | 19 |
| Roster lines | | | 14 | 12 | 14 |

## Results and Discussion

Table 3 displays the results. It presents the average net arrival rate for each time period (shift), the ceiling values of the staffing plan and the optimal **d** reached by applying each of the two experimental heuristics and Algorithm 1 of Mason *et al*. (1998). Figs. 2-4 display these **d** respectively. Table 3 also displays for each **d**, the minimum man hours/ week it implies(that is, $8*\Sigma_i d_i$), the variance in staffing levels across time periods within that **d**, the optimal team size(that is, $\Sigma_j x_j$) needed to achieve that **d**, and the number of distinct roster lines that the team will be scheduled into, which is the same as the number of non-zero $x_j$ in the **x**.

Heuristic 1 leads to a markedly lower weekly man hour requirement (648) than those of Heuristic 2 and Algorithm 1 (704 and 712 respectively). Clearly, the optimal **d** reached by solving the staffing sub-problem depends upon the heuristic deployed. Heuristic 1 also yielded the best team size (17) vis-à-vis those of Heuristic 2 (19) and Algorithm 1 (18).

### The Role of Variance

We make a couple of observations. The optimal team size resulting from Heuristic 1 is only slightly better (lower) than that obtained by Algorithm 1 (17 vs. 18) even though Heuristic 1 led to markedly lower weekly man hour requirement (648 vs. 712). Further, the optimal team size

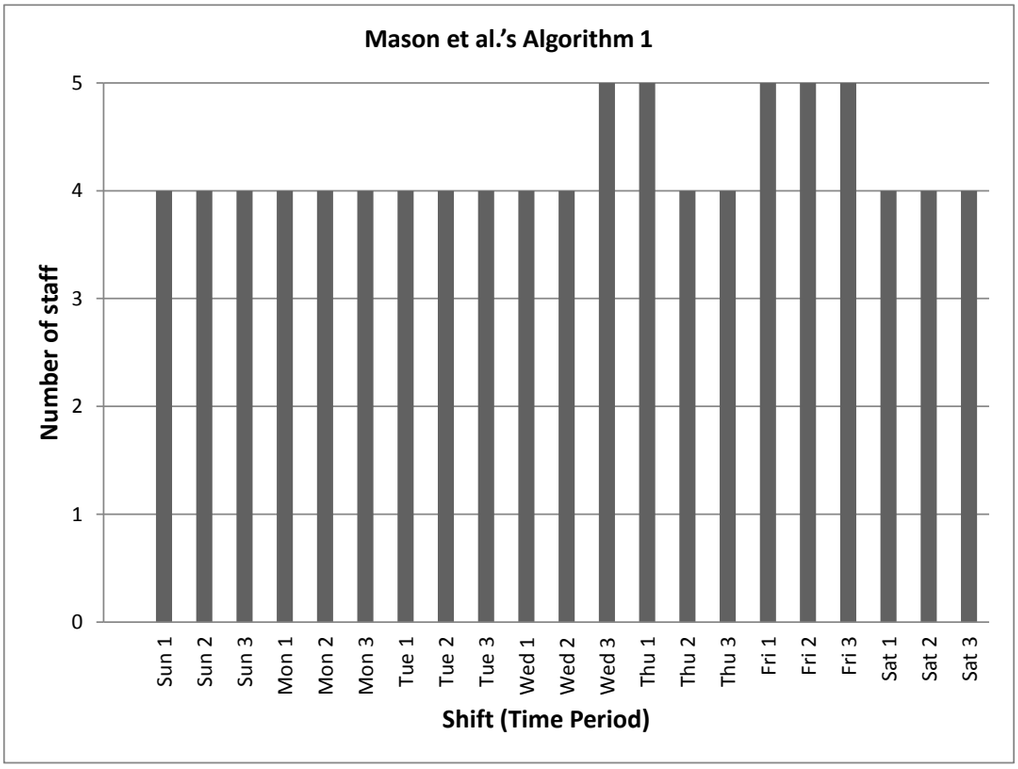


**Fig. 2: Staffing Solution Obtained by Mason, Ryan, & Panton's (1998)Algorithm 1**

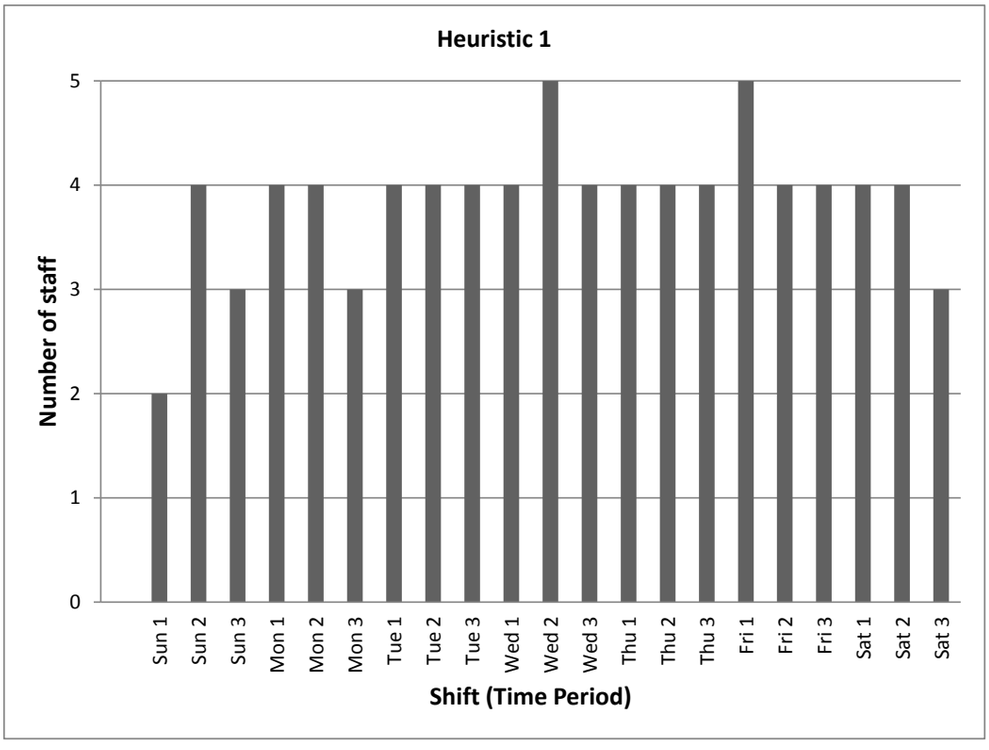


**Fig.3: Staffing Solution Obtained by Heuristic 1**

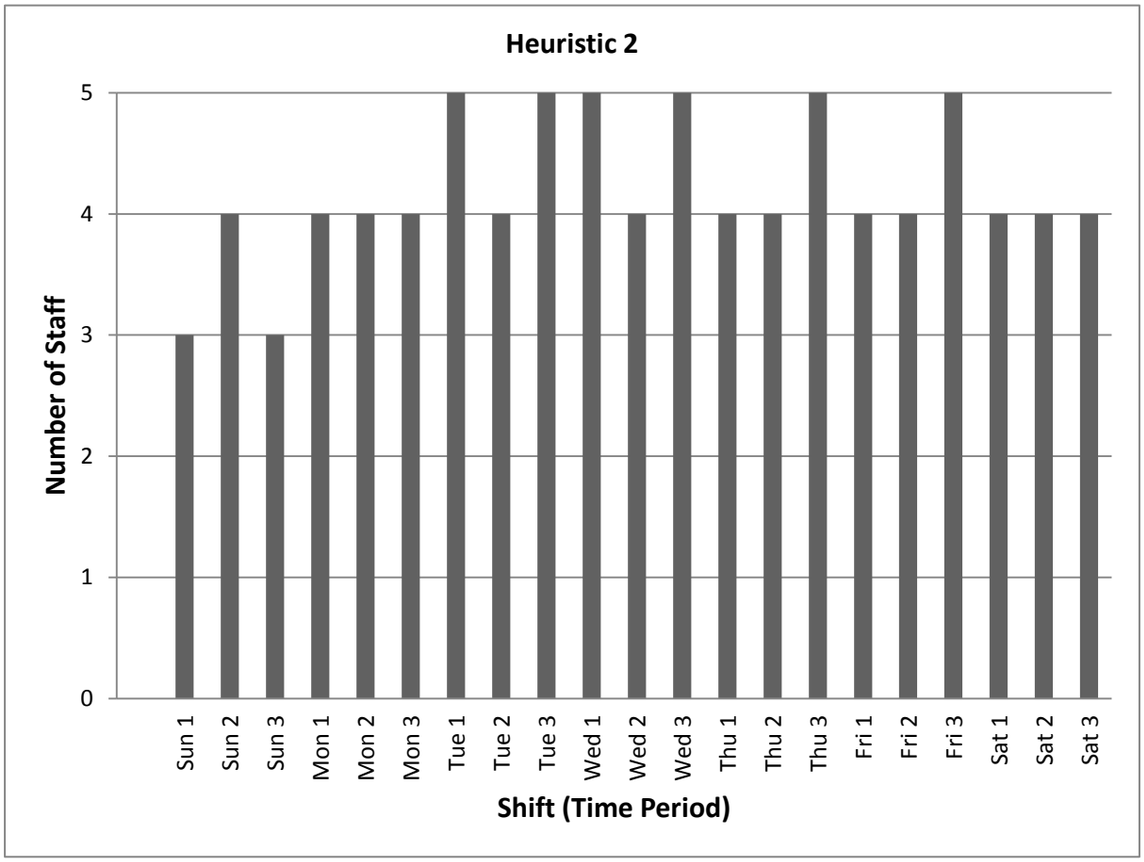


**Fig. 4: Staffing Solution Obtained by Heuristic 2**

resulting from Heuristic 2 is one unit worse (higher) than that obtained by Algorithm 1 (19 vs. 18) despite Heuristic 2 having better (lower) weekly man hour requirement (704 vs. 712). These observations can be explained by the variance of the period-wise staffing values ($d_i$) within the **d**. The solution obtained by Heuristic 1 has a variance of 0.429, while that from Heuristic 2 has a lower variance (0.362) and that from Algorithm 1 has the lowest variance (0.190). We recollect that the $d_i$ values become right-hand sides of the set-covering ILP in the second sub-problem. When the variance of $d_i$ is more, the ILP needs to use a larger set (team) to provide the required minimum cover to all periods. More generally, we can expect that given two staffing solutions with equal weekly man hour requirements, the solution with lower internal variance is likely to lead to a lower team size. This means that the effectiveness of the staffing heuristic should be judged not only by the total weekly man hour requirement that it leads to but also by the variance in its staffing across periods. Naturally, a heuristic that gives us both the least weekly man hour requirement as well as the lowest variance across shifts can be expected to give us the lowest team size.

## Number of roster lines

The last column of Table 3 shows the number of distinct roster lines that emerge from the ILP's solutions. As noted earlier, this is the same as the number of non-zero $x_j$ in the optimal roster schedule **x**. From the perspective of managerial convenience in scheduling, having lesser roster lines (greater parsimony) might be easier to handle. Again, Heuristic 1 does best with 12 distinct lines vis-à-vis the 14 lines that Heuristic 2 and Algorithm 1 yield.

## Algorithm 1 vs. Heuristics 1 and 2

There seems to be an inherent limitation in Algorithm 1, which our heuristics attempt to overcome. While selecting the next period to reduce staff, Algorithm 1 picks that period in the candidate list with the largest number of

staff in it. When there is a tie, it picks the chronologically earlier period. Given that the initial feasible solution has equal staffing in all periods, Algorithm 1 begins with the first chronological time period and proceeds by selecting periods chronologically. As a result, it can induce lower staffing levels in earlier time periods of the scheduling horizon and higher ones in later periods. Though, this may seem problematic, Mason *et al.*'s (1998) study deals with scheduling customs staff in the departure process of an international airport, and the authors propose that such biasing is advantageous as it gives, "… a more robust solution should delays occur" (pg. 164).

While adapting Algorithm 1, we recognised that a product support centre differs from an international airport in some ways. Though demand is flexible in both cases, the scheduling horizon for us is not just a day as in the airport; it could be a week, ten days, or even longer, depending upon how arrival patterns repeat themselves over time. Consequently, the bias that results from picking chronologically earlier periods first can be disadvantageous as it can arbitrarily restrict the final solution to the scheduling problem from reducing below a certain point. In other words, the sequence in which the periods are picked for reduction can matter to the dominant minimum reached. As seen, Heuristic 1 performed distinctly better than Heuristic 2 (Table 3).

In modifying Algorithm 1 to arrive at our heuristic, we were aware that our guiding intuition that shifts with lower arrival rates will need lower staffing cannot be generalised owing to dependencies between periods. Arrivals that occur in a given period may 'spill over' and be processed by staff in subsequent periods and this is typical for arrivals that occur close to the end of a given period. As noted earlier, the staffing level in one period influences service levels achieved in periods that immediately follow it. For this reason, even though it is possible to compare heuristics with each other and identify one that is better than others, it is difficult to firmly ascertain whether we actually reach the optimal solution by either heuristic. Alternate heuristics may exist that can yield better minima and lower cost rosters. Future experiments can examine other variations of these heuristics. For example, one variation could be to choose the next period to pick as the one with the lowest arrival rate for the highest priority arrival and yet another could be with the lowest arrival rate for the highest priority arrival.

All the three heuristics compared in this study work by reducing one staff at a time in a given period, before moving to the next period. The distinctive feature of Heuristic 1 is that it first chooses periods that have lower average arrival rates. This seems to be its strength. We explained all three heuristics to the company and showed them the differences in results.

## Conclusion

The purpose of our study was to find a method for ERP Inc. to determine the optimal number of staff that the support centre should employ, along with the scheduling of those staff into rosters. Did our efforts serve that purpose? The answer is: yes, but in a more fundamental way than the numerical results that our method provides. The significant positive that managers at ERP Inc perceived, stemmed from the realisation that they now had a scientific procedure to justify the number of staff they employ and the rosters they develop. They no longer had to depend only on the intuitive and ad-hoc procedure they had been following all along. In the year-long period since this study concluded, the product support centre team has regularly used the split procedure that we discussed in this paper, and are satisfied with it. The procedure has provided a scientific support for intuition and common sense in actual practice.

From an academic perspective, several research studies have examined the scheduling problem in the call centre domain. However, our study has certain novel features. It is perhaps amongst the first studies on staff scheduling in product support centres. As such, it incorporates multiple arrival priority categories and multiple service level constraints within each category, something not generally seen in the call centre scheduling research. It also offers a simulation-based split approach to the call centre scheduling problem along with a heuristic not presented earlier. Thus it makes a useful contribution to research on scheduling. Notwithstanding this, more work is necessary to understand how our procedure compares to competing procedures in terms of efficiency. Indeed, we believe that finer insights and stronger conclusions are possible, by repeating the above trials with multiple datasets.

During this study, we manually changed staffing levels in the model, while applying the heuristic. As a result, each trial of the heuristic took several hours to conduct. Nonetheless, the exercise gave us a glimpse of how the

heuristics work and helped us gain some early insight into why one heuristic may be better than another. Going forward, we are exploring the possibility of developing a code that can automate the entire heuristic descent and give us the dominant minimum with the press of a key. We expect that run time would reduce to a few minutes, and make it practical for us to conduct an exhaustively large number of trials within a few weeks. Armed with this code, we intend to experiment with several sets of input data (arrival rate, processing time, SLCs) and compare the performances of the heuristics with each other. We feel that that such work can lead to notable contributions to research on staff scheduling in call centres and product support centres.

## References


Atlason, J., Epelman, M. A., & Henderson, S. G. (2004). Call center staffing with simulation and cutting plane methods. *Annals of Operations Research*, 333-358. Retrieved from http://doi.org/10.1023/B:ANOR.0000019095.91642.bb

Atlason, J., Epelman, M. A., & Henderson, S. G. (2008). Optimizing call center staffing using simulation and analytic center cutting-plane methods. *Management Science*, 54(2), 295–309. Retrieved from http://doi.org/10.1287/mnsc.1070.0774

Avramidis, A. N., Chan, W., & L'Ecuyer, P. (2009). Staffing multi-skill call centers via search methods and a performance approximation. *IIE Transactions*,6, 483-497. Retrieved from http://doi.org/10.1080/07408170802322986

Bhulai, S., Koole, G., & Pot, A. (2008). Simple Methods for Shift Scheduling in Multiskill Call Centers. *Manufacturing & Service Operations Management*, 10(3), 411-420. Retrieved from http://doi.org/10.1287/msom.1070.0172

Cezik, M. T., & L'Ecuyer, P. (2008). Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2), 310-323. Retrieved from http://doi.org/10.1287/mnsc.1070.0824

Dileepan, P., & Ettkin, L. P. (2010). Not just for large companies: benefits of simulation modeling for a small telephone call center. *Production and Inventory Management Journal*, 46(1), 54-64.

Ernst, A. T., Jiang, H., Krishnamoorthy, M., & Sier, D. (2004). Staff scheduling and rostering: A review of applications, methods and models. *European Journal of Operational Research*, 153(1), 3-27. Retrieved from http://doi.org/10.1016/S0377-2217(03)00095-X

Green, L. V., & Kolesar, P. J. (1995). On the accuracy of the simple peak hour approximation for Markovian queues. *Management Science*, 41, 1353-1370.

Green, L. V., Kolesar, P. J., & Soares, J. (2001). Improving the SIPPapproach for staffing service systems that have cyclic demands. *Operations Research*, 49(4), 549-564. Retrieved from http://doi.org/10.1287/opre.49.4.549.11228

Green, L. V., Kolesar, P. J., & Soares, J. (2003). An improved heuristic for staffing telephone call centers with limited operating hours. *Production and Operations Management*, 12(1), 46-61. Retrieved from http://doi.org/10.1111/j.1937-5956.2003.tb00197.x

Koole, G., & van der Sluis, E. (2003). Optimal shift scheduling with a global service level constraint. *IIE Transactions*, 35(11), 1049-1055.

Mason, A. J., Ryan, D. M., & Panton, D. M. (1998). Integrated simulation, heuristic and optimisation approaches to staff scheduling. *Operations Research*, 46(2), 161-175. Retrieved from http://doi.org/10.1287/opre.46.2.161

Pot, A., Bhulai, S., & Koole, G. (2008). A simple staffing method for multiskill call centers. *Manufacturing & Service Operations Management*, 10(3), 421-428. Retrieved from http://doi.org/10.1287/msom.1070.0173

Robbins, T. R., & Harrison, T. P. (2010). A stochastic programming model for scheduling call centers with global service level agreements. *European Journal of Operational Research*, 207(3), 1608-1619. Retrieved from http://doi.org/10.1016/j.ejor.2010.06.013