August 2023

Comparing Elevator Strategies for a Parking Lot

Naveed Arafat
University of Windsor, arafatn@uwindsor.ca

Comparing Elevator Strategies for a Parking Lot

by

Naveed Arafat

A Major Research Paper
Submitted to the Faculty of Graduate Studies
through the Department of Mathematics and Statistics
in Partial Fulfillment of the Requirements for
the Degree of Master of Science at the
University of Windsor

Windsor, Ontario, Canada

2023

Comparing Elevator Strategies for a Parking Lot

by

Naveed Arafat

APPROVED BY:

M. Belalia
Department of Mathematics and Statistics

M. Hlynka, Advisor
Department of Mathematics and Statistics

July 25, 2023

Author's Declaration of Originality

Abstract

In this paper, we compare elevator strategies for a parking garage. It is assumed that the parking garage has several floors and there is an elevator which can stop on each floor. We begin by considering 4 strategies detailed in page 23. For each strategy, we loop the program 100 times, and get 100 mean values for wait times. Welch's test confirms highly significant differences among the 4 strategies. Repeating the analysis multiple times we see that the best of the 4 strategies is strategy 2, which places the elevator on floor 2 (the median floor) after use.

Acknowledgements

I would like to express my gratitude to my supervisor Professor Hlynka for his guidance and help throughout the creation of my major paper. I also want to express my thanks for the kindness and patience he always showed to me throughout my study at the University of Windsor. Thank you also to my departmental reader, Dr. Belalia.

I would like to thank the Department of Mathematics and Statistics, my family, friends and especially my mother, for her love and support.

Contents

List of Figures

CHAPTER 1

Elevator Strategies

In this chapter, we discuss elevator strategies in general and give a historical review. There are many companies that manufacture elevators. Two of the best known are Otis Elevators, and TK (ThyssenKrupp). The first passenger elevator was built in 1856 in New York.

Elevators for people are common in many settings: office buildings, hotels, apartment buildings, gymnasia, retail outlets, hospitals, schools, universities, government buildings. Some places have a single elevator and others have a battery of elevators. Some elevators are restricted to certain levels. Most elevators have buttons to select a floor on the inside cabin. Others have the selection external to the elevator.

There is a classic story about a slow elevator system in a building. There were numerous complaints from users. Various solutions were proposed. Add another elevator. Put most commonly used businesses on lower floors. Move vacant elevators to the main floor. The eventual solution was to add mirrors in and near to the elevator. This distracted users. They adjusted their clothing and tried to improve their own appearance. They could look at their fellow users without being conspicuous. Remarkably, most complaints stopped.

Often, elevators have some computer algorithms built in to help with decision making. For example, if an elevator is on floor 1 and two riders enter, one of whom wants floor 2 and the other wants floor 5, the elevator algorithm knows to stop on floor 2 first and then proceeds to floor 5 regardless of which request was made first.

For the pair of elevators in Lambton Tower, at the University of Windsor, the elevator pair is programmed so that there is an attempt to keep at least one of the two elevators at floor 1. So if the current vacant pair is sitting at (1,5) (i.e. one of the elevators is currently at floor 1 and the other is currently at floor 5) and if a rider enters the elevator at floor 1, and pushes the elevator button to go to floor 7, the other empty elevator (at floor 5) automatically and simultaneously begins to move to floor 1, even though no such request was made by a rider. The idea is that there should be an elevator kept on floor 1 as it is the most commonly requested floor (for arrival or departure). This would allow for a simplified Markov chain transition matrix. Given that there are 11 floors (Ground plus 10 others), that would mean there are $11^2 = 121$ states. If we consider state (a, b) to be equivalent to (b, a), then there would only be $\binom{}{11,2+11}$ different states. If we restrict states to have form $(1, a)$, then we would only have to consider 11 possible states. This would be reasonable if the elevator was empty most of the time - say late at night. There is the classic issue for whether one should use the elevator or use the stairs. Elevator use may have lower expected time but stair use may have lower expected variability of time. Which is more important? How many people have been stuck in a broken elevator?

In tall buildings, elevators are sometimes given restricted floors. For example, one elevator services floors $1, 2, ..., 10$, and another services floors $1, 11, 12, ..., 20$.

CHAPTER 2

Literature Review

There is very little literature on parking lot elevators. Almost all literature is for standard buildings.

The evaluation of the elevator round-trip time has been, and still is, fundamental to the design of elevator systems (Al-Sharif et al., 2014, [6]). Historically, round-trip time has been evaluated using analytical methods based on equations. The equations are derived using the expected value of critical parameters that constitute the round-trip time. Numerical and modern analytical methods have been introduced for evaluating round-trip time, such as the Monte Carlo simulation method and Markov Chains. Al-Sharif et al. (2014) list and discuss six analytical, numerical, and simulation methods for evaluating round-trip time. They also compare the six methods, highlighting the advantages and disadvantages of each method and areas of application. Monte Carlo simulation is a practical way of evaluating the round-trip time by randomly generating a large number of scenarios and taking the average value of the round-trip time resulting from each scenario.

Crites and Barto (1994, [8]) describe a new reinforcement learning approach called "RL," which uses reinforcement learning to solve a vast, unstated, and intractable problem. In this paper, the authors present a reinforcement learning approach to the elevator dispatcher problem. They use an RL framework to predict the behaviour of an eight-story building with four elevator cars. The goal of the RL system is to reduce the average waiting time by as much as possible while keeping the overall cost of the operation low.

In Pepyn and Cassandras (1998, [20]), an online adaptive dispatching control algorithm was designed for elevator systems during peak passenger traffic. The concurrent estimation dispatching algorithm allows us to observe the elevator system unobtrusively while it operates. This paper is a companion to their previous paper, where they proved that the structure of the optimal dispatching policy for elevator systems in peak traffic is a threshold-based policy with threshold parameters that change as a function of the passenger arrival rate. Their objective is to observe an actual elevator system while it is operating under some arbitrary dispatching threshold. They use the TWA to construct hypothetical sample paths and estimate the passenger waiting times that would have resulted if the elevator system had operated under various thresholds. These concurrently estimated waiting times are used to adapt to the operating thresholds.
Al-Sharif et al. (2013, [5]) present a step-by-step automated design methodology that gives the optimum number of elevators in specific, constrained arrival situations. It uses the round-trip time calculated using other tools to arrive at an optimal design for a building. The design of an elevator system heavily relies on calculating the round-trip time under up-peak (incoming) traffic conditions. The round-trip time can either be calculated analytically or by the use of Monte Carlo simulation. However, the round-trip time calculation is only part of the design methodology. Evaluating the round-trip time is critical to the design of elevator traffic systems. This paper introduces the Markov Chain Monte Carlo method as a numerical tool. The method is compelling in cases where analytical equations do not exist for unique building conditions. This paper introduces a methodology for evaluating the elevator round-trip time using the Markov Chain Monte Carlo method. The method introduced in this paper is restricted to the case of a single-entrance building. In order to calculate the first term of the round-trip, the traveling time, it is necessary to find the kinematic matrix. This represents the

time required for the elevator to travel between two floors starting at rated speed, acceleration, and jerk. Six passengers were included in the analysis.

A wide variety of problems in diverse areas, ranging from manufacturing to computer communications, involve sequential decision-making under uncertainty. In Das et al. (1999, [9]), they present a reinforcement learning algorithm for semi-Markov decision problems with a high degree of confidence. It is based on an earlier version of the Markov reinforcement learning approach originally proposed by Mahadevan. In this paper, the authors use a reinforcement learning framework to develop an efficient and cost-effective method for predicting future events. They use a real-world inventory problem as an example. The results of the proposed method are compared with those obtained from two heuristics. In recent times, there has been more interest in a stochastic approximation method that is more simulation-based, called reinforcement learning (RL), for computing near-optimal policies for MDPs. RL has been successfully applied to a large number of problems, such as elevator scheduling and dynamic channel allocation of cellular telephone systems.

Additionally, the authors aim to extend reinforcement learning to a general class of decision tasks that are referred to as semi-Markov decision problems (SMDPs). Specifically, they focus on SMPDs under the average-reward criterion, presenting a new model-free RL algorithm called SMART (Semi-Markov Average Reward Technique). Furthermore, they present a detailed study of SMART on a combinatorically large problem of determining the optimal preventive maintenance schedule of a product inventory system.

SMART was implemented and tested on a commercial discrete-event simulator, CSIM. The effectiveness of the SMART algorithm was demonstrated by first comparing its results with optimal results for a minor problem and then applying it to

a large-scale problem and comparing its results with those obtained from two suitable heuristic methods. This was the first large-scale implementation of average-reward reinforcement learning and was part of a major ongoing cross-disciplinary study that involves industrial process optimization using reinforcement learning. For more complex systems consisting of numerous interrelated subsystems of machines, it may be more appropriate to design a hierarchical control system where each subsystem is controlled using separate agents instead of having a single agent governing the whole system. The elevator problem is an example of a multi-agent system, where the agents are homogeneous and control identical subsystems.
In another paper, Patino-Forero et al. (2012, [19]) proposed a novel architecture for elevator systems based on a DeviceNet® industrial network for connecting several industrial devices, such as a Programmable Logic Controller (PLC) which implements the Local Control Systems (LCSs), Panel View for implementing Destination Control System (DCS), frequency inverters and inductive sensors. The OLE for Process Control (OPC) was developed in Java language sending data between the PLC and an Elevator Group Control System (EGCS) based on fuzzy logic (FEGCS) and division zoning techniques that runs on PC. The EGCS was designed to increase the performance of the system of elevators and takes advantage of the division zoning techniques and the traffic pattern identification performed by FEGCS, as well as the previous information provided by DCS, avoiding uncertainties of a conventional system. The LCS was designed to be implemented in only one PLC, saving industrial instrumentation and decreasing energy consumption and implementation cost. By accomplishing the proposed architecture, they hope that the elevator system takes advantage of the security, versatility, and robustness provided by industrial instrumentation. The DeviceNet® industrial network allows connecting up to 64 nodes, increasing the system's scalability in the number of elevators and floors. The OPC protocol provides high versatility

and flexibility in transmitting information, allowing exploration of new approaches for controlling automation integrated systems in modern buildings.

Marja-Liisa Siikonen (1993) reports that this paper describes a method for predicting the performance of an elevator in a building. It uses a combination of Monte Carlo and Bayesian methods to predict the effect of different types of traffic on the performance, or "call allocation," of the elevator. This paper presents a new approach to the problem of predicting elevator performance by assuming that only one car in a group has passenger waiting times. It uses a novel approach to predict the behavior of an elevator with respect to time intervals and load values.

In Brand and Nikovski (2004, [7]), the authors consider the problem of optimally parking empty cars in an elevator group to anticipate and intercept the arrival of new passengers and minimize their waiting times. Two solutions are proposed for the down-peak and up-peak traffic patterns. They demonstrate that matching the distribution of free cars to the arrival distribution of passengers was sufficient to produce savings of up to 80

Nikovski and Brand (2004, [18]) present an efficient algorithm for exact calculation and minimization of expected passenger waiting times using a bank of elevators. The dynamics of the system are represented by a discrete-state Markov chain embedded in the continuous phase space diagram of a moving elevator car. The chain was evaluated efficiently using dynamic programming to compute measures of future system performance, such as expected waiting time, averaged over all possible future scenarios. An elevator group controller based on this method significantly outperforms benchmark algorithms and, although slower than them, was entirely within the computational capabilities of currently existing elevator bank controllers.

Brand and Nikovski (2004, [7]) describe the exact calculation of expected waiting times for Group elevator control. Group elevator scheduling is a complex optimal control problem that has been researched extensively due to its high practical significance. The usual performance criterion to be optimized is the average waiting time (AWT) of all passengers in the system. They demonstrate that such a simplification is optional and describe an algorithm that can compute the proper expectation of AWT over all possible paths of an elevator car. New passengers arrive at a bank of elevators at random times and floors. The dynamics of the system are represented by a discrete-state Markov chain.

In Sun, Zhao, and Luh (2010, [24]), a two-level formulation of the group elevator scheduling problem with advanced traffic information is developed. A new door action control method and a hybrid nested partitions and genetic algorithm method for the passenger-to-car assignment are proposed. A team from the University of Connecticut led by Jin Sun (2010) reports that in this paper, they address the problem of building elevators in a high-rise building and how to effectively use advanced information to control the speed of the door. Sun and colleagues present three examples of how well the methods work:

1) using near-Optimality of the solution,

2) using the value of forward traffic information,

3) using genetic algorithm

4) using single-car dispatching.

This paper presents a two-level approach to predicting future traffic behavior. The proposed methods are based on a two-stage framework, which uses detailed car dynamics to predict future behavior. Six passengers were included in the analysis.

Feng and Redner (2020, [10]) focus on the idealized case of a building with a single unlimited-capacity elevator. With a single infinite capacity elevator, a steady state was eventually achieved where the average time for the elevator to complete a single cycle equals the average number of people who arrive in the lobby during a cycle. A single cycle of an elevator involves the following steps: The elevator arrives on the ground (lobby) floor; it delivers each passenger to her/his destination floor in ascending order, and passengers with this destination floor exit. When the elevator empties, it returns to the ground floor, and a cycle begins anew. The researchers investigated the transport of people by elevators in a tall building during the start of its daily operation within the framework of a minimal probabilistic model. There were 21 elevators involved in the analysis. The authors' results strengthen earlier work on this topic: "We know that the average clearing time is infinite, and its distribution asymptotically decays as $n^{-3/2}$, where n is the number of cycles. This condition corresponds to the lobby being cleared," Feng argued.

Tanida (2022, [25]) considered a downward elevator system motion during peak loads. The group examined the order parameter of the elevator motion and round-trip time for various inflow rates of passengers and the proportion of passengers set to ride the first-arriving elevator. The effects of the two parameters on the round-trip time are different. Dynamics and order formation in nonequilibrium systems have garnered widespread interest in physics. They investigated differences between the dynamics of elevators when they were isolated or coupled. They introduced a mathematical model to explain the behavior of R.

Zhang and Tsiligkaridis (2022, [27]) mention how Group Elevator Control (GEC) is a demanding industrial control problem that needs to be solved repeatedly. The group proposed a Predictive Group Elevator Scheduler that uses predictive information about passengers' arrivals from a Transformer-based destination predictor. Empirical experiments validated the effectiveness and efficiency of the approach. Kwon et al. described a sensor-aware GEC method that places reservation calls for future passengers yet to arrive. They found savings of waiting time for a traditional myopic GEC algorithm on the order of 5

Al Sukkar et al. (2017, [2]) attempts to understand the reasons for the difference in the round-trip time between calculation and simulation. It is deemed that the main reason for the difference is the combination of two factors: the limited car capacity and the randomness in the behavior of the elevator traffic system, thus leading to reduced efficiency in car loading due to a smaller number of passengers in the car. There are three sources of randomness in the behavior of the system - the randomness of the passenger destinations (thus making the value of the round-trip time a random variable), the randomness of the passenger arrival (driven by a Poisson passenger arrival model), the effect of elevator bunching (thus making the value of the interval a random variable). Using a MATLAB-based simulator, the value of the round-trip time is plotted against the system loading level for the case of a single entrance and incoming traffic only. Different conditions are simulated, including constant and random passenger arrivals, queues-allowed and queues-not-allowed conditions. Varying these conditions provides an insight into the variation of the round-trip time and its reasons. The effect of the number of passengers boarding the elevator on the value of the round-trip time (and thus on the value of the system handling capacity) is investigated in more detail.

Hakonen and Siikonen, (2008, [11]) states in their paper that the passenger service level in an elevator system depends on the group control and cannot be calculated directly. With conventional control, waiting times and intervals correlate to up-peak. With a destination control system (DC), interval and waiting times do not correlate similarly to conventional collective control. Therefore, simulation has become essential in determining passenger waiting times with DC. Passenger arrivals follow a Poisson distribution, and simulation results vary depending on the random seed number of the simulation. This article studies different simulation procedures and the consistency of the simulation results.

Two performance measures commonly used in planning elevators are Up-peak Handling and Waiting Time. The first measure was to compare the actual waiting times and time to the destination. When comparing the actual average waiting times, they see that the differences between average and standard deviations are much more significant than the typical deviations. This paper discusses two of the most commonly used performance measures in elevator planning: up-peak handling ability and up-peak interval. The first is to find out the highest acceptable Waiting Time with an 80 percent car load. The second is to study how well the group can handle peak traffic. Passenger Waiting Times are shorter than Interference when the arrival time is below the handling capacity. Simulations allow us to determine how many passengers can fit in an average car based on the number of people arriving. Handling Capacity is calculated from the arrival rate. A higher loading factor will be more accurate since it considers the passenger arrival size.

Robert H. Crites, Andrew G. Barto (1994) describes a new reinforcement learning approach, called "RL," which uses reinforcement learning to solve a very big, unstated, and intractable problem. It uses a novel kind of top-down decomposition to achieve extremely large performance. In this paper, the authors present a novel reinforcement learning approach to the elevator dispatcher problem. They use a RL framework to predict the behavior of an eight-story building with four elevator cars. The goal of the RL system is to reduce the average waiting time by as much as possible while keeping the overall cost of the operation low.

In Pepyn et al. an on-line adaptive dispatching control algorithm was designed for use in elevator systems during uppeak passenger traffic. The concurrent estimation dispatching algorithm allows us to observe the elevator system, in an unobtrusive way, while it operates. This paper is a companion to their previous paper, where they proved that the structure of the optimal dispatching policy for elevator systems in uppeak traffic is a threshold-based policy with threshold parameters that change as a function of the passenger arrival rate. Their objective is to observe an actual elevator system while it is operating under some arbitrary dispatching threshold. They use the TWA to construct the hypothetical sample paths and estimate the passenger waiting times that would have resulted if the elevator system had been operating under the various thresholds. These concurrently estimated waiting times are used to adapt to the operating thresholds.

Al-Sharif, L., Algzawi, Hammodeh, A. T. (2013) presents a step-by-step automated design methodology which gives the optimum number of elevators in very specific, constrained arrival situations. It uses the round-trip time calculated by the use of other tools to arrive at an optimal design for a building.

The design of an elevator system heavily relies on the calculation of the round-trip time under up-peak (incoming) traffic conditions. The round-trip time can either be calculated analytically or by the use of Monte Carlo simulation. However, the calculation of the round-trip time is only part of the design methodology.

Evaluating the round-trip time is critical to the design of elevator traffic systems. This paper introduces the Markov Chain Monte Carlo method as an additional numerical tool. The method is very powerful in cases where analytical equations do not exist for the special building conditions. This paper introduces a methodology for evaluating the elevator round trip time by using the Markov Chain Monte Carlo method. The method introduced in this paper is restricted to the case of a single entrance building. In order to calculate the first term of the round trip, the travelling time, it is necessary to find the kinematic matrix. This represents the time required for the elevator to travel between any two floors starting at rated speed, rated acceleration and rated jerk. 6 passengers were included in the analysis.

CHAPTER 3

Assumptions for our Elevator Model

We assume that the parking garage has n levels and that the elevator can stop at each level. Assume that each level can hold k vehicles. The ground floor is called level 1. Assume that the interarrival time of vehicles is exponentially distributed with rate λ. Assume that the time that a single vehicle stays in the parking garage is exponential with rate μ.

The exponential distribution is used for the model because the exponential distribution is memoryless. This means that we need to keep track of less information in our notation and analysis.

CHAPTER 4

R program

Before presenting our R program to simulate and analyse elevator strategies, we state a well known result about exponential random variables.

PROPOSITION 4.1. *(Memoryless Property) Let X be an exponential random variable with pdf $f(x) = \lambda e^{-\lambda x}$, $x \geqslant 0$. Then for fixed $k > 0$, $P(X > x + k|X > k) = P(X > x)$.*

PROOF.

$$P(X > x) = 1 - F(x) = 1 - (1 - e^{-\lambda x}) = e^{-\lambda x}.$$

$$\begin{aligned} P(X > x + k|X > k) &= \frac{P(X > x + k \text{ and } X > k)}{P(X > k)} = \frac{P(X > x + k)}{P(X > k)} \\ &= \frac{e^{-\lambda(x+k)}}{e^{-\lambda k}} = e^{-\lambda x} = P(X > x) \end{aligned}$$

□

The memoryless property means that if we are looking at processes that have exponential interarrival and service times, then our simulation programs do not need to keep track of the time since the last event, and we only need to keep track of the current system state. This simplifies our programming task.

We present an R program and analyze it in some detail. We will be assuming exponentially distributed interarrival times and exponentially distributed parking times. Because the exponential distribution is memoryless, we do not have to store excessive information. We need to know the configuration of the parking garage. Let N = number of floors. Let K = number of parking positions on each floor.

We assume that K is the same for each floor though we could allow the number to depend on which floor is considered.

Events consist of arrivals or departures. We need to keep track of the number of cars on each floor, the position (floor) of the elevator at the beginning of an event, the distance that the elevator needs to move to adjust to the event. An elevator strategy is a rule by which we choose to move the elevator after the occurrence of an event, in order to prepare for the next event.

Possible elevator strategies could include

(a) leave elevator at position resulting from an event.

(b) always move elevator to the ground level (floor 1)

(c) Always move elevator to the median floor position

Our R program works as follows. at each time point, we have

(a) a vector giving the number of vehicles on each floor

(b) the current time

(c) the current elevator floor

plus other information. From this, generate

(i) generate an interevent time (which depends on (a) since a larger total number of vehicles suggests a shorter interevent time)

(ii) compute an updated time from (b) and (i)

(iii) generate type of event -arrival or departure (which depends on (a)

(iv) an updated elevator end position (depending on strategy)

(v) calculate distance moved by elevator to respond to call.

In our R program, we begin with the special case of $K = 5$ spots per floor, and $N = 4$ floors. Of course, $K = 5$ is extremely small but we wish to examine output to see what happens as floors fill up. We create a matrix which will store all the information that we are interested in. Each row will need the count for each floor. The first entry in each row will be the event time (of either an arrival or a departure). The first row of A sets the start time to 0. The next N entries in each row will be the updated number of vehicles on each floor.

Before we explain our R program, let us show some output.

```
          [,1] [,2] [,3] [,4] [,5] [,6]      [,7] [,8] [,9] [,10]
[1,] 0.0000000    0    0    0    0    1 0.8118205    1    0     1
[2,] 0.8118205    1    0    0    0    1 1.8259720    1    0     1
[3,] 2.6377925    2    0    0    0    1 1.3522721    1    0     1
```

We show 3 lines of output. The first entry of each row gives the current time. The next 4 entries (columns 2,3,4,5) of each row indicate the number of cars parked on each of the floors. In our case, we have 4 floors. At time 0. we have 0 parked cars on each floor. The next entry (column 6) in each row shows the event type ET (=1 if arrival and =2 if departure). The 7th column for each row gives the interevent time (time until the next event). The 8th column entry gives EL, the event location (floor). The 9th entry gives EM, the distance that the elevator must move to handle the event. Finally, the tenth column gives EE, the elevator end position, after the event.

The entry EM is the entry of major interest, which gives an indication of the amount of time needed for the elevator to arrive to service the customer need.

In the program, we define indicator variables $B1$ (=1 if system is empty; otherwise 0), $B2$ (=1 if system is full), $B3$ (=1 if system is neither empty nor full). The program follows.

```
#N=number of floors    K=number of cars per floor
#M= number of rows of matrix output
N=4;K=5;M=50
# matrix A will store all information at each time step
A=matrix(1:(M*(6+N))  ,M,(N+6))
# We fill A with numbers to initialize but they will change over time
# T= time
#  C[1] C[2]... C[N] = configuration (number of cars on each floor)
# ET= event type (1 means arrival, 2 means departure)
# IT=interevent time
# EL=event location
# EM = elevator move distance
# EE=elevator end position
    #initial configuration
    # Note that rexp(1,.5) generates 1 exponential random value with
    # rate .5
T=0; C=0*(1:N); ET=1; IT=rexp(1,.5); EL=1; EM=0;  EE=1
A[1,]=c(T,C, ET,IT, EL, EM,EE)
# updates
#time update
    F1=function(T,IT){T+IT}
#F6 event type update
F6=function(C){S=sum(C);B1=(S==0); B2=(S==(N*K));
B3=1-B1-B2; R=runif(1);
    ET=1*B1+2*B2+1*B3*(R<(1-S/(K*N*1.5)))+2*B3*(R>(1-S/(K*N*1.5)));
     return(ET)}
#F8    event location
```

```
F8=function(C){ s=sum(C);D=(C<K); D=c(D,1);
 mn=(s<(K*N))*min(which(D==1)); C=C+c(.00001,0*(2:N));
  s1=sum(C);E=sample(1:N, 1,prob=C/s1);
EL=(ET==1)*mn+(ET==2)*E; return(EL)}
#F2    Configuration update (number of vehicles pere floor)
F2=function(C,ET,EL){C+1*((1:N)==EL)*(ET==1)-1*((1:N)==EL)*(ET==2) }
# F7    interevent time update
F7=function(C){sum=sum(C); B1=1*(sum==0);
B2=1*(sum==(N*K)); B3=1-B1-B2;
IT=B1*rexp(1,.5)+B2*rexp(1,(1/3))+B3*rexp(1,1/(2+3*sum/(N*K)));
 return(IT)}
#F9    elevator move distance
F9=function(EE,EL){abs(EE-EL)}
#F10   Elevator end positions
F10=function(ET,EL){(ET==1)*1+(ET==2)*EL}
for (i in 2:M)
{T=F1(T,IT); ET=F6(C); EL=F8(C); C=F2(C,ET,EL) ;
IT=F7(C); EM=F9(EE,EL); EE=F10(ET,EL);
A[i,]=c(T,C, ET,IT,EL, EM,EE)}
A
```

We look at 50 lines of typical output.

```
         [,1] [,2] [,3] [,4] [,5] [,6]      [,7] [,8] [,9] [,10]
[1,]  0.000000    0    0    0    0    1 1.4728057    1    0     1
[2,]  1.472806    1    0    0    0    1 1.7692348    1    0     1
[3,]  3.242040    2    0    0    0    1 0.5145856    1    0     1
[4,]  3.756626    3    0    0    0    1 1.3588579    1    0     1
[5,]  5.115484    4    0    0    0    1 2.8878279    1    0     1
```

```
[6,]   8.003312    5    0    0    0    1  3.5836669    1    0     1
[7,]  11.586979    5    1    0    0    1  4.1903493    2    1     1
[8,]  15.777328    5    2    0    0    1  1.6168756    2    1     1
[9,]  17.394204    5    3    0    0    1  5.1755735    2    1     1
[10,]  22.569777    5    2    0    0    2  6.9558115    2    1     2
[11,]  29.525589    5    3    0    0    1  6.2533741    2    0     1
[12,]  35.778963    5    4    0    0    1  0.2426509    2    1     1
[13,]  36.021614    5    5    0    0    1  1.8132013    2    1     1
[14,]  37.834815    5    5    1    0    1  4.4004731    3    2     1
[15,]  42.235288    4    5    1    0    2  2.1250158    1    0     1
[16,]  44.360304    5    5    1    0    1  3.8353506    1    0     1
[17,]  48.195655    5    4    1    0    2  1.9204525    2    1     2
[18,]  50.116107    5    3    1    0    2  0.4757502    2    0     2
[19,]  50.591857    5    4    1    0    1  1.6118420    2    0     1
[20,]  52.203699    5    5    1    0    1  8.4289138    2    1     1
[21,]  60.632613    5    5    2    0    1  2.4285073    3    2     1
[22,]  63.061120    5    5    3    0    1  5.8300665    3    2     1
[23,]  68.891187    5    5    2    0    2  8.6006187    3    2     3
[24,]  77.491805    5    5    3    0    1 11.2977932    3    0     1
[25,]  88.789599    5    5    4    0    1  1.7638560    3    2     1
[26,]  90.553455    5    5    5    0    1  2.8941195    3    2     1
[27,]  93.447574    5    5    5    1    1 12.4981289    4    3     1
[28,] 105.945703    5    4    5    1    2  1.1684248    2    1     2
[29,] 107.114128    5    5    5    1    1 19.5313561    2    0     1
[30,] 126.645484    5    5    5    0    2  2.8652371    4    3     4
[31,] 129.510721    4    5    5    0    2  0.4529262    1    3     1
[32,] 129.963647    4    5    4    0    2  0.3243088    3    2     3
```

```
[33,] 130.287956    5    5    4    0    1 13.0648844    1    2     1
[34,] 143.352840    5    4    4    0    2  1.5182460    2    1     2
[35,] 144.871086    5    5    4    0    1  1.3547604    2    0     1
[36,] 146.225847    5    5    5    0    1  0.2293114    3    2     1
[37,] 146.455158    5    4    5    0    2 10.5926918    2    1     2
[38,] 157.047850    5    5    5    0    1  1.1498218    2    0     1
[39,] 158.197672    5    5    5    1    1 11.7491841    4    3     1
[40,] 169.946856    5    5    5    2    1  3.8285380    4    3     1
[41,] 173.775394    5    4    5    2    2  5.1141779    2    1     2
[42,] 178.889572    5    4    5    1    2  4.9293930    4    2     4
[43,] 183.818965    5    5    5    1    1  5.6661687    2    2     1
[44,] 189.485134    5    5    5    2    1  0.4470715    4    3     1
[45,] 189.932205    4    5    5    2    2  5.3980902    1    0     1
[46,] 195.330295    5    5    5    2    1 11.1463773    1    0     1
[47,] 206.476673    5    5    5    1    2 14.0239537    4    3     4
[48,] 220.500626    5    5    4    1    2  3.3946964    3    1     3
[49,] 223.895323    5    5    5    1    1  0.6588260    3    0     1
[50,] 224.554149    5    5    5    2    1  6.3572581    4    3     1
```

An examination of the output seems to indicate that the program is performing as it should. At time 0, the number of customers on floors (1,2,3,4) is (0,0,0,0). The next event has to be an arrival and will take place 1.47 minutes later (col 7). The elevator is left at floor 1.

Row 2 (line 2) of the matrix shows the time of the first event (col 1). That event is an arrival (col 6) on floor 1 (col 8). Immediately after the event, the configuration is (1,0,0,0) for floors 1,2,3,4 as indicated in columns 2,3,4,5. The elevator position is floor 1 (col 8) after the first event. The elevator movement to respond to the first event is 0 (col 9). The elevator position after the first event is floor 0 (col 10). The time until the 2nd event is 1.769 minutes. (col 7).

Line 41 looks at the system 173.77 minutes after start time, when the 40th event occurs. That event is a departure(col 6) from floor 2 (col 8). Immediately after the 40th event, there were (5,4,5,2) vehicles parked on floors 1,2,3,4. It shows the elevator position (floor 2, col 8) after the 40th event. It indicates the elevator movement distance (1 floor, col 9) to respond to the 40th event. The 41st event will take place 5.114 minutes (col 7) later.

A few additional comments about the program should be made. Consider the part of the code

```
F6=function(C){S=sum(C);B1=(S==0); B2=(S==(N*K));
B3=1-B1-B2; R=runif(1);
ET=1*B1+2*B2+1*B3*(R<(1-S/(K*N*1.5)))+2*B3*(R>(1-S/(K*N*1.5)));
return(ET)}
```

If the system is full, then the next event must be a service completion (exiting vehicle). If the system is empty then the next event must be an arrival to the parking lot. If almost empty, the next event is an arrival with probability near 1. However, in other cases, if the system is almost full, then S is close to $N * K$ so $S/(K * N * 1.5)$ is close to 2/3. So the probability of an arrival is close to 1/3 and the probability of a departure is close to 2/3. We have chosen these values arbitrarily and we can adjust the system being modeled by choosing something other than 1.5. Arrival and service rates can also be adjusted through changes to the interevent time parameters.

We next loop the program 100 times, and get 100 mean values for column 9. These values are placed in groups of 10 and averaged, giving 10 averages. The Central Limit Theorem suggests that the 10 values (averages) should be approximately normally distributed. The looping is done by placing the code

```
s1=c(); for(i1=1:100){
```

at the beginning of the major program followed at the end by

```
s1=c(s1,mean(A[,9]))}
 S1=matrix(s1,10,10); v1=apply(S1,1,mean)
 v1
```

CHAPTER 5

Comparing strategies

We assume that there are 4 floors for parking cars. The first floor is ground level. Assume that if people park on floor 2 or 3 or 4, they call for the elevator to take them to ground level and they exit on foot. When they come to retrieve their car parked on level 2 or 3 or 4, they arrive on ground level, call for the elevator, and take it to appropriate level. We measure the distance for the elevator to move from its current position to the floor on which the elevator is called. The distance travelled also indicates the time needed for the elevator to respond to the call.

We can look at a variety of strategies as to where to place the elevator immediately (using 0 time) after its last use. We begin by considering 4 strategies.

Strategy 1: Elevator is moved to floor 1 after use, to wait for next call.

Strategy 2: Elevator is moved to floor 2 after use, to wait for next call.

Strategy 3: Elevator is moved to floor 3 after use, to wait for next call.

Strategy 4: Elevator stays where it was left on its last use.

The necessary code follows

```
#F10   Elevator end positions
#Strategy 1 Elevator moves to floor 1 after use
#      F10=function(ET,EL){1}
#Strategy 2 Elevator moves to floor 2 after use
#      F10=function(ET,EL){2}
#Strategy 3 Elevator moves to floor 3 after use
#      F10=function(ET,EL){3}
# Strategy 4 Elevator stays in its last position after use
#F10=function(ET,EL){(ET==1)*1+(ET==2)*EL}
```

In order to compare our different strategies, we assume a reasonable setting of $K = 25$ cars per floor. We also set M (the number of arrival/departure events considered for the day) to 200 to reflect the daily traffic. After some point in the day, arrivals will drop, and most events will be departures so the best elevator position at that time would be on floor 1.

Of course, we can set our parameters to any desired values and this cold have a large effect on the "best" strategy.

For each strategy, we loop the program 100 times, and get 100 mean values for column 9. These values are placed in groups of 10 and averaged, giving 10 averages. The Central Limit Theorem suggests that the 10 averages should be approximately normally distributed. The looping is done by placing the code

```
s1=c(); for(i1=1:100){
```

at the beginning of the major program followed at the end by

```
s1=c(s1,mean(A[,9]))}
S1=matrix(s1,10,10); v1=apply(S1,1,mean)
v1
```

Here are some values from the output of the R program in the previous chapter.

```
> x=c(x1,x2,x3,x4)
> y=c(rep(1,10),rep(2,10), rep(3,10), rep(4,10))
> boxplot(x~y)
> y=as.factor(y)
> df=data.frame(x,y)
> output=aov(x~y,df)
> summary(output)
Df Sum Sq Mean Sq F value Pr(>F)
```

```
y             3 1.8057  0.6019   580.3 <2e-16 ***
Residuals    36 0.0373  0.0010
```

We note that the Analysis of Variance result shows high significance (pvalue << .01).

Recall that the assumptions for ANOVA are that

(a) each group has values which are normally distributed

(b) the variances of each group are equal

However the distance that the elevator travels for strategy 2 is either 0 or 1, while the distance that the elevator travels for strategy 1 or strategy 3 is 0 or 1 or 2. This suggests that the variances may not be equal.

We next look at the box plots to see the nature of the differences and to check the variances.

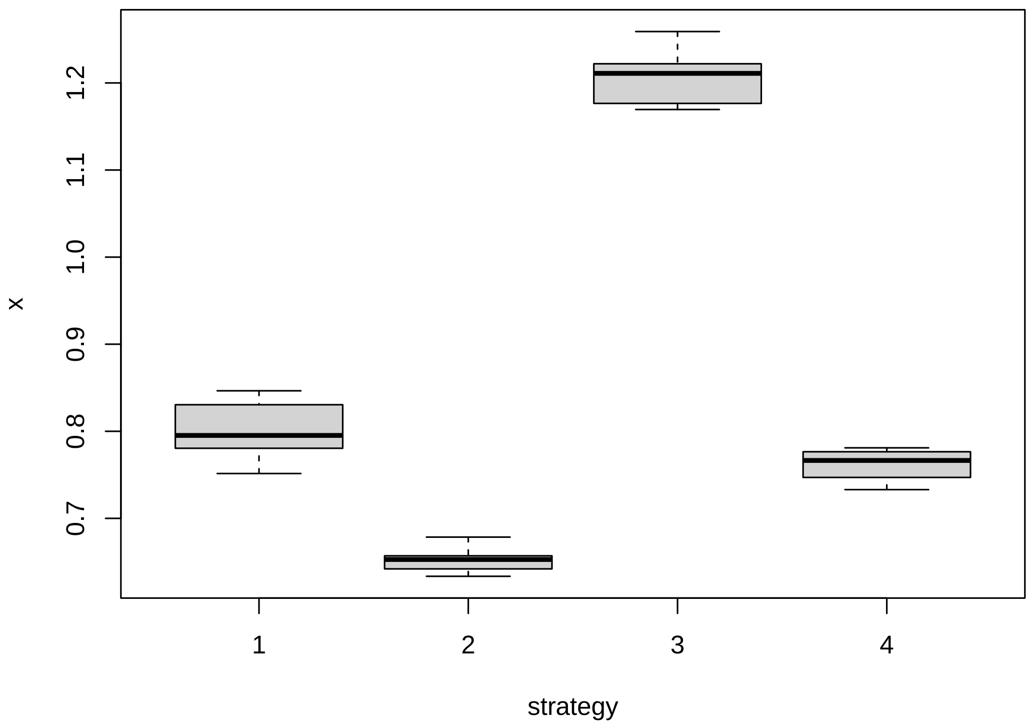

FIGURE 5.1. #moves vs Strategy

We also applied Kruskal Wallis nonparametric test on the same simulated data, with the same conclusion. The advantage of the Kruskal Wallis test is that we do not need the normality assumption. However, we still need the equal variance assumption.

Because the variances do not appear to be equal, we use Welch's one way ANOVA instead.

Results (from R) are:

```
> oneway.test(x~y, data = df, var.equal = FALSE)


One-way analysis of means (not assuming equal variances)


data:  x and y
F = 500.51, num df = 3.000, denom df =
19.027, p-value < 2.2e-16
```

Welch's test confirms highly significant differences among the 4 strategies. Repeated runs indicate that the best of the 4 strategies is strategy 2, which places the elevator on floor 2 after use.

CHAPTER 6

Hybrid strategies

Results in the previous chapter indicate that the strategy which always moves the elevator to floor 2 after each use, gives the lowest average move required for new uses. However, we should clearly be able to do better than that strategy, by keeping the elevator on floor 1 at the beginning stages, while the first floor of the parking garage is filling up.

So we propose hybrid strategy, which put the elevator on floor 1 for some initial amount of time, and then switches to placing the elevator on floor 2 after the initial time, after each use.

This brings us to an optimization question as to how long to maintain the initial floor 1 portion of the strategy. We will use another simulation run in order to compare the results, for our chosen parameters.

We propose 3 different switching points of time and compare with the base case of moving to floor 2 strategy immediately.

Our 4 strategies switch from floor 1 status to floor 2 status at times 0, 30, 60, 90 time units.

```
#F10   Elevator end positions

F10A=function(ET,EL){2}

F10B=function(ET,EL){2*(T>30)+1*(T<30)}

F10C=function(ET,EL){2*(T>60)+1*(T<60)}

F10D=function(ET,EL){2*(T>90)+1*(T<90)}
```

This time we used the same generated data for each strategy. The boxplots appear as follows.

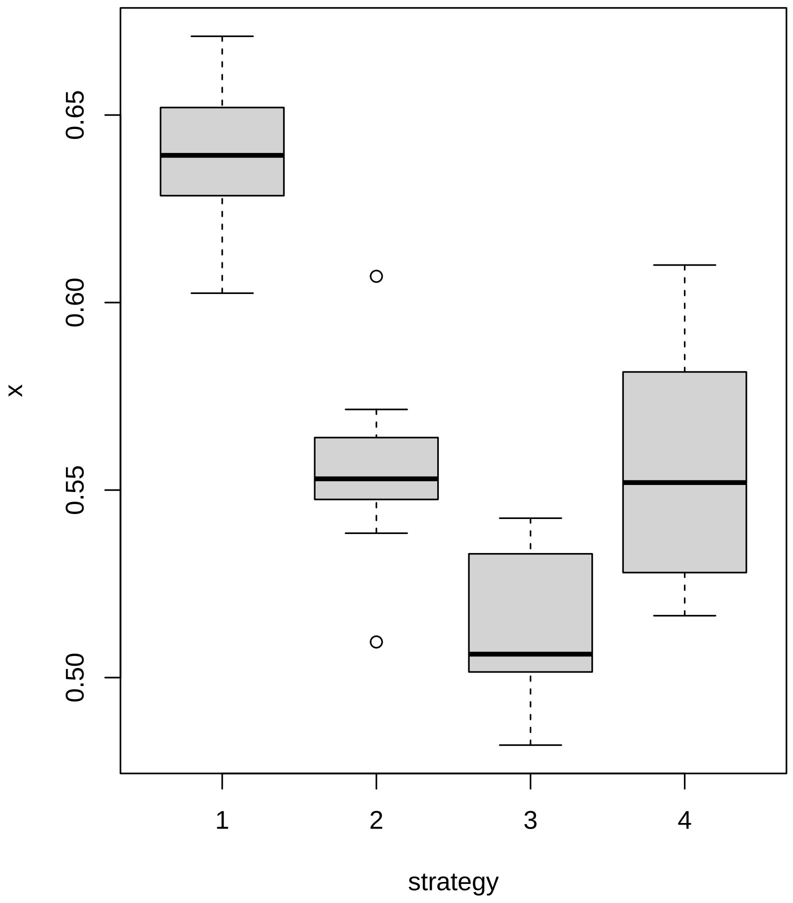

FIGURE 6.1. #moves vs Strategy

From this boxplot, we see that strategy 3 (switch after 60 time units) gives the best (lowest) average number of steps. We repeated the same experiment 10 times with newly generated data, and each time, strategy 3 was the best. We could get more precise results by searching among switching times near 60 (for example 50, 55, 60, 65 70). However, we have indicated our method, which could be used for a variety of parameters (number of floors, number of cars per floor, etc.).

CHAPTER 7

Conclusion and Future Work

We have given a method that allows us to compare strategies for a vehicle parking structure. Our focus was to minimize the time or distance that the elevator must travel once a call is made for the elevator. We made assumptions about interarrival times and about occupancy times, and assumed that an exponential distribution was appropriate.

We could make our methodology more general by relaxing the exponential assumptions. But the cost would be great since we would have to keep track of the time spent in each parking position at each time update.

A simpler modification to our assumptions would be to keep the exponential assumptions, but to allow the rates to change with time. For example, in the morning the arrival rate could be large but might decline later in the day.

Bibliography

[1] Ahmed, S., Shekha, M., Skran, S., Bassyouny, A. (2022). *Investigation of Optimization Techniques on the Elevator Dispatching Problem.* arXiv, 2022.13092v1, 13 pp.

[2] Al Sukkar, G., Al-Sharif, L., Mansour, M., Gharbieh, M., Farraj, E., Jarrah, