Staff Scheduling in Service Systems with Non-Stationary Arrival Processes

by

Samira Shirzaei

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
December 11, 2021

Keywords: non-stationary, non-Poisson, staff scheduling, deep reinforcement learning, service systems

Copyright 2021 by Samira Shirzaei

Approved by

Jeffrey S. Smith, Chair, Joe W. Forehand Jr. Professor of Industrial and Systems Engineering
Konstantinos Mykoniatis, Assistant Professor of Industrial and Systems Engineering
Daniel Silva, Assistant Professor of Industrial and Systems Engineering
Aleksandr Vinel, Associate Professor of Industrial and Systems Engineering

Abstract

There are significant issues that cause customers' dissatisfaction with service systems. One of the most critical frustrations deals with the unpredictable waiting time in the queue before getting service. This issue is even more noticeable in the systems with time-varying arrival rates. Variation in arrival rates demands appropriate staffing level responses. Understaffing leads to higher waiting times in the line, and over-staffing causes higher system costs. This dissertation covers different methods used to solve staff scheduling in this particular type of service system. The first approach is to discretize the time horizon into adjacent blocks. These periods are stationary queuing systems, and we can apply queuing methods to solve these problems. Next, we apply simulation-based optimization to find the staffing level appropriate for the system. Finally, we can use a staff scheduling algorithm to determine each staff member's start and end working time. In the second method, we apply Deep Reinforcement Learning (DRL) methodology to perform staff scheduling. This method has many advantages over the previous one. Firstly, there is no need to estimate the location of change-points in the arrival process since the neural network can learn when and how to change the staffing levels. Next, unlike other approaches that require determining staffing levels first and then require carrying out staff scheduling, staff scheduling can be completed in one step by applying the DRL approach. Lastly, since all studies in the service systems area require data for analysis of their performance, the final objective of this dissertation is to model non-stationary processes by characterizing the arrival process and using the resulting models to generate the data with the same properties with matching the dispersion ratio. We will develop data generation algorithms when the arrival data are independent (renewal). The finding of this work and its future extensions can potentially help service systems like airports and call centers to improve their performance and increase their customers' satisfaction.

Acknowledgments

This dissertation is dedicated to my daughter, Anita, the most important person in my life. She has made me stronger, better, and more fulfilled than I could have ever imagined. I am grateful to my faithful parents, Marziyeh and Mohammad Sharif, whose encouragement and constant love have sustained me throughout my life. I am thankful to my older brother, Dr. Manoochehr Shirzaei, for his constant support, kindness, and encouragement. I am very grateful to my loving husband, Brian, for his supports, especially during the last year.

I would like to express my appreciation to my adviser, Dr. Jeffrey Smith, for his constant support during my PhD career. Indeed, without having his support and mentorship, this journey was not possible. I would also like to thank the committee members, Dr. Jeffrey Smith (chair), Dr. Konstantinos Mykoniatis, Dr. Daniel Silva, and Dr. Aleksander Vinel, for their support and enriching insights.

Table of Contents

List of Figures

List of Tables

List of Abbreviations

DRL	Deep Reinforcement Learning
IID	Independent and Identically Distributed
NHPP	Non-Homogeneous Poisson Process
NN	Neural Network
NSNP	Non-Stationary Non-Poisson
NSP	Non-Stationary Poisson
SBO	Simulation-Based Optimization

Chapter 1

Introduction

1.1 Problem description and significance

There are extreme inefficiencies in the performance of service systems due to staffing levels. However, with proper staff scheduling, we can improve a system's performance. When the arrival rate exceeds the service rate in a system, a queue builds up, and customers are delayed, which causes a short-term imbalance in the system. In this research, we assume the system is stable. If the long-term arrival rate exceeds the long-term service rate, the queue builds infinitely, leading to long-term instability, which is out of the scope of this study. The waiting time of the customers is negatively related to their satisfaction with the service. Since the service rate is a function of the staffing level, controlling the staffing levels is the key to controlling the service rate.

Our focus is on non-stationary non-Poisson processes since non-stationary Poisson processes are well-studied and well-known. However, the models and methods we will develop in this dissertation are general, and they can be used in both systems with non-stationary Poisson and non-stationary non-Poisson arrival processes. Some examples of non-stationary non-Poisson service systems are airports, healthcare systems, and call centers. For instance, arrivals to the security check counters at the airport are generally non-stationary non-Poisson processes since the service time in the previous station, check-in counters, does not necessarily follow an exponential distribution and affects the arrival pattern of the next station.

We characterize the arrival process and determine staffing levels such that arriving customers experience *predictable* waiting time in the queue. Focusing on *predictable* waiting time is not necessarily focusing on obtaining minimum waiting time in the queue. It indicates that the customers experience almost constant waiting times throughout the day.

The unpredictable waiting time in the queue is a significant issue for service systems. In the systems with non-stationary arrivals, this is more noticeable. For instance, at the airport, it forces customers to arrive earlier than necessary to the systems to make sure they will not be late for their flights.

Figure 1.1 represents a queuing system, including the arrival process, queue structure, and service system. The arrival process is the density function that determines the arrival of customers to the queuing system. This distribution function can change during time, which indicates that this process is non-stationary. The time-varying process could have different density functions or the same distribution with different parameters during the time horizon. The arrival rate represents the number of arrivals per time unit. Queue structure is an essential element of the queuing system that shows the order customers are picked from the queue to get service like first-in first-served policy. The service mechanism characterizes the number of available resources and the distribution of the service time. Service rate is the number of customers that can be served by each server per time unit.

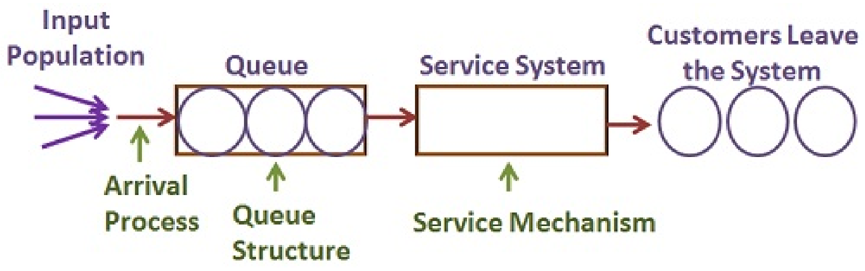


Figure 1.1: Queuing system

Figure 1.2 shows an example of two arrival processes to the system, with the same mean arrival rates in a day. The arrival process on the left,(a), represents the non-stationary process, while the one on the right,(b), is stationary. In case (a), the arrival rate changes noticeably over time, while for the process (b), this rate remains constant at 250 arrivals per hour during the period. In processes like process (a), the numbers of servers should not remain constant over the planning horizon and instead needs to match the arrival rate pattern. The mismatch between arrival rate and service rate either results in under-staffing or over-staffing. Under-staffing results in long customers' waiting times, while over-staffing increases the system costs.

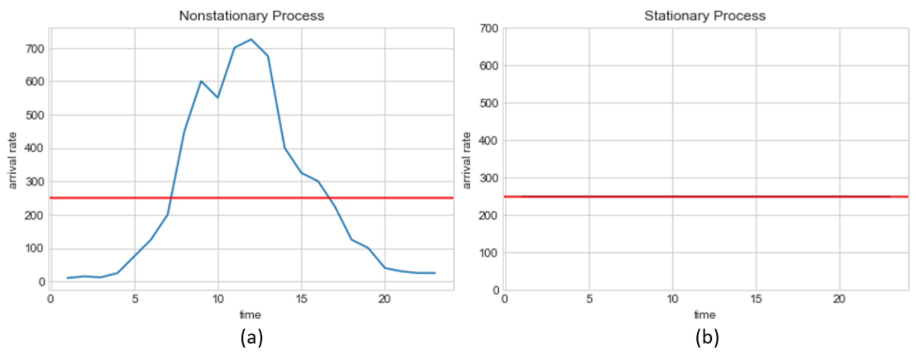


Figure 1.2: Nonstationary and stationary arrival process over a day

For staff scheduling in the system (b), if we assume each server can serve one customer every 3 minutes on average, a constant level of 13 servers per hour will result in relatively short waiting times as shown in figure 1.3 which is the output of this system's simulation model for three replications. Whereas, in system (a), a fixed setting of 13 servers per hour,

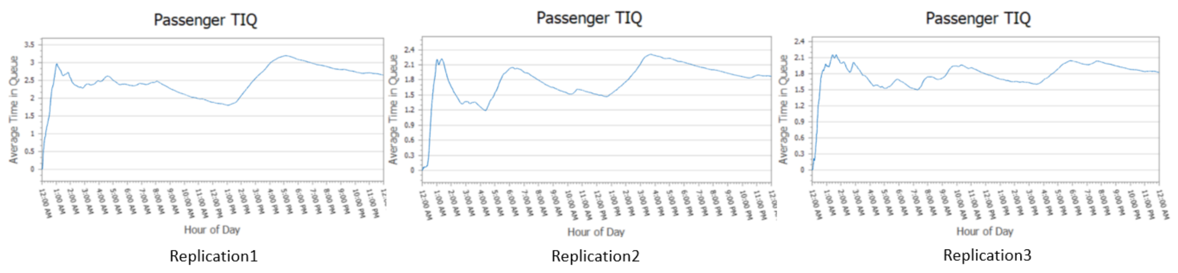


Figure 1.3: The example of airport check-in counters for the average time-in-queue in the system, with stationary arrival process with constant staffing levels over entire day [1]

over a day, will cause several idle resources at the beginning of the day and then a long line of waiting for customers as the arrival rate increases. This phenomenon causes dissatisfaction for customers. Figure 1.4 shows the number of customers in the queue resulting from simulating the system (a) for three replications when the staffing level is constant over the day. When the arrival process is time-varying, the most common approach (we will discuss the rest in chapter 2) to deal with non-stationarity is discretizing the time horizon into disjointed, shorter periods for arrival rates assumed to be stationary. For this purpose, we have to detect the change points from the boundaries of the periods in the discretized time horizon.

In the non-stationary arrival process, the problem of staff scheduling is more complicated. Figure 1.5 shows a piece-wise stationary graph of the arrival process in system (a),

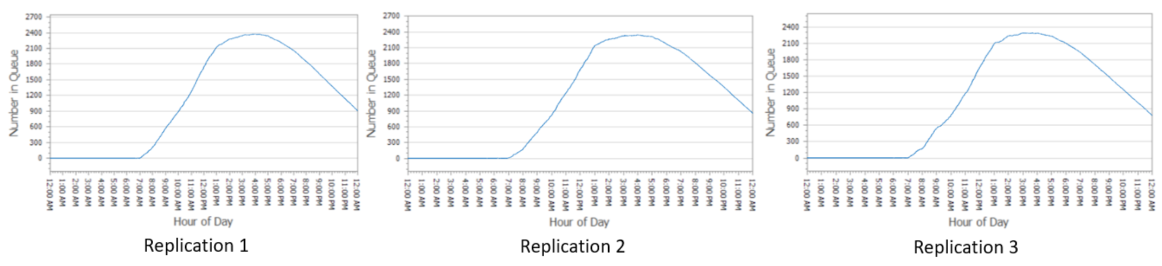


Figure 1.4: Average number-in-queue in the non-stationary system with constant staffing levels during the day

in which we divided the day into 24 equal-length segments. Next, we can consider each of these parts as a system, like a system (b), and determine the staffing level for each one independently. In this system, we assumed that the changing points in the arrival process are hourly. However, in real-world systems, these change-points are not always equally spaced and generally do not start and stop at known times (i.e., on-the-hour as shown above), and that is where the change-point detection process comes in. In chapter 3, we will apply the proposed algorithm in [2] to detect these change-points.

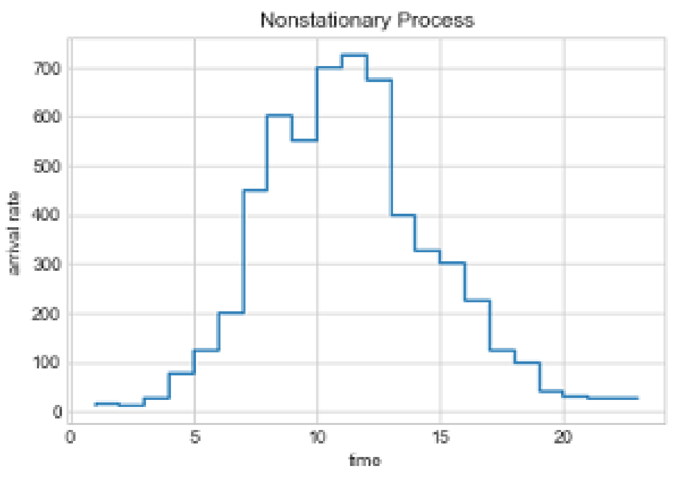


Figure 1.5: Discretizing the arrival process of system (a)

Some non-parametric approaches detect change-points in the time series without any information about the type of distribution function. These algorithms estimate the number of change-points and of the place they occurred [2]. The non-stationary process consists of changes in the type of distribution function or some of the parameters. Figure 1.6 shows data sets with some change-points. The X-axis represents the time, and the y-axis illustrates the value of parameters in each case. The vertical lines in each graph illustrate the position of these changes. Data set 1 represents the case that the mean of the arrival process distribution is changing over time. Data set 2 illustrates the case with changes in the variance of the arrival time. Changes in the co-variance are presented in data set 3. Finally, data set 4 presents the changes in the frequency of the sinusoidal function in the data set with the sinusoidal pattern. These examples are about situations in which the distribution function parameters change over time, but the type of distribution family does not change (i.e., the distribution function is a normal distribution in the entire time horizon). Related works

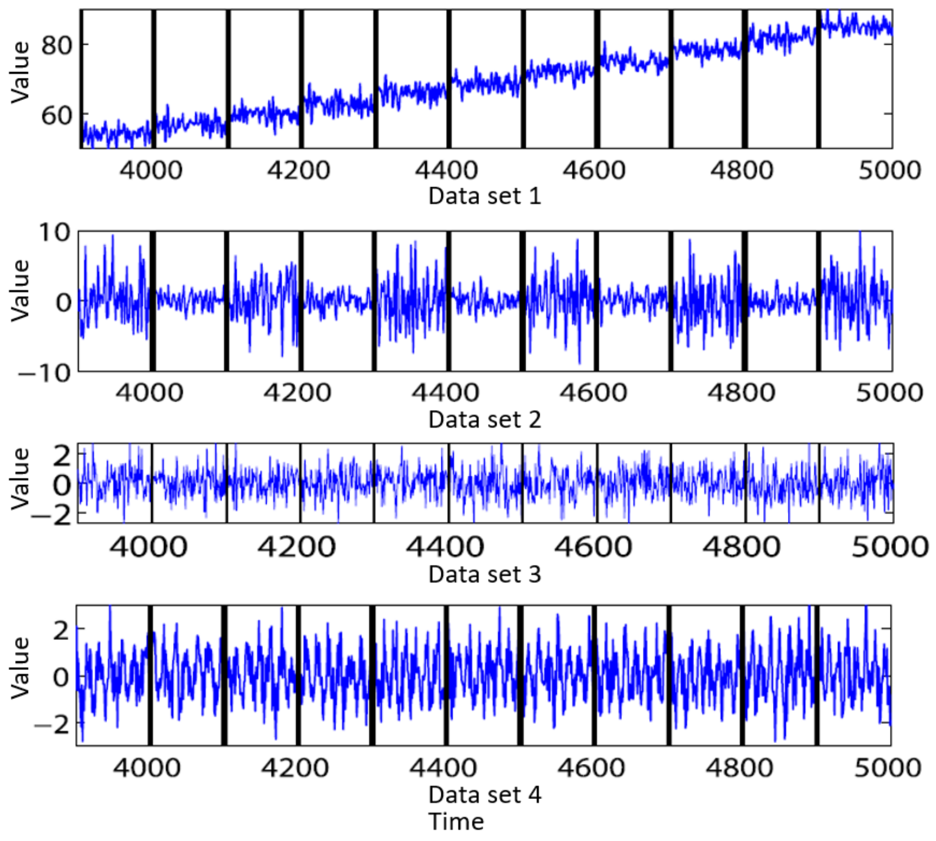
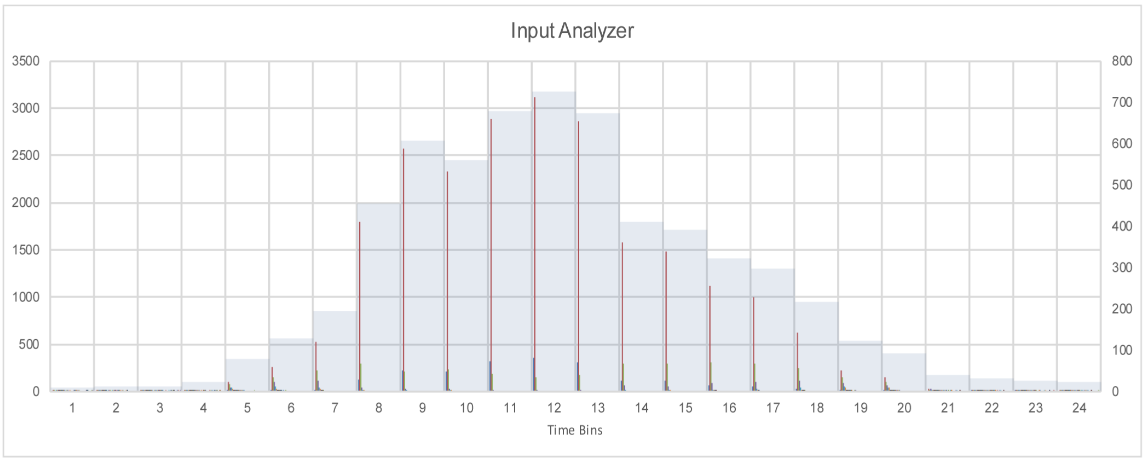


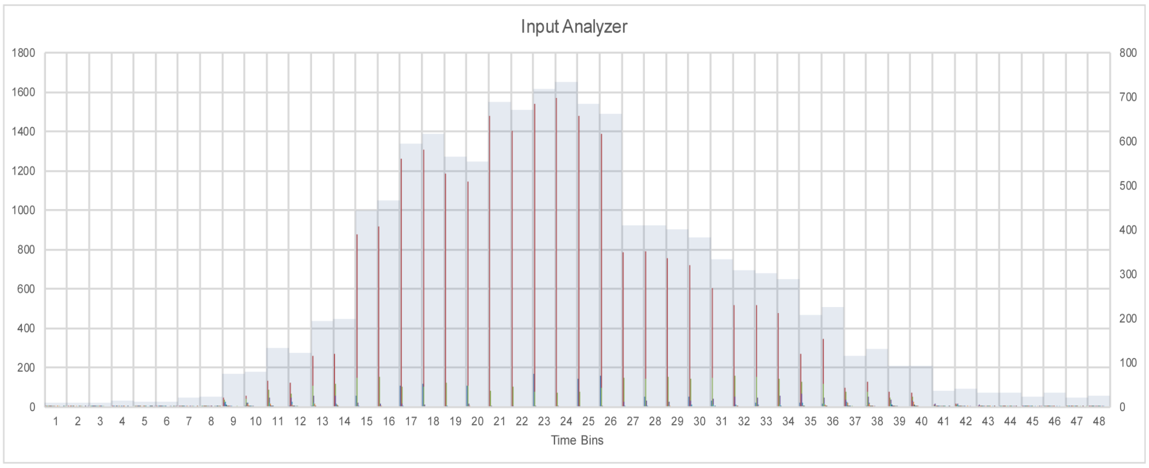
Figure 1.6: Examples of time series with change-points [3]

considered the equal-length segments when converting the time horizon into a piecewise stationary time series. In the first method used in this research, we will use the algorithm proposed in [4] to determine the change points in the time-varying arrival process. This algorithm can identify changes in the distribution function. Then, by applying methods for density function estimation, we can determine the distribution function of the data in these segments. In our research, we assume the arrival process follows a general, time-varying distribution function.

In some non-stationary arrival processes, the arrival rates change gradually over time. Therefore, change-point detection might not detect the changes immediately at the exact location that those changes took place. In this case, we suggest using some visual tools like histograms. Ansari et al. [5] proposed a tool called HistoRIA that can plot various rate histograms of the arrival data with user-defined time block lengths. This tool can help estimate the location of change-points in the type of data sets that changes in rates occur slowly. The figure 1.7 extracted from our paper [1] shows the histogram of interarrival times for two cases, first 1-hour time block and then 30-minutes time buckets. For instance, time blocks 27 and 28 have almost the same arrival rates in the bottom image. However, rates in time buckets 15 and 16 are slightly different. If the changes are in the distribution function, visual analysis cannot detect changes, and a change-point detection method is required.



1- hour time block



30-minutes time block

Figure 1.7: Example of arrival rates in 24 hours with 1-hour and 30-minutes lengths time blocks

Another approach to solving staff scheduling problems is to apply learning methods without partitioning the arrival process into approximately stationary segments. This approach is based on reinforcement learning. First, it trains the Neural Network (NN) to find the best possible policy. Then it applies the trained policy into the simulation model to find the optimal staff scheduling. This training process improves the efficiency of the staff scheduling procedure since it can be completed in one step, and it does not require performing change-point detection on the arrival process.

Most research related to determining staffing levels needs arrival times data or arrival rates over the planning horizon as the input. This dissertation endeavors to characterize the arrival process by developing models and then, by applying these models, propose algorithms to generate data with the same properties. Motivated by the above, this dissertation aims to provide an integrated and joint analysis of non-stationary processes, including developing models and data generation algorithms as well as staff scheduling of service systems. Specifically, the stream of research performed in this dissertation investigates the following research questions that the current literature left unanswered:

1. What is the effect of using the average of waiting times in the entire day to staff scheduling instead of the average of each time block on the customers' waiting times?
2. Does performing change-point detection change the result of staffing levels and customers' waiting times?
3. How predictable waiting times can be achieved in the service systems with a non-stationary arrival process?
4. How training is performed in the service system's staff scheduling specifically with the non-stationary arrival process?
5. What are the inputs to the training model to develop staff scheduling algorithm in non-stationary service systems?
6. How to use the DRL to solve staff scheduling problems?
7. How to develop models to generate non-stationary data?
8. How to generalize the process of data generation and modeling of non-stationary processes in the way it can produce both non-stationary non-Poisson and non-stationary Poisson arrival data?

1.2 Research objectives and contributions

Three main objectives are considered in the present research. First, we determine the non-stationary service systems' staffing levels to achieve predictable waiting times. If the only constraint is customer satisfaction, we will expect the shape of the arrival process and the staffing levels' distribution functions to look almost the same. Figure 1.8 shows this situation. On the left is the arrival rate distribution, and on the right are the final staffing levels, which follow the same behaviors.

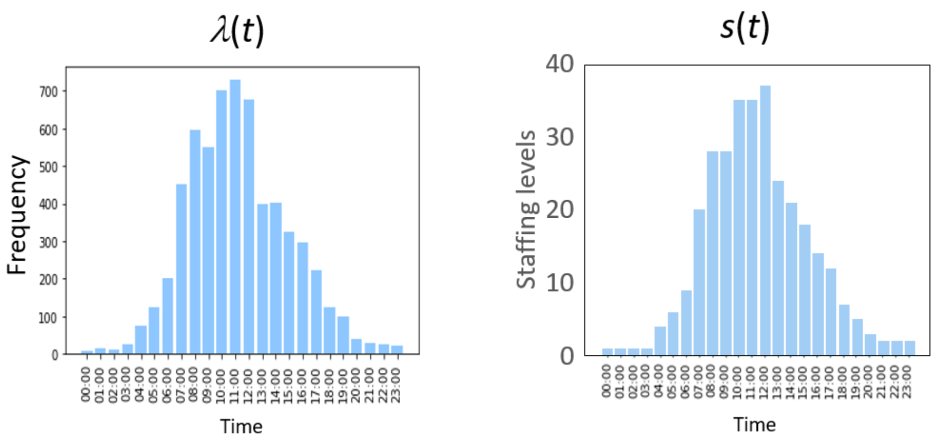


Figure 1.8: Arrival rate and corresponding resource schedule

After performing change-point detection, we define the number of servers in each time bucket. Then, we add the constant shift length constraint in order to determine the staff scheduling. We consider another constraint in the model to achieve predictable waiting times. This constraint will minimize the variations in the waiting times and force the waiting times in time blocks to be as close as possible. We consider the average and percentile of the waiting times as the functions that define the service level. These functions are the most common in the reviewed literature. For the average, there is an approximation formula used to solve the staffing scheduling problem. Since there is no estimation formula to calculate the percentile of waiting time over a day, we will use the simulation-based optimization (SBO) method to get the optimal solution.

The second objective will expand the problem to include flexible shift scheduling with job rotation. By flexible shift scheduling, we mean the workers can start working at any time of the day, and there are no predetermined starting times. Job rotation indicates a worker can switch a job that it is currently performing after l hours and start another job in the network. We will determine the beginning of the shift for each server with working hour limitations at a specific job. In this case, we do not consider the predetermined beginning of the shift for all servers. However, instead, we determine the start time of each server. One example is the scheduling of doctors' shifts at the hospital. Instead of having predefined, fixed start times like 7:00 a.m. and 7:00 p.m., we determine each doctor's beginning of the shift. This model will allow us to have flexible start times. We will set another constraint to obtain predictable waiting times in the queue. To accomplish this, We set an upper limit for the variation of waiting times over a day.

Stationary approximation approaches are most commonly applied for performance evaluation in time-varying systems [6]. In these approaches, they segment the time and use a series of stationary models to determine the staffing levels [7, 8, 9, 10, 11, 12, 13]. This dissertation is the only research that solves the staffing problem without stationary approximation to the best of our knowledge. We apply Deep Reinforcement Learning (DRL) to solve the staffing problem. In this approach, the Neural network (NN) learns from the environment (simulation model) and makes necessary changes to the number of staff members.

The third objective considers modeling and generating data of the non-stationary processes. The arrival process is one of the main components of the simulation models' input to simulate systems like airports, hospitals, and supply chain systems. Since the arrival process to some systems like call centers and healthcare systems are not Poisson [14], there is a need to go beyond the Poisson process. We develop general models to characterize both the non-stationary non-Poisson and non-stationary Poisson processes. Our proposed model and data generation methodologies are applicable for the case the arrival data are independent.

1.3 Organization of dissertation

The chapters of this dissertation are organized in the following way. Existing related works are presented in chapter 2. A method of staff scheduling in a service system with a non-stationary arrival process is discussed in chapter 3. Chapter 4 will describe the DRL methodology used to perform flexible staff scheduling in a non-stationary service system. The proposed method will achieve predictable waiting times objective. Modeling and generating data of the non-stationary arrival process are developed in chapter 5. This chapter will establish methods to model and generate independent data from renewal processes. Concluding remarks and future works will be discussed in chapter 6.

Chapter 2

Literature Review

2.1 Introduction

This chapter presents an analysis of the gap and strength of the stream of research on four main areas, namely input analysis, change-point detection, non-stationary data generation, and determining staffing levels in service systems with non-stationary processes. Obtaining the optimal staffing levels has a direct relationship with the arrival process to the system. In the time-varying arrival process, characterizing the input process to the system is more critical since any departures from the stationary arrival process require different staffing levels during the planning horizon.

When the arrival rate is higher than the service rate, more servers are needed to continue working without facing a short-term imbalance. Since data from the real world is not available most of the time, or even if it is, the available data is not sufficient to replicate enough number of the simulation model to reduce the variability in the outputs, generating data to analyze the system is an important subject. There is a long history of generating data from Poisson distribution [15], but when it comes to generating data from non-stationary non-Poisson processes, there are a few works in the literature. This chapter describes the related existing works and how we will improve their shortcoming in this dissertation.

2.2 Input process analysis

The input modeling process is how the arrival process in the real-world system is translated into mathematical models. In the simulation, there are several places where specifying probability function is needed to represent random numerical inputs, like inter-arrival times, service times, machine failures, travel times, the demand of the products, and many other examples. Input modeling consists of four main steps: 1- Assessing the IID assumptions. 2- Select some candidates for density function according to the characteristics of the process and graphical experiment of the data. 3- Determine the value of the model's parameters. 4- Assessing the goodness of fit [16]. Most simulation textbooks assume either the IID assumptions are met, or the process is non-stationary. The process that has random parameters and random observations is stochastic. Stochastic processes can be classified into stationary and non-stationary ones [17]. In many queuing systems, the arrival rate changes over time. Examples are the passengers arriving at the check-in counter in the airport, fast-food restaurants over a day, and emergency rooms over a flu season. Ignoring non-stationarity in the arrival process can cause underestimation or overestimation in the output results, just as ignoring correlation across inputs can cause the same issue in determining output values [16]. To characterize this non-stationarity, Green et al. [18] numerically investigated the effect of non-stationarity on the performance of multi-server queuing systems with exponential service times and sinusoidal arrival rates. They determined when and how stationary model approximation can be used for the non-stationary process. The results showed that stationary models could noticeably underestimate the delays when the system is only non-stationary. In addition, they showed expected delay and probability of that increase by increasing the amplitude of the sinusoidal arrival rate function. Another finding is that the expected queue length and probability of delay also increase when the sinusoidal arrival rate function frequency increases. A modification of the simple peak hour approximation (SPHA) for estimating peak congestion in the systems with exponential service times and time-varying periodic Poisson arrivals is proposed in [19]. By SPHA, they mean the peak in the curve of the point-wise stationary approximation (PSA). They considered $M(t)/M/s$ systems with a sinusoidal arrival rate function for the analysis. They compared the lagged point-wise stationary approximation (lagged PSA) with SPHA using the probability of delay and found that lagged PSA is always more accurate than SPHA, and the results have smaller errors when average service times are greater than half an hour in 24 hour period. They concluded that PSA could support better capacity decisions than SPHA to achieve a targeted probability of delay when the service rate is less than two since the SPHA is inaccurate. However, this method is applicable for the cases the arrival process is Poisson and can not be used in general arrival processes. While simulating a process, there is a need to generate data. A non-parametric estimate for the mean-value function of non-homogeneous Poisson processes (NHPP) with the long-term trend or cyclic effects that display non-trigonometric characteristics in some cases was developed in [20]. They proposed a multi-resolution algorithm that uses the inversion method to simulate the non-stationary Poisson process. They continue this work in another one which is presented in [21]. They considered modeling and simulating arrival processes that may show a long-term trend with periodic phenomena (like daily and weekly cycles) or both types to automate the proposed algorithm of [20]. They developed some steps in order to achieve this purpose. First, they assumed the arrival process follows non-homogeneous Poisson processes. Second, to show the accuracy and flexibility of the automated multi-resolution procedure, they used 100 independent replications of eight selected test processes, applied comprehensive experimental performance evaluation, and developed an estimation for the mean-value function when the arrival process has a long-term trend. A nonstationary non-Poisson arrival process was developed in [22]. Simulation of time-varying non-Poisson processes is not straightforward, and for generating this type of process, we have to use the inverse of the cumulative density function. Because the inverse function is often unavailable, they constructed an accurate approximation of it and discussed application issues. In some cases, simulation models of real-life systems assume a stationary Poisson process, and when the process is time-varying, it is natural to assume a non-stationary Poisson process. However, this assumption causes an inaccurate representation of the arrival process in many systems, leading to higher or lower variability in the arrival process than its reality.

2.2.1 Conclusion

Present works analyzed several options when the arrival process is a non-stationary Poisson process, including different arrival rate patterns. However, there is still a gap in characterizing non-stationary non-Poisson arrival processes with various trends like long-term and short-term ones. The arrival process in most of these systems does not follow the Poisson distribution, and assuming it is Poisson, it can reduce the outputs' accuracy.

2.3 Generating data of non-stationary processes

In the stochastic simulation modeling of complex systems, the arrival processes are one of the parts that must receive special attention. Sometimes, the arrival rates change over time. Depending on the system, these rates could change hourly, daily, weekly, yearly, or follow some other trends [23]. In the non-stationary Poisson process's case, when the rate function and associated mean-value function are given, two well-known methods in the literature are inversion and thinning to generate data. The thinning method in the simplest form was proposed in [15]. This method is based on deleting the points whose corresponding rate function is greater than the upper bound of the rate function. The thinning method with a given rate function can be implemented computationally simply to construct a homogeneous process. The rate function of this process is the maximum value of the given rate function. Because of the additive properties of the Poisson process, the method of thinning is working. Most commercial simulation packages used the thinning-based method to generate non-stationary Poisson data. There are many systems where the arrival process does not exhibit Poisson distribution and even is not approximately close. Gerhardt and Nelson [24] extended techniques for transforming a stationary Poisson process into a non-stationary Poisson arrival process (NSPP) by transforming a stationary renewal process into a non-stationary, non-Poisson process (NSNP). They provided an algorithm for using thinning and inversion methods. The base process in their inversion algorithm has the general cumulative distribution function, G. They claim X_1, the time until the first event, may not follow distribution G. They used G_e to represent the cumulative distribution function of the first arrival event. $\lambda(t)$ represents the arrival rate function and its cumulative distribution function is $\Lambda(t) = \int_0^t \lambda(u)du$.

1. Set $V_0 = 0,$ index counter $n = 1$. Generate $S_1 \sim G_e$. Set $V_1 = \Lambda^{-1}(S_1)$
2. Return interarrival time $W_n = V_n - V_{n-1}$
3. Set $n = n + 1$. Generate $X_n \sim G$. Set $S_n = S_{n-1} + X_n$ and $V_n = \Lambda^{-1}(S_n)$
4. If $S_n \geq T$ stop; otherwise go to step 2.

Where for $s \in \mathbb{R}, \quad \Lambda^{-1}(s) \equiv inf\{t : \Lambda(t) \geq s\}$.

The transformation preserves specific properties of the marginal variance and dependence structure of the base process. The researchers' proposed method is useful when the mean-value function is easily invertible, $\mu(t)$, $\mu(t) = \int_0^t \lambda(u)du$ [23]; otherwise, we can not use this method directly to model the process. One approach to solving this issue could be to divide the time horizon into lower piece-wise linear arrival rates, easily invertible. As a particular case, they analyzed the resulting process in the non-stationary Poisson process.

They began by setting a value $\lambda^* \geq max_{t\geq 0}\lambda(t)$. Next, they assigned a stationary renewal process the arrival rate λ^* and distribution G (G_e for the first event) with mean arrival time $(\lambda^*)^{-1}$ and variance equal to $C^2/(\lambda^*)^2$.
Their proposed thinning algorithm for a non-stationary renewal process is as follows:

1. Set index counters $n = 1$, $k = 1$, and $T_0 = 0$. Generate $S_1 \sim G_e$
2. Generate $U_1 \sim$ Uniform$[0, 1]$. If $U_1 \leq \lambda(S_1)/\lambda^*$, then
    a. Set $T_1 = S_1$
    b. Return interarrival time $Y_1 = T_1 - T_0$
    c. Set $k = 2$
3. Set $n = n + 1$. Generate $X_n \sim G$. Set $S_n = S_{n-1} + X_n$.
4. Generate $U_n \sim$ Uniform$[0, 1]$. If $U_n \leq \lambda(S_n)/\lambda^*$, then

a. Set $T_k = S_n$

b. Return interarrival time $Y_k = T_k - T_{k-1}$

c. Set $k = k + 1$

5. Go to step 3.

One disadvantage of this method is that it may be computationally inefficient if $\lambda(t) \ll \lambda^*$ over a substantial range of values for t, resulting in a relatively large number of rejections [23]. In general, for non-Poisson base distribution, C (dispersion ratio) of generated data from a thinning method does not converge to C of the base process.

For each thinning and inversion method, they proposed a technique for specifying a renewal base process. However, the proposed method does not cover the cases with dependent arrivals to the system. For this purpose, in another work, they developed a tool for generating and defining the time-varying, non-renewal arrival process for simulation was constructed in [25]. The user's inputs required to provide are desired piece-wise constant arrival rate, cv^2, and lag-1 autocorrelation of the base process. This tool is appropriate for evaluating the effect of non-stationary, non-exponential, and non-independent arrivals on simulation outputs and performance. The drawback of this tool is that it requires a piece-wise stationary arrival process or easily invertible arrival function, which makes the application of this tool very limited.

Liu et al. [23] proposed an algorithm to improve drawbacks in [24]. In their work, they showed the importance of dispersion ratio (variance-to-mean ratio; $C(t) = \frac{Var[N(t)]}{E[N(t)]}$ in which $N(t)$ is representing the arrival process) for modeling non-stationary non-Poisson processes. They developed the model by combining thinning and inversion methods. First, they approximated the majorizing rate function, which is piece-wise constant. Second, they computed its associated mean-value function. The next step generated an equilibrium renewal process whose noninitial inter-renewal times follow Weibull distribution with variance equal to dispersion ratio and the mean equal to one. Then they inverted the majorizing mean-value function to generate majorizing non-stationary non-Poisson process (NSNPP) and used the thinning method to have NSNPP with the given arrival rate function. We will present more details about their proposed algorithm in section 5.

2.3.1 Conclusion

One of the limitations of the proposed method in [24] is that it is computationally inefficient if $\lambda(t) \ll \tilde{\lambda}$ for noticeable range of t values; in which $\lambda(t)$ is the time-varying arrival rate and $\tilde{\lambda}$ is the upper bound for $\lambda(t)$ [26].

For constructing piece wise-constant majorizing function of the arrival rate function, $\lambda(t)$ in [23], they divided the time horizon to 200 equal-length sub-intervals. One drawback of this approach is, depending on the complexity of the arrival rate function, this number of sun-interval may vary, and it demands more computation to determine the optimal number in each specific arrival rate function. Chapter 5 will develop models and data generation algorithms to overcome these limitations in non-Poisson non-stationary data generation.

2.4 Change-point detection

Change-points are when the probability distribution of a time series of stochastic process changes, and change-point analysis is the process of detecting these changes. Detecting these change points is helpful in modeling and predicting time series. Changepoint detection has broad application areas such as human activity analysis, climate change detection, and image analysis [27]. Liu et al. [3] developed an algorithm to detect change-points in non-stationary time series. They used relative Pearson divergence as a divergence measure using direct density-ratio estimation for their purpose. The proposed relative unconstrained least-square importance fitting (RULSIF) method and experiments on artificial and real-world data sets proved its performance. A method to implement multiple change point analysis of multivariate observations when they are independent was proposed in [2]. This method can detect any type of changes in distribution without any assumptions of the existence of αth absolute moment (α is the moment index used for determining the distance between and within segments). Their method can also estimate the number of changes and their locations, and it does not need prior knowledge or additional analysis. They assumed the unknown number of change points is constant, using the maximization of goodness-of-fit statistics to estimate that.

2.4.1 Conclusion

After executing change-point detection algorithms in section 2.4, we realized method proposed in [3] has a delay in detecting the change-points. In addition, when the arrival rate changes over time, but their values are close to each other, for instance, 12 arrivals in first time block and 15 arrivals over the next time bucket, both methods performed poorly in detecting the location of change-points. Developing an algorithm to detect change points is out of this dissertation's scope since this dissertation focuses on staff scheduling and non-stationary arrival data generation.

2.5 Determining staffing levels

Staffing drives costs and service quality in most service systems. Therefore, determining staffing levels for these systems plays a critical role, as real-life systems have many sources of variability. A literature review on staffing and scheduling approaches when the demand is non-stationary was performed in [6]. They considered references published during 1991-2013, classified them by system assumptions, performance evaluation methods and performance metrics, optimization approach, and application area. The performance evaluation methods and metrics are a stationary approximation, simulation, numerical methods, fluid models, and empirical methods. In the optimization approach, they classified the references to staffing optimization and shift schedule optimization classes. We applied their idea for classification by performance evaluation methods and performance metrics.

2.5.1 Stationary approximation

Stationary approximations are the most commonly applied approach for performance evaluation in time-varying systems [6]. These approaches segment the period and use a series of stationary models to determine the staffing levels.

Izady et al. [7] set the minimal hour-by-hour medical staffing levels for achieving the government target (reducing patients' waiting times) in the presence of complexities like time-varying demand, multiple types of patients, and resource sharing. They applied hourly arrival rates to the total annual attendance of 87,000 patients (an average-sized A&E in the UK). A day-of-week effect was not observed in the survey data. They suggested an iterative scheme that uses infinite server networks, the square root staffing law, and simulation to develop a good solution. Square root staffing law was proposed to achieve the target probability of α, and it can be calculated as follows: $s(t) = \lceil m_\infty(t) + \beta\sqrt{m_\infty(t)}\rceil$, where $m_\infty(t)$ is the time-dependent offered load, and β is the quality of service parameter. The time-dependent offered load function $m_\infty(t)$, estimated by the average number of busy servers in $M_t/G/\infty$ queue. In this system, the arrival process is a non-homogeneous Poisson process, service time follows general distribution, and there are infinite servers.

An optimal staffing level in the multi-skill call center by minimizing the cost of servers and service level constraint by using an iterative cutting-plane algorithm was determined in [8]. They expressed service levels as staffing functions for a fixed sequence of random numbers driving the simulation. They assumed the arrivals are stationary Poisson process with rate λ. They explored the difficulties encountered with more significant problem instances and developed (heuristic) methods to develop these problems practically. These heuristics include getting an initial set of constraints by solving a max-flow problem, computing finite differences with steps larger than one (selected adaptively), and rounding up the solution of an LP instead of solving the exact IP. In addition, the methodology replaces the unknown expectations in the constraints of the original problem by sample averages. They relaxed the nonlinear service-level constraints and progressively added linear constraints until the optimal solution satisfied all service-level constraints. They showed their proposed approach's performance by testing it on moderate and large centers. This method is not applicable in this dissertation as both service time and interarrival times follow exponential distribution. A staffing problem with uncertain arrival rates was considered in [9]. The objective is to minimize the total cost of agents under some chance constraints. They assumed the arrival process over a day for a specific type of call is time-homogeneous Poisson which its rate can vary day-to-day. The service level in their problem is a fraction of answered calls over a specific time block. As the first step, they determine initial staffing based on the imperfect forecast of the arrival rates. This staffing can be corrected with recourse actions by adding or removing agents at the price of some penalty costs. For the solution method, they combine simulation, mixed integer programming, and cut generation. They consider chance constraints that require the QoS over a day to satisfy a minimum probability threshold instead of constraints defined on the expectations, which are never observed in reality. They assume a day with only one period, and the arrival rate stays constant throughout the day. For probability functions of the chance constraints, they used simulation to approximate their empirical values. Another study in staffing problems with probabilistic constraints in an emergency call center was investigated in [10]. In this problem, there is one set of staff groups. They assumed these characteristics for the system: (i) All agents are identical and can answer all call types, (ii) the average service time is low compared to other call centers, and the buildup of a queue is very rare. They formulated an average sample approximation (SAA) and then proposed a fast and straightforward simulation-based heuristic algorithm to obtain a good solution for this SAA. The idea of the method is to first find the ”right” number of agents period by period, then adjust for interaction and global constraints via simulation. The arrival processes can be arbitrary but are usually non-stationary Poisson with random arrival rates, dependent across periods. Finally, the day is divided into P periods of equal length. The performance measure in this problem is average waiting time which in chapter 3, we will show is not a good enough performance measure as the overall average is unbiased in terms of an outlier (in this case high) waiting times in some time blocks. Some ways to cope with time-varying demand in determining staffing levels problems were discussed in [11]. To present a single-skill varying demand, they consider the $M_t/GI/s_t + GI$ model, a non-homogeneous Poisson process with varying arrival rates and independent and identically distributed service times. Staffing requirements in service systems have been considered in [28]. Their focus was on the queuing system, $M/GI/s_t + GI$, which has a non-stationary Poisson arrival process, general service time, and general abandonment probability distribution. They adapted stationary queuing models for use in non-stationary environments to capture time-dependent performance and set staffing requirements. A non-Poisson non-stationary arrival process that includes, as a particular case, the non-stationary Cox process (a modification of a Poisson process in which the rate itself is a non-stationary stochastic process) was investigated in [13]. They modified the many-server heavy-traffic approximation for non-Poisson arrivals and tried to determine staffing levels to achieve the same service level as the non-stationary Poisson arrival case. They used simulation experiments with non-stationary Markov modulated Poisson arrival processes with sinusoidal arrival rate functions to demonstrate that the staffing algorithm effectively stabilizes the time-varying probability of delay at designated targets.

Methods to perform time-dependent staffing for many-server queues were investigated in [29] to achieve time-stable performance when the arrival process is time-varying. It resulted in that it suffices to target a stable probability of delay. Their focus was on the $M_t/M/s_t + G$ model with customer abandonment and a non-homogeneous Poisson arrival process. They showed simulation experiments that the iterative-staffing algorithm (ISA) leads to time-stable delay probabilities across many target delay probabilities. In addition, they showed that with ISA, other performances like agent utilization, abandonment probability, and average waiting times are stable. They compared the performance of ISA with PSA (Point-wise stationary approximation). Our problem in chapter 3 considers the average and percentile of the waiting time in each time block instead of the overall average over a day. We will discuss how that affects the performance. Green et al. [30] evaluated the practice of determining staffing requirements in service systems with random cyclic demands using a series of stationary queueing models. They considered Markovian models with sinusoidal arrival rates and used numerical methods to show that the commonly used ”stationary independent period by period” (SIPP) approach to set staffing requirements is inaccurate for parameter values corresponding to many real situations. They showed that SIPP could result in staffing levels that did not meet target delay probability and identified domains for which SIPP will be accurate. They proposed two simple modifications of SIPP that will produce reliable staffing levels in models that span a wide range of practical situations. The relative amplitude was identified as the major influencing factor in the reliability of the method. A quick overview of issues like building realistic models and developing tools to simulate these models, and finding a good approximation for staffing levels in call center management was given in [31]. First, they discussed a two-step approach for determining the staffing levels in a multi-skill call center, which first computes the optimal staffing for each period and second to cover this staffing by a set of working shifts to achieve minimum cost. They mentioned that this approach is producing sub-optimal solutions. They considered an upper bound for the waiting times as the service level. This paper did not execute any change-point detection on the arrival process and assumed changes are every half an hour. The problem of dynamic assignment of resources in the presence of time-varying demand with application to tactical control of scarce resources at a busy airport was investigated in [32]. They developed heuristics approaches based on an approximation of the queuing dynamics considering busy times of the day.

2.5.2 Simulation-based methods

An overall view of call center simulation models and focuses on typical inputs, challenges in modeling, and key outputs were presented in [33]. They defined different scenarios for the simulation model, and they figured out the correct levels of cross-training to meet the service level goals with the current staffing level, and they examined trade-offs between different scenarios regarding critical outputs of the model. Finally, they performed a what-if analysis for situations like the impact of an increase in call volume on the system or the value of adding an outsource to help over peak months. Another methodology that applies system simulation combined with optimization to determine an optimal number of lab technicians, nurses, and doctors that maximizes throughput and reduces patient time in the system with considering the budget as the constraint was presented in [34]. Their main objective in this paper is to provide a decision support tool to evaluate the impact of different staffing levels on the system's efficiency. Using experimental scenarios showed that optimal staffing generated from their model generates a 28% increase in patient throughput and, on average 40% reduction in patients' waiting time. In this work, for optimization purposes, they traced the average time in the system of the patients. Our research will show that if the average is less than the upper bound, it does not mean necessarily all individual time in systems is less than the upper bound, and there could be some very high values of time in systems and still the average be acceptable. An Arena simulation model was considered in [35] for a specific hospital's emergency department consisting of two areas: a full-service emergency room and emergency care express. The model allowed having 13 various types of patients and evaluated different feasible schedules for doctors, nurses, and technicians. They used the model to evaluate the schedules and not determine them. The concern was about the average patient time in the system, which was about 142 minutes. Data for the model building was collected on treatment types for different patients, patient arrival, task duration for various required treatment tasks, and resources. The primary performance metric used for the evaluation process is the average patients' length of stay in the emergency department. As the input to the simulation model, they did the experiments with five schedules. Only the numbers of nurses and technicians varied in these schedules, and the number of doctors on duty was kept constant. They ran each of these schedules for five replications and compared the results regarding time-in-system for patients.

2.5.3 Numerical methods

The heuristic SIPP (stationary independent period by period) approach was improved in [36] for those call centers with limited hours of operation on workdays. They showed that the staffing suggested from the SIPP approach is considered too low to achieve the targeted customer service levels in critical periods. They mentioned two main reasons for that. First, SIPP fails to consider the time lag between the peak in customer demand and when system congestion peaks, and second, SIPP's assumption about constant arrival rate during the period. Their result showed that the same approximation for lag used in the system working 24 hours of 7 days of the week is good enough for most of the limited hour systems and the lag approach is a reliable, easy way to implement for scenarios in which SIPP is possible to be unreliable. Another method that iterates between the schedule generator and schedule evaluator is introduced in [37]. Schedule generator solves a series of integer programs to produce improved schedules and, by adding constraints, tries to eliminate infeasible solutions without removing optimal solutions. They mentioned that the most apparent application of their suggested approach is that the approximate approach generates infeasible solutions (service level constraints are not satisfied). Therefore, they tried to test their method in situations where the approximate approach is least reliable. Also, their approach can be used for verification. If the approximate approach is accurate sufficiently, their suggested method should terminate after one iteration.

The performance of seven methods was compared in [38] in computing or approximating service levels for time-varying M(t)/M/s(t) queuing systems, including exact method, randomization method, closure approximation, direct infinite-server approximation, modified-offered-load infinite-server approximation, effective-arrival-rate approximation, and lagged stationary approximation. They classified these methods into three main categories. First, the methods are slow but highly accurate. Second, methods that are fast but less accurate than those in the first category and third the intermediate category with modest computation times and acceptable accuracy. They claimed that exact and randomization methods are expected to fall in the first category. The effective-arrival-rate approximation, infinite-server-based approximations, and lagged stationary approximation to be in the second category. Finally, closure approximation is expected to be in the third category. A methodology was presented in [39] to solve the problem of flexible shift scheduling when hospital managers can accomplish flexible start times, overtime, and variable shift length to answer the demand to find minimum assignment cost in the planning horizon of up to 6 weeks. They considered a wide range of legal constraints, individual preferences, and on-call requirements during the week. The proposed model can construct the shift implicitly instead of starting with a predefined set of various shift types. The Branch-and-Price (B&P) algorithm is developed to find high-quality rosters. In this work, they did not consider and evaluate the waiting time of patients in the system. Another method for scheduling staffing levels was developed in [40]. A genetic algorithm uses the implementation of searching good schedules and evaluating the service level resulting from a schedule. The algorithm solved the equations of motion for a non-stationary queuing system and considered a bi-criteria problem, where two criteria are minimum instantaneous service level and cost. They compared their approach with using traditional SIPP to set staffing requirements and an integer program(IP) to choose shifts, and they concluded that the traditional approach could overestimate the service level. Their method sometimes generates schedules that result in higher service levels and lower labor costs than the SIPP-IP approach. Another benefit of this approach is its usage in situations where the system capacity to serve customers is temporarily less than the arrival rate to the system (rush hours). The number of staff was determined in [41] in an organization responsible for blood collection in the Netherlands to be donor-friendly, and part of this perception is the experience of waiting time. Therefore, they considered M/M/s queuing system and suggested the two-step procedure to determine staff capacity. First, they applied queue theory to compute the minimum staffing levels for every 30 minutes. Then, they used these minimum staffing levels to determine the optimal length and starting times of the shifts by applying integer linear programming. In this paper, they considered the average waiting time during the entire planning horizon, and they also did not consider the effect of staffing in each time block on its previous and next one. Under-staffing in one time block causes customers to stay longer and affects the following time blocks' number of staff members and waiting times of customers.

2.5.4 Fluid models

A simple non-stationary model for a call center was studied in [42]. They simulated a call center with 32 servers with exponential service times with an average of 240 seconds. They simulated two cases, both with the first overload and then underloaded. In the first case, they used a Poisson distribution for the arrivals with 270 calls for the first half an hour and an average of 225 calls per 30 minutes for the remaining simulation time. So, the first 30 minutes is an overload situation, and then the system is stable. In the second case, for the first 30 minutes, they considered an average of 331 calls and then 223 calls for the next half an hour and 162 calls per 30 minutes for the rest of the simulation. Again, there was an overload situation in the first 30 minutes, and then the system was stable. They concluded that the lower bound, based on the fluid limit and the stationary approach for the minimum arrival rates, is substantially better than applying only the fluid limit.

2.5.5 Empirical methods

An economic optimization model was developed in [43] for scheduling staffing levels in the telephone center in a large telemarketer and mail-order in catalog house for quality outdoor sporting goods and apparel. First, they obtain the total cost objective function using the regression model by applying the queuing theory. Second, they tried to predict the relationship between the abandonment rate and the telephone service factor. Finally, they described the expected-profit-maximization analysis that provides economic advantages and disadvantages of deviating from predetermined customer service level, and this helps the decision-maker have an essential tool for sensitivity analysis. A two-stage model was proposed in [44] to identify the effect of staff availability on customer sales and maximize the store profit. For the first stage, they applied the salesperson model to determine hourly staff requirements. Using the output of the first step as the input of a mixed integer programming, they obtained the optimal assignment of the staff to the daily shift. Next, they used a simulation model for validation and revising the model for more accurate results. Finally, they applied the proposed model in three types of apparel sector retail chains. Their results showed that the proposed model maximizes the scheduling efficiency and increases sales realization. Although this study is staff scheduling, they did not solve the problem when the arrival process was non-stationary. They considered only a random variable to present the arrival rates.

2.6 Deep Reinforcement Learning

Deep reinforcement learning (DRL) is the combination of deep learning (DL) and reinforcement learning (RL) to solve more complex problems. It has many applications in healthcare, finance, smart grids, and many more. The main idea is that the learning agent (neural network) should learn from its environment (simulation model) to optimize the cumulative reward. This approach can be applied in any decision-making problem that relies on experience [45].

Recently, RL has become popular because of its ability to solve complex sequential decision-making problems. Its combination with deep learning is most applicable in problems with high dimensional state-space. In general, the application of artificial intelligence in service systems is very limited in the existing literature. Li et al. [46] developed an end-to-end framework for the multi-objective problem with applying deep reinforcement learning (DRL) and called it DRL-MOA. They modified the decomposition idea to decompose the multi-objective problem into a set of subproblems. They solved the multi-objective traveling salesman problem (MOTSP) by using the DRL-MOA method by modeling the subproblem as a Pointer Network. They showed that their proposed framework could be generalized strongly and fast solving speed compared to other algorithms. Arel et al. [47], applied a multi-agent system and reinforcement learning (RL) to determine an efficient traffic signal control policy. The objective is to minimize the average delay and probability of cross-blocking of the intersection. They employed two types of agents, a central agent, and an outbound agent, in a five-intersection traffic network. In their proposed methodology, for function approximation, they utilized a feed-forward neural network with a Q-learning algorithm. Their experimental results show that their novel methodology outperforms the longest-queue-first (LQF) algorithm. Waschneck et al. [48] used cooperative Deep Q Network (DQN) agents for production scheduling. They tested and improved rules in simulation. By applying deep reinforcement learning, they achieved Industrie 4.0, a decentralized and self-optimizing system for production control. The advantages of their approach are flexibility, global optimization, global transparency, and automation.

2.7 Summary of literature

In modeling the non-stationary queuing systems, some methods use the queuing model to approximate the performance metrics. When comparing simple stationary approximation, point-wise stationary approximation, and stationary period-by-period approaches, the difference is the length of the interval in which the steady-state queuing model is applied.

The disadvantage of approaches that applied infinite server approximation and modified offered load approximation is that they test their approach in sinusoidal arrival rate function situations. To apply them to non-sinusoidal cases, there is a need to apply polynomial approximation. All methods related to applying a steady-state queuing model cannot deal with overloading situations (when a queue with finite-size becomes full or the queue size continues to grow without stopping, the system is overloaded).

The accuracy of these approximation methods depends on the characteristics of the queuing system. More exact results are achieved by using numerical integration methods, randomization methods, and discrete-event simulation. Simulation methods seem more exact than others because the service time can be any general distribution, so very small errors and the most flexibility seem achievable.

In terms of computation time, discrete-event simulation usually has the highest time compared to other methods because the number of replications is essential in the quality of the results. The square root staffing formula is inaccurate when the average service time is the high or low-performance target. The iterative staffing algorithm is based on simulation. Therefore, it can be used in any queuing system case. We will use this method in chapter 3.

For staff scheduling in a service system with non-Poisson non-stationary arrivals, the work in [13] considered the Cox process as the arrival rate distribution. To show that their model worked adequately, they applied the sinusoidal arrival rate function. Regarding virtual waiting time, they measured it at the fixed time points Δt, $2\Delta t$,... and averaged over all the replications. They used it to estimate the probability of delay, and they did not measure it according to waiting time. The disadvantage of this work is that there is no analysis for the virtual waiting of each entity. The work in [49] showed that decision making according to average waiting time over replications could be risky for customers, as it underestimates variations of the virtual waiting time.

The tour-scheduling method seems to generate dominant schedules for shift scheduling, compared to the SIPP-IP approach, regarding labor cost and service level. Additionally, SIPP-IP may overestimate the service level provided by the schedule, and it is not appropriate for situations with low arrival rates and service completion. Examples include equipment service repair and air ambulance service. The tour-scheduling model considered only M(t)/M/s(t) queuing systems.

Related studies in the literature, while determining staffing level, have not considered the virtual waiting time of each customer. The flexible shift scheduling work in [39], they considered cost as the criteria, and they did not include the waiting time of the entities to determine staffing levels. The piece-wise stationary approach always divided the study period into equal-length intervals and assumed the arrival rate changes after a fixed amount of time. In existing service systems, this assumption is not necessarily valid.

According to the literature review, none of the studies considered the virtual waiting time for individual customers while determining the staffing level. In addition, the relevant works did not consider the minimum variation of waiting times over the day as a criterion to achieve predictable waiting times in the queue. In addition, in the case of flexible staff scheduling, they did not consider waiting time to determine the number of servers. In terms of staff scheduling in the service system problem, no work applies deep reinforcement learning to achieve flexible staff scheduling. Chapter 3 of this dissertation will propose an algorithm to overcome the shortcoming of the related researches (for example, see [34], and [44]) in determining staffing level to achieve predictable waiting time. For this purpose, we will evaluate the percentile and average of waiting times in each time block. An algorithm will be proposed in chapter 4 of this research to improve the staff scheduling performance of related works like [39] by setting an upper bound for variance of waiting times during a planning horizon as a constraint in the optimization model. In addition, in chapter 4, we will propose a training approach to flexible staff scheduling. This approach will improve the related works that applied piece-wise stationary approximation to determine staffing levels (for example, see [8, 7, 9]) since this proposed approach does not require partitioning time into a smaller disjoint time interval. Chapter 5, will improve the shortcoming of related works (see [24] and [23]) in modeling and generating non-stationary arrival processes. Chapter 5 will develop a model to characterize the arrival process to generate new arrival data with the same characteristics. None of the related studies develop such a model in their researches.

Chapter 3

Staffing Levels in Service Systems with Constant Shift Length and Start Time

3.1 Introduction

This chapter aims to optimize a service system's performance by controlling customers' waiting times. Over-staffing causes an increase in operational costs, and under-staffing leads to customer dissatisfaction. We will extend the work in [1], and [50] by first, using the method described in [2] to find the locations of change-points at which the density function of arrivals is changing. Using this method, we can divide the non-stationary time series into a set of stationary series (this methodology does not require these partitions to be equal in length) that cover the same time frame. These sections are queuing systems in the arrival processes, and service time follows the general distribution. Our goal is to minimize the cost of using resources over the day, with the constraint of satisfying the service level. We are interested in determining the staffing schedule to experience predictable waiting times in the queue. To implement the simulation-based optimization for solving this problem, we use the "queue computer" simulation package in R [51].

3.2 Problem Description and Formulation

Equation 3.1 represents the formulation of the optimization problem to solve the staffing problem. The function $f()$ in the constraint could be any function of the waiting times. We will consider average and percentile, as these functions are more common in the literature.

The formulation of the problem to determine staffing levels is as follows:

$$\begin{aligned} & min \sum_{i=1}^{n} c_i s_i \\ & s.t; \\ & f(w_i^q) \leq a, \quad i \in [1, n] \end{aligned} \tag{3.1}$$

Where n is the number of time buckets in our piece-wise stationary approximation, s_i is the number of servers in segment i which is the decision variable, c_i is the unit cost of using server i, a is the maximum desired level of customers' waiting in the queue, w_i^q is the waiting time in that segment, and $f(w_i^q)$ is the function of waiting times.
Some key assumptions for this work are as following:

1. We do not consider customer abandonment in this model.
2. The average service time is shorter than the minimum time block length.
3. We do not consider customer priority.
4. We model the system using a terminating model where the run length is one day, and the system works 24 hours.
5. The arrival process is a non-stationary non-Poisson process.
6. The service time follows a general distribution.

3.3 Discussion about waiting time function

The function in the constraint could be any function like median, average, percentile, mode, or any other function of the customers' waiting times. In this section, we consider average and percentile functions and justify the model according to them.

3.3.1 Considering average as the function for waiting time

We consider average as the function of waiting times in equation 3.1, so, we can substitute $f(w_i^q)$ by $E(w_i^q)$. The average waiting time in the $G/G/s$ queuing model could be estimated based on the Kingman/Kollerstrom-formula as follows:

$$E(W_{G/G/s}) \approx \frac{\rho}{1-\rho} \times \frac{\frac{C_A^2}{\rho^2} + C_S^2}{2s_i} \tag{3.2}$$

Where $C_A = \frac{\sqrt{Var(A)}}{E(A)}$ and A is inter-arrival time and $C_S = \frac{\sqrt{Var(S)}}{E(S)}$ and S is the service time and s_i is the number of servers in time block i. In equation 3.2 we replace ρ with $\frac{E(S)}{s_i E(A)}$, where $E(S)$ is expected value of the service time. Since we do not consider the shift length constraint in this step, by using this function, we can solve the optimization model analytically. Nonlinear optimization techniques can be applied to solve this problem which is not of interest for this dissertation.

3.3.2 Considering percentile as the function for waiting time

Here we consider the percentile as the function of waiting times in the constraint of equation 3.1, and it means that the probability of nth percentile of customers experiences less than a units of time, waiting at queue be greater than b. Since there is no approximation to estimate this percentile, we need to apply simulation-based optimization methods to obtain the optimal solution. In this case, the model to get the staffing levels is as follows:

$$\begin{aligned} &min \sum_{i=1}^{n} c_i s_i \\ &s.t; \\ &p(w_i^q \leq a) > b \quad for \quad i \in [1, n] \end{aligned} \tag{3.3}$$

Since we solve the optimization model in two steps, this model does not include shift constraints.

3.4 Simulation-based Optimization

The simulation-based optimization (SBO) approach solves a problem for which there is no exact solution. Simulation is applied to estimate the output. For instance, after simulating the model, we find the output percentile, and with repeating the simulation, we estimate the confidence interval of the output. Since the direct solution is not available, realizations of a random variable which are replications of simulation outputs, are observed [52]. The SBO techniques are mainly applied to discrete-event simulations or systems of the stochastic non-linear differential equation. Discrete-event simulation is used to model systems like queues and operations, while stochastic differential equation is used to model and control nonlinear systems [53]. Various algorithms like ranking and selection, meta-heuristics, response surface methodology, gradient-based methods, and discrete search are applied to problems with discrete or continuous variables to find local or global optimum solutions. This chapter develops an iterative algorithm in a simulation environment to determine the optimal staffing levels.

3.5 Description of the staffing level problems

In real-world systems, there are more constraints like the constant shift lengths and minimum working hours for each server. This section will consider each of these constraints and justify the model accordingly for each case. To achieve predictable waiting times in the queue, we determine staffing in the way the waiting times' values get closer to each other in the entire time horizon (in this research, we arbitrarily assume it is a day).

It is important to consider the probability of average waiting time in the entire day to be less than a constant number like a minutes, and it is not equivalent to the average waiting time being less than a minutes for each time block. If there is a very short waiting time in some time buckets and there is a very long waiting time in others, there still is a possibility that the probability of overall average waiting time be less than a minutes over the day meets our criteria since average is unbiased about outlier values.

Here is an example of this situation. We created a queuing system for the check-in counters of the airport in Simio simulation software, the same system in our work in [1]. Figure 3.1 illustrates the arrival process.

We computed the waiting time in the queue for each customer in different time buckets. We replicated the simulation model 200 times and calculated the average waiting time once in each time block and another time overall average of waiting times over the entire day.

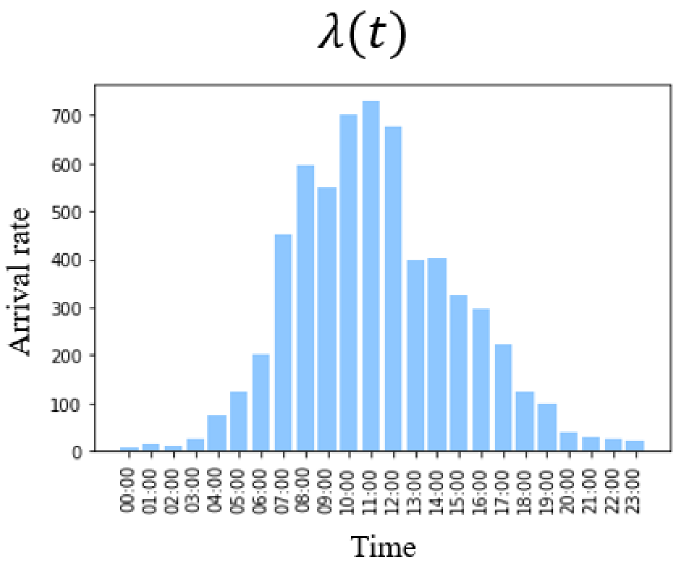


Figure 3.1: The arrival process to the system

Table 3.1 shows the staffing levels that cause customers to experience long waiting times in two time buckets, 6:00 to 7:00 a.m. and 12:00 p.m. to 01:00 p.m., compared to others. The average waiting time in the entire day is still less than 10 minutes, and the probability of having a waiting time of fewer than 10 minutes is more than 90 percent. The average waiting time for the entire day is 7.75 minutes.

Table 3.1: staffing levels in a day with hourly time buckets

Start Time	End Time	Staffing Levels	Average Waiting Time
12:00 a.m.	01:00 a.m.	1	1.87
01:00 a.m.	02:00 a.m.	1	4.59
02:00 a.m.	03:00 a.m.	2	0.52
03:00 a.m.	04:00 a.m.	2	1.42
04:00 a.m.	05:00 a.m.	4	2.46
05:00 a.m.	06:00 a.m.	7	2.53
06:00 a.m.	07:00 a.m.	3	26.36
07:00 a.m.	08:00 a.m.	30	5.02
08:00 a.m.	09:00 a.m.	29	2.45
09:00 a.m.	10:00 a.m.	30	2.16
10:00 a.m.	11:00 a.m.	36	1.46
11:00 a.m.	12:00 p.m.	36	2.60
12:00 p.m.	01:00 p.m.	26	13.09
01:00 p.m.	02:00 p.m.	35	5.79
02:00 p.m.	03:00 p.m.	21	1.18
03:00 p.m.	04:00 p.m.	17	2.59
04:00 p.m.	05:00 p.m.	15	4.03
05:00 p.m.	06:00 p.m.	14	2.57
06:00 p.m.	07:00 p.m.	7	3.32
07:00 p.m.	08:00 p.m.	6	3.11
08:00 p.m.	09:00 p.m.	3	2.85
09:00 p.m.	10:00 p.m.	2	3.10
10:00 p.m.	11:00 p.m.	2	2.34
11:00 p.m.	12:00 a.m.	2	1.86

The average waiting time in the time block from 6:00 a.m. to 7:00 a.m. and then from 12:00 p.m. to 1:00 p.m. is very high compared to other blocks, but still, the probability of experiencing the average waiting time of fewer than 10 minutes in a day is more than 90 percent. To solve this issue, we suggest considering the average (or percentile in equation 3.3) waiting times in each time block rather than the overall average in the entire time horizon.

Since this is not reasonable to have high variations in consecutive staffing levels, for instance, 30 servers between 10:00 a.m. to 11:00 a.m. and ten servers from 11:00 a.m. and 12:00 p.m. and 20 until 1:00 p.m., we will consider shifts to staff scheduling. Here we assume there are four shifts of 6 hours length over a day (24 hours). So, the problem will be determining the staffing levels for these shifts. We assume the beginning of the shifts is known and fixed.

3.5.1 Percentile as the function of waiting time

Since there is no analytical method to estimate the percentile of waiting times, we need to use simulation. The simulation model gives the waiting time for each customer, and we will be able to calculate the bth percentile of the waiting times. We perform the simulation-optimization approach to find the best staffing levels. We used the "queue computer" package in R software to simulate the discrete event queuing system. We included its source code in appendix B. The procedure of finding the optimum staffing levels is as below:

Algorithm 1 Simulation-optimization algorithm

```
1. Apply the change-point detection algorithm to identify the change-points. 2. Initialize the staffing levels by 1 for all time blocks.
3. Run an experiment using the discrete-event simulation and calculate the customers' virtual waiting time in each bucket.
4. Calculate the $bth$ percentile of each time block.
5. If the $bth$ percentile of the waiting time is more than $a$ units, increase the staffing level of the corresponding time block by one and go to step 3. Otherwise, Keep the staffing level of that segment constant.
6. If all time blocks satisfy the service level constraint, report the staffing levels as the optimal solution of replication $i$.
```

We use data in [1] as the input to the simulation-based optimization algorithm. However, this algorithm works in the case of the general non-stationary arrival process. Chapter 5 will propose an algorithm to generate non-stationary non-Poisson arrival data and use them as the input to the simulation model. We assumed the service time follows the symmetric triangular distribution with these parameters; ($a = 1$ minutes, $c = 3$ minutes, $b = 5$ minutes). We run the simulation model for 100 replications. We performed constraint of at least 90% of customers experience less than a minutes waiting time in the queue. In most of the time blocks, the waiting times are less than 10 minutes. Table 3.2 illustrates the results for 10 replications as well as the average staffing levels for each time bucket.

Table 3.2: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes

#	Rep1	Rep2	Rep3	Rep4	Rep5	Rep6	Rep7	Rep8	Rep9	Rep10	Average
1	1	1	1	1	3	1	1	1	1	1	1.2
2	1	2	1	2	2	2	1	2	1	2	1.6
3	1	1	1	2	1	2	2	2	1	2	1.5
4	2	2	2	1	2	2	2	2	2	2	1.9
5	4	4	4	4	4	3	4	3	5	5	4
6	6	7	6	6	6	5	6	7	6	7	6.2
7	10	10	10	11	10	12	9	10	10	10	10.2
8	21	21	18	22	18	19	26	23	20	21	20.9
9	29	30	26	29	29	29	30	28	27	28	28.5
10	29	28	27	26	26	26	31	29	28	29	27.9
11	37	34	35	36	33	35	32	35	36	36	34.9
12	35	36	34	35	36	37	35	36	34	35	35.3
13	34	33	32	38	34	37	35	34	36	37	35
14	20	21	20	21	24	19	22	23	20	21	21.1
15	20	18	19	20	21	23	20	19	21	20	20.1
16	17	15	16	17	18	17	16	16	17	15	16.4
17	15	17	15	15	15	16	15	15	14	14	15.1
18	10	11	10	13	11	12	11	12	13	12	11.5
19	8	7	8	7	7	6	7	7	8	6	7.1
20	5	5	6	6	5	5	4	6	5	6	5.3
21	2	2	3	2	2	2	3	4	3	4	2.7
22	2	2	2	2	2	3	5	2	2	3	2.5
23	3	2	2	2	2	2	4	2	2	2	2.3
24	2	3	2	2	2	2	2	2	2	2	2.1

Since the staffing level has to be an integer, we perform some analysis to decide which staffing level is optimal. We apply rounding to the nearest number, rounding up, and rounding down. Table 3.3 shows the staffing level, the $90th$ percentile, and maximum waiting time in each option.

As it is shown in the table 3.3, by applying rounding to the nearest number, sometimes it violates the threshold of the problem, which is that at least 90% of customers do not experience more than 10 minutes. By applying to round down, more time blocks have high waiting times. In rounding up, all of the time blocks meet the waiting time requirement. However, in this case, the cost of using resources is higher than the other two cases since the number of resources is higher than those other ones. That is the manager's decision to satisfy customers entirely or pay less for the cost of resources. For the rest of our experiments, we will use rounding up the staffing levels.

Table 3.3: Staffing levels in each time block with different types of rounding

Block#	Round	90%	Max	Round up	90%	Max	Round down	90%	Max
1	1	5.32	6.18	2	0	0	1	5.32	6.18
2	2	1.44	1.62	2	0.67	1.22	1	5.4	8.63
3	2	0.00	0	2	0	0	1	5.26	5.34
4	2	0.88	2.97	2	0.61	7.21	1	7.37	10.48
5	4	3.87	4.97	4	3.64	7.68	4	1.42	5.51
6	6	4.04	4.93	7	2.4	3.02	6	2.31	3.16
7	10	4.25	6.01	11	2.28	3.13	10	3.91	4.94
8	21	7.92	8.88	21	3.56	3.99	20	6.61	7.43
9	29	7.52	8.89	29	5.18	5.99	28	7.97	9.33
10	28	9.13	10.47	28	6.57	7.46	27	12.46	13.81
11	35	4.48	5.04	35	0.99	2.14	34	11.75	13.14
12	35	3.52	5.74	36	0.69	1.61	35	8.47	9.69
13	35	4.06	4.68	35	5.41	9.52	35	7.95	8.85
14	21	2.10	2.95	22	8.42	10.31	21	5.57	7.45
15	20	0.41	1.35	21	6.2	7.36	20	8.43	10.76
16	16	1.33	2.33	17	1.1	2.99	16	11.28	12.14
17	15	3.13	3.97	16	1.65	2.65	15	4.1	5.19
18	12	1.97	2.62	12	4.24	5.35	11	6.78	8.65
19	7	2.40	4.71	8	3.19	3.99	7	8.71	10.68
20	5	2.07	3.34	6	0.69	1.91	5	6.41	7.64
21	3	4.47	7.22	3	2.43	3.39	2	25.4	27.35
22	3	1.37	2.77	3	1.66	2.64	2	30.27	32.49
23	2	23.74	25.43	3	6.32	8.46	2	8.64	10.4
24	2	27.04	27.97	3	2.28	4.94	2	0.35	0.89

After obtaining the optimal staffing levels that satisfy the constraint that the $90th$ percentile of the customers does not experience the waiting time of more than 10 minutes in the queue, we plot the histogram of all waiting times during 24 hours. Figure 3.2 shows 99% percent of customers did not experience more than 8.66 minutes in the queue.

Table 3.4 shows the result of staffing levels for each time block that satisfies the constraint that at least 95 percent of customers do not experience the waiting time of more than 10 minutes in the queue for ten replications.

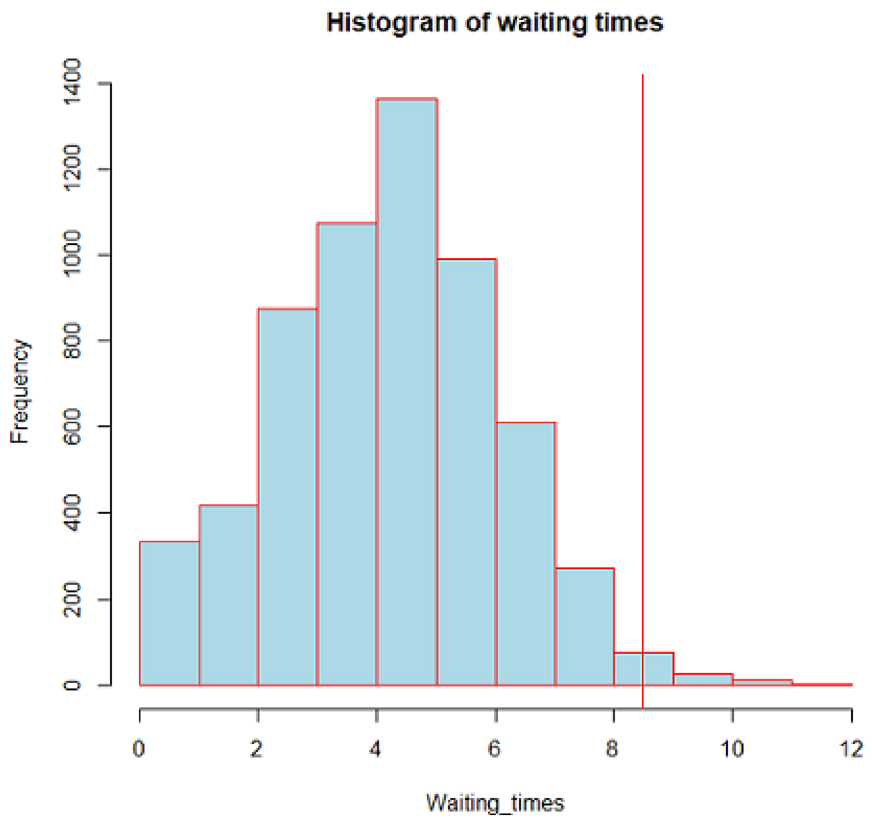


Figure 3.2: Customer's waiting times in a day and $99th$ percentile of waiting times

Table 3.4: Staffing levels in a day with hourly time buckets considering $95th$ percentile of customers do not experience waiting time more than 10 minutes

Rep1	Rep2	Rep3	Rep4	Rep5	Rep6	Rep7	Rep8	Rep9	Rep10	staffing level
1	2	1	1	2	1	1	1	1	2	2
1	1	2	1	2	1	1	2	2	2	2
1	2	1	2	2	2	1	1	1	2	2
1	2	2	2	2	2	2	2	2	2	2
5	4	4	3	4	3	4	4	4	3	4
7	9	6	6	6	6	7	9	7	7	7
11	10	11	10	12	11	9	11	10	10	11
21	23	20	23	20	21	27	22	22	25	23
33	30	29	31	29	31	30	33	27	29	31
28	28	27	27	28	26	27	30	28	28	28
34	34	33	37	33	35	35	34	32	37	35
37	36	34	38	35	40	35	37	33	35	36
34	31	31	34	36	36	36	33	35	37	35
21	19	20	21	22	21	23	20	20	21	21
20	20	21	20	21	20	19	22	22	23	21
17	14	15	18	16	19	17	17	17	16	17
14	14	16	14	18	15	15	16	15	16	16
11	12	12	12	11	12	12	12	13	12	12
6	6	6	9	7	6	6	7	7	9	7
6	4	6	5	5	6	4	6	6	7	6
3	2	3	3	2	2	2	2	2	3	3
3	3	2	2	2	4	4	2	2	2	3
2	3	3	2	2	2	3	2	2	2	3
2	2	2	2	2	2	3	2	2	2	3

After comparing 95*th* percentile, 90*th* percentile, and a maximum of waiting times in each time block, and according to Figure 3.2, 90*th* percentile generated the waiting times that 99 percent of them are less than 8.5 minutes. Because of this, we decided to consider the service level as at least 90 percent of the customers do not experience a waiting time of more than 10 minutes in the queue.

3.5.2 Determining staffing levels inside of the simulation process

Since there are variations in the arrival times, the staffing level of some time blocks may differ noticeably from one replication to another. To reduce this variability, we define the number of replications before the simulation, *n*, then we repeat the simulation-based optimization process *n* times. In this case, the output of the process is the confidence interval for the number of staff in each time block. Algorithm 2 describes this process.

Algorithm 2 Simulation-optimization algorithm to replicate inside of the SBO process

```
1. Determine the required number of replications, n to achieve 95% confidence interval for staffing levels.
2. $i \leftarrow 1$
3. while $i \leq n$ do
    4. Apply the change-point detection algorithm to identify the change-points.
    5. Initialize the staffing levels by 1 for all time blocks.
    6. Run an experiment using the discrete-event simulation and calculate the virtual waitingtime of the customers arriving in each time bucket.
    7. Calculate the $bth$ percentile of each time block.
    8. If the $bth$ percentile of the waiting time is more than $a$ units, increase the staffing level of the corresponding time block by one and go to step 3. Otherwise, Keep the staffing level of that segment constant.
    9. If all time blocks satisfy the service level constraint, report the staffing levels as the optimal solution of replication i.
    10. $i \leftarrow i + 1$
end
```

After applying this algorithm on data in [1], we get the following staffing level for each time block.

Table 3.5: Staffing levels after replicating inside of the SBO process

Time block	Lower Level	Upper Level	Average
1	1.09	1.82	1.46
2	1.19	1.32	1.26
3	1.17	1.29	1.23
4	1.67	1.79	1.73
5	4.12	4.28	4.20
6	6.39	6.56	6.48
7	9.44	9.69	9.57
8	21.34	21.60	21.47
9	28.98	29.29	29.14
10	27.44	27.78	27.61
11	34.08	34.45	34.27
12	35.67	36.06	35.87
13	34.26	34.66	34.46
14	21.38	21.72	21.55
15	18.99	19.29	19.14
16	16.45	16.71	16.58
17	14.79	15.04	14.92
18	11.11	11.36	11.24
19	6.55	6.83	6.69
20	5.18	5.38	5.28
21	2.68	2.84	2.76
22	2.16	2.27	2.22
23	2.03	2.10	2.07
24	2.00	2.05	2.03

Comparing these results with table 3.2 shows the outputs of the two algorithms are very similar. The advantage of algorithm 2 is that it can cover cases with higher variety in the arrival process. This example was about a data set whose variety from one replication to another was not high. Based on the number of replications, the results of the two algorithms might not be the same in all cases. Since the output of algorithm 2 is a confidence interval for the required number of staff in each time block, it provides more information for managers to decide their staffing requirements.

3.5.3 Staff scheduling and defining shifts

To obtain the staffing levels in each shift with l hours length, we need to choose the maximum of optimal staffing levels in that shift. In this work, we solve the optimization model for the constant 6 hours length shifts and determine the best time for the beginning of the shift to achieve the minimum servers' cost. To reach optimal staffing levels in each shift, we determine the beginnings of the shifts that the variation among the staffing levels of each time block constructing that shift is minimum. We use enumeration to investigate all the possible combinations and calculate the variance of the number of staff in each shift for each combination. Table 3.6 represents the staff scheduling for 6 hours shift lengths of staffing levels in table 3.3 with 1-hour time blocks.

Table 3.6: Optimal number of staffs in each shifts and beginning times

	Shift 1	Shift 2	Shift 3	Shift 4
Beginning Time	1:00 a.m.	7:00 a.m.	1:00 p.m.	7:00 p.m.
Number of Staffs	10	35	22	4

Since the time blocks in this example are equal length and equal to 1, determining staffing level first and then shift scheduling might not be necessary. Here we present another example where the time blocks are of different lengths.

Table 3.7: Staffing levels after replicating inside of the SBO process

Time block	Time	Lower Level	Upper Level	Average	Round up
1	12:00- 01:00	1.05	1.17	1.11	2
2	01:00- 01:30	1.03	1.46	1.245	2
3	01:30 -02:30	1.41	1.61	1.51	2
4	02:30-04:00	3.19	3.47	3.33	4
5	04:00-05:00	6.39	6.71	6.55	7
6	05:00-08:00	7.51	7.87	7.69	8
7	08:00-09:00	10.53	10.91	10.72	11
8	09:00-09:30	12.43	12.89	12.66	13
9	09:30-10:30	18.1	18.66	18.38	19
10	10:30-12:00	21.43	21.98	21.705	22
11	12:00-12:30	24.32	24.92	24.62	25
12	12:30-01:30	25.67	26.23	25.95	26
13	01:30-02:30	27.69	28.41	28.05	29
14	02:30-03:00	26.89	27.51	27.2	28
15	03:00-04:30	22.03	22.61	22.32	23
16	04:30-05:30	16.27	16.79	16.53	17
17	05:30-06:00	13.39	13.89	13.64	14
18	06:00-08:30	10.04	10.44	10.24	11
19	08:30-10:00	4.99	5.33	5.16	6
20	10:00-10:30	3.07	3.33	3.2	4
21	10:30-11:30	2.2	2.38	2.29	3
22	11:30-12:00	1.15	1.27	1.21	2

Table 3.8: Optimal number of staffs in each shifts and beginning times

	Shift 1	Shift 2	Shift 3	Shift 4
Time	10:00-04:00	04:00-10:00	10:00-04:00	04:00-10:00
Number of Staffs	4	19	29	23

3.5.4 Average as the function of waiting time

Figure 3.3 represents the idea of dividing a planning horizon into a shift that each includes time blocks. Depending on the change-point locations, the shift's start time can be either at the beginning of a time block or sometimes in between.

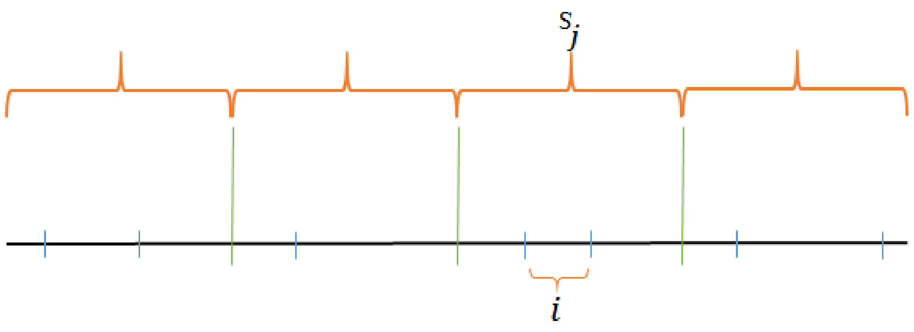


Figure 3.3: Dividing time horizon into fixed length shifts

In equation 3.2, if we replace ρ and C_A by $\frac{E(S)}{s_i E(A)}$ and $\frac{\sqrt{Var(A)}}{E(A)}$, respectively. We will have the equation 3.4:

$$E(W) = \frac{s_i^2 Var(A) + var(S)}{2s_i^2 E(S)E(A) - 2s_i E(S)^2} \tag{3.4}$$

To find the server's schedule level in shifts, we can consider two approaches as described below.

We can assume all segments in a shift have the same impact in determining required servers in that shift, and since $E(W) \leq a$ (equation 3.5 shows this inequality), s_i^{min} is the minimum required server in segment i which meets the minimum service level.

$$\frac{s_i^2 Var(A) + var(S)}{2s_i^2 E(S)E(A) - 2s_i E(S)^2} \leq a \tag{3.5}$$

By repeating these calculations for all segments, we can find the s_i^{min} for each of them. After solving the optimization problem for each time block, the staffing levels will be as follows in Table 3.9.

Table 3.9: Staffing levels

Staffing levels	1 1 1 2 4 7 10 24 31 28 36 36 35 20 20 17 15 9 6 5 3 2 2 2

So s_{min} for entire shift can be calculated as equation 3.6:

$$s_j^{min} = \max_i \{s_i^{min}\}, \quad \text{for all } j \tag{3.6}$$

As figure 3.3 illustrated, s_j represents the number of staff members in shift j. Number of shifts in a day is shown by j.

Table 3.10: Number of staffs in each shifts and beginning times

	Shift 1	Shift 2	Shift 3	Shift 4
Beginning Time	1:00 a.m.	7:00 a.m.	1:00 p.m.	7:00 p.m.
Number of Staffs	10	36	20	5

Table 3.10 represents the number of required workers in each shift to at least 99 percent of customers who experience waiting time less than 10 minutes in the queue. Figure 3.4 shows the corresponding waiting times histogram. It indicates that 99 percent of customers experienced less than 5 minutes of waiting time in the queue.

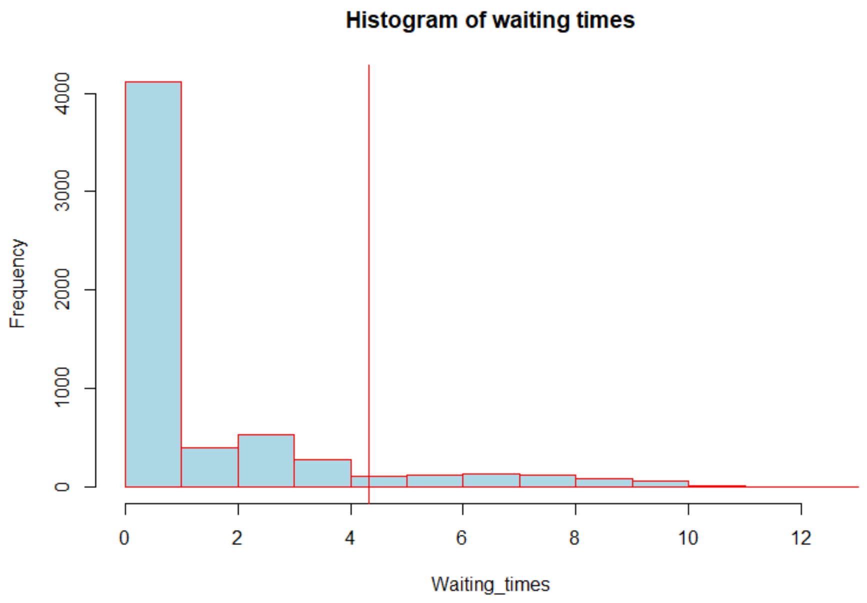


Figure 3.4: Waiting times in a day and $99th$ percentile of waiting times

3.6 Performance evaluation

In this section, we investigate the performance of the proposed algorithm to determine staffing levels by applying it to some data sets with different arrival process patterns generated by Simio simulation software. The set of patterns includes step, sinusoidal, and bi-modal. In addition, we will evaluate the performance of the algorithm with the real-world data set from arrivals to the check-out to the grocery store retrieved from the work in [54]. Figure 3.5 illustrates different patterns we consider in the arrival process.

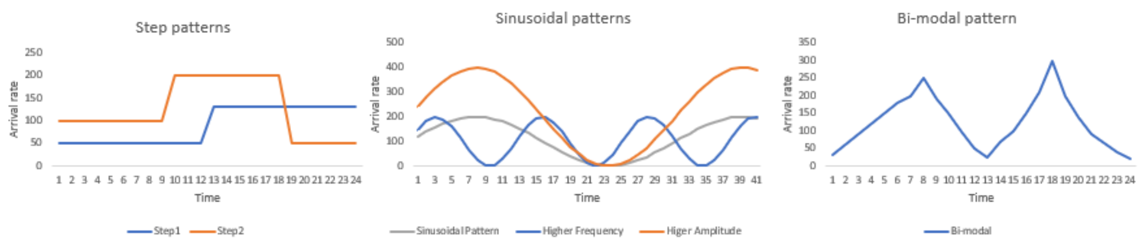


Figure 3.5: Arrival process patterns

3.6.1 Arrival processes with step patterns

Tables 3.11 represents 10 replications of the simulation model to determine the staffing levels along with the number of required workers (average rounded up) and the $90th$ percentile of the waiting times in each time block for the case the arrival process follows step pattern 1. Table 3.11 indicates that by using the number of workers resulting from the algorithm, at least 90% of the customers in each time block will experience less than 10 minutes of waiting time in the line. In addition, it shows the step pattern in the staffing levels, which is as expected.

Table 3.11: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with step pattern 1 in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
3	3	3	4	3	4	2	3	2	3	3	2.27
3	3	3	3	2	3	3	3	3	3	3	0.73
4	4	4	3	3	3	2	4	3	3	4	2.39
3	3	2	2	3	4	3	5	4	2	4	0.64
3	4	3	3	3	3	5	4	4	3	4	3.01
4	4	4	4	4	3	3	3	2	3	4	2.17
3	4	3	3	3	3	3	4	3	2	4	0.00
2	3	3	2	3	3	3	3	3	3	3	4.63
3	4	3	3	4	3	3	4	3	3	4	3.87
3	3	3	3	2	3	3	3	3	2	3	2.83
3	2	2	2	3	3	3	3	3	3	4	5.27
5	4	3	4	4	3	3	3	3	3	4	5.14
6	7	7	7	7	5	6	6	6	6	7	2.75
7	8	7	8	9	5	7	6	7	7	8	2.80
7	5	6	6	6	7	5	7	7	7	7	2.65
7	8	8	8	6	7	7	6	7	5	7	2.21
6	7	7	7	8	8	7	6	7	7	7	3.65
7	7	7	5	7	6	7	7	7	7	7	8.04
7	7	7	7	7	6	8	8	8	6	8	9.61
7	7	8	7	8	7	5	7	6	6	7	3.62
6	6	6	6	6	7	6	8	8	6	7	1.62
7	7	6	6	6	6	8	7	9	6	7	4.64
6	6	7	6	7	6	7	7	8	7	7	1.69
7	6	6	7	5	7	6	7	7	7	7	1.81

Table 3.12 represents the staffing levels for each time block for step pattern 2 for 10 replication, the number of required workers, and $90th$ percentile of waiting times. It shows the algorithm results to the number of workers that satisfy the constraint of at least 90% of customers experience waiting time less than 10 minutes in the queue.

Table 3.12: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with step pattern 2 in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
5	5	6	5	4	5	5	5	4	5	5	2.93
7	6	6	6	6	6	6	6	7	6	7	1.73
4	5	5	5	5	4	6	6	5	5	5	9.53
6	6	6	6	5	7	6	5	6	6	6	3.40
6	5	6	5	5	7	5	5	5	5	6	2.68
5	6	6	7	6	5	5	5	6	6	6	9.10
5	5	6	6	5	7	6	5	7	6	6	1.80
6	7	5	8	5	5	5	6	5	6	6	0.70
7	7	7	7	6	7	7	7	6	6	7	0.26
11	7	11	9	9	9	9	10	8	10	10	2.60
10	10	9	9	8	11	10	9	11	10	10	6.86
10	11	10	10	11	9	10	11	10	10	11	4.86
10	10	9	10	10	10	11	12	11	9	11	2.90
9	8	9	10	11	10	9	10	8	9	10	1.96
10	11	10	9	8	9	9	11	10	9	10	3.38
8	9	8	10	9	10	9	10	8	9	9	9.38
9	11	11	10	11	11	11	11	10	10	11	8.56
10	11	10	9	10	9	10	10	11	10	10	6.67
6	4	5	4	2	4	3	8	3	4	5	7.81
3	2	2	2	3	3	3	2	3	3	3	9.60
3	3	3	4	2	4	4	3	3	2	4	2.48
3	2	2	2	3	2	3	3	2	2	3	7.48
2	3	3	3	3	3	2	2	3	3	3	9.54
4	3	3	4	4	4	3	3	4	4	4	8.07

3.6.2 Sinusoidal pattern in arrival process

Tables 3.13, 3.14, and 3.15 present the sinusoidal patterns in the arrival process. We consider a sinusoidal pattern with arrival rate function $\lambda(t) = \bar{\lambda}(1 + \sin(0.2t))$, we assume $\bar{\lambda} = 100$ as the first sinusoidal pattern. Second sinusoidal pattern represents the previous sinusoidal function with higher frequency. Its arrival function is $\lambda(t) = \bar{\lambda}(1 + \sin(0.5t))$. The last sinusoidal pattern has a higher amplitude with arrival rate function $\lambda(t) = \bar{\lambda}(1 + \sin(0.2t))$, we assume $\bar{\lambda} = 200$. Table 3.13 shows that resulted staffing levels satisfy the optimization problem's constraints.

Table 3.13: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with sinusoidal pattern 1 in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
5	6	7	8	7	7	8	6	7	8	7	1.53
4	3	5	3	4	4	4	4	3	4	4	0.11
2	2	2	2	2	2	2	2	2	2	2	0.00
2	2	2	2	2	2	2	2	2	2	2	0.00
2	2	2	2	2	2	2	2	2	2	2	4.77
4	4	3	4	3	3	4	4	4	4	4	4.26
7	6	7	6	6	7	6	7	6	7	7	3.16
9	9	8	9	9	9	8	9	9	9	9	8.94
10	11	11	10	10	10	10	11	10	10	11	6.92
9	8	10	10	10	9	9	10	11	10	10	2.80
7	7	9	9	8	8	9	9	8	8	9	1.20
7	8	6	6	5	6	6	6	6	6	7	2.77
4	4	4	4	4	4	3	4	3	4	4	2.80
2	2	2	3	3	2	2	2	2	2	3	0.00
2	2	2	2	2	2	2	2	2	2	2	1.16
5	4	5	4	4	4	3	4	4	3	4	7.30
7	7	6	6	7	5	7	7	7	7	7	2.60
7	9	8	9	8	8	7	9	7	9	9	1.91
11	10	8	10	9	10	9	10	9	9	10	2.14
7	9	10	10	10	9	9	10	9	9	10	4.73
9	9	8	8	8	8	8	10	9	9	9	2.09
7	5	6	5	5	5	5	5	5	5	6	3.67
2	5	3	3		3	4	3	3	4	4	2.08
2	2	2	2	2	2	2	2	2	2	2	2.22

Table 3.14 represents the required staffing levels for the system with the sinusoidal arrival patterns with higher amplitude than the previous one. The table indicates that when there is a higher frequency in the sinusoidal pattern, there is a need for more staff members to satisfy the constraint.

Table 3.14: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with sinusoidal pattern 2 in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
11	12	11	12	13	11	13	12	12	11	12	9.76
4	6	8	5	5	6	5	5	8	7	6	9.95
2	2	2	2	2	2	2	2	2	2	2	2.67
2	2	2	2	2	2	2	2	2	2	2	0.81
4	3	2	3	3	3	3	2	3	3	3	1.17
8	9	8	9	8	9	7	8	9	8	8	5.05
11	12	12	11	13	12	11	11	12	12	12	5.93
16	17	17	18	18	17	19	19	17	17	18	4.58
19	22	21	19	21	22	22	21	18	21	22	4.10
20	19	20	20	18	18	20	19	21	18	19	9.02
15	16	16	14	16	14	16	17	16	15	16	8.44
11	12	11	12	11	12	10	12	13	13	12	9.11
8	5	6	7	6	7	7	6	5	6	7	9.59
3	2	3	2	3	3	3	2	3	2	3	0.34
3	3	2	3	3	2	4	3	4	3	3	1.09
7	7	8	7	7	5	7	6	7	6	7	3.64
11	11	9	11	10	10	10	10	11	11	11	2.45
15	16	15	15	17	15	15	19	15	16	17	3.16
20	21	20	21	21	20	20	18	20	19	20	3.52
17	17	19	17	16	16	18	17	17	18	18	7.17
16	17	16	16	15	14	15	16	15	16	16	9.16
10	11	12	10	11	12	11	11	11	13	12	11.07
7	6	6	5	7	5	5	5	5	5	6	4.75
2	2	3	3	3	2	2	2	3	2	3	3.26

Table 3.15 shows the number of required workers in the case that arrival process follows the sinusoidal pattern with higher frequency.

Table 3.15: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with sinusoidal pattern 3 in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
7	7	7	8	9	8	7	8	7	7	8	1.66
2	2	2	2	2	2	2	2	2	2	2	0.00
4	4	4	4	4	4	5	4	5	4	5	0.22
9	9	9	10	10	9	9	9	8	9	10	3.53
9	9	9	8	8	8	8	8	9	8	9	9.71
2	2	3	2	2	4	3	3	2	2	3	3.00
2	2	2	2	2	2	2	2	2	2	2	0.87
9	9	8	8	8	7	8	8	8	8	9	0.98
10	10	10	11	10	10	10	10	10	10	11	1.59
5	5	4	3	4	4	5	4	5	5	5	0.43
2	2	2	2	2	2	2	2	2	2	2	0.00
5	5	5	5	5	4	4	5	5	5	5	1,18
10	9	9	9	8	12	9	9	9	10	10	2.23
7	9	7	10	8	8	8	8	7	7	8	3.70
4	4	4	2	5	2	2	4	4	4	4	0.00
9	8	8	7	8	8	8	7	7	8	8	4.82
9	9	9	9	8	9	8	9	11	9	9	6.89
4	4	5	4	5	4	4	6	4	4	5	7.75
2	2	2	2	2	2	2	2	2	2	2	0.00
6	5	6	6	6	6	6	5	6	5	6	3.60
10	10	10	10	9	10	9	9	9	8	10	7.54
5	5	5	5	5	6	6	7	6	6	6	9.75
2	3	2	2	2	2	2	2	2	2	2	0.00
3	2	3	2	3	3	2	2	3	2	3	2.99

3.6.3 Bi-modal pattern in arrival process

Table 3.16 represents the staffing level in each time block and its corresponding $90th$ percentile of waiting time. The results show that at least 90% of customers arriving in each time block experienced waiting time less that 10 minutes.

Table 3.16: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with bi-modal pattern in arrival process

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
2	2	2	2	2	2	3	2	2	3	3	1.33
3	3	3	3	3	3	3	3	3	4	4	1.40
6	5	5	6	5	6	7	6	5	5	6	3.02
7	6	6	7	6	6	6	6	6	6	7	2.58
8	7	8	8	8	8	7	8	8	8	8	1.73
11	10	10	10	9	10	10	9	11	10	10	3.32
10	12	12	10	9	11	10	10	11	10	11	1.78
12	12	13	14	12	12	14	13	13	12	13	1.85
10	10	12	11	10	11	10	10	11	11	11	1.59
7	7	8	9	8	7	7	8	9	7	8	2.38
6	6	6	5	6	6	5	7	6	6	6	6.64
3	3	3	4	3	3	4	3	3	3	4	2.17
2	2	3	2	2	2	2	2	3	2	3	0.19
4	4	3	4	4	4	4	3	4	4	4	1.15
5	5	4	5	4	5	4	5	5	5	5	7.24
8	7	7	7	7	7	7	6	7	6	7	6.13
11	9	10	10	11	10	9	11	10	9	10	1.56
17	15	14	16	14	12	14	14	14	14	15	4.26
11	12	11	11	11	12	13	12	11	12	12	4.95
6	7	7	7	8	7	8	6	8	6	7	5.92
5	4	4	5	5	5	5	4	4	5	5	7.73
3	3	3	2	3	3	3	3	3	3	3	9.02
3	3	4	3	4	3	3	3	3	3	4	3.55
2	2	2	2	2	2	2	2	2	2	2	2.09

3.6.4 Point of sale arrival data

Figure 3.6 represents the arrival rate into the check-out counters of the grocery store. This data is real-world data that we retrieved from work in [54]. Table 3.17 shows the staffing levels and the 90th percentile of the waiting times in each time block.

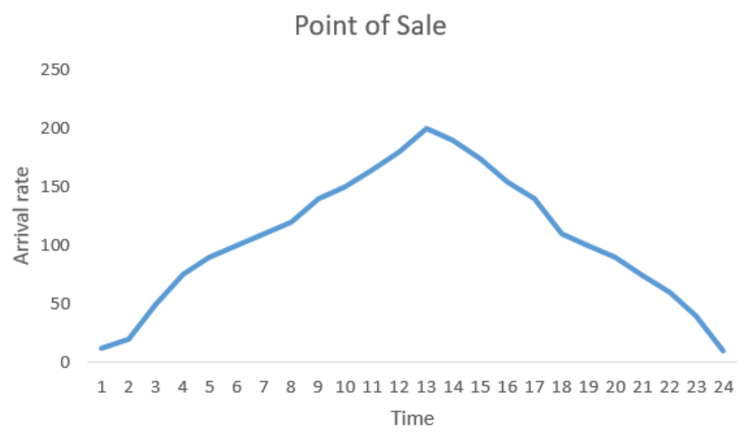


Figure 3.6: Arrival rate to the point of sale in grocery store

Table 3.17: Staffing levels in a day with hourly time buckets considering $90th$ percentile of customers do not experience waiting time more than 10 minutes with point of sale arrival data

#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#Workers	90%
1	1	1	1	1	1	2	1	1	1	2	6.68
2	2	2	2	2	2	2	2	2	2	2	1.09
2	3	3	3	3	2	3	2	3	3	3	3.05
6	6	6	6	6	6	5	6	6	7	6	0.90
6	6	6	6	6	6	8	5	6	6	7	1.75
9	10	9	10	9	10	10	10	9	9	10	8.80
12	13	12	13	13	12	11	12	12	12	13	7.58
13	15	15	14	16	15	14	15	16	15	15	7.27
14	15	14	14	14	13	16	14	15	14	15	2.97
10	12	14	11	12	10	13	11	10	10	12	2.94
10	10	9	9	8	10	9	9	10	8	10	1.88
7	6	7	7	7	6	6	6	6	5	7	2.71
5	4	5	4	4	5	5	5	5	6	5	5.68
2	3	3	3	3	2	3	2	3	3	3	1.80

The results of applying our proposed algorithm to determine staffing levels on various generated data sets and the data set from the real world in this section indicate that it can work adequately on similar data sets with the same arrival rate patterns.

3.7 Conclusion

This chapter solved the optimization problem of a queuing system with a non-stationary arrival process and single-task server. In this system, the workers working hours and shifts' start times are constant. We were interested in staff scheduling in the way customers experience predictable waiting times in the queue.

To achieve this goal, first, we applied the change-point detection algorithm to determine the location of change-points over time, and then we used the "queue computer" simulation package in R to simulate the optimization problem. We investigated two functions, average and percentile of waiting times, to characterize customer service level. We performed a two-step optimization approach to determine the staff scheduling for this problem. To determine the staffing schedules, we determine the start of the shift because there are fewer variations in the number of workers in that shift. Since this approach results in fewer workers in a day, we did not consider other shifts' starting times.

As figure 3.4 shows, in this approach, 99% of customers experienced less than 5 minutes in the queue even though the upper bound in the optimization problem was set up 10 minutes. The reason is, we determined the staff levels first, and then we performed staff scheduling. That results in more staff in the final scheduling problem than the total number of workers determined in staffing levels. In the next chapter, we will overcome this issue by flexible staff scheduling that allows workers to start working at different times with different total working hours. In addition, chapter 4 will consider the option that workers have minimum and maximum working hours in a specific job.

To evaluate the performance of our proposed algorithm, we conduct some more experiments by applying other patterns in the arrival process. The results indicate that this chapter's algorithm can solve different problems to achieve predictable waiting times. Finally, we applied the arrival data of the check-out counters in a grocery store to our model to determine the staffing levels and evaluate the solution approach's performance.

Chapter 4

Flexible Staff Scheduling in Service Systems with Non-Stationary Arrival Processes by Applying Deep Reinforcement Learning (DRL)

4.1 Introduction

This chapter proposes a method to perform flexible staff scheduling in service systems with a non-stationary arrival process. We characterize the arrival process, and we determine staffing levels such that arriving customers experience *predictable* waiting time in the queue. By flexible, we mean each staff member can start working at any time of the day. As described in the previous chapter, focusing on *predictable* waiting time is not necessarily focusing on obtaining minimum waiting time in the queue. It indicates that the customers experience almost constant waiting times at different times of the day.

We set an upper bound for the waiting time variance over the time horizon to achieve a predictable waiting time. We determine this inequality as the constraint in the optimization problem. Unfortunately, since there is no closed-form expression for these variances, we cannot use an analytical method to solve this optimization problem. As such, we use the simulation for its estimation.

We propose to use the Deep Reinforcement Learning (DRL) approach to solve this problem. One of the advantages of using this technique is that performing input analysis to detect change-points in the non-stationary arrival process is not required. The learning agent (the artificial neural network) observes the environment and learns when to add or remove staff members from the available resource pool.

There is a need for an environment in which the learning agent can learn from the system in this solution approach. The simulation model provides such an environment for the learning agent. We define possible actions that the learning agent can take. According to the earned “reward”, the learning agent decides which action it should take in each step in order to maximize the possible cumulative reward.

The rest of this chapter is organized accordingly. In section 4.2 we represent the general overview of the problem and its formulation to get the staff schedule. Section 4.3 describes the deep reinforcement learning technique. In the next section 4.4, we explain both how to use this approach as a tool for flexible staff scheduling, and we explain the inputs to the tool. In section 4.5 we talk briefly about the verification and validation of the simulation model. Section 4.6 discusses experiments used to test the performance of the proposed solution method. We also include different possible variations in the arrival process. Then we represent the result of each simulation run in section 4.6, and we interpret them in the appendix A section.

Next, we test this technique on the use case, including testing the arrival data to a grocery store in Europe in section 4.7. Section 4.8 includes the comparison between our proposed solution approach and simulation-based Optimization (SBO). Ultimately. We conduct a statistical test to check the hypothesis that the proposed algorithm outperforms the SBO approach. Finally, in section 4.9, we summarize our finding.

4.2 Overview of the problem

In the previous chapter, we solved the staffing problem with the objective function of minimizing the number of servers. We considered two cases as the constraint. One was the percentile of waiting times, and another was the average waiting times in the queue. In that problem, the shifts had a specific start and end time. For example, the beginning of the shift is 7:00 a.m., and the end is 3:00 p.m. for all servers working in that shift. We applied a simulation package in R software to perform simulation-based optimization (SBO).

This chapter aims to determine flexible staff scheduling in a service system with the non-stationary arrival process. The servers perform a single task, and each can start working at any time over the day. In addition, there is a penalty function that we present it by step function which enforces a penalty for the number of customers waiting in the line. The servers of queuing system in chapter 3 had a shared queue, while the servers in this chapter have their individual lines. Other assumptions of the queuing system are the same as chapter 3 in this chapter. Table 4.1 represents an example of what we mean by staffing level and staff rostering:

Table 4.1: Staffing level and staff rostering example

Working hours of each staff member in each one-hour time block						
	8:00-9:00	9:00-10:00	10:00-11:00	11:00-12:00	12:00-1:00	1:00-2:00
Staff 1	1	1	1	1	1	0
Staff 2	0	1	1	1	1	1
Staff 3	0	1	1	1	1	1
Required staffing level	1	3	3	3	3	2

In this example, there are three workers with a total of 5 hours working. They can start working in any time block, but they need to work for 5 hours once they have started working. For instance, worker 1 started working at 8:00 a.m., and staff 2 and 3 started at 9:00 a.m. Here is the problem formulation for a staff rostering in a single task server's queuing system.

$$\begin{aligned}
&\text{Minimize} \sum_{j=1}^{n} c_j s_j + p_1 f(w_1 \leq w_j^q \leq w_2) + p_2 f(w_j^q > w_2)\\
&\quad \text{Subject to :}\\
&\qquad var(w^q) \leq b\\
&\qquad s_j = \textstyle\sum_i x_{ij}, \quad \text{for all } j\\
&\qquad y_{ij} = x_{ij}(1 - x_{ij-1}), \quad \text{for all } i \text{ and } j\\
&\qquad hy_{ij} \leq \textstyle\sum_j L_j.x_{ij} \leq Dy_{ij} \quad \text{for all } i\\
&\qquad L_j \leq h \leq D \quad \text{for all } j
\end{aligned} \tag{4.1}$$

$$\begin{aligned}
x_{ij} &= \begin{cases} 1 \text{ if server } i \text{ works in time bucket } j \text{ in this job,} \\ 0 \text{ otherwise.} \end{cases}\\
y_{ij} &= \begin{cases} 1 & \text{if server } i \text{ begins working in this job in time block } j, \\ 0 & \text{otherwise.} \end{cases}
\end{aligned} \tag{4.2}$$

Where $f()$ is a function representing the number of customers that waited for a certain amount of time, s_j is the number of staff members in time block j, c_j is the cost of using each unit, or staff member, per hour, w_j^q is the waiting time in the queue in time bucket j, w_1 and w_2 are the bounds for the waiting time in queue, w_j^q is the waiting time in the queue in time block j, $var(w^q)$ is the variance of waiting times over the planning horizon, L_j is the length of the time bucket j in hours, h is the minimum time the server has to work on a particular job, and D is the length of the shift for each staff member. Each worker can start working at any time during the day. However, once their workday begins, they need to continue until the end of the shift.

Note that the variance of waiting times is included in the problem constraints. Unfortunately, there is no closed-form expression for these variances, and there is no analytical method to solve this optimization problem. In response, we use simulation to estimate these variances. This is a staff rostering problem in a service system like the grocery store or airport check-in counters. This problem aims at determining staff rostering in the way customers experience *predictable* waiting time.

4.2.1 Solution approaches

As we mentioned, there is no analytical method to solve this problem. In order to determine staff rostering, we will compare two solution approaches:

1. Deep Reinforcement Learning (DRL)

2. Simulation-based Optimization (SBO)

There are many differences in the way we implement these two techniques. First, there is no need to perform change-point detection in the arrival data set before putting it into the DRL approach's simulation model, but SBO does require this step. Second, the output of the DRL is the staff rostering, while SBO output gives the staffing level. Last, we need to perform one more step to determine staff rostering. Here are the steps to develop the DRL approach:

1. Building simulation model

2. Setting up requirements to implement DRL
3. Setting up training experiment
4. Simulation model Verification and Validation
5. Setting up an experiment to replicate the outputs

There are some differences between the two SBO approaches in chapters 3 and 4, as listed below:

- The queuing system is different. In chapter 3, there is one common queue for all active servers. In this chapter, each active server has its line.
- The objective function in chapter 3 is the total cost of resources, while in this chapter, it is the total cost of using servers and a step function that penalties the number of customers who waited more than a predetermined amount of time.
- The constraint in chapter 3 is satisfying the *nth* percentile of waiting times in each time block. Instead of percentile as the function of waiting times in the constraint, this chapter's constraint is to satisfy the variance of waiting times over a planning horizon (a day).
- The shifts in chapter 3 have a constant start time and length, while the start of each shift in this chapter depends on each individual server. The servers in this chapter have a flexible start time.
- In chapter 3, there were no minimum and maximum working hours, while in this chapter, there are both minimum and maximum working hours in the current job.

4.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) combines a neural network with reinforcement learning, enabling the learning agent to acquire the best possible action in a virtual environment (simulation model).
Here are some key terminologies of the DRL approach [55]:

Environment: The environment is the real or simulated world in which the learning agent operates. The learning agent interacts with and learns from the environment.

Learning Agent: The learning agent, or more accurately, the artificial neural network (ANN), is the agent that interacts with the environment (simulation model) and learns from it by making observations, performing specific actions, and receiving rewards based on certain actions.

State, *s*: The state is the information that is knowable in the environment that the agent needs to observe in order to be able to take the best possible action.

Actions *a*: The possible options that the learning agent can take in the given environment. Similar to the real world, it can take only certain types of actions in the simulation model.

Reward, *r*: The reward is the feedback from the environment that reinforces or punishes the learning agent's behavior. This reward comes in the form of a number (positive or negative), altering how the learning agent chooses its actions.

Policy, $\pi_\theta(s, a)$: Policy is the learning agent's strategy used to decide which action to take based on the current observation. After training, a learned policy is outputted. A "good" policy tells us which action is optimal (or near-optimal) based on the system's given state.

We used AnyLogic simulation software to provide the environment and the training experiment to guide the learning agent. The next part will describe the procedure we implemented to perform DRL for flexible staff scheduling.

4.3.1 Implementation of the Deep Reinforcement Learning technique to flexible staff scheduling

The simulation model is built with the necessary functions to communicate the model and the reinforcement learning framework. Specifically, two functions are added to the simulation model, consisting of both making the observation and taking an action. Further details of these functions are provided in the following section.

4.3.1.1 Get State function

In this problem, the ”GetState” is a function that summarizes information such as the number of staff members, the variance of waiting time in the queue, and the queue length in a vector of seven variables. This research contains the information that the learning agent observes in each time epoch. The vector is:

S[0]: waiting time of the last customer that arrives at the cashier to get service

S[1]: variance of waiting times

S[2]: current number of staff

S[3]: number of customers waiting between a and b minutes

S[4]: number of customers waiting between b and c minutes

S[5]: number of customers waiting more than c minutes

S[6]: queue length (maximum line length among active cashiers).

Since the actual state of a system typically contains an excess of information, we abstracted the overall system state into seven numbers. This action assumes that this array contains enough information about the system's state for the ANN to find an optimal policy. The training experiment setup is how each t seconds the ANN updates the status vector.

4.3.1.2 Do Action function

This function requires a numerical input (referred to as an action) and, based on that value, and will perform a specific action in the model. In our problem, there are three actions, do nothing (if the action argument is 0), remove a worker (if the action argument is 1), and add a worker (if the action argument is 2). To make the action function clearer, we first consider a problem without any constraints for removing a worker. In this case, according to the trained policy, there are three actions during the simulation, including adding a worker, removing a worker, or doing nothing. We trained the model for a point of the sale grocery store for this case, and we recorded the number of staff every 15 minutes. Table 4.2 shows the output of the number of staff members for this case.

There are some conditions for removing a staff member in this research problem, including their maximum working hours, D and minimum working hours, h.

Table 4.2: Number of staff member for each 15-minutes time block

Time block	Workers	Time block	Workers	Time block	Workers	Time block	Workers
1	1	15	11	29	11	43	14
2	1	16	11	30	11	44	14
3	1	17	11	31	11	45	14
4	1	18	11	32	11	46	14
5	1	19	11	33	10	47	14
6	1	20	11	34	6	48	14
7	1	21	11	35	5	49	14
8	1	22	11	36	14	50	14
9	5	23	11	37	14	51	14
10	11	24	11	38	14	52	14
11	11	25	11	39	14	53	14
12	11	26	11	40	14	54	14
13	11	27	11	41	14	55	14
14	11	28	11	42	14	56	14

Adding a new worker: Whenever a new staff member starts working, the new customers will check out at their counter. The previous customers continued waiting in the station they initially entered at the arrival time.

Removing a worker: There are two criteria for removing a worker:

1. The working hours of the staff member should be more than *h* hours and less than *D* hours.

2. When the number in queue is less than D, and when the worker is approaching the end of their shift, that counter stops accepting new customers. However, they continue serving the customers that are already in their line.

4.3.1.3 Reward

A reward is calculated based on the observations taken from the environment before and after taking action. We set the reward function as the step function. Since the action function includes removing workers, this prevents the DRL from increasing staff numbers into infinity. Here is the definition of reward function:

1. Waiting time less than w_1 minutes:

   $r(s, a) =$ number of staff members $- var(w^q)$

2. Waiting time between w_1 and w_2 minutes:

   $r(s, a) =$ number of staff members $- p_1 f(w_1 < w_i < w_2) - var(w^q)$

3. Waiting time more than w_2 minutes:

   $r(s, a) =$ number of staff members $- p_1 f(w_1 < w_i < w_2) - p_2 f(w_i \geq w_2) - var(w^q)$

Where $f(x)$ is the number of customers waiting for a certain amount of time. The p_1 and p_2 are penalty parameters, and w_i is the waiting time in the time block *i*. We present variance of waiting times over a day by $var(w^q)$.

4.3.1.4 Training experiment

We use the custom experiment in the researcher version of AnyLogic simulation software to train the best possible policy. We need to specify hyper-parameters (parameters set before the learning process related to the neural network). As a whole, these values determine how the learning occurs – which ranges from how many steps in each epoch (or episode), to how often to update the policy, to how much the network explores its environment.

Random seed: 1
Max step By epoch: 2880
Max step: 6000
Max size of experience replay: 28800
Number of step noop warmup: 10
Number of steps for episode greedy aneal: 8640
According to the "Traffic light example model" [56], we set these parameters that the AnyLogic company made available on their website for others to use. If we conduct systematic parameter tuning, that might lead us to a better quality of training. It is an area for further research to select the training's hyperparameters and investigate their impact on the quality of training.

Next is the need to set up the structure of the neural network. As the observation consists of seven values, the input layer has seven neurons (or the input layer is built based on the observation space's size). The neural network with more than one hidden layer is considered a deep neural network. The number of hidden layers in deep learning depends on the complexity of the problem. Problems with more features and complex relationships between inputs and outputs require a greater number of hidden layers. It is worth noting that an increase in the number of hidden layers increases the possibility of over-fitting. The network we used to train in the present research has two hidden layers and an output layer with three neurons (number of possible actions). We did not perform more exploration for choosing the number of hidden layers, as this problem is not complicated to train. However, it can be a direction for future research. After training, the neural network determines whether or not to take any action. The network's overall structure is a simple, fully connected, feed-forward network. A diagram of it can be seen in figure 4.1 below.

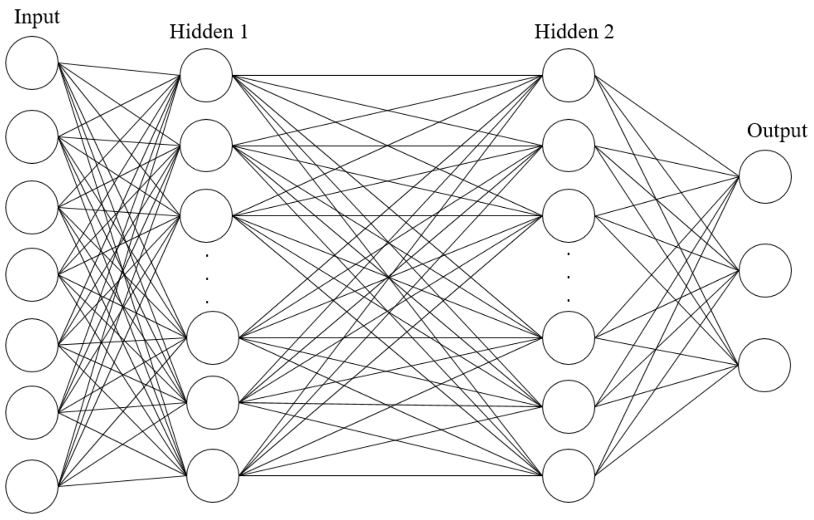
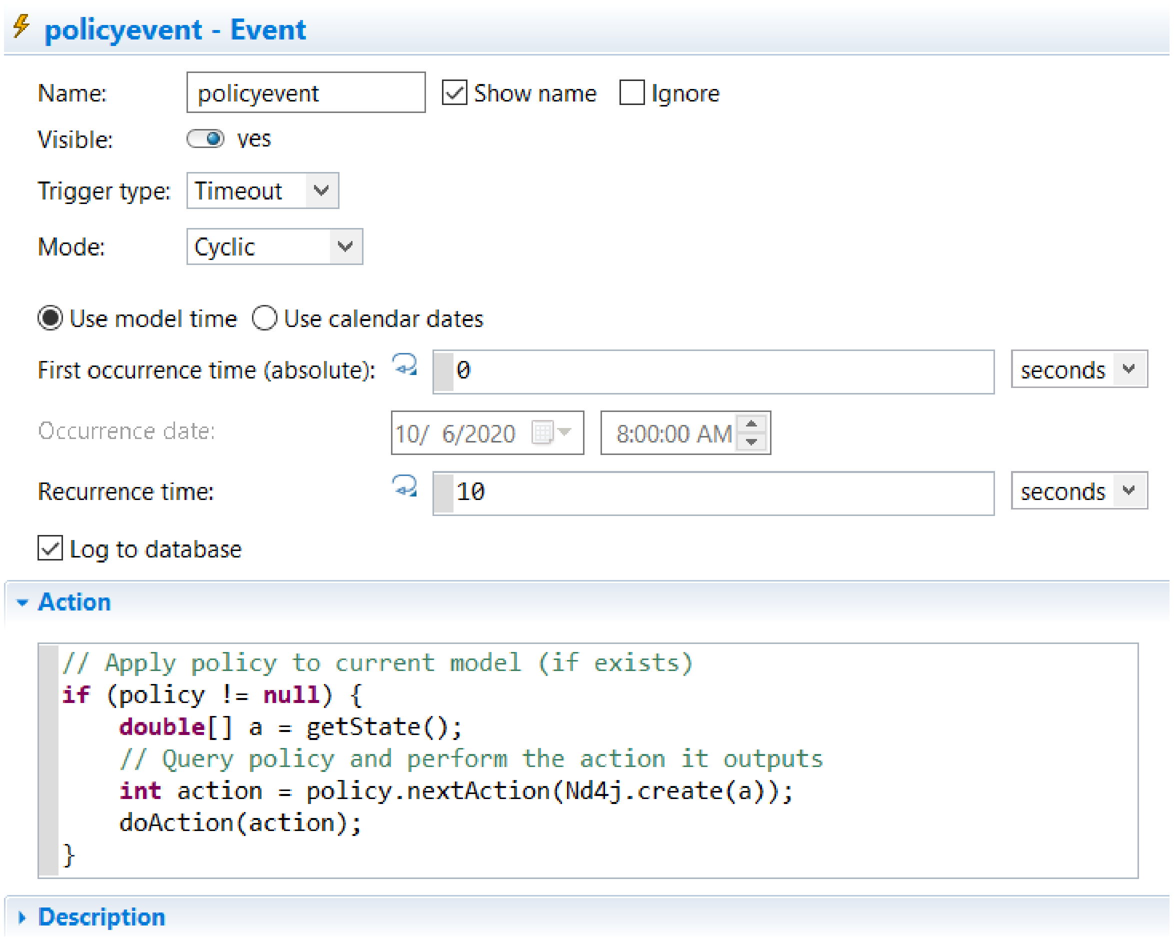


Figure 4.1: Structure of artificial neural network

To get the staff scheduling, first, we need to train the model and then use the best possible policy in a simulation run. The result is the staff scheduling as the simulation's output. To replicate the model and reduce the output variability, we design experiments in the "parameter variation" experiment of the AnyLogic software.

The output of this training is a set of weights for the links of the neural network. To use this trained policy in the simulation, we define an event to connect with AnyLogic's simulation environment. Figure 4.2 illustrates a screenshot view of this event in the AnyLogic software.


policyevent - Event

Name:	policyevent [✓ Show name] [ ] Ignore
Visible:	yes
Trigger type:	Timeout
Mode:	Cyclic

[● Use model time] ○ Use calendar dates

First occurrence time (absolute):	0	seconds
Occurrence date:	10/ 6/2020 8:00:00 AM	
Recurrence time:	10	seconds

[✓ Log to database]

Action

```
// Apply policy to current model (if exists)
if (policy != null) {
    double[] a = getState();
    // Query policy and perform the action it outputs
    int action = policy.nextAction(Nd4j.create(a));
    doAction(action);
}
```

Description

Figure 4.2: Event in AnyLogic to perform the trained policy and output the action

4.4 Description of how to use Deep Reinforcement Learning as a tool to determine staff rostering

To describe the procedure of flexible staff scheduling using Deep Reinforcement Learning, we need to determine the inputs for the software tool and the procedure. Figure 4.3 illustrates the procedure of staff rostering with this tool.

Two inputs are used with the tool, including the arrival process and the target waiting time that we describe in the following sections. We use the simulation model as the environment to train the best possible policy. This best possible policy is the set of weights of the neural network links after training. Reinforcement learning uses these link weights to find the best actions which will lead us to the best policy. The inputs to the neural network at each time epoch are the states of the system that the learning agent observes from the environment, and the output is the action. The action function is the set of possible actions. Based on the system's state, the learning agent takes action and earns a corresponding reward. According to the observed states and earned reward, if it is necessary, the action will be modified. After the process of learning is completed, the output is the trained policy. We simulate with this trained policy as the input, and the output of the simulation will be the staff scheduling. We can replicate the simulation model to reduce the output variability and build the confidence interval for the average number of staff in each time block over a day.

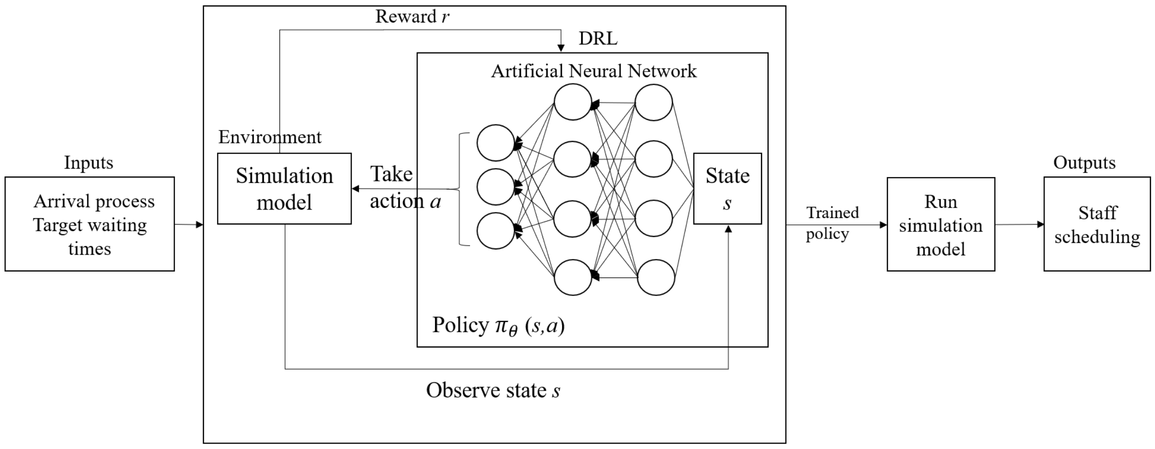


Figure 4.3: Process of staff scheduling with using DRL

4.4.1 Definition of inputs

There are two inputs in this process, including the target waiting time and the arrival process.

Definition of target waiting time: The target waiting time is a number that determines the maximum waiting time we expect a customer to experience before getting service. For example, if the target waiting time is 15 minutes, serving each customer must begin before the waiting time in the queue reaches 15 minutes.

Definition of arrival process: The arrival process represents the mean customers' arrival rates each day; we show the arrival rate function with $\lambda(t)$. The assumptions about the arrival process are as follows:

1. Customers arrive individually (not in batches) to the system.

2. We use the mean arrival rate per time block during a day. These time buckets are not required to be equal in length.

3. The arrival rate function ($\lambda(t)$) can follow different patterns during the time horizon, such as sinusoidal, bi-modal, and step patterns. We will illustrate these patterns in section 4.6.1. We put the mean arrival rates over the day into the simulation software as the input, and it generates data that follows Poisson distribution in each time block, for each pattern, regardless of the overall arrival rate function's pattern (most commercial simulation software uses a thinning-based approach to generate observations from a non-stationary Poisson process).

Table 4.3 shows the sample of the arrival process to the system.

Table 4.3: Arrival process example for one day

Time	$\lambda(t)$
08:00-09:00	10
09:00-10:00	25
10:30-11:00	30
11:00-11:30	50
11:30-12:00	80
12:00-01:00	150
01:00-02:00	200
02:00-03:00	330
03:00-04:00	150
04:00-05:00	95
05:00-06:00	70
06:00-07:00	120
07:00-08:00	175
08:00-09:00	200
09:00-10:00	230

Figure 4.4 presents the process we follow, based on the availability of arrival data, to construct the arrival process. According to this process, if the data is available, we need to check if the data distribution for different days is the same. If it is the same, we can determine staff rostering for one day and then apply the same staff scheduling for each day. Note that we assumed that each staff member's working hours in each day are independent of another day. In other words, we do not consider the case where a staff member worked more in one day and needed to work less in the following day.

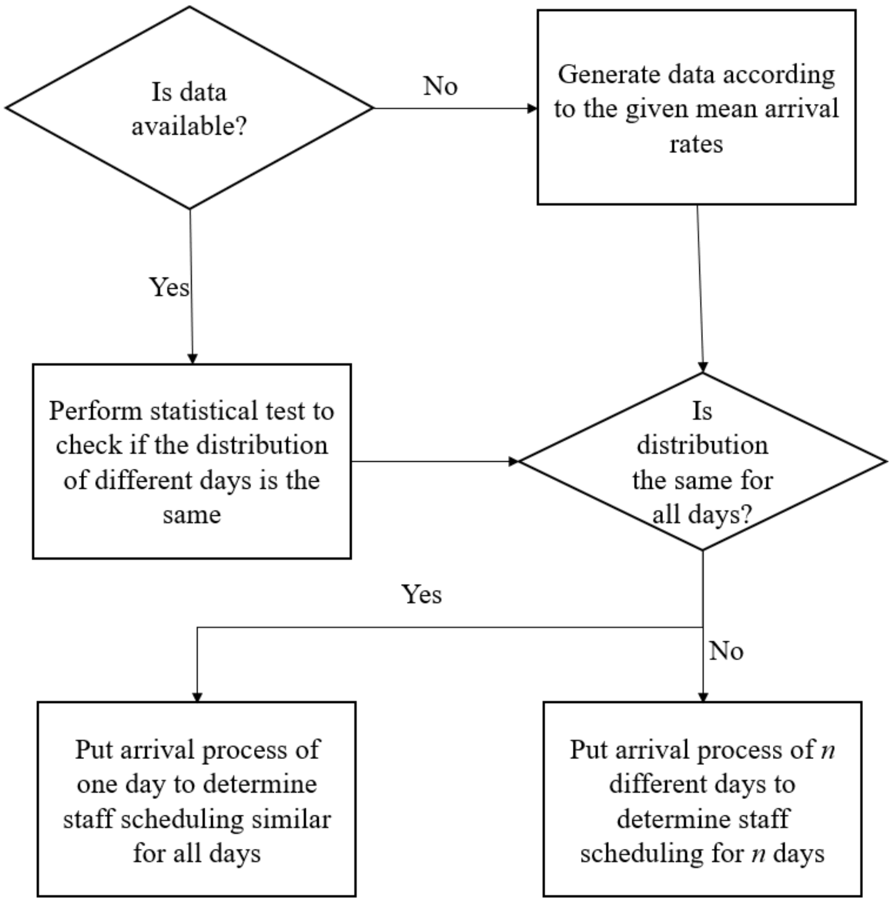

Is data available?
No
Generate data according to the given mean arrival rates
Yes
Perform statistical test to check if the distribution of different days is the same
Is distribution the same for all days?
Yes
No
Put arrival process of one day to determine staff scheduling similar for all days
Put arrival process of n different days to determine staff scheduling for n days

Figure 4.4: Process to follow based on availability of the data

Figure 4.5 illustrates the process we followed to use the arrival data we retrieved from a paper that published the point of sale of a grocery store [54]. First, we aggregate the data from different files and break down the data into individual days. This data set includes three days of the week, including Thursday, Friday, and Saturday. Then we calculate the mean arrival rate per hour for each day. After this step, we test two hypotheses. First, we test if the distribution of the same day of each week is the same. For instance, where all Thursdays are the same versus when they are not. The second hypothesis deals with if the different days of a week have the same distribution. For example, if Thursday and Friday's distribution is the same.

We conducted a Kolmogorov–Smirnov test for this purpose. If the days of a week have the same distribution, we put the mean arrival rate of one day as the input to the process, determine staff scheduling for a day, and use the same staff scheduling for other days. If the distribution of days is not the same, we determine staff scheduling for each day.

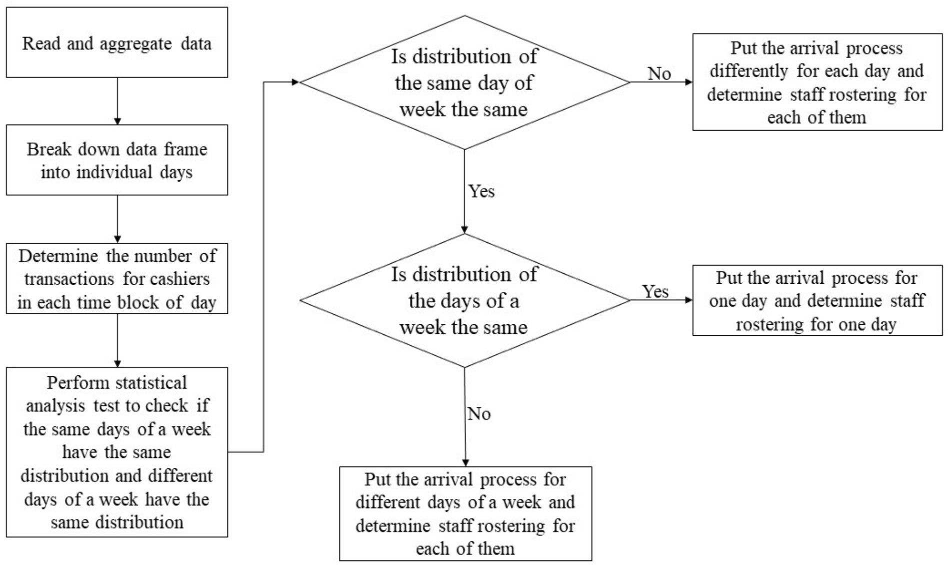

Read and aggregate data
Break down data frame into individual days
Determine the number of transactions for cashiers in each time block of day
Perform statistical analysis test to check if the same days of a week have the same distribution and different days of a week have the same distribution
Is distribution of the same day of week the same
No
Put the arrival process differently for each day and determine staff rostering for each of them
Yes
Is distribution of the days of a week the same
Yes
Put the arrival process for one day and determine staff rostering for one day
No
Put the arrival process for different days of a week and determine staff rostering for each of them

Figure 4.5: Process to prepare arrival process of point of sale data set

4.4.1.1 The case arrival data is not the same from day-to-day

In this case, we have two options to choose from as the inputs that go into the process. The first step is to set each day's arrival process and to train the policy. Next, one must simulate to get the staff rostering for that day separately. Another option is to put all days' arrival processes as the input, train the model once, run the simulation, and get the staff scheduling for all days. To decide which option is more efficient, we compare the training time of each case. We trained the model with one day, three days, ten days, and 100 days of arrival data with two cases, with the same distribution and pattern and completely different distributions and patterns.
Table 4.4 illustrates the training time that did not change significantly in different cases.

According to this table, we hypothesize that the training time is not dependent on the simulation's run length for this set of experiments. We test this hypothesis by comparing the staff rostering in two cases. First, we train and simulate the model with the same arrival

Table 4.4: Training time for different cases

Length of model time	Distribution	Replication number	training time
1 day		1	1:46.80
1 day		2	1:49.17
3 days	Same	1	1:49.64
3 days	same	2	1:52.44
3 days	Different	1	1:52.49
3 days	Different	2	1:50.86
10 days	same	1	1:49.08
10 days	same	1	1:48.02
10 days	Different	1	1:49.51
10 days	Different	1	1:50.62
100 days	same	1	1:52.12
100 days	same	2	1:49.89
100 days	Different	1	1:51.52
100 days	Different	2	1:49.38

pattern. Next, we train the model with a different arrival process than the one we will simulate. Then, we compare the output of these two cases together.

In this problem, there are three actions that the learning agent can take. They include doing nothing, decreasing or increasing the number of staff members. This increase or decrease is based on the waiting time of customers, the working hours of each staff member, and the line length. As soon as the training agent learns when to take the best possible action, the training will stop, giving the best possible policy as the output. As a result, having various days in simulation with different distributions and patterns will not significantly affect the training time in these sets of experiments.

Based on a small pilot experiment, we hypothesize that the trained neural net will perform well for any non-stationary arrival process regardless of the specific arrival process used to train the model. To test this hypothesis, we designed the following sets of experiments.

We use the step arrival pattern and the inverse U-shaped pattern to train the best possible policy and then get the staff rostering for sinusoidal and bi-modal patterns. Also, we use these trained policies to get staff scheduling for point of sale input data of the grocery store. Tables 4.5, 4.6, and 4.7 show the rounded-up average number of staff needed in each time block in three different cases. The first average is when we trained and ran the simulation with the same arrival process. The second and third columns are for the cases in which the model has been trained with a step arrival pattern and an inverse U-shaped pattern. Then it is simulated with another arrival process.

Table 4.5: The rounded up average number of staff for the case in which we trained the model with a step arrival rate pattern, a reversed U-shaped pattern, and then ran it with a bi-modal arrival process pattern

Time	trained and ran with the same arrival process	trained with the step arrival process	trained with the Inverse U-shape pattern
08:00-09:00	1	1	1
09:00-10:00	11	11	11
10:00-11:00	13	11	13
11:00-12:00	13	13	13
12:00-01:00	9	9	9
01:00-02:00	13	13	13
02:00-03:00	13	13	13
03:00-04:00	11	11	11
04:00-05:00	9	9	9
05:00-06:00	9	9	9
06:00-07:00	12	12	12
07:00-08:00	13	13	13
08:00-09:00	12	12	12
09:00-10:00	11	11	11

Table 4.6: The rounded up average number of staff for the case in which we trained the model with a step arrival rate pattern, a reversed U-shaped pattern, and then ran it with a sinusoidal arrival process pattern

Time	trained and ran with the same arrival process	trained with the step arrival process	trained with the Inverse U-shape pattern
08:00-09:00	13	13	13
09:00-10:00	13	13	13
10:00-11:00	13	13	13
11:00-12:00	7	7	7
12:00-01:00	8	8	8
01:00-02:00	12	12	12
02:00-03:00	12	12	12
03:00-04:00	12	12	12
04:00-05:00	9	9	9
05:00-06:00	12	12	12
06:00-07:00	12	12	12
07:00-08:00	11	11	11
08:00-09:00	11	11	11
09:00-10:00	10	10	10

After we simulate the trained model with a step arrival process pattern for bi-modal and sinusoidal arrival patterns, we use the same trained policy to get the staff rostering for point of sale data of a grocery store. The result of staff scheduling in table 4.7 shows we result in the same staff scheduling if we train the model with another arrival pattern, and then run the simulation for this arrival process of customers to the grocery store, with the condition that all other settings of the simulation model and the training stay the same.

According to this small pilot set of experiments, we suggest to train the policy with one non-stationary arrival process pattern the same way as the ones we have tested above, and then use it for other patterns (among the ones we tested) as long as other settings and model conditions are the same. So when there are different arrival rate patterns, the same as ones we tested here, for *n* days, in terms of training time, it is preferred to set the arrival pattern of all days as the input to the simulation to train the best possible policy at once. If there

Table 4.7: The rounded up average number of staff for the case model has been trained with a step arrival rate pattern and a reversed U-shaped pattern, and then it has been run with point of sale arrival data

Time	trained and run with the same arrival process	trained with the step arrival process	trained with the Inverse U-shape pattern
08:00-09:00	1	1	1
09:00-10:00	1	1	1
10:00-11:00	7	7	7
11:00-12:00	11	11	11
12:00-01:00	12	12	12
01:00-02:00	10	10	10
02:00-03:00	11	11	11
03:00-04:00	14	14	14
04:00-05:00	13	13	13
05:00-06:00	12	12	12
06:00-07:00	12	12	12
07:00-08:00	11	11	11
08:00-09:00	11	11	11
09:00-10:00	11	11	11

are some changes in the model, like varying the target waiting time, or maximum working hours of each worker, we need to train the model again.

4.4.2 Definition of components of the simulation model

- Service rate: The mean service time in this problem follows a non-Poisson distribution function. The service rate shows how many customers can be served in the time unit.
- Queuing discipline: Each server has its line in this problem, and the new customer will choose the line with the minimum number of customers. For closing a line, we will consider a server that worked for a certain number of hours. The server will continue serving the customers that have already been in their line.
- Length of each day: In this problem, each day starts at 8:00 a.m. and stops accepting new customers at 10:00 p.m., but it will continue working to serve the customers who arrived at the system before 10:00 p.m.
- Replication of simulation model: We use an experiment in the simulation software to replicate the simulation results of staff rostering.

4.5 Model validation and verification

Model verification has been implemented in several steps. We used several features of the ”Analysis” and ”Agent” parts of the Palette view of AnyLogic simulation software to verify the model, including statistics, plot, functions, variables, and parameters. For instance, we used ”traceln()” in many places to print the results during the run. Figures 4.6 and 4.7 show two screen-shots of the verification process. Figure 4.6 represents various functions, statistics, and features of AnyLogic we have used to verify the model's performance. These features help to collect the waiting time of each customer in self-check-outs and cashiers, the start time of each cashier, and the total working hours of each cashier. In addition, functions calculate average waiting time, the variance of waiting time, and the number of customers who waited for a specific time amount.

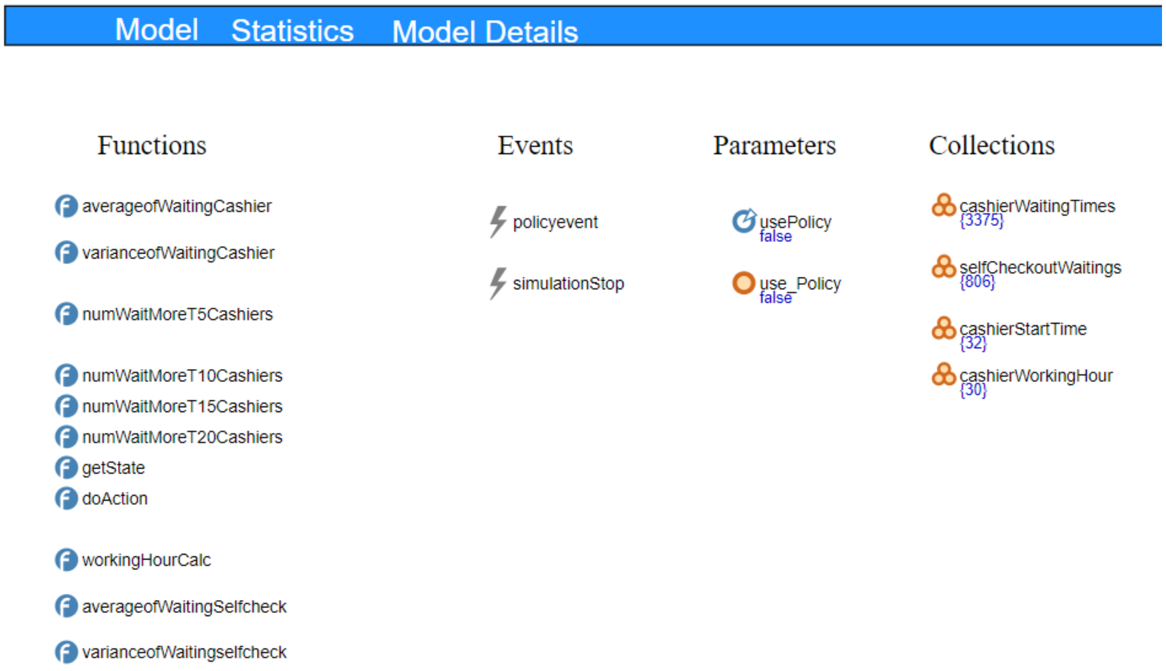
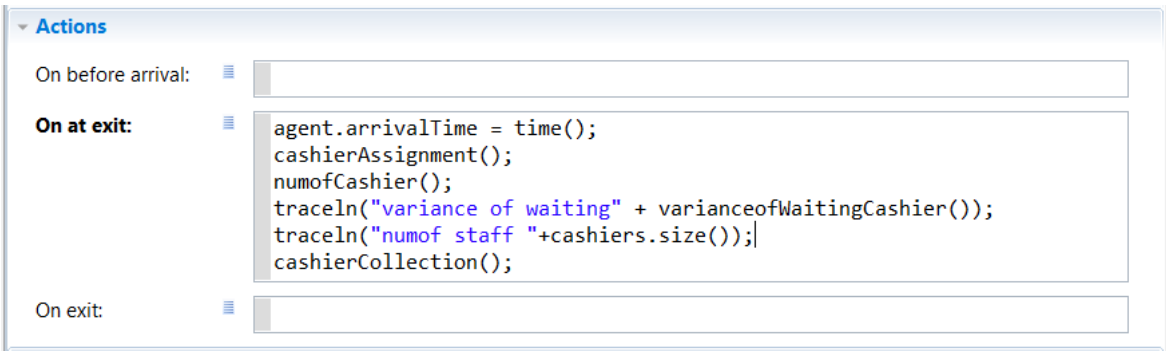


Figure 4.6: Functions, parameters, and variables in simulation model



```
agent.arrivalTime = time();
cashierAssignment();
numofCashier();
traceln("variance of waiting" + varianceofWaitingCashier());
traceln("numof staff "+cashiers.size());
cashierCollection();
```

Figure 4.7: One trace line view of the simulation model

Model verification has been implemented gradually during model building. In this part, we tried to give only a general overview representing model verification, and it does not include a complete verification process.

We also investigate the performance of two solution approaches, Deep Reinforcement Learning (DRL) and Simulation-based Optimization (SBO). In this case, the arrival process is constant over the day. Table 4.8 shows that the staffing is the same for both solution approaches, and it is as we expected. Since the arrival rate does not vary during the day, the expectation is to have constant staffing levels.

Table 4.8: Result of two solution approaches when average arrival rate is constant during a day

Time	**DRL**	**Time**	**SBO**
08:00-09:00	3	08:00-09:00	3
09:00-10:00	3	09:00-10:00	3
10:00-11:00	3	10:00-11:00	3
11:00-12:00	3	11:00-12:00	3
12:00-01:00	3	12:00-01:00	3
01:00-02:00	3	01:00-02:00	3
02:00-03:00	3	02:00-03:00	3
03:00-04:00	3	03:00-04:00	3
04:00-05:00	3	04:00-05:00	3
05:00-06:00	3	05:00-06:00	3
06:00-07:00	3	06:00-07:00	3
07:00-08:00	3	07:00-08:00	3
08:00-09:00	3	08:00-09:00	3
09:00-10:00	3	09:00-10:00	3

For simulation model validation, we tested the performance of the simulation model using data set of arrivals to the grocery store. Figure 4.8 shows the overall view of a grocery store's point of sale. We use this as a use case in this research. In this system, some customers prefer to check out in cashiers and some in self-check-out counters. The self-check-out counters have shared queues, while cashiers have individual queues in this store.

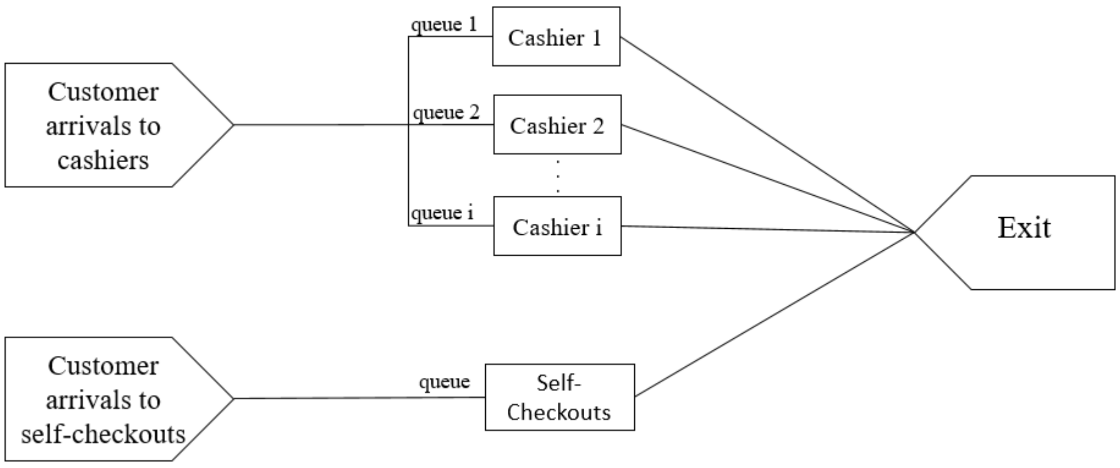


Figure 4.8: General overview of the point of sale in grocery store

Figure 4.9 illustrates a view of the model of this grocery store use-case. The simulation model with this data set works as expected in the real-world grocery store. For instance, a cashier's line gets closed when approaching its upper bound of working hours. In addition, all customers who chose to be served at the self-checkout counter stayed in one line. The system's simulation time is 10 hours, and it stops generating new customers when the simulation's clock is 10:00 p.m., but it continues serving customers that arrived before 10:00 p.m. as expected.

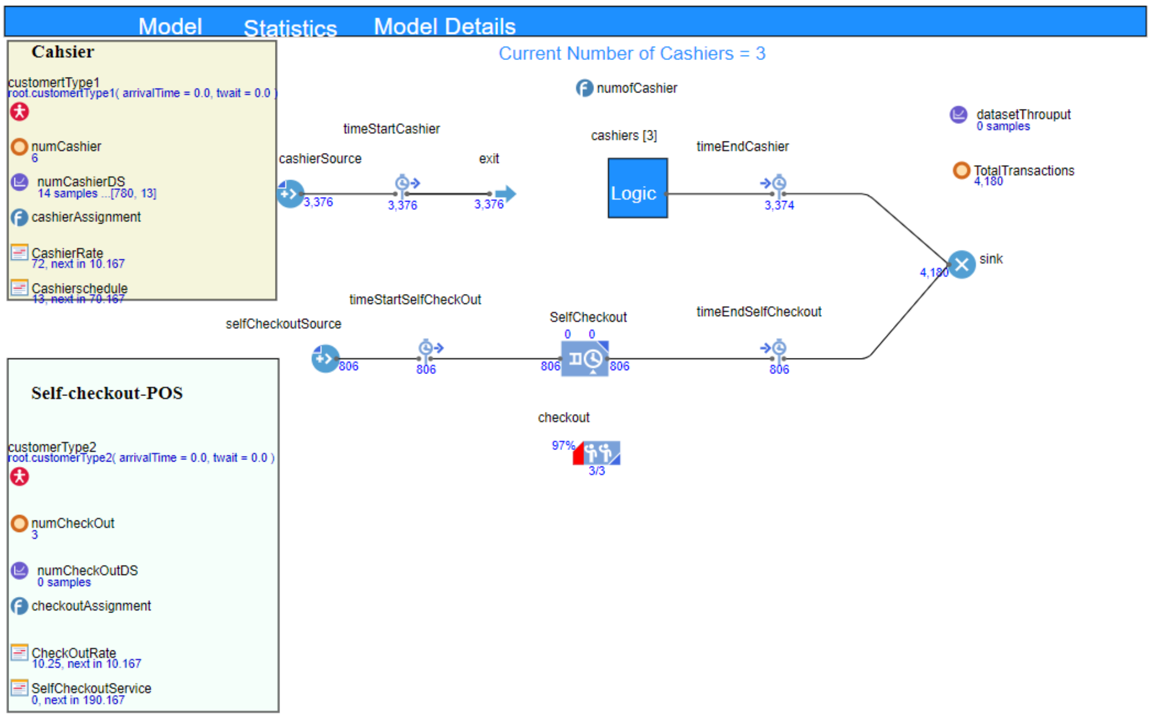


Figure 4.9: Simulation view of one experiment's run

4.6 Performance evaluation of the solution approach

Like other service systems, the possible input is the arrival rate pattern, and the possible outputs are the waiting time in the queue, line length, number of workers, and total working hours of them.

Table 4.9 illustrates inputs and outputs in the model with single task servers.

Table 4.9: Factors and responses in the single task queuing system model

Inputs	Outputs
Arrival rates	Time in queue
Penalty parameters	Number of required staff
	Queue length
	Variance of Waiting
	Staff start time
	Total working hours
	Total throughput

4.6.1 Arrival process

We talked about the arrival process and its definition in section 4.4.1. Since this problem is a queuing system problem, we can apply queue concepts and rules to determine characteristics of the arrival process, such as low and high variability in the arrival process over a planning horizon.

To evaluate the performance of the proposed approach to staff scheduling, we consider constant, step, sinusoidal, bi-modal, and uni-modal patterns in the arrival process. These patterns are representative of the variation in the mean arrival rates during a day.

If the peak arrival rate is much higher than the average arrival rate during a day, the arrival process has a high variability [28]. To define variability in the arrival process, we use the coefficient of variation ($\hat{C_v} = \frac{s}{\bar{x}}$). Distributions with $c.v < 1$ have variation less than Poisson distribution, and distributions with $c.v > 1$ are more variable than Poisson distribution.

Appendix A presents the results of performing the Deep Reinforcement learning algorithm to staff scheduling for different arrival process patterns, including constant (stationary), uni-modal, sinusoidal, step, and bi-modal arrival patterns to evaluate our proposed algorithm's performance. Figure 4.10 represents example of these arrival process patterns. We present the bi-modal arrival pattern in the top left part of this figure. As we can see, there are two peaks of arrivals to the system over a day. There are two cases with the step arrival process patterns that we applied as the input into the Deep Reinforcement Learning algorithm to determine the staff rostering over a day. On the bottom left of the figure are examples of the rival process with the uni-modal pattern with low and high variability in the arrival process.

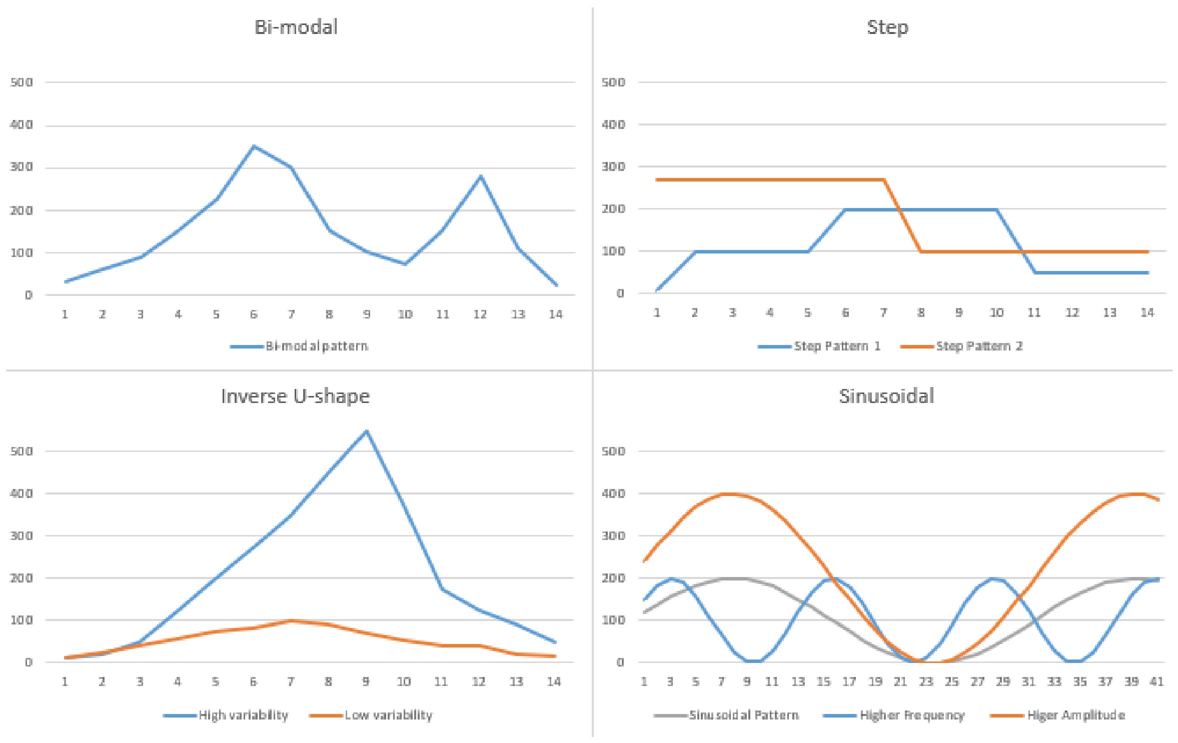


Figure 4.10: Example of arrival process patterns

We present three cases with sinusoidal arrival pattern. For first pattern, we used sinusoidal arrival rate as: $\lambda(t) = \bar{\lambda}(1 + \sin(0.2t))$ we assume $\bar{\lambda} = 100$ the same as the base model in existing related work [13]. For higher frequency and higher amplitude sinusoidal pattern we applied $\lambda(t) = 100(1 + \sin(0.5t))$ and $\lambda(t) = 200(1 + \sin(0.2t))$ arrival rate functions, respectively.

4.6.2 Penalty parameters levels

Penalty parameters are the ones included in the objective function of the nonlinear programming problem that was described in section 4.2. We designed the experiment with three levels, low, medium, and high penalty rates. This penalty is defined as the step function in the objective function, as we stated in equation 4.1. With an increase in customers' waiting time, the penalty cost will increase by a higher value.

By decreasing the value of penalty parameters, we enforce lower levels of staff to the system. In this case, the customers' waiting time increases the objective function less than the case penalty parameter values are high. Since this is a minimization problem, assigning lower staffing levels to the system over the planning horizon. According to the discussion in section 4.6.1 and this section, we will conduct a set of experiments as shown in table 4.10.

Table 4.10: List of the set of experiments

Run(i)	input 1	input 2	output
1	low constant arrival rate	big	R1
2	medium constant arrival rate	big	R2
3	high constant arrival rate	big	R3
4	uni-modal with low variability	small	R4
5	uni-modal with low variability	medium	R5
6	uni-modal with low variability	big	R6
7	uni-modal with high variability	small	R7
8	uni-modal with high variability	medium	R8
9	uni-modal with high variability	big	R9
10	oscillating pattern	big	R10
11	oscillating pattern with higher frequency	big	R11
12	oscillating pattern with higher amplitude	big	R12
13	step pattern 1	big	R13
14	step pattern 2	big	R14
15	U-shape with bi-modal pattern	big	R15

The results in appendix A indicate that DRL performs properly in staff scheduling. For instance, while staff scheduling with bi-modal arrival rate pattern, considering the working hour constraint, the output is as expected. The results show that the values of staffing levels will increase with an increase in the penalty parameters. The total number of workers in the case with high penalty parameters is higher than medium penalty parameters. Similarly, the total number of workers in medium penalty parameters is higher than in low penalty parameters.

4.6.3 Deciding about the number of required replications

Since these problems are stochastic, we can not rely on only one replication to determine the value of responses. To decrease the variability in the responses, we replicate the simulation experiment *n* times and construct a confidence interval.

If we wish to have a narrower confidence interval, we need to reduce *h*, which is the confidence interval length. For this purpose, we increase the number of replications, *n*. To do so we solve the following approximation equation for *n*. $h \simeq t_{1-\frac{\alpha}{2},n-1} \times s/\sqrt{n}$, then we have $n \simeq (\frac{t_{1-\frac{\alpha}{2},n-1} \times s}{h})^2$. Because still t-value is dependent on n, as we replicate enough (usually more than 30), we can substitute that with Z distribution. So we have, $n \simeq (\frac{z_{\frac{\alpha}{2}} \times s}{h})^2$. In this research, we replicate enough to reach the confidence interval length arbitrarily close to 1. In this way, we are 95% confident that the true mean, in this case, the staffing level is in that range LCL (Lower Confidence Interval) and UCL (Upper Confidence Interval), which are two consecutive numbers, and with rounding up to the nearest integer, the mean will always be equal to LCL and/or UCL.

4.7 Applying the DRL approach to the use case, point of sale data of the grocery store

This section uses a European grocery store's point of sale data set and applies our proposed approach to staff scheduling. The data are retrieved from the Point of Sale (POS) transaction times recorded for each customer at the cashier counters and the self-checkout machines in a grocery store, [54]. The studied store works from 8:00 a.m. to 10:00 p.m. on Thursdays, Fridays, and Saturdays. The data set consists of four Thursdays, four Fridays, and four Saturdays. These data allow us to obtain the number of transactions for each checkout option and the recorded transaction times for any given time frame. We split recorded transaction times into the POS data set into two different portions. The first data set represents the transaction time and, in particular, the time required to scan the items and accept the payment. The second data set represents the break time required for a customer to receive the receipt or the change, the time for bagging the items, and the idle time until the next customer's arrival. Figure 4.11 shows the number of transactions in cashiers on Thursdays, Fridays, and Saturdays. We perform the Kolmogorov- Smirnov test to test the hypothesis that these transactions are following the same distribution. The $p - value$ of the test for all combinations of comparing two days is between 0.63 and 0.999, indicating there is not enough evidence to assume the distribution of the number of transactions during Thursdays is not identical. We apply the same analysis for Fridays and Saturdays. The results show that the distribution on Fridays is the same, and distribution on Saturdays is the same with $p - value$ of in range (0.348,0.999) and (0.06, 0.91), respectively. For more investigation, we performed the Kolmogorov–Smirnov test to hypothesize if the distribution of transactions is different from day-to-day. That is it, from Thursday to Friday, from Friday to Saturday, and from Saturday to Thursday. The result shows $p - value$ is between 0.348 and 0.912 for different combinations, indicating there is not enough evidence to assume the distribution of the different days is not identical.

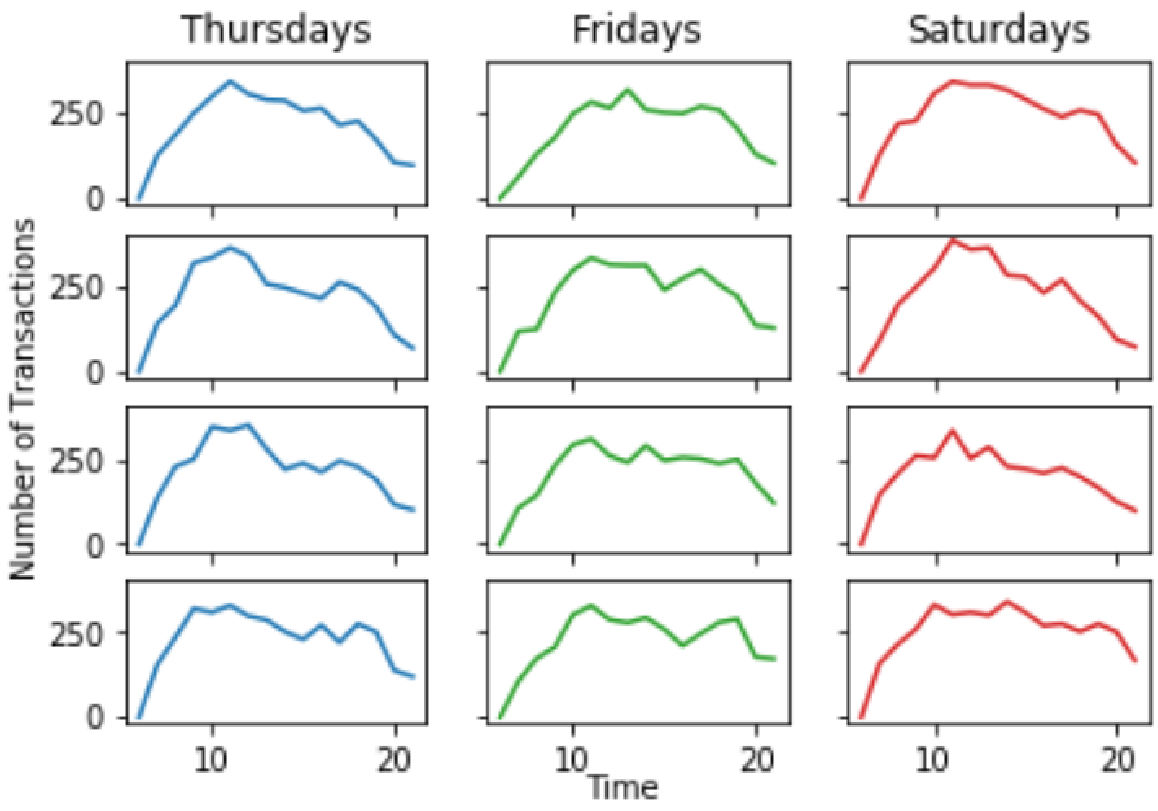


Figure 4.11: Cashiers' number of transactions during the day on Thursdays, Fridays, and Saturdays

After characterizing the arrival process, according to section 4.3, we implement the Deep Reinforcement Learning approach to get the number of cashiers over a day for point of sale in the grocery store. Table 4.11 illustrates these results.

Table 4.11: Experiment result for point of sale grocery use case, big penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded-Up Mean	LCL	UCL
08:00-09:00	1	0	1	1	1	1	1
09:00-10:00	1	0	1	1	1	1	1
10:00-11:00	6.05	0.9	5.15	6.95	7	6	7
11:00-12:00	10.9	0.37	10.53	11.27	11	11	12
12:00-01:00	11.05	0.37	10.68	11.42	12	11	12
01:00-02:00	9.02	0.35	8.67	9.37	10	9	10
02:00-03:00	10.44	0.34	10.1	10.78	11	11	11
03:00-04:00	13.03	0.29	12.74	13.32	14	13	14
04:00-05:00	12.98	0.54	12.44	13.52	13	13	14
05:00-06:00	11.21	0.56	10.65	11.77	12	11	12
06:00-07:00	11.37	0.77	10.6	12.14	12	11	13
07:00-08:00	11.74	0.8	10.94	12.54	12	11	13
08:00-09:00	11.23	0.81	10.42	12.04	12	11	13
09:00-10:00	10.13	0.82	9.31	10.95	11	10	11

At the beginning of the day, the mean arrival rate was low, and the required cashier was one in all replications of the simulation without any variability among replications.

4.8 Comparison between DRL and SBO approaches' outputs

In this section, we compare the resulted staff rostering of two solution approaches. We compare them in two aspects:

1. Efficiency
2. Performance

4.8.1 Efficiency comparison

Here is the comparison between two approaches in terms of efficiency:

1. DRL approach does not require performing change-point detection to detect changes in input data.
2. The DRL approach's output is a staff rostering, but the output of the SBO method is staffing levels, and there is a need to perform one more step to get staff rostering.

According to these cases, the DRL approach seems to be more efficient. We need to apply an algorithm that performs well enough to estimate change-point close to their actual location to perform change-point detection. Also, not performing staff rostering at the same step as determining staffing levels causes some other differences between the two approaches output. For instance, by two-step staff scheduling, there is a possibility of over-staffing, as discussed in section 3.5. In this case, first, we apply the problem's constraints to determine staffing level and then apply working hours constraints on the optimal staffing level. This will increase the possibility of over-staffing in a staff scheduling problem.

4.8.2 Performance

To compare the two methods' performance, we compare the total number of staff during a day and the variance of waiting times over a day. Note that in chapter 3, we needed to consider the average of waiting times for each time block instead of the overall average. The reason is that if some of the time blocks have a high value of waiting times, but there are some other time blocks with low waiting times, the overall average will not be affected by those high values. Unlike, variance is sensitive to the large values, and if there are some of the time blocks with high variance, the overall variance will be high. Because of this, we will consider the overall variance of waiting time over a day and not in each time block to evaluate the proposed methodology to achieve predictable waiting times. To test if statistically there is any difference between the two methods' performance, we conducted two sample t-Test. For this purpose, we simulated the system with each pattern's staff schedule 30 times, and we compared two metrics, the average number in the system and average time in the system.

4.8.2.1 Uni-modal arrival pattern with low variability in arrival process

Table 4.12 represents the staff rostering for both approaches. The total number of staff resulting from DRL is 73, while the resulted one from SBO is 118, which indicates a higher staffing cost of SBO methodology. In addition, the variance of waiting times is lower in DRL than SBO, which indicates the waiting times resulting from the DRL approach are less variable.

Table 4.12: Mean staff rostering resulted from DRL and SBO- Low variability in arrival process and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	1	1
09:00-10:00	2	1
10:00-11:00	3	4
11:00-12:00	8	15
12:00-01:00	9	15
01:00-02:00	9	15
02:00-03:00	8	13
03:00-04:00	8	14
04:00-05:00	7	10
05:00-06:00	6	9
06:00-07:00	4	6
07:00-08:00	3	5
08:00-09:00	3	5
09:00-10:00	2	5
Total staff	73	118
Variance of waiting times	1.66	2.29

Tables 4.13 and 4.14 present the average performance metrics and the t-test for two approaches. the p-value of the test for both performance metrics is higher than 0.05, which indicates there is no statistical difference between the average number in the system and the average time in the system of the two schedules.

Table 4.13: Performance metrics comparison

	DRL	**SBO**
Average number in system	1.42	1.45
Average time in system(minutes)	2.05	2.08

Table 4.14: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.59	0.56
Average time in system	-0.17	0.86

4.8.2.2 Uni-modal pattern with high variability in arrival process

Table 4.15 shows the staff rostering and the total number of staff as well as the overall variance of waiting time over a day. The total number of staff is lower in DRL than SBO, which results in a lower total staffing cost than SBO. In addition to the lower staff cost, DRL methodology resulted in the lower variability in the waiting times than SBO. Tables 4.16 and 4.17 indicate there is not significant difference between the average number in system and average time in system of two schedules.

Table 4.15: Resulted staff rostering of DRL and SBO- High variability in arrival process and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	1	1
09:00-10:00	1	1
10:00-11:00	6	12
11:00-12:00	6	15
12:00-01:00	13	15
01:00-02:00	14	15
02:00-03:00	15	10
03:00-04:00	16	13
04:00-05:00	16	13
05:00-06:00	15	13
06:00-07:00	13	16
07:00-08:00	13	16
08:00-09:00	13	16
09:00-10:00	12	9
Total staff	154	165
Variance of waiting times	7.69	22.06

Table 4.16: Performance metrics comparison

	DRL	**SBO**
Average number in system	3.38	3.42
Average time in system(minutes)	1.89	1.90

Table 4.17: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.23	0.81
Average time in system	-0.38	0.70

4.8.2.3 Oscillating pattern

In this part, we investigate three sinusoidal patterns. We apply sinusoidal arrival pattern with arrival rate function: $\lambda(t) = \bar{\lambda}(1+\sin(0.2t))$, we assume $\bar{\lambda} = 100$ the same as the base model in existing works in literature [13]. The next two oscillating patterns are with higher frequency and higher amplitude than this pattern, respectively. Table 4.18 represent the staff rostering results from DRL and SBO methods when the arrival pattern is sinusoidal. As shown, the DRL approach ended up with a lower staffing cost and variance of waiting times.

Table 4.18: Resulted staff rostering of DRL and SBO- Sinusoidal arrival process and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	13	10
09:00-10:00	13	16
10:00-11:00	13	16
11:00-12:00	7	16
12:00-01:00	8	16
01:00-02:00	12	16
02:00-03:00	12	6
03:00-04:00	12	6
04:00-05:00	9	14
05:00-06:00	12	14
06:00-07:00	12	14
07:00-08:00	11	14
08:00-09:00	11	9
09:00-10:00	10	13
Total staff	155	180
Variance of waiting times	3.56	5.58

Table 4.19: Performance metrics comparison

	DRL	**SBO**
Average number in system	2.30	2.34
Average time in system(minutes)	1.81	1.84

Table 4.20: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.55	0.29
Average time in system	-0.45	0.65

4.8.2.4 Oscillating pattern- higher frequency

In this experiment, we compare the performance of DRL and SBO for the arrival process with a sinusoidal pattern and higher frequency than the arrival process in the last part. The arrival rate function is $\lambda(t) = \bar{\lambda}(1 + \sin(0.5t)$. Table 4.21 represent staff scheduling of two methodologies. The required number of workers and variability in waiting times from applying DRL is lower than the SBO method. These outputs indicate that DRL performed better in this example.

Table 4.21: Resulted staff rostering of DRL and SBO- Sinusoidal arrival process with higher frequency and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	12	16
09:00-10:00	8	16
10:00-11:00	12	16
11:00-12:00	11	6
12:00-01:00	10	16
01:00-02:00	10	16
02:00-03:00	9	15
03:00-04:00	10	6
04:00-05:00	11	15
05:00-06:00	10	15
06:00-07:00	1	5
07:00-08:00	8	15
08:00-09:00	11	15
09:00-10:00	11	15
Total staff	134	186
Variance of waiting times	0.68	4.96

Table 4.22: Performance metrics comparison

	DRL	**SBO**
Average number in system	3.36	3.45
Average time in system(minutes)	3.30	3.38

Table 4.23: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.38	0.70
Average time in system	-0.39	0.69

4.8.2.5 Oscillating pattern- higher amplitude

In this example, the amplitude is higher than the base sinusoidal arrival pattern. The arrival rate function is $\lambda(t) = \bar{\lambda}(1 + \sin(0.2t)$, where $\bar{\lambda}$ is 200. Table 4.24 shows staff rostering resulted from applying these approaches. The DRL reported a lower number of staff members and variance of waiting times than SBO.

Table 4.24: Resulted staff rostering of DRL and SBO- Sinusoidal arrival process with higher amplitude and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	11	9
09:00-10:00	11	11
10:00-11:00	11	16
11:00-12:00	10	16
12:00-01:00	9	9
01:00-02:00	9	9
02:00-03:00	11	16
03:00-04:00	15	16
04:00-05:00	16	16
05:00-06:00	15	16
06:00-07:00	15	16
07:00-08:00	15	16
08:00-09:00	16	16
09:00-10:00	14	16
Total staff	178	205
Variance of waiting times	3.78	5.35

Table 4.25: Performance metrics comparison

	DRL	**SBO**
Average number in system	4.24	4.25
Average time in system(minutes)	1.80	1.81

Table 4.26: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.05	0.96
Average time in system	-0.07	0.94

4.8.2.6 Bi-modal arrival process pattern

In this case, the arrival pattern is bi-modal with the two peaks of arrivals. Table 4.27 represents the staff rostering of two methods. The DRL approach resulted in a lower staffing requirement than SBO. Also, we can observe a pattern in staff rostering. When the arrival rate was higher, more workers, and when it is lower, fewer staff members. However, SBO did not perform well in this example. Both required staffing levels and variance of waiting times are higher in SBO than in the DRL approach.

Table 4.27: Resulted staff rostering of DRL and SBO- bi-modal arrival process and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	2	2
09:00-10:00	8	7
10:00-11:00	9	10
11:00-12:00	9	10
12:00-01:00	11	10
01:00-02:00	15	13
02:00-03:00	15	16
03:00-04:00	15	16
04:00-05:00	11	16
05:00-06:00	9	16
06:00-07:00	12	16
07:00-08:00	13	16
08:00-09:00	12	16
09:00-10:00	11	16
Total staff	152	180
Variance of waiting times	0.93	10.02

Table 4.28: Performance metrics comparison

	DRL	**SBO**
Average number in system	3.41	3.42
Average time in system(minutes)	1.76	1.76

Table 4.29: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.07	0.94
Average time in system	-0.15	0.88

4.8.2.7 Step pattern 1 in arrival process

In this example, we investigate the arrival process with a step pattern. Table 4.30 represents staff rostering of two methodologies. The DRL ended up with a lower staff requirement and variance of waiting times. Table 4.32 shows there is a significant difference between the average time in system of two schedules.

Table 4.30: Resulted staff rostering of DRL and SBO- step pattern 1 in arrival process and big penalty parameters

Time	**DRL**	**SBO**
08:00-09:00	1	1
09:00-10:00	14	14
10:00-11:00	14	14
11:00-12:00	14	14
12:00-01:00	10	14
01:00-02:00	12	14
02:00-03:00	12	13
03:00-04:00	11	8
04:00-05:00	12	15
05:00-06:00	12	15
06:00-07:00	12	15
07:00-08:00	11	15
08:00-09:00	10	15
09:00-10:00	10	15
Total staff	155	182
Variance of waiting times	1.01	2.16

Table 4.31: Performance metrics comparison

	DRL	**SBO**
Average number in system	2.63	2.65
Average time in system(minutes)	2.05	2.06

Table 4.32: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.1	0.90
Average time in system	-0.08	0.94

4.8.2.8 Step pattern 2 in arrival process

Table 4.33 shows staff scheduling resulted from two methods, DRL and SBO. This step pattern includes two uniform arrival rates; the second has the lower arrival rates. Like previous examples, the DRL approach resulted in fewer workers and lower variance of waiting times.

Table 4.33: Resulted staff rostering of DRL and SBO- step pattern 2 in arrival process and big penalty parameters

Time	DRL	SBO
08:00-09:00	15	14
09:00-10:00	15	14
10:00-11:00	15	14
11:00-12:00	14	14
12:00-01:00	14	14
01:00-02:00	10	14
02:00-03:00	10	12
03:00-04:00	10	12
04:00-05:00	10	12
05:00-06:00	10	12
06:00-07:00	10	12
07:00-08:00	10	12
08:00-09:00	10	12
09:00-10:00	10	12
Total staff	163	180
Variance of waiting times	2.31	7.65

Table 4.34: Performance metrics comparison

	DRL	SBO
Average number in system	2.24	4.30
Average time in system(minutes)	2.62	4.45

Table 4.35: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	0.21	0.83
Average time in system	0.18	0.86

4.8.2.9 Point of sale in grocery store

After performing the DRL with different arrival process patterns and comparing the outputs with SBO, we use the point of sale data in the grocery store to get staff scheduling. In table 4.36, we represent staff rostering of two methods. The DRL gave the lower total number of workers. Also, the variability in the waiting times is lower in the staff schedule resulting from the DRL approach than the SBO.

Table 4.36: Average staff rostering of DRL and SBO- point of sale arrival process and big penalty parameters

Time	DRL	SBO
08:00-09:00	2	1
09:00-10:00	2	1
10:00-11:00	7	8
11:00-12:00	11	14
12:00-01:00	12	14
01:00-02:00	10	14
02:00-03:00	11	16
03:00-04:00	14	14
04:00-05:00	15	14
05:00-06:00	12	14
06:00-07:00	13	16
07:00-08:00	12	16
08:00-09:00	13	16
09:00-10:00	11	10
Total staff	145	168
Variance of waiting times	2.40	4.55

Table 4.37: Performance metrics comparison

	DRL	SBO
Average number in system	3.23	2.10
Average time in system(minutes)	2.09	3.24

Table 4.38: Statistical test to compare performance metrics

	t-Stat	*p-value*
Average number in system	-0.12	0.90
Average time in system	-0.11	0.91

4.9 Conclusion

This chapter proposed a methodology in DRL to flexible staff scheduling in a non-stationary service system. This method uses a neural network to train the best possible policy to determine when and how to change the number of workers. Unlike other methods, this approach does not need information about the change-points of the non-stationary arrival process. We compared DRL, and SBO approaches to get the staff rostering in the system with single task servers and a non-stationary arrival process. We performed a set of experiments with different inputs to investigate the performance of two solution approaches. We tested the DRL methodology on the use-case, including the point of sale data at the grocery store. The results show that this approach can perform appropriately on the various arrival processes we tested and the use case. We considered variance of waiting times to achieve predictability of waiting times. The evidence indicates that DRL performs better in both efficiency and performance. It produced better schedules in the cases that we tested. In addition, it resulted in schedules with lower variation in waiting times than the SBO method. The DRL's performance as a methodology is noticeable when the variation in the arrival rate is higher. The results of performing the DRL and SBO staff scheduling on two samples T-test on the average time in the system and the average number in the system indicates considering the average is not a good measure to capture differences between two approaches. As we discussed in chapter 3, there could be a high waiting time in one time block, but the overall average of waiting times over a day still satisfies the constraint.

Chapter 5

Modeling and Generating Independent Data for Non-Stationary Processes

5.1 Introduction

In this chapter, we focus on developing the models and generating independent random observations from non-stationary processes. As in real-world applications, there is no information about the distribution function of arrivals to each part of the service system; our interest is practical applications. We focus on the notion of having an input, and we try to match the output. Our objective is to generate data such that the arrival rate and dispersion ratio match those of the input data.

Since both the stationary Poisson process and non-stationary Poisson process have a dispersion ratio equal to 1, the dispersion ratio provides a measure of deviation from the Poisson process. We use dispersion ratio and arrival rate function as criteria to evaluate our data generation process as they are two keys to characterize the arrival process [57]. We define the non-stationary process model to generate data as following:

$M = [P, \Lambda, C]$, where:

$P = [P_1, P_2, ..., P_k]$ and P_k is the ending time for block k

$\Lambda = [\lambda_1(t), \lambda_2(t), ..., \lambda_k(t)]$ and $\lambda_k(t)$ is the arrival rate during block k

$D = [D_1, D_2, ..., D_k]$ and D_k is the dispersion ratio for block k

We are interested in two goals including:

1. Creating a set of representative models from a given set of arrival data

2. Generating observations from the models that match the dispersion ratio and arrival rate function.

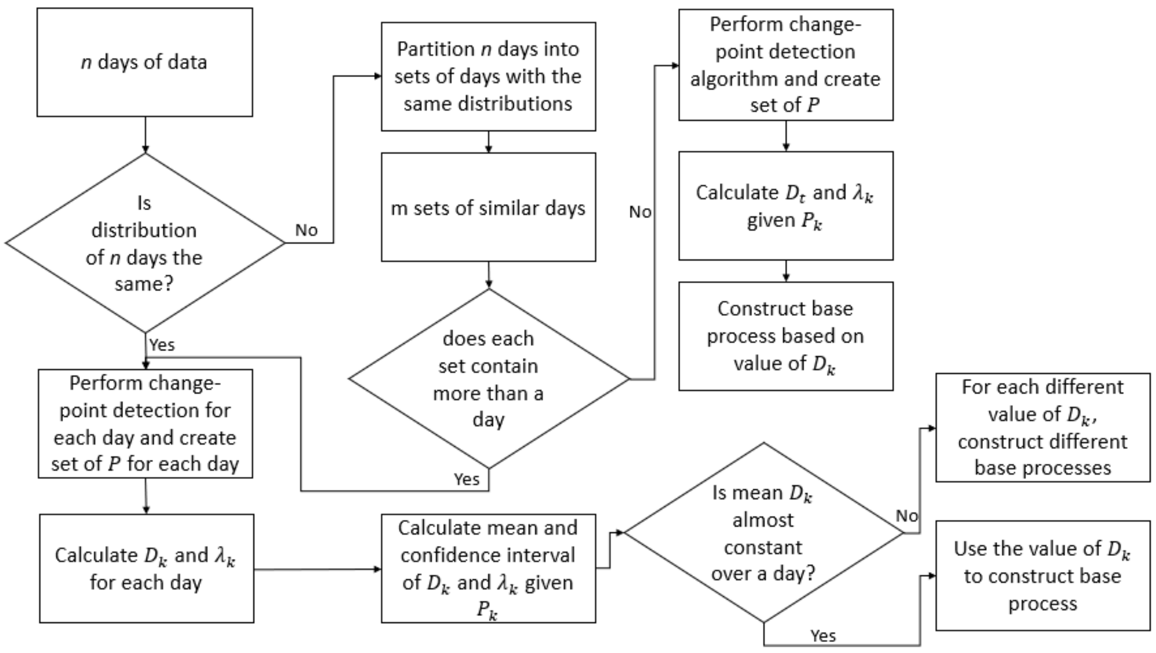
Both $\Lambda(t)$ and $D(t)$ are continuous functions of time. Given a period of a specific length (like a shift, day, or a week), first, we discretize the time horizon by change-point detection into *k* adjacent blocks. We call each of these time blocks a *phase*. As discussed in chapter 1, for time series with gradual changes in arrival rate over time, more analysis is needed, like plotting a histogram of the interarrival times in various time bucket lengths. Each phase, k, is characterized by an arrival rate, λ_k, and a dispersion ratio, D_k. In this chapter, we arbitrarily assume the period is one day.

In general, in terms of variance-to-mean (dispersion) ratio, processes are classified into three sets. If arrival process is Poisson (interarrival times are exponentially distributed), then $D_k = 1$. If the arrival process variability is less than the Poisson distribution (underdispersed), then $D_k < 1$. For example, hyper-Erlang distribution is in this category. Lastly, the variability of an arrival process with $D_k > 1$ is more than the Poisson distribution (overdispersed) [24]. The hyperexponential and negative binomial distributions are instances of this type of process.

The value of D_k is used to characterize the process, which will be transformed into a non-stationary process. This process is called the *renewal base process*, and it has a mean equal to one and a variance equal to D_k. We will describe this process in more detail in section 5.2.

The first step in building the model is estimating the values of D_k and λ_k in each phase based on the available data set. If there is no data available, we use our judgment and domain knowledge to specify those values according to similar processes. Here is the process that describes how to develop the model. Assuming n ($n \geq 1$) days of arrival times to a service system are available, we start with partitioning *n* days into *m* sets of days with observations following the same distribution. Next, for each partition, we perform change-point detection on each day by applying the method proposed in [2] to determine a set of change-points, P.

Because of randomness, each time bucket of similar days can contain a different number of arrivals. Therefore, we calculate D_k and λ_k for each day, and we build the confidence interval for them. Figure 5.1 illustrates the process of building a model according to the sample available data set.



n days of data
Is distribution of n days the same?
No
Yes
Partition n days into sets of days with the same distributions
m sets of similar days
does each set contain more than a day
No
Yes
Perform change-point detection algorithm and create set of P
Calculate D_t and λ_k given P_k
Construct base process based on value of D_k
Perform change-point detection for each day and create set of P for each day
Calculate D_k and λ_k for each day
Calculate mean and confidence interval of D_k and λ_k given P_k
Is mean D_k almost constant over a day?
No
Yes
For each different value of D_k, construct different base processes
Use the value of D_k to construct base process

Figure 5.1: The process of developing the model based on the available data set

In the present research, we apply different non-Poisson distributions (where $D_k \neq 1$) as the base process to provide a specific D_k values. One category of these distributions is phase-type distributions. It has been shown that phase-type distributions have a relatively short warm-up period to converge to the given dispersion ratio [23]. The first step in generating nonstationary arrival data is to build a stationary point process according to the specific renewal base process. Methods to generate stationary point processes are well-defined (for example, see [58] and [59]). After generating stationary sample observations from the base process, we transform this produced point process, based on the inverse of the cumulative arrival rate function, to build the arrival times of the nonstationary arrival process. We present an example to make the process described in figure 5.1 clear.

We generated non-stationary arrival times as an available data set in Simio simulation software. The data set arbitrarily includes ten days with similar distributions. Each day contains eight working hours. Figure 5.2 shows the arrival rate over a day.

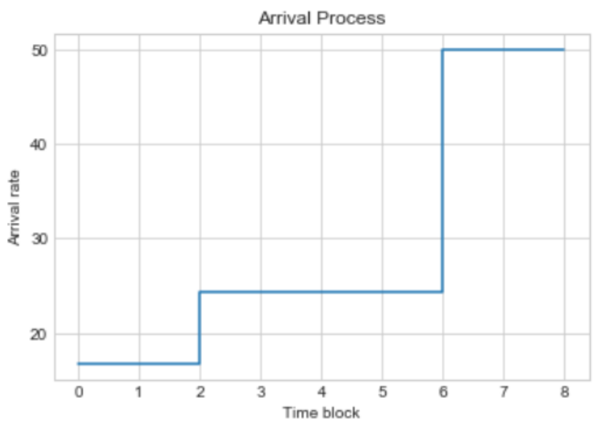


Figure 5.2: The average arrival rate during 8 hours of simulation

We generated data from a triangular distribution with parameters (min= 1.2, mode= 3.6, max= 6) minutes for the first two hours. The generated data for the next four hours are from a truncated normal distribution with parameters ($\mu = 1.98$, σ= 1.8) minutes. Since normal distribution includes negative value, and our goal is generating arrival times, we use truncated normal distribution instead of normal distribution to generate values greater than zero. Figure 5.3 shows truncated normal distribution we sampled from to generate random observations. The produced data for the last two hours are from an exponential distribution with an interarrival time equal to 1.2 minutes. We included exponential distribution to show the generality of the developed approach to generate nonstationary data when the dispersion ratio is equal to 1 and when it is not equal to 1.

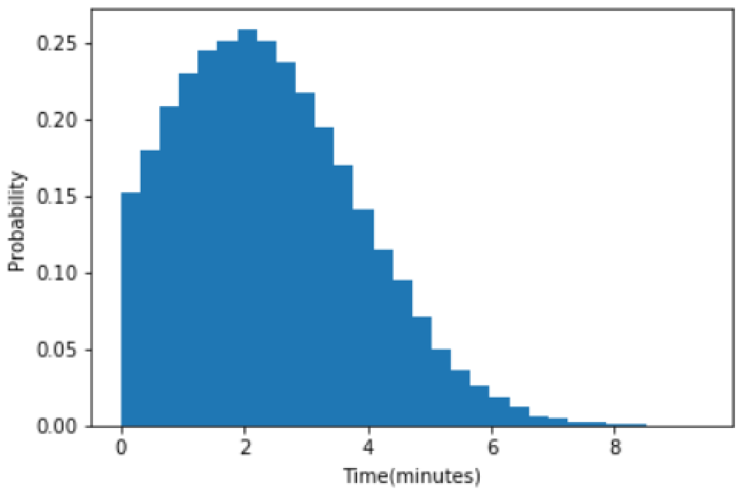


Figure 5.3: Truncated normal distribution to generate non-negative values

We follow the procedure described in figure 5.1 to determine change-points and then estimate the values of D_k and λ_k. Table 5.1 represents the values of D_k, λ_k, and P_k for each time block in each day from the 10-day sample of generated data.

Table 5.1: The values of D_k, λ_k, and P_k for each day

Day	λ_1	λ_2	λ_3	D_1	D_2	D_3	P_1	P_2	P_3
1	16.5	25.75	43.5	0.036	0.23	1.05	1.95	5.97	8.00
2	17	22.75	52.5	0.023	0.38	1.06	1.97	5.96	8.00
3	17	24	51.5	0.082	0.46	0.96	1.99	5.97	8.00
4	16	26.25	48	0.078	0.91	1.08	1.97	5.98	8.00
5	18	25.5	62.5	0.057	0.16	0.96	1.98	5.93	8.00
6	17.5	24.5	49	0.097	0.28	1.04	1.98	5.97	8.00
7	16.5	24.25	48.5	0.023	0.29	0.99	1.96	5.94	8.00
8	17	26.25	51	0.09	0.52	1.02	1.96	5.94	8.00
9	15.5	25.25	48	0.162	0.52	0.99	1.94	5.91	8.00
10	16	23.75	39.5	0.10	0.21	1.02	1.97	5.99	8.00

Since λ_k, D_k, and P_k are random variables, we need to use a function of these values over different days to characterize the model's parameters. As the average is the unbiased estimator of a population's mean, we choose to use the averages of these values to estimate the model's parameters. Here is the model we developed from the sample data:

$P = [1.97, 5.96, 8.00]$

$\Lambda = [16.70, 24.81, 49.40]$

$D = [0.076, 0.410, 1.006]$

After describing terms and methodologies to generate non-stationary arrival data in the following sections, we will use this model in section 5.7 to generate data with the same properties of data described in this section. We match the dispersion ratio for the generated data with the sample input data.

The organization of the rest of this chapter is as follows. First, we present the definition of the base process in section 5.2. Section 5.3 presents existing methodologies to generate data from the NSNP process and its drawbacks. In section 5.4 we represent numerical inversion and an example to clarify how we apply this approach. Then in sections 5.5 and 5.6 we talk about algorithms and different base processes used for non-stationary data generation. Section 5.7 presents an example to generate non-stationary data. Finally, in section 5.8

we represent examples and evaluate the performance of the proposed algorithm to generate non-stationary data.

5.2 Definition of base process

Renewal base process is a set of stationary sample observations from a desired distribution, where *renewal process* is an arrival process in which the inter-arrival intervals are positive, independent, and identically distributed random variables. We define a set of stationary non-negative interarrival times. For instance, to generate stationary interarrival times from the uniform distribution with parameters a and b, first we generate a random variable from uniform (0,1) distribution and we show it u_i. Then put it in $X_i = a+u_i \times (b-a)$, where X_i is the *ith* observation of the uniform distribution. Let $\{X_n, n \geq 1\}$ be a set of n samples and let S_n be the time of *nth* arrival; $S_0 = 0$, $S_k = \sum_{i=1}^{k} X_i$. For $n = 1, 2, ...$, let $N(t)$ be the number of arrivals that occurred on or before time *t*; $N(t) = max\{n \geq 0 : S_n \leq t\}$ [24]. *N(t)* is assumed to have a given asymptotic dispersion ratio:

$$D \equiv \lim_{t \to \infty} \frac{Var[N(t)]}{E[N(t)]} \tag{5.1}$$

Since both stationary Poisson processes and non-stationary Poisson processes have dispersion ratios equal to 1, the process with a dispersion ratio not equal to 1 is a non-Poisson process [23]. Since a base process is a stationary rate-1 (arrival rate equal to 1) process, the mean inter-arrival time is equal to 1, and the variance is equal to D.

5.3 Existing method, combined inversion-and-thinning approach (CIATA) to generate non-stationary non-Poisson process

Thinning and inversion methods to generate data of non-stationary non-Poisson processes were proposed in [24]. There are some drawbacks to applying these two methods to generate NSNP arrival data. Their proposed inversion method is only applicable for the cases where the arrival rate function is easily invertible. One disadvantage of their proposed thinning method is that it may be computationally inefficient if $\lambda(t) \ll \lambda^*$ over a substantial range of values for t, resulting in a relatively large number of rejections [23]. Another downside is that the dispersion ratio of the generated data does not necessarily converge to the target value of D. Liu et al. [23] extended the methods proposed by Gerhardt and Nelson [24] for modeling the NSNP process with a given arrival rate function ($\lambda(t)$) and an asymptotic dispersion ratio by dividing the time horizon into Q smaller time blocks. The researchers claim the proposed approach can handle broader forms of arrival rate functions $\lambda(t)$ than the inverse method, unlike the thinning approach, CIATA-Ph (CIATA with phase-type distribution as the renewal base process) can achieve any value of D. In this algorithm, they set two assumptions:

1. The given rate function $\lambda(t)$ for $t \geq 0$ has a finite upper bound λ^*. Moreover, each finite time interval [0, t] has a (finite) partition such that $\lambda(y)$, $y \in [0, t]$, is constant or quasiconcave on each subinterval of the partition.

2. The arrival rate function $\lambda(t)$ is continuous at every $t \geq 0$.

They showed if assumption 1 holds and Q is sufficiently large, then $E[N(t)] = \mu(t) \equiv \int_0^t \lambda(y)dy, t \geq 0$. By this, they showed that their NSNP process generated by CIATA achieves the mean-value function. As the first step, researchers constructed a piece-wise constant majorizing rate function, $\tilde{\lambda}_Q(t)$, which converges to $\lambda(t)$ when $Q \to \infty$. For more details about constructing majorizing rate function, see their algorithm 1. They called the corresponding majorizing mean-value function in each section, $\tilde{\mu}_Q$. Here is their proposed algorithm to generate NSNP process:

Algorithm 3 Algorithm to generate data proposed in [23]

```
Construct the majorizing rate function $\tilde{\lambda}_Q(t)$
According to the value of $D$, construct the base process and call it $G$
Set $n \leftarrow 1$, $l \leftarrow 0$, $S_0^o \leftarrow 0$, $\tilde{S}_0 \leftarrow 0$, and $S_0 \leftarrow 0$. Generate $X_n^o \sim G_e$ and set $S_n^o \leftarrow S_{n-1}^o + X_n^o$ and set $\tilde{S}_n \leftarrow \tilde{\mu}_Q^{-1}(S_n^o)$
While $\tilde{S}_n \leq S$ do;
    Generate $U_n \sim$ uniform$[0,1]$
    If $U_n \leq \lambda(\tilde{S}_n)/\tilde{\lambda}_Q(\tilde{S}_n)$, then
        Set $l \leftarrow l+1$ and $S_l \leftarrow \tilde{S}_n$
    end if
    Set $n \leftarrow n+1$. Generate $X_n^o \sim G$. Set $S_n^o \leftarrow S_{n-1}^o + X_n^o$ and $\tilde{S}_n \leftarrow \tilde{\mu}_Q^{-1}(S_n^o)$.
end while
```

In this algorithm, X_n^o is the nth inter-renewal time. They calculated the inverse function of the Cumulative Distribution Function (CDF) of arrival rates for each partition. To include each point, they applied thinning, and if it was accepted, they used the inverse of the CDF to generate a new sample observation.

5.4 Numerical inversion

If the CDF of the arrival rate function, $\Lambda(t)$, is not invertible, we can also use the numerical inversion method instead of the methodology described in section 5.3. The output of the inversion method is a table. Later in this section, we will present an example to make this approach more clear. One of the advantages of numerical inversion is that we can replicate the model several times by looking up the previously created table. We build this table and reuse it several times according to the arrival rate function outside the simulation.

Ma and Whitt [22] proposed an algorithm to get the numerical inverse of the Cumulative Distribution Function (CDF), which we discussed in detail formerly in this section. The goal of the numerical approximation of the inverse function is to construct an approximation efficiently, that we show it with J of the function Λ^{-1} mapping interval $[0, \Lambda(t)]$ to $[0, T]$ with specific accuracy.

$$||J - \Lambda^{-1}|| \equiv \underset{0 \leq t \leq \Lambda(t)}{sup} \{|J(t) - \Lambda^{-1}(t)|\} \leq \epsilon \tag{5.2}$$

Their general strategy is to partition two intervals $[0, T]$ and $[0, \Lambda(t)]$ into n_x and n_y evenly sub-intervals of length η and δ, respectively. Since numerical inversion maps each point on y axis to a proper point on x axis, they define $J(i\delta)$ to be an appropriate $j\eta$ on x axis, for each i, $0 \leq i \leq n_y$. The main parameters in their algorithm are:

$$\rho \equiv \frac{\lambda_u}{\lambda_l}, \quad \eta = \frac{\epsilon}{1+\rho}, \quad \delta = \lambda_u \eta \tag{5.3}$$

Where λ_u and λ_l are the upper and lower bound of arrival rate function, $\lambda(t)$. Therefore, ρ is the slope ratio, $1 \leq \rho \leq \infty$. They calculate $\Lambda(t)$ or each point in $[0, T]$ as follows:

$$a(j) \equiv \Lambda(k\eta), \quad 0 \leq j \leq n_x \tag{5.4}$$

To approximate the value of $\lambda^{-1}(y)$ in each point in $[0, \Lambda(T)]$ within the point in $[0, T]$, they use the following equation:

$$b(i) \equiv inf\{j \geq 0 : a(j) \geq i\delta\} \tag{5.5}$$

Then $J(i\delta) = b(i)\eta$ for all i, $0 \leq i \leq n_y$. Figure 5.4 illustrates the numerical inversion that maps each point on the y axis to the appropriate one on the x-axis.

To make the numerical inversion more clear, we create an example and present it here. We use the arrival rate function $\lambda(t) = 4t$ and its corresponding CDF is, $\Lambda(t) = 2t^2$. After applying equation 5.3, the values of ρ, η, and δ are 1.5, 0.0004, and 0.024, respectively. Next, we use equations 5.4 and 5.5 to create a lookup table to generate non-stationary arrival data of a process with $D = 0.3$ and uniform distribution as the base process. The parameters of this uniform distribution are (0.05,1.95) according to section 5.6.2 to have a mean equal to one and variance equal to 0.3. Then we generate a stationary random sample from the characterized uniform distribution. To find the index that we need to map into vector b, we use this equation: $k = \frac{\lfloor times[i] \rfloor}{\delta}$, where times$[i]$ is the ith element of the generated stationary random sample set. We show a part of the look-up table and the indexes to look up in table 5.2.

Table 5.2: The table look up resulted from inversion method

index	b	k	$b[k]$	time	$\Lambda(t)$
0	0	0	0	0.00	0.00
1	64	680	1650	0.83	1.36
2	90	1205	2196	1.09	2.41
3	110	1479	2433	1.22	2.96
4	127	2449	3130	1.57	4.89
5	142	3022	3477	1.74	6.04
6	155	3525	3755	1.88	7.05
7	168	4057	4029	2.01	8.11
8	179	4848	4404	2.20	9.69
9	190	5386	4642	2.32	10.77

The figure 5.4 illustrates the points that transformed to build the arrival times. $\Lambda(t) = 2t^2$ is the CDF of the arrival rate function ($\lambda(t)$). The point on the y axis are arrival times in the rate-1 base process, and the points on the x-axis are the arrival times in the nonstationary arrival process.

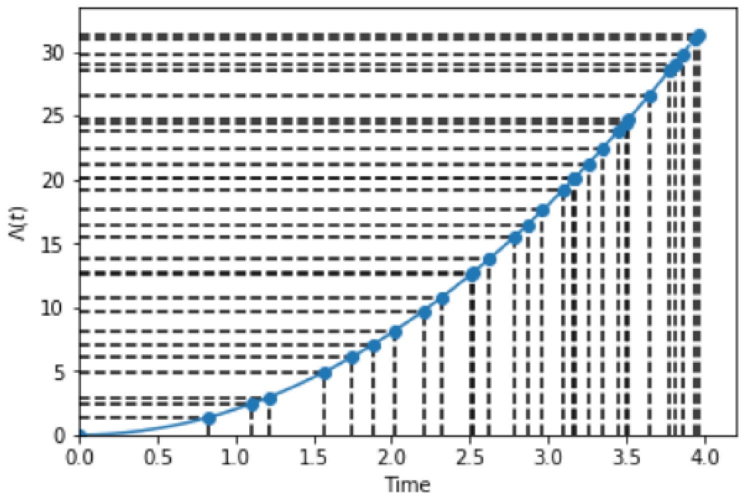


Figure 5.4: Illustration of the numerical inversion method described in the example

Table 5.2 shows only ten rows of the table resulting from numerical inversion. For example, the value of k for time[1] is 680, and the corresponding value in vector b is 1650. We use this equation to obtain the corresponding arrival time: arrival time $=k \times \eta$. For this example, the second arrival time is at time 0.83.

5.5 Overview of the algorithm used to generate data, in order to simulate the non-stationary independent arrival process

If the mathematical inverse of the cumulative rate function is available, which is called the invertible function, generating non-stationary arrival data can be done based on the algorithm proposed in [24]. In this research, we focus on the case in which the inversion of the cumulative arrival rate function is not available (non-invertible function). Table 5.3 summarizes the choices of the base processes according to the value of D_t. We emphasize that we use these distribution functions to generate the stationary data transformed into non-stationary sample data. The resulting non-stationary sample data does not necessarily follow the same distribution as the initial stationary data. The characteristic that matches is the dispersion ratio. Later in section 5.7, we will provide more details about the reason we are choosing these distributions.

Table 5.3: Base process's distribution and its parameters

Value of D_t	Base process, $G(x)$
$D_m \simeq 0$	normal
$D_m < 1$	Hyper-Erlang
$D_m = 1$	Exponential
$D_m > 1$	Hyperexponential

When the value of D_t is close to zero, the hyper-Erlang distribution is not efficient since the value of its k parameter gets large [23]. Our example in section 5.7 indicates that in this case, the estimation of target D_t is not good, and it requires using another distribution to generate non-stationary sample observations. Table 5.4 illustrates the value of k of Erlang distribution regarding values of $D_t \leq 0.2$, which is growing fast.

Table 5.4: The values of hyper-Erlang parameter k for different values of $D_t \leq 0.2$

D_t	k
0.2	5
0.1	10
0.08	13
0.06	17
0.04	25
0.02	50
0.01	100
0.005	200

More comprehensive experiments are required to determine the exact threshold for the value of D_t, that is necessary to substitute hyper-Erlang distribution with the normal distribution. Performing such experiments is a direction for future work.

Algorithm 4 Overview of data generation algorithm for non-stationary renewal processes

```
A model, $M = [P, \Lambda, D]$, with $m$ phases is given,
Create an empty list, all-observation=[]
for $i \leftarrow 1$ to $m$ do
    if CDF of $\lambda_i$ is not mathematically invertible
        1. Build the table lookup for the inverse of the arrival rate function's CDF
    else
        1. Invert the CDF of $\lambda_i$ mathematically
    2. Choose the base process, $G(x)$ according to the table 5.3
    3. Generate random observations of base process $G(x)$
    4. Create an empty list, observations=[]
    5. Transform resulting data from step 3 using either inverted function or from table from
    step 1
    6. Append all generated data in step 5 into observation list
    7. Add the observation list into all-observations list
end
```

We will describe the detail of how to construct each base process in the following sections.

5.6 Renewal base processes

In this case, we assume the interarrival times are independent of each other. The distributions we investigate include the phase-type distribution, uniform distribution, and normal distribution with various values of the dispersion ratios.

5.6.1 Phase-type distributions

Gerhardt and Nelson [24] suggested using phase-type distribution as the base process to model the NSNP processes. We construct a process with a given arrival rate function and an asymptotic dispersion ratio. For the case $D \geq 1$, we use hyperexponential distribution and hyper-Erlang distribution, when $0 < D < 1$.

Case $D > 1$:

The density function of 2-phase hyperexponential(H_2) distribution is as follows:

$$f_{H_2}(x; p, \lambda_1, \lambda_2) = p\lambda_1 e^{-\lambda_1 x} + (1-p)\lambda_2 e^{-\lambda_2 x} \quad p \in (0, 1) \tag{5.6}$$

The cumulative distribution function is:

$$F_{H_2}(x; p, \lambda_1, \lambda_2) = 1 - pe^{-\lambda_1 x} - (1-p)e^{-\lambda_2 x} \tag{5.7}$$

The base process has a mean interarrival time equal to 1, and variance equal to an asymptotic dispersion ratio, D. Liu et al. [23], characterized this process, the following equation is based on their reported results:

$$p = \frac{1 + D \pm \sqrt{D^2 - 1}}{2(D+1)} \tag{5.8}$$

Additionally, two-phased balanced-means hyperexponential distribution has $\lambda_1 = 2p$ and $\lambda_2 = 2(1-p)$. In order to generate arrival data of the base process using this setup, with a probability of p we sample from an exponential distribution with the parameter $2p$, we call it $G_1(x)$, and with the probability of $(1-p)$, we sample from an exponential distribution with the parameter $2(1-p)$, we call it $G_2(x)$.

Case $D < 1$: In this case, we sample from the hyper-Erlang distribution with cumulative distribution function as following:

$$F_{Er}(x; k, \lambda) = \int_0^t \frac{t^{k-1} e^{(-t/\lambda)}}{(k-1)!\lambda^k} dt \tag{5.9}$$

Liu et al. [23] characterized the parameters when the mean interarrival time and variance are equal to 1 and D, respectively, as below:

$$k = \lceil 1/D \rceil, \quad p = \frac{kD - \sqrt{k(1+D) - k^2 D}}{1+D}, \quad \lambda = 1/(k-p) \tag{5.10}$$

We generate a new sample from the Erlang distribution with $F_{Er}(x; k-1, \lambda)$ cumulative distribution function with probability p, and we call it $G_1(x)$, and with probability $(1-p)$ we sample from Erlang distribution with $F_{Er}(x; k, \lambda)$ cumulative distribution function, and we call it $G_2(x)$.

5.6.2 Considering uniform distribution as base process

We set the rate-1 base process as the continuous uniform distribution ($u \sim$ uniform$[a, b]$) with the mean and variance equal to 1 and D_t, respectively. Here we characterize the parameters of this distribution. We let $b = 1 \pm \sqrt{3D_k}$ and $a = 1 \mp \sqrt{3D_k}$ uniform distribution has a dispersion ratio less than exponential distribution. We can only model NSNP processes with a small dispersion ratio ($D_k \leq 0.3$ since $t \geq 0$, for the bigger values of D_t, a and/or b will be negative). Figure 5.5 illustrates the dispersion ratio estimated for 200 replications of the algorithm when the uniform distribution is the rate-1 base process. The result shows uniform distribution could be a good candidate as a base process, but it has a limitation for the values of dispersion ratio it covers. There is the same limitation in considering triangular distribution as the base process. Because of this, we will not use this distribution further.

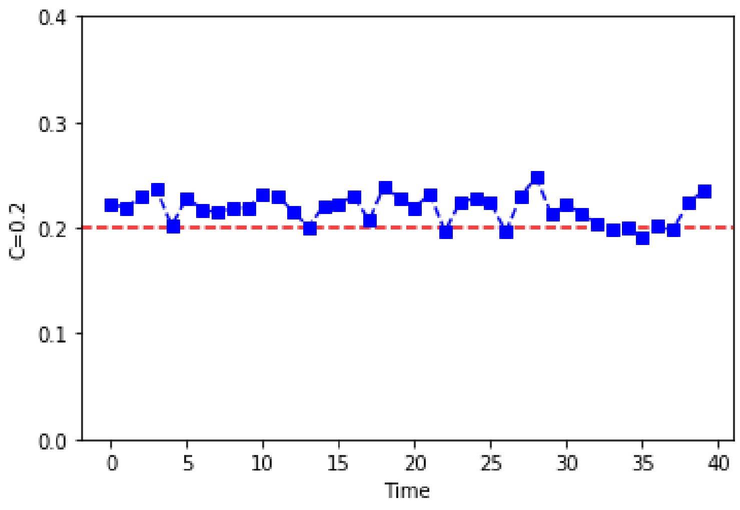


Figure 5.5: Mean dispersion ratio estimation for $D = 0.2$

5.6.3 Truncated normal distribution as the base process

In this part, we consider truncated normal distribution as the base process and investigate the estimation of the dispersion ratio from the generated arrival data. In this case, the mean and variance of the truncated normal distribution are set to 1 and D_t, respectively. The normal distribution has an easy setup in Python programming software as the rate-1 base process. We will show the results of applying this distribution in section 5.8. Algorithm 3 illustrates the procedure to generate random independent observations from different base processes.

Algorithm 5 Generating sample observation from base process

```
1. According to table 5.3, set the base process as the input function, $G(x)$
2. Set $n \leftarrow 1$
3. Generate $u \sim \text{uniform}(0,1)$
if $u \leq p$ then
  | Generate a random observation from $G_1(x)$ and put it in $S_n$
else
  | Generate a random observation from $G_2(x)$ and put it in $S_n$
end
4. Construct $T_k$, the $k$th arrival time in process $N(t)$ by letting $T_k = \sum_{i=1}^{k} S_n$ and set $n \leftarrow n+1$
```

5.7 Example to describe the methodology to generate non-stationary renewal random observations

In this section, we resume the example we explained in section 5.1 in order to generate random independent observations with approximately the same dispersion ratio of described arrival times data. Recall the developed model, M is:

$P = [1.97, 5.96, 8.00]$

$\Lambda = [16.70, 24.43, 49.40]$

$D = [0.076, 0.410, 1.006]$

Now, based on values of D_t, we construct the base processes. Since the base process is a stationary rate-1 process, its mean is equal to 1, and according to the equation 5.1, the variance is approximately equal to the value of D_t. As these D_t values are changing over a day, we need three base processes. Table 5.5 presents the base processes that we use with their corresponding parameters.

Table 5.5: Base process's distribution and its parameters

Time block	Base process's distribution	Mean	Variance
1	Hyper-Erlang	1	0.076
2	Hyper-Erlang	1	0.41
3	exponential	1	1.00

The next step is to generate observations from each of these base processes and then transform them according to the inverse of the cumulative rate function. Figure 5.6 represents the mean and the confidence interval of estimation of target dispersion ratios in smaller 0.4 hours time blocks.

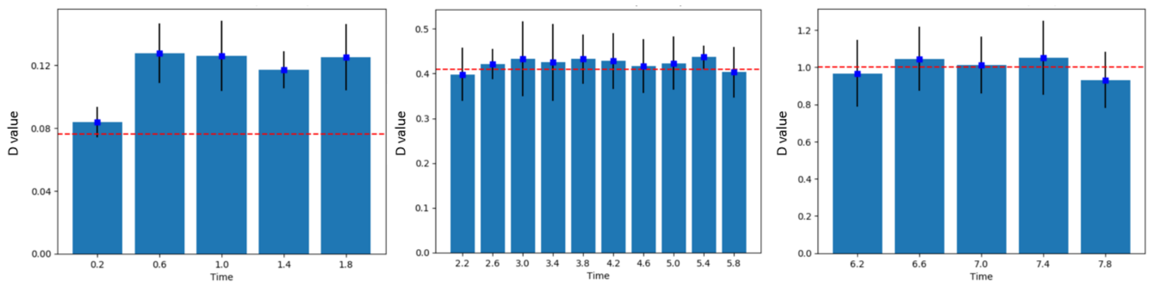


Figure 5.6: The dispersion ratio estimation and its confidence interval when $D_1 = 0.076$, $D_2 = 0.41$, and $D_3 = 1$, red dashed line shows the target D value.

The value of the dispersion ratio is very close to zero in the first time block. As described in section 5.5, the hyper-Erlang distribution has a poorer performance in generating data with close to zero values of the dispersion ratio [23]. Cases with $D_2 = 0.41$ and $D_3 = 1$ have estimated values very close to the target dispersion ratio.

To investigate the performance of other distributions for the first time bucket, we consider normal and uniform distributions as the base process. The parameters of uniform distribution are (0.52, 1.48) to achieve the target D_t value, 0.076 (we described how to calculate uniform distribution's parameters in section 5.6.2). In terms of normal distribution, the mean is one, and the variance is 0.076.

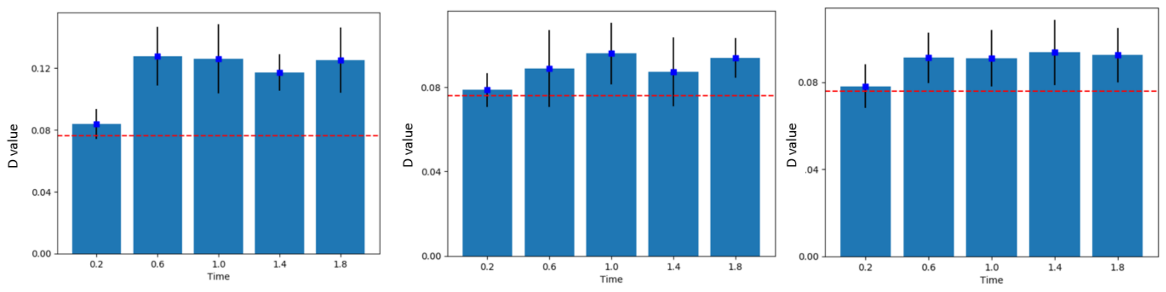


Figure 5.7: The dispersion ratio estimation comparison in case of different base processes, left presents hyper-Erlang distribution, the middle one is normal distribution, and the one in the right shows Uniform distribution

As it is shown in figure 5.7, the normal and uniform distributions give a better estimation of D_t since the estimations are closer to the target D_t value. Normal distribution performed better than uniform distribution in the fourth time block. Therefore, for the values of D_t close to zero, we suggest using the normal distribution as it has an easy setup to generate stationary arrival data and performs better than uniform and hyper-Erlang distributions. We summarized our proposed base processes for different values of D_t in section 5.5.

5.8 Performance evaluation of numerical inversion approach

In some situations, the arrival data is not available, but the model is given. We considered a case that the input data was available and the arrival rate was invertible in section 5.7. In this section, we focus on given model with not mathematically invertible cumulative arrival rate function to evaluate performance of numerical inversion. In the given model, we assume the dispersion ratio is constant during the time horizon, and the arrival rate function is sinusoidal. We use the same arrival rate function as in work by Liu et al. [23], to be able to compare the results. The sinusoidal arrival rate function is as follows:

$$\lambda(t) = \bar{\lambda}[1 + \gamma sin(t)], \quad t \in [0, 8] \tag{5.11}$$

where $\bar{\lambda}$ = 50. We will consider two values for gamma, $\gamma = 0.2, 0.5$. The cumulative function of this arrival rate function is not mathematically invertible; therefore, we apply numerical inversion discussed in section 5.4. We replicate the 200 replications of algorithm 4 for generating non-stationary independent arrival data 30 times and build the confidence interval for the dispersion ratio for each time block with a length of 0.2 hours (40 time blocks in 8 hours of the experiment). Figure 5.8 represents arrival rate over 8 hours for two cases, $\gamma = 0.2$ and $\gamma = 0.5$. In the first case, the maximum arrival rate is 60, and the lowest is 40, while in the second case maximum is about 80 and the minimum is almost 20.

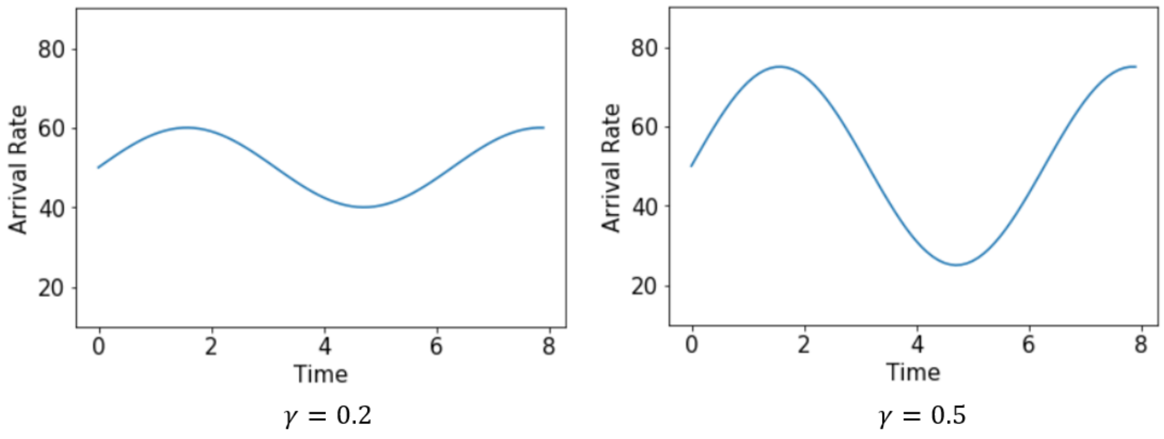


Figure 5.8: Arrival rate function, with $\gamma = 0.2$ and $\gamma = 0.5$

We apply phase-type distribution and normal distribution for the renewal rate-1 base process. In phase-type distribution, if $D \geq 1$, we apply balanced hyperexponential distribution as the base process, and when $D < 1$, we use hyper-Erlang distribution. Figures 5.9 and 5.10 show the confidence interval for each time block's asymptotic dispersion ratio using balanced hyperexponential distribution and hyper-Erlang distribution as the based process, respectively. At the beginning of each realization, three is warm-up time.

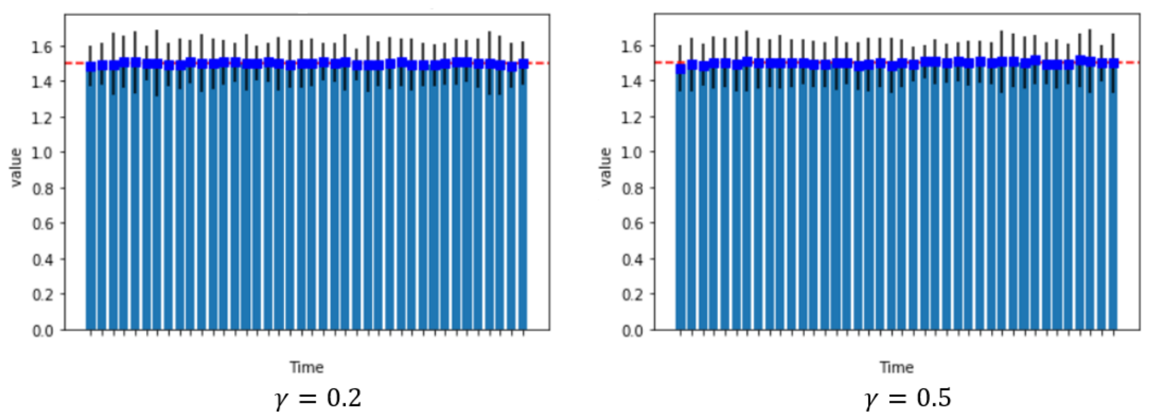


Figure 5.9: Confidence interval estimation for $D = 1.5$, hyper-exponential distribution as the base process

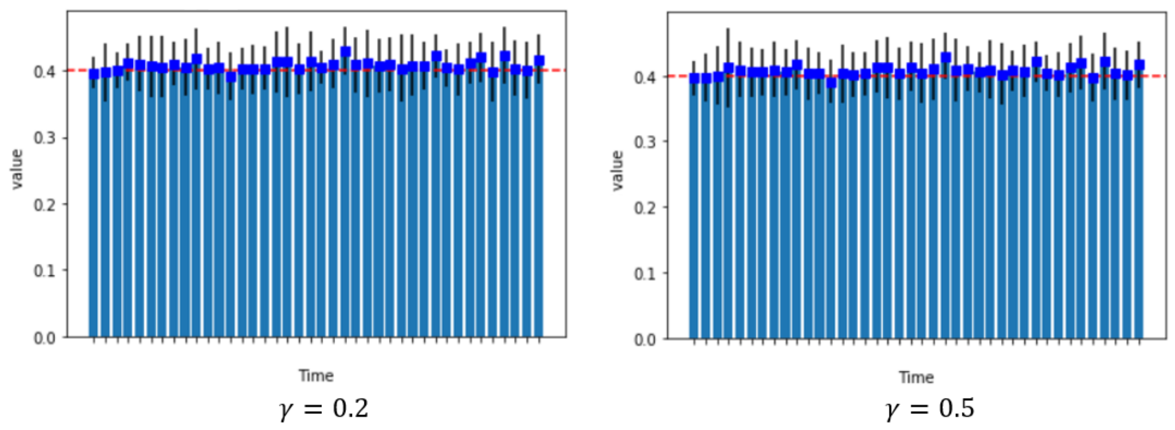


Figure 5.10: The confidence interval estimation for $D = 0.4$, hyper-Erlang distribution as base process

Figure 5.11 and 5.12 show the confidence interval estimation for the case normal distribution is the base process to generate non-stationary arrival data for both cases, $D > 1$ and $D < 1$. The results show that normal distribution can have comparable performance as the phase-type distributions. In the related existing works, there is no theoretical proof of the performance of this distribution. However, in practice, its performance could be very good.

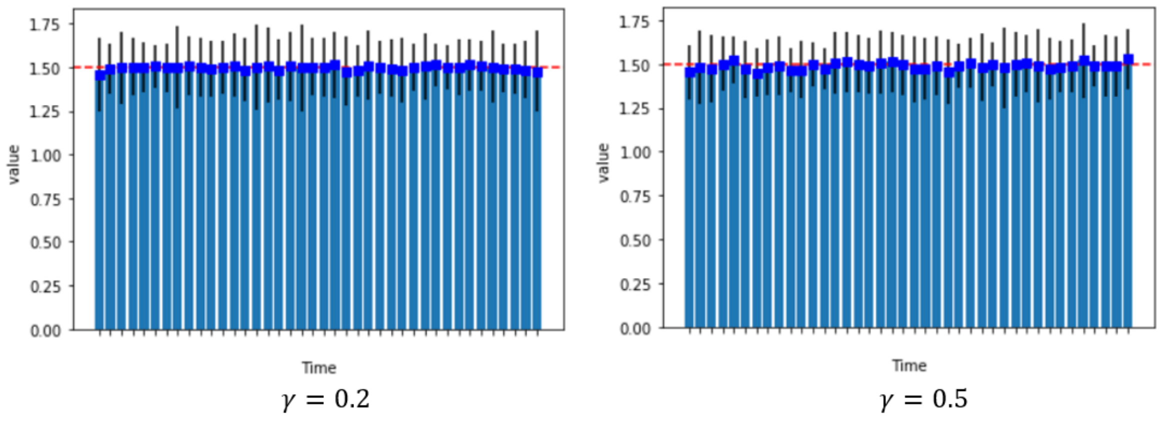


Figure 5.11: Confidence interval estimation for $D = 1.5$, normal distribution as base process

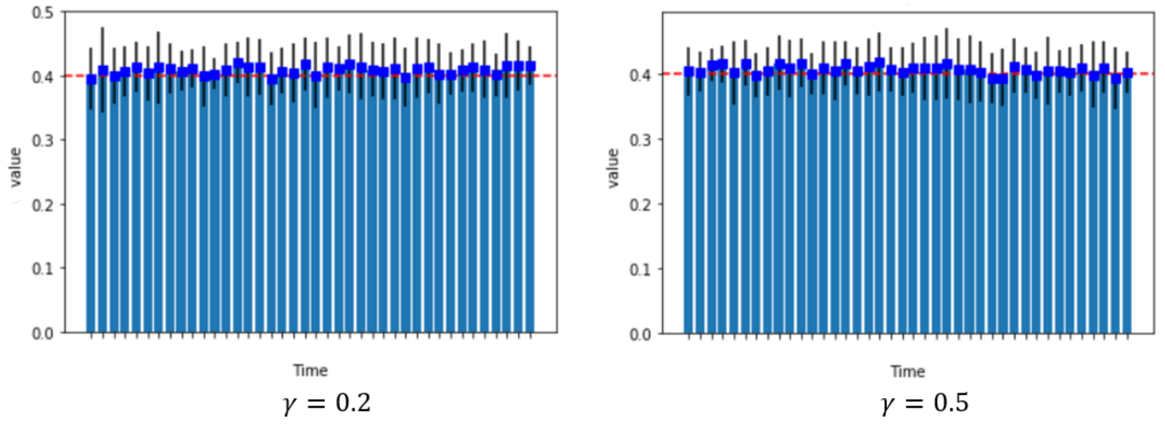


Figure 5.12: Confidence interval estimation for $D = 0.4$, normal distribution as base process

5.8.1 Comparison between the proposed methodology in this chapter and CIATA

This section compares the proposed algorithm to generate non-stationary processes and CIATA proposed in [23]. As we mentioned earlier, the general idea of the CIATA approach is that the researcher divided the arrival rate function into Q smaller partitions. Then they applied the thinning and inversion methods proposed in [24] to generate NSNP random observation in each partition. The reasons that they did this include:

1. By dividing the arrival rate function into Q partitions, they can cover a broader range of arrival rate functions that are not invertible. This action overcomes the drawback of the inversion method in [24] which was applicable only on invertible arrival rate functions.

2. By applying thinning in each partition, they improved the inefficiency of the proposed thinning approach to generate NSNP random observations in [24] for the case which $\lambda(t) \ll \lambda^*$ over a substantial range of values for t.

The proposed algorithm by Liu et al. [23] required applying thinning in each partition in addition to inversion because they used the majorizing rate function in each partition instead of using the actual arrival rate function of that partition.

According to the preceding discussion, there are some inefficiencies in the proposed methodology in [23] as are listed below:

1. While applying CIATA, one needs to specify the optimal number of Q in order to partition the time horizon into sub-intervals in which the arrival rate is constant or quasi-concave. In practice, this approach is not efficient as with any changes in the arrival rate function, determining again the optimal value of Q is required since the performance of the data generation algorithm depends highly on the value of Q. In addition, this value changes by changing the simulation run length.

2. In CIATA, there is a need to construct mean-value function, $\tilde{\mu}_Q(t)$ (CDF of $\tilde{\lambda}_Q(t)$) for each sub-interval.

3. Since in the CIATA approach, the authors did not perform input analysis, including the change-point detection, they did not cover the cases when the dispersion ratio value changes over time. Therefore, they suggested that one direction for future work in CIATA is to adapt their algorithm to generate an NSNP process whose dispersion ratio is piece-wise constant over the planning horizon.

4. When the value of Q is not big enough, CIATA generates data that either under-estimate or over-estimate the target value of D. Figure 5.13 illustrates this situation which is extracted from their supplementary material [23]. This indicates that CIATA is computationally expensive.

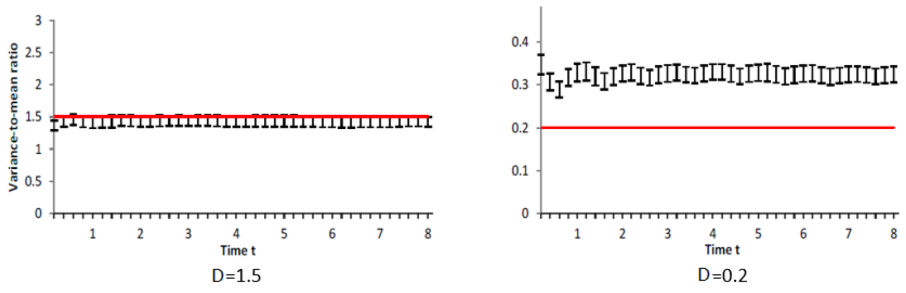


Figure 5.13: Estimation of dispersion ratio when $Q = 40$ [23]

The proposed algorithm in this dissertation has properties that make it more efficient as follows:

1. It can be applied to generate data with any type of arrival rate function.
2. This algorithm applies a lookup table based on the arrival rate function, and this table can be reused many times to generate non-stationary data with the same arrival rate function but different dispersion ratios.
3. Unlike CIATA, it does not require dividing time horizon to smaller partitions (Q) and creating majorizing rate functions and mean-value functions for each section.
4. CIATA assumes information like dispersion ratio value and arrival rates are given. This dissertation develops a model to specify this information as the input into the CIATA approach.

Figures 5.14 and 5.15 illustrates the comparison between the performance of CIATA and our proposed method to generate random observations from the non-stationary arrival process. We considered two sinusoidal arrival processes as the equation 5.11 with $\gamma = 0.2$ and $\gamma = 0.8$ for two values of dispersion ratios, 0.2 and 1.5. The results indicate that our proposed method is competitive in achieving the target dispersion ratio for the generated random observations, with less expensive computational requirements.

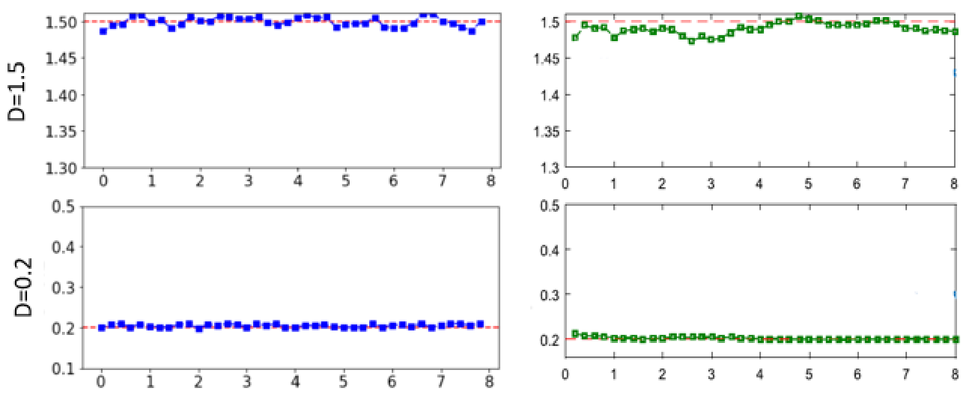


Figure 5.14: Comparison of our proposed method, left panel, and CIATA, right panel, in estimating the target value of D with sinusoidal pattern in arrival process and $\gamma = 0.2$

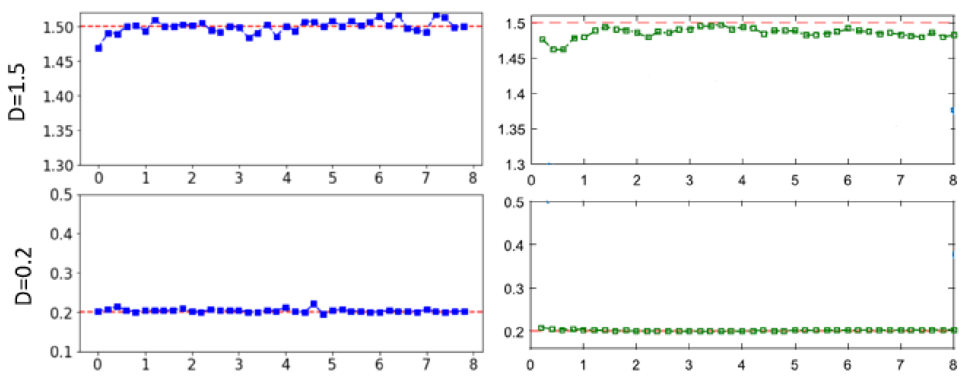


Figure 5.15: Comparison of our proposed method, left panel, and CIATA, right panel, in estimating the target value of D with sinusoidal pattern in arrival process and $\gamma = 0.8$

5.9 Conclusion

In this research, we developed models and presented the algorithms to generate data of the non-stationary process. The model we developed is general since it can model both NSNP and NSP processes. The main focus was on the numerical inversion technique as it does not require the arrival rate function to be linear or piece-wise constant linear. We investigated the ability of the numerical inversion methods to generate non-stationary arrival data with the estimation of the dispersion ratio of generated arrival data. To do this, we considered renewal processes as the rate-1 stationary process. This study represented the method to generate independent arrival data where the arrival rate function is time-varying and not necessarily invertible. The confidence interval estimation for the dispersion ratio shows that the numerical inversion method can yield good performance for renewal processes. We compared the performance of our proposed method with the CIATA [23]. The results showed that our proposed algorithm performs as well as CIATA with fewer computational requirements to perform the algorithm. However, one of the limitations of this method and other existing ones to generate the nonstationary arrival data is that the distribution of the generated data is not necessarily the same as the distribution of the input data since we are targeting the dispersion ratios generated from a finite sample form the population. In real-world service systems, since there is limited sample input data to estimate the dispersion ratio, therefore, there will generally be a sampling error in the estimate. Our objective is to match this input data, regardless of the possible sampling error.

Chapter 6

Concluding Remarks and Future Work

This dissertation studied methods of staff scheduling in order to minimize the operational cost of the system while achieving predictable customers waiting times. We developed two models to find optimal staffing levels. In the first optimization model, we considered the average and the percentile of waiting times in each time block over a day in order to characterize the service level. The objective function in this problem is minimizing the staffing cost. We proposed applying a change-point detection algorithm to find change points in the arrival process. Then we performed simulation-based optimization using a package in R software to find the optimal staffing levels. Lastly, we determined the staff schedule, assuming the workers have constant working hours and the shift start times are constant.

In the second model, we proposed staff scheduling using Deep Reinforcement Learning. This model's objective function included two parts. The first is the cost of using resources, and the second is the penalty cost for the number of customers that waited in the queue more than the target waiting time. To achieve predictable waiting times, we proposed to use the upper bound for variation of waiting times over the entire day as the constraint. We assumed there are lower and upper bound limits for staff members' working hours in this problem. Our results highlighted that we could improve both the performance and efficiency of the SBO method by using the DRL method. We applied our proposed model and methodology to the use-case of grocery store's arrival times data. Our analysis showed that this method works appropriately for staff scheduling in service systems.

We developed a model to generate data of the non-stationary processes. We used several distributions as the base process, and we showed the performance of each distribution for data generation purposes through examples. We investigated both cases where the dispersion ratio is less than 1 and greater than 1. These algorithms are beneficial to improve the performance of research projects in the non-stationary processes area. One of the biggest challenges in modeling these types of systems is the lack of observational data. Being able to replicate the results is a step forward in reducing the variation among output responses.

This dissertation aimed to provide a comprehensive study on the non-stationary service system. We believe our proposed methods and algorithms can significantly improve the performance of service systems. In future studies, there are some promising extensions of our work, including:

1. Staff scheduling in a network of servers: Since real-world service systems like airports have more than one station, we can extend our proposed flexible staff scheduling using DRL into a network of servers.

2. Neural network structure and training parameters: We applied the parameters used in the stoplight example of AnyLogic. We believe conducting some parameter tuning for training variables can improve the quality of the results. Furthermore, running more experiments with various numbers of NN's hidden layers can possibly affect the quality of the trained policy.

3. To develop the model described in chapter 5 in order to cover the non-renewal non-stationary processes. This model will help extend the non-stationary data generation algorithm to the cases that the data are dependent on (non-renewal processes).

We believe models and algorithms presented in this dissertation and future work extensions will improve service systems' efficiency and customer satisfaction.

Bibliography

[1] Samira Shirzaei and Jeffrey S Smith. “Resource scheudling in non-stationary service systems”. In: *2018 Winter Simulation Conference (WSC)*. IEEE. 2018, pp. 537–547.

[2] David S Matteson and Nicholas A James. “A nonparametric approach for multiple change point analysis of multivariate data”. In: *Journal of the American Statistical Association* 109.505 (2014), pp. 334–345.

[3] Song Liu et al. “Change-point detection in time-series data by relative density-ratio estimation”. In: *Neural Networks* 43 (2013), pp. 72–83.

[4] Nicholas A James and David S Matteson. “ecp: An R package for nonparametric multiple change point analysis of multivariate data”. In: *arXiv preprint arXiv:1309.3295* (2013).

[5] Mohammadnaser Ansari et al. “HistoRIA: A new tool for simulation input analysis”. In: *Proceedings of the Winter Simulation Conference 2014*. IEEE. 2014, pp. 2702–2713.

[6] Mieke Defraeye and Inneke Van Nieuwenhuyse. “Staffing and scheduling under nonstationary demand for service: A literature review”. In: *Omega* 58 (2016), pp. 4–25.

[7] Navid Izady and Dave Worthington. “Setting staffing requirements for time dependent queueing networks: The case of accident and emergency departments”. In: *European Journal of Operational Research* 219.3 (2012), pp. 531–540.

[8] Mehmet Tolga Cezik and Pierre L’Ecuyer. “Staffing multiskill call centers via linear programming and simulation”. In: *Management Science* 54.2 (2008), pp. 310–323.

[9] Wyean Chan et al. “Two-stage chance-constrained staffing with agent recourse for multi-skill call centers”. In: *Proceedings of the 2016 Winter Simulation Conference*. IEEE Press. 2016, pp. 3189–3200.

[10] Thuy Ta, Pierre L’Ecuyer, and Fabian Bastin. “Staffing optimization with chance constraints for emergency call centers”. In: *MOSIM 2016-11th International Conference on Modeling, Optimization and Simulation*. 2016.

[11] Ward Whitt. “What you should know about queueing models to set staffing requirements in service systems”. In: *Naval Research Logistics (NRL)* 54.5 (2007), pp. 476–484.

[12] Otis B Jennings et al. “Server staffing to meet time-varying demand”. In: *Management Science* 42.10 (1996), pp. 1383–1394.

[13] Beixiang He, Yunan Liu, and Ward Whitt. “Staffing a service system with non-Poisson non-stationary arrivals”. In: *Probability in the Engineering and Informational Sciences* 30.4 (2016), pp. 593–621.

[14] Athanassios N Avramidis, Alexandre Deslauriers, and Pierre L’Ecuyer. “Modeling daily arrivals to a telephone call center”. In: *Management Science* 50.7 (2004), pp. 896–908.

[15] PA W Lewis and Gerald S Shedler. “Simulation of nonhomogeneous Poisson processes by thinning”. In: *Naval research logistics quarterly* 26.3 (1979), pp. 403–413.

[16] J Smith, D Sturrock, and D Kelton. “Simio and simulation: modeling, analysis, applications”. In: *Sewickley, PA: Simio LLC* (2017).

[17] Sachin Sumant. “An automated procedure for simulating complex arrival processes: A web-based approach”. In: (2003).

[18] Linda Green, Peter Kolesar, and Anthony Svoronos. “Some effects of nonstationarity on multiserver Markovian queueing systems”. In: *Operations Research* 39.3 (1991), pp. 502–511.

[19] Linda V Green and Peter J Kolesar. “The lagged PSA for estimating peak congestion in multiserver Markovian queues with periodic arrival rates”. In: *Management Science* 43.1 (1997), pp. 80–87.

[20] Michael E Kuhl and James R Wilson. “Modeling and simulating Poisson processes having trends or nontrigonometric cyclic effects”. In: *European Journal of Operational Research* 133.3 (2001), pp. 566–582.

[21] Michael E Kuhl, Sachin G Sumant, and James R Wilson. “An automated multiresolution procedure for modeling complex arrival processes”. In: *INFORMS Journal on Computing* 18.1 (2006), pp. 3–18.

[22] Ni Ma and Ward Whitt. “Efficient simulation of non-Poisson non-stationary point processes to study queueing approximations”. In: *Statistics & Probability Letters* 109 (2016), pp. 202–207.

[23] Ran Liu et al. “Modeling and simulation of nonstationary non-Poisson arrival processes”. In: *INFORMS Journal on Computing* 31.2 (2019), pp. 347–366.

[24] Ira Gerhardt and Barry L Nelson. “Transforming renewal processes for simulation of nonstationary arrival processes”. In: *INFORMS Journal on Computing* 21.4 (2009), pp. 630–640.

[25] Barry L Nelson and Ira Gerhardt. “Modelling and simulating non-stationary arrival processes to facilitate analysis”. In: *Journal of Simulation* 5.1 (2011), pp. 3–8.

[26] Ran Liu et al. “Modeling and Simulation of Nonstationary Non-Poisson Processes.” In: (2013).

[27] Samaneh Aminikhanghahi and Diane J Cook. “A survey of methods for time series change point detection”. In: *Knowledge and information systems* 51.2 (2017), pp. 339–367.

[28] Linda V Green, Peter J Kolesar, and Ward Whitt. “Coping with time-varying demand when setting staffing requirements for a service system”. In: *Production and Operations Management* 16.1 (2007), pp. 13–39.

[29] Zohar Feldman et al. “Staffing of time-varying queues to achieve time-stable performance”. In: *Management Science* 54.2 (2008), pp. 324–338.

[30] Linda V Green, Peter J Kolesar, and João Soares. “Improving the SIPP approach for staffing service systems that have cyclic demands”. In: *Operations Research* 49.4 (2001), pp. 549–564.

[31] Pierre L’Ecuyer. “Modeling and optimization problems in contact centers”. In: *Quantitative Evaluation of Systems, 2006. QEST 2006. Third International Conference on*. IEEE. 2006, pp. 145–156.

[32] Rob Shone, Kevin Glazebrook, and Konstantinos Zografos. “Resource allocation in congested queueing systems with time-varying demand: An application to airport operations”. In: *European Journal of Operational Research* (2019).

[33] Vijay Mehrotra and Jason Fama. “Call center simulation modeling: methods, challenges, and opportunities”. In: *Proceedings of the 35th conference on Winter simulation: driving innovation*. Winter Simulation Conference. 2003, pp. 135–143.

[34] Mohamed A Ahmed and Talal M Alkhamis. “Simulation optimization for an emergency department healthcare unit in Kuwait”. In: *European journal of operational research* 198.3 (2009), pp. 936–942.

[35] Gerald W Evans, Edward Unger, and Tesham B Gor. “A simulation model for evaluating personnel schedules in a hospital emergency department”. In: *Proceedings Winter Simulation Conference*. IEEE. 1996, pp. 1205–1209.

[36] Linda V Green, Peter J Kolesar, and Joao Soares. “An improved heuristic for staffing telephone call centers with limited operating hours”. In: *Production and Operations Management* 12.1 (2003), pp. 46–61.

[37] Armann Ingolfsson et al. “Combining integer programming and the randomization method to schedule employees”. In: *European Journal of Operational Research* 202.1 (2010), pp. 153–163.

[38] Armann Ingolfsson et al. “A survey and experimental comparison of service-level-approximation methods for nonstationary M (t)/M/s (t) queueing systems with exhaustive discipline”. In: *INFORMS Journal on Computing* 19.2 (2007), pp. 201–214.

[39] Jens O Brunner, Jonathan F Bard, and Rainer Kolisch. “Midterm scheduling of physicians with flexible shifts using branch and price”. In: *Iie Transactions* 43.2 (2010), pp. 84–109.

[40] Armann Ingolfsson, Md Amanul Haque, and Alex Umnikov. “Accounting for time-varying queueing effects in workforce scheduling”. In: *European Journal of Operational Research* 139.3 (2002), pp. 585–597.

[41] SPJ van Brummelen et al. “Waiting time-based staff capacity and shift planning at blood collection sites”. In: *Health systems* 7.2 (2018), pp. 89–99.

[42] Tania Jiménez and Ger Koole. “Scaling and comparison of fluid limits of queues applied to call centers with time-varying parameters”. In: *OR Spectrum* 26.3 (2004), pp. 413–422.

[43] Bruce Andrews and Henry Parsons. “Establishing telephone-agent staffing levels through economic optimization”. In: *Interfaces* 23.2 (1993), pp. 14–20.

[44] Özgür Kabak et al. “Efficient shift scheduling in the retail sector through two-stage optimization”. In: *European Journal of Operational Research* 184.1 (2008), pp. 76–90.

[45] Vincent François-Lavet et al. “An introduction to deep reinforcement learning”. In: *arXiv preprint arXiv:1811.12560* (2018).

[46] Kaiwen Li, Tao Zhang, and Rui Wang. “Deep reinforcement learning for multiobjective optimization”. In: *IEEE transactions on cybernetics* (2020).

[47] Itamar Arel et al. “Reinforcement learning-based multi-agent system for network traffic signal control”. In: *IET Intelligent Transport Systems* 4.2 (2010), pp. 128–135.

[48] Bernd Waschneck et al. “Optimization of global production scheduling with deep reinforcement learning”. In: *Procedia Cirp* 72 (2018), pp. 1264–1269.

[49] Jeffrey S Smith and Barry L Nelson. “Estimating and interpreting the waiting time for customers arriving to a non-stationary queueing system”. In: *2015 Winter Simulation Conference (WSC)*. IEEE. 2015, pp. 2610–2621.

[50] Konstantinos Mykoniatis et al. “Society 5.0: A Simulation Study of Self Checkout Operations in a Grocery Store”. In: (2020).

[51] Anthony Ebert et al. “Computationally efficient simulation of queues: The R package queuecomputer”. In: *arXiv preprint arXiv:1703.02151* (2017).

[52] Michael C Fu et al. *Handbook of simulation optimization*. Vol. 216. Springer, 2015.

[53] Satyajith Amaran et al. “Simulation optimization: a review of algorithms and applications”. In: *Annals of Operations Research* 240.1 (2016), pp. 351–380.

[54] Tomasz Antczak and Rafał Weron. “Point of sale (POS) data from a supermarket: Transactions and cashier operations”. In: *Data* 4.2 (2019), p. 67.

[55] Maxim Lapan. *Deep Reinforcement Learning Hands-On: Apply modern RL methods, with deep Q-networks, value iteration, policy gradients, TRPO, AlphaGo Zero and more*. Packt Publishing Ltd, 2018.

[56] *Webinar: AnyLogic Environment for Deep Reinforcement Learning*. 2019. URL: `http://https://www.anylogic.com/resources/educational-videos/webinar-anylogic-environment-for-deep-reinforcement-learning/`.

[57] Yunan Liu. “Staffing to stabilize the tail probability of delay in service systems with time-varying demand”. In: *Operations Research* 66.2 (2018), pp. 514–534.

[58] Luc Devroye. “Nonuniform random variate generation”. In: *Handbooks in operations research and management science* 13 (2006), pp. 83–121.

[59] Sheldon M Ross et al. *Stochastic processes*. Vol. 2. Wiley New York, 1996.

[60] Gerard Cachon and Christian Terwiesch. *Matching supply with demand*. McGraw-Hill Publishing, 2008.

Appendices

Appendix A

Results of Deep Reinforcement Learning Approach

In this part, we present the result of the Deep Reinforcement Learning (DRL) solution approach for experiments we discussed in section 4.6.

A.1 Results of experiments for Run 1, Run 2, and Run 3; stationary arrival process

To experience a stationary arrival process over a day, we need to divide the day into smaller intervals [60]. To do this, we divide arrivals to 15- minutes time blocks. Our goal is to reach to half interval length of about 0.5. In this case, the mean will be equal to either LCL and/or UCL. Here is the result for running this experiment:

Table A.1: Experiment result for R1-stationary arrival process, small penalty parameters

Time	Mean	Half Interval	LCL	UCL	Rounded Up Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	1.00	0.00	1.00	1.00	1	1	1
11:00-12:00	1.00	0.00	1.00	1.00	1	1	1
12:00-01:00	1.00	0.00	1.00	1.00	1	1	1
01:00-02:00	1.00	0.00	1.00	1.00	1	1	1
02:00-03:00	1.00	0.00	1.00	1.00	1	1	1
03:00-04:00	1.00	0.00	1.00	1.00	1	1	1
04:00-05:00	1.00	0.00	1.00	1.00	1	1	1
05:00-06:00	1.00	0.00	1.00	1.00	1	1	1
06:00-07:00	1.00	0.00	1.00	1.00	1	1	1
07:00-08:00	1.00	0.00	1.00	1.00	1	1	1
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1

We experimented to examine other options when the required number of servers is more than 1. We considered a case when the the arrival rate is 180 per hour.

In this experiment, we consider arrival rate 600 per hour with service ate 40 per hour.

Table A.2: Experiment result for R2-medium stationary arrival process, high penalty parameters

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	3.00	0.00	3.00	3.00	3	3	3
09:00-10:00	3.00	0.00	3.00	3.00	3	3	3
10:00-11:00	3.00	0.00	3.00	3.00	3	3	3
11:00-12:00	3.00	0.00	3.00	3.00	3	3	3
12:00-01:00	3.00	0.00	3.00	3.00	3	3	3
01:00-02:00	3.00	0.00	3.00	3.00	3	3	3
02:00-03:00	3.00	0.00	3.00	3.00	3	3	3
03:00-04:00	3.00	0.00	3.00	3.00	3	3	3
04:00-05:00	3.00	0.00	3.00	3.00	3	3	3
05:00-06:00	3.00	0.00	3.00	3.00	3	3	3
06:00-07:00	3.00	0.00	3.00	3.00	3	3	3
07:00-08:00	3.00	0.00	3.00	3.00	3	3	3
08:00-09:00	3.00	0.00	3.00	3.00	3	3	3
09:00-10:00	3.00	0.00	3.00	3.00	3	3	3

Table A.3: Experiment result for R3-high stationary arrival process, big penalty parameters

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	15.11	0.08	15.03	15.19	16	16	16
09:00-10:00	15.11	0.08	15.03	15.19	16	16	16
10:00-11:00	15.11	0.08	15.03	15.19	16	16	16
11:00-12:00	15.11	0.08	15.03	15.19	16	16	16
12:00-01:00	15.11	0.08	15.03	15.19	16	16	16
01:00-02:00	15.11	0.08	15.03	15.19	16	16	16
02:00-03:00	15.11	0.08	15.03	15.19	16	16	16
03:00-04:00	15.11	0.08	15.03	15.19	16	16	16
04:00-05:00	15.11	0.08	15.03	15.19	16	16	16
05:00-06:00	15.11	0.08	15.03	15.19	16	16	16
06:00-07:00	15.11	0.08	15.03	15.19	16	16	16
07:00-08:00	15.11	0.08	15.03	15.19	16	16	16
08:00-09:00	15.11	0.08	15.03	15.19	16	16	16
09:00-10:00	15.11	0.08	15.03	15.19	16	16	16

A.2 Interpreting the results of R1, R2, and R3 experiments

In this set of experiments, the arrival rate is constant during the day. When the arrival rate is low, there is no variability in the output and it is reported equal to one for all time blocks. By increasing the arrival rate, for example in table A.3, there is some variability in the results. We can reduce this variability by replication.

A.3 Output of the experiments for Run 4, Run 5, and Run 6

In this section, we show the results of experiments for the cases the arrival rate has low variability during the day and three options for penalty parameters, small, medium, and big.

Table A.4: Experiment result for R4- low variability, small penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	2.01	0.45	1.56	2.46	3	2	3
11:00-12:00	6.01	0.43	5.58	6.44	7	6	7
12:00-01:00	5.89	0.52	5.37	6.41	6	6	7
01:00-02:00	5.77	0.56	5.21	6.33	6	6	7
02:00-03:00	4.88	0.45	4.43	5.33	5	5	6
03:00-04:00	4.11	0.54	3.57	4.65	5	4	5
04:00-05:00	4.66	0.62	4.04	5.28	5	5	6
05:00-06:00	4.19	0.45	3.74	4.64	5	4	5
06:00-07:00	3.98	0.49	3.49	4.47	4	4	5
07:00-08:00	3.54	0.49	3.05	4.03	4	4	5
08:00-09:00	3.32	0.49	2.83	3.81	4	3	4
09:00-10:00	3.23	0.51	2.72	3.74	4	3	4

Table A.5: Experiment result for R5- low variability, medium penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	2.45	0.55	1.90	3	3	2	3
11:00-12:00	6.55	0.52	6.03	7.07	7	7	8
12:00-01:00	6.43	0.44	5.99	6.87	7	6	7
01:00-02:00	6.21	0.43	5.78	6.64	7	6	7
02:00-03:00	5.11	0.26	4.85	5.37	6	5	6
03:00-04:00	4.76	0.43	4.33	5.19	5	5	6
04:00-05:00	5.01	0.34	4.67	5.35	6	5	6
05:00-06:00	4.87	0.55	4.32	5.42	5	5	6
06:00-07:00	4.31	0.49	3.82	4.80	5	4	5
07:00-08:00	3.91	0.52	3.39	4.43	4	4	5
08:00-09:00	3.76	0.54	3.22	4.30	4	4	5
09:00-10:00	3.67	0.63	3.04	4.30	4	4	5

Table A.6: Experiment result for R6- low variability, big penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	2.83	0.67	2.16	3.50	3	3	4
11:00-12:00	7.13	0.42	6.71	7.55	8	7	8
12:00-01:00	7.68	0.38	7.3	8.06	8	8	9
01:00-02:00	6.91	0.32	6.59	7.23	7	7	8
02:00-03:00	5.83	0.26	5.57	6.09	6	6	7
03:00-04:00	5.13	0.2	4.93	5.33	6	5	6
04:00-05:00	5.74	0.29	5.45	6.03	6	6	7
05:00-06:00	5.39	0.51	4.88	5.90	6	5	6
06:00-07:00	4.82	0.56	4.26	5.38	5	5	6
07:00-08:00	4.29	0.57	3.72	4.86	5	4	5
08:00-09:00	4.3	0.60	3.70	4.90	5	4	5
09:00-10:00	4.28	0.63	3.65	4.91	5	4	5

A.4 Interpretation of the results of Run 4, Run 5, and Run 6

When the arrival rate is low, for example, in time blocks 1, 2, 12, 13, and 14, there were not reported any variability in the resulting staffing levels and then staff rostering. So the half interval was reported equal to zero as well. The output shows by increase in the penalty parameters, the staffing level increases as well which was expected.

A.5 Experiment results for Run 7, Run 8, and Run 9

These set of experiments are for the cases with high variability among arrival rates during the day and low, medium, and high penalty parameters.

Table A.7: Experiment result for R7- high variability, small penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Man	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	5.64	0.52	5.12	6.16	6	6	7
11:00-12:00	9.56	0.53	9.03	10.09	10	10	11
12:00-01:00	9.01	0.55	8.46	9.56	10	9	10
01:00-02:00	7.52	0.43	7.09	7.95	8	8	8
02:00-03:00	10.65	0.36	10.29	11.01	11	11	12
03:00-04:00	12.22	0.55	11.67	12.77	13	12	13
04:00-05:00	12.34	0.38	11.96	12.72	13	12	13
05:00-06:00	12.33	0.51	11.82	12.84	13	12	13
06:00-07:00	10.34	0.62	9.72	10.96	11	10	11
07:00-08:00	10.55	0.51	10.04	11.06	11	11	12
08:00-09:00	10.21	0.49	9.72	10.70	11	10	11
09:00-10:00	9.33	0.56	8.77	9.89	10	9	10

Table A.8: Experiment result for R8- high variability, medium penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	6.51	0.52	5.99	7.03	7	6	8
11:00-12:00	10.23	0.41	9.82	10.64	11	10	11
12:00-01:00	10.78	0.44	10.34	11.22	11	11	12
01:00-02:00	8.63	0.32	8.31	8.95	9	9	9
02:00-03:00	11.78	0.36	11.42	12.14	12	12	13
03:00-04:00	13.64	0.34	13.3	13.98	14	14	14
04:00-05:00	13.87	0.31	13.56	14.18	14	14	15
05:00-06:00	13.28	0.51	12.77	13.79	14	13	14
06:00-07:00	11.67	0.34	11.33	12.01	12	12	13
07:00-08:00	11.75	0.57	11.18	12.32	12	12	13
08:00-09:00	11.35	0.55	10.8	11.9	12	11	12
09:00-10:00	10.56	0.51	10.05	11.07	11	11	12

Table A.9: Experiment result for R9- high variability, big penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up Man	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	1.00	0.00	1.00	1.00	1	1	1
10:00-11:00	7.01	0.67	6.34	7.68	8	7	8
11:00-12:00	11.38	0.42	10.96	11.8	12	11	12
12:00-01:00	11.94	0.38	11.56	12.32	12	12	13
01:00-02:00	9.77	0.32	9.45	10.09	10	10	11
02:00-03:00	12.52	0.26	12.26	12.78	13	13	13
03:00-04:00	14.78	0.2	14.58	14.98	15	15	15
04:00-05:00	14.79	0.29	14.5	15.08	15	15	16
05:00-06:00	14.28	0.51	13.77	14.79	15	14	15
06:00-07:00	12.59	0.56	12.03	13.15	13	13	14
07:00-08:00	12.28	0.57	11.71	12.85	13	12	13
08:00-09:00	12.07	0.60	11.47	12.67	13	12	13
09:00-10:00	11.69	0.63	11.06	12.32	12	12	13

A.6 Interpreting results of experiment 7, 8, and 9

Since the arrival rates are relatively low at the beginning of the day, the variability in resulted staffing is equal to zero in all three experiments. As we expected, by increasing the penalty parameters, the staffing level increased. Higher penalty parameters enforce the model to increase the number of staff to reduce the number of customers waiting for a certain amount of time.

A.7 Experiment result for Run 10- sinusoidal pattern

In this experiment, we apply sinusoidal arrival rate as: $\lambda(t) = \bar{\lambda}(1 + \sin(0.2t))$ we assume $\bar{\lambda} = 100$ the same as the base model in existing works [13]. As there is working hour limit for each staff, the resulted staff rostering does not follow the same sinusoidal pattern as the staffing level does.

Table A.10: Experiment result for R10- sinusoidal arrival process, big penalty parameter

Time	Mean	Half Interval	LCL	UCL	Rounded Up Man	LCL	UCL
08:00-09:00	12.12	0.54	11.58	12.66	13	12	13
09:00-10:00	12.12	0.50	11.58	12.66	13	12	13
10:00-11:00	12.12	0.51	11.58	12.66	13	12	13
11:00-12:00	6.01	0.54	5.47	6.55	7	6	7
12:00-01:00	7.4	0.49	6.86	7.94	8	7	8
01:00-02:00	11.52	0.52	10.98	12.06	12	11	13
02:00-03:00	11.56	0.37	11.19	11.93	12	12	12
03:00-04:00	11.53	0.62	10.91	12.15	12	11	13
04:00-05:00	8.76	0.38	8.38	9.14	9	9	10
05:00-06:00	11.56	0.27	11.29	11.83	12	12	12
06:00-07:00	11.45	0.27	11.18	11.72	12	12	12
07:00-08:00	11	0.27	10.73	11.27	11	11	12
08:00-09:00	10.13	0.3	9.83	10.43	11	10	11
09:00-10:00	9.97	0.48	9.49	10.45	10	10	11

A.8 Experiment result for Run 11- sinusoidal pattern with higher frequency

In this experiment, we put the arrival process into the model as the oscillating patter with higher frequency than the previous experiment.

Table A.11: Experiment result for R11- sinusoidal arrival process with higher frequency

Time	Mean	Half Interval	LCL	UCL	Rounded Up Man	LCL	UCL
08:00-09:00	11.33	0.55	10.78	11.88	12	11	12
09:00-10:00	7.79	0.48	7.31	8.27	8	8	9
10:00-11:00	11.33	0.61	10.72	11.94	12	11	12
11:00-12:00	10.13	0.49	9.64	10.62	11	10	11
12:00-01:00	9.69	0.46	9.23	10.15	10	10	11
01:00-02:00	9.47	0.5	8.97	9.97	10	9	10
02:00-03:00	8.47	0.48	7.99	8.95	9	8	9
03:00-04:00	9.49	0.56	8.93	10.05	10	9	11
04:00-05:00	10.13	0.5	9.63	10.63	11	10	11
05:00-06:00	10	0.43	9.57	10.43	10	10	11
06:00-07:00	1	0.52	0.48	1.52	1	1	2
07:00-08:00	7.84	0.45	7.39	8.29	8	8	9
08:00-09:00	10.46	0.47	9.99	10.93	11	10	11
09:00-10:00	10.48	0.49	9.99	10.97	11	10	11

A.9 Experiment result for Run 12- sinusoidal pattern with higher amplitude

Table A.12: Experiment result for R12- sinusoidal arrival process with higher amplitude

Time	Mean	Half Interval	LCL	UCL	Rounded Up Mean	LCL	UCL
08:00-09:00	13.64	0.5	13.14	14.14	14	14	15
09:00-10:00	13.64	0.5	13.14	14.14	14	14	15
10:00-11:00	13.64	0.5	13.14	14.14	14	14	15
11:00-12:00	10.63	0.55	10.08	11.18	11	11	12
12:00-01:00	11.32	0.48	10.84	11.8	12	11	12
01:00-02:00	11.36	0.46	10.9	11.82	12	11	12
02:00-03:00	13.5	0.51	12.99	14.01	14	13	15
03:00-04:00	13.5	0.5	13	14	14	13	14
04:00-05:00	13.96	0.49	13.47	14.45	14	14	15
05:00-06:00	14.14	0.53	13.61	14.67	15	14	15
06:00-07:00	14.45	0.25	14.2	14.7	15	15	15
07:00-08:00	14.34	0.45	13.89	14.79	15	14	15
08:00-09:00	13.66	0.57	13.09	14.23	14	14	15
09:00-10:00	13.9	0.56	13.34	14.46	14	14	15

A.10 Interpretation of the results of sinusoidal experiments (run 10, run 11, and run 12)

According to the figure 4.10, bottom right panel, that represents the arrival processes for three different sinusoidal patterns and also the result of DRL in tables A.10, A.11, and

A.12, by increasing the amplitude in run 12 than the amplitude in run 10, we observe the number of required staff were increased which matches our expectation. As this is rostering problem and there is a constraint of working at least 6 hours per day for each staff member, we do not see the sinusoidal pattern in staffing the same as the arrival process.

A.11 Experiment result for Run 13, 14

Table A.13 shows the result of staff scheduling for step pattern 1 in the arrival process.

Table A.13: Experiment result for R13- step pattern 1 in arrival process

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Mean	LCL	UCL
08:00-09:00	1.00	0.00	1.00	1.00	1	1	1
09:00-10:00	13.44	0.48	12.96	13.92	14	13	14
10:00-11:00	13.44	0.48	12.96	13.92	14	13	14
11:00-12:00	12.82	0.48	12.34	13.3	13	13	14
12:00-01:00	9.32	0.48	8.84	9.8	10	9	10
01:00-02:00	11.36	0.47	10.89	11.83	12	11	12
02:00-03:00	10.94	0.46	10.48	11.4	11	11	12
03:00-04:00	10.66	0.24	10.42	10.9	11	11	11
04:00-05:00	11.11	0.71	10.4	11.82	12	11	12
05:00-06:00	11.33	0.51	10.82	11.84	12	11	12
06:00-07:00	10.65	0.5	10.15	11.15	11	11	12
07:00-08:00	9.66	0.5	9.16	10.16	10	10	11
08:00-09:00	9.02	0.51	8.51	9.53	10	9	10
09:00-10:00	9.09	0.44	8.65	9.53	10	9	10

Table A.14 shows the result of staff scheduling for step pattern 2 in the arrival process.

Table A.14: Experiment result for R14- step pattern 2 in arrival process

Time	Mean	Half Interval	LCL	UCL	Rounded Up		
					Man	LCL	UCL
08:00-09:00	10.34	0.37	9.97	10.71	11	10	11
09:00-10:00	10.31	0.33	9.98	10.64	11	10	11
10:00-11:00	10.31	0.33	9.98	10.64	11	10	11
11:00-12:00	10.3	0.39	9.91	10.69	11	10	11
12:00-01:00	10.3	0.32	9.98	10.62	11	10	11
01:00-02:00	10.3	0.4	9.9	10.7	11	10	11
02:00-03:00	10.3	0.35	9.95	10.65	11	10	11
03:00-04:00	6.77	0.45	6.32	7.22	7	7	8
04:00-05:00	6.75	0.59	6.16	7.34	7	7	8
05:00-06:00	6.71	0.6	6.11	7.31	7	7	8
06:00-07:00	6.63	0.62	6.01	7.25	7	7	8
07:00-08:00	6.6	0.64	5.96	7.24	7	6	8
08:00-09:00	6.72	0.51	6.21	7.23	7	7	8
09:00-10:00	6.61	0.4	6.21	7.01	7	7	8

A.12 Interpretation of the results of run 13 and run 14

Table A.13 shows the staff rostering for the step pattern 1 which is a type of U-shape arrival pattern that the arrival rate remains constant for a few time blocks. The table shows the desired staff scheduling is almost constant for some consecutive time buckets. Again because of the working hour constraint for the staff members, the overall shape of output is not the same as arrival process shape. Table A.14 represent the step pattern that the arrival rate is constant for half of the period and then decreases and stay constant until the end of period. The staff rostering resulted from DRL algorithm remains constant with some variability over the day.

Table A.15: Experiment result for R15- bi-modal arrival pattern

Time	Mean	Half Interval	LCL	UCL	Rounded Up Mean	LCL	UCL
08:00-09:00	1.20	0.22	0.98	1.42	2	1	2
09:00-10:00	7.49	0.75	6.74	8.24	8	7	9
10:00-11:00	8.41	0.33	8.08	8.74	9	9	9
11:00-12:00	8.41	0.33	8.08	8.74	9	9	9
12:00-1:00	10.02	0.33	9.69	10.35	11	10	11
1:00-02:00	14.24	0.27	13.97	14.51	15	14	15
02:00-03:00	14.97	0.27	14.7	15.24	15	15	16
03:00-04:00	14.25	0.23	14.02	14.48	15	15	15
04:00-05:00	10.62	0.65	9.97	11.27	11	10	12
05:00-06:00	8.39	0.61	7.78	9	9	8	9
06:00-07:00	11.38	0.71	10.67	12.09	12	11	13
07:00-08:00	12.54	0.85	11.69	13.39	13	12	14
08:00-09:00	11.12	0.62	10.5	11.74	12	11	12
09:00-10:00	10.43	0.38	10.05	10.81	11	11	11

A.13 Experiment result for Run 15

Table A.15 shows the staffing scheduling for the bi-modal arrival pattern. As in this arrival process the arrival rate experiences two peaks of arrival, we expect to see the to increasing patterns in the staff rostering as well, but as there is minimum working hours constraint, these increases so not happen at the same location that arrival rate did. The result in table shows that the staff rostering is as we expected.

Appendix B

Resource code for algorithm 1 in chapter 3

This section represents the source code we developed in R to determine staffing levels in chapter3. This code is based on algorithm 1 of chapter 3.

```
library("queuecomputer")
library(dplyr)
library("randomNames")
library("ggplot2")
library(readr)
library(RTriangle)
library("xlsx")
require(triangle)
#set.seed(1)
#Read the arrival time data
arrivals1<- read.csv
(file.choose(), header = TRUE, sep = ",", dec = ".")
#Creating service time random
##observations for exponential distribution
service1<-data.frame(rexp(nrow(arrivals1), 20))
#Creating service time random observations
##for triangular distribution
#service1<-data.frame(rtriangle(nrow(arrivals1), 1, 5, 3))
#Creating arrival time list in required format
arrivals <- arrivals1[1:nrow(arrivals1),]
#Creating service time list in required format
service<-service1[1:nrow(arrivals1),]
###Initiate the Staffs, 24 hours a day
#Staffs<-c(1,1,1,1,1,1,1,1,1,1,
1,1,1,1,1,1,1,1,1,1,1,1,1,1)
###Initiate the Staffs, 14 hours a day
Staffs<-c(1,1,1,1,1,1,1,1,1,1,1,1,1,1)
#Defining the time blocks, 24 hours a day
#TimeBlocks<- c(1,2,3,4,5,6,7,8,9,10,11,
12,13,14,15,16,17,18,19,20,21,22,23)
#Defining the time blocks, 14 hours a day
TimeBlocks<- c(1,2,3,4,5,6,7,8,9,10,11,12,13)
#Queuing system,
```

```r
##existing definitions in "queuecomputer" package
resource_schedule <- as.server.stepfun(TimeBlocks, Staffs)
queue_obj<- queue_step(arrivals =
arrivals, service = service, servers = resource_schedule)
#arrival times
ArrivalsTimes<-queue_obj$departures_df$arrivals
#Converting waiting times to minutes, rounding waiting times
WaitingTimes<-round((queue_obj$departures_df$waiting)*60, 3)
####putting waiting times in the data frame
Waiting_df<-data.frame(ArrivalsTimes, WaitingTimes)
#Calculating variance of waiting times
total_variance<-var(Waiting_df$WaitingTimes)
#Calculating standard deviation of waiting times
total_sd<-sd(Waiting_df$WaitingTimes)
#summary(queue_obj)
WaitingtimeList<-list()
###Collecting the waiting times of the
###arrivals in each time block
WaitingtimeList[[1]]<- Waiting_df
[Waiting_df$ArrivalsTimes <TimeBlocks[1],2]
#Initializing the percentile set,
##maximum and variance lists for waiting times in each time blocks
qualtileset<-0
Maxset<-0
varianceset<-0
qualtileset[1]<-quantile(WaitingtimeList[[1]],0.9)
Maxset[1]<-max(unlist(WaitingtimeList[1]))
varianceset[1]<-var(WaitingtimeList[[1]])
for ( i in 2:(length(TimeBlocks)+1)) {
  WaitingtimeList[[i]]<- Waiting_df
    [(Waiting_df$ArrivalsTimes<TimeBlocks[i]
    & Waiting_df$ArrivalsTimes> TimeBlocks[i-1]),2]

  WaitingtimeList[[length(TimeBlocks)+1]]
  <- Waiting_df[ Waiting_df$ArrivalsTimes>
        TimeBlocks[length(TimeBlocks)],2]
  qualtileset[i]<-quantile(WaitingtimeList[[i]],0.9)
  qualtileset[length(TimeBlocks)+1]
  <-quantile(WaitingtimeList[[length(TimeBlocks)+1]],0.9)
  Maxset[i]<-max(unlist(WaitingtimeList[[i]]))
  Maxset[length(TimeBlocks)+1]
  <-max(unlist(WaitingtimeList[[length(TimeBlocks)]]))
  varianceset[i]<-var(WaitingtimeList[[i]])
  varianceset[length(TimeBlocks)+1]
  <-var(WaitingtimeList[[length(TimeBlocks)+1]])
}
print(total_sd)
print(total_variance)
print(varianceset)
print(qualtileset)
########Getting the number of staff members to satisy
#####the service level, case of percentile
########as the function of waiting
##times and maximum as the function of waiting times
for ( i in 1:(length(TimeBlocks)+1)) {
  while ( qualtileset[i] >10){
  #while ( Maxset[i] >10){
    Staffs[i]<-Staffs[i]+1
    resource_schedule <- as.server.stepfun(TimeBlocks, Staffs)
    queue_obj<- queue_step(arrivals = arrivals,
    service = service, servers = resource_schedule)
    ArrivalsTimes<-queue_obj$departures_df$arrivals
    WaitingTimes<-round((queue_obj$departures_df$waiting)*60, 3)
    Waiting_df<-data.frame(ArrivalsTimes, WaitingTimes)
    WaitingtimeList<-list()
    WaitingtimeList[[1]]<-
    Waiting_df[Waiting_df$ArrivalsTimes <TimeBlocks[1],2]

    for ( j in 2:length(TimeBlocks))
    {
      WaitingtimeList[[j]]<-
        Waiting_df[(Waiting_df$
        ArrivalsTimes <TimeBlocks[j]
       & Waiting_df$ArrivalsTimes> TimeBlocks[j-1]),2]
      WaitingtimeList[[j+1]]<-
      Waiting_df[ Waiting_df$ArrivalsTimes> TimeBlocks[j],2]
    }
    qualtileset[i]<-quantile(WaitingtimeList[[i]],0.9)
    #Maxset[i]<-max(unlist(WaitingtimeList[[i]]))
  }
}
############Printing the number of staffs in each time block
print(Staffs)
##########Getting the summary of queue components
summary(queue_obj)
##sdset[i]<-sd(unlist(WaitingtimeList[i]))}
##var(unlist(WaitingtimeList[2]))
##max(unlist(WaitingtimeList[2]))
#######Finding the beginning
######of the shift with minimum variations in the staffing levels in ea
Staff_shifts<-list()
Staffs1<-c(Staffs,Staffs)
Variances<-0
for (i in 1:length(Staffs)){
  Staff_shifts[[i]]<-Staffs1[i:(i+7)]
  Variances[i]<-var(unlist(Staff_shifts[[i]]))
}
print(which.min(Variances))
```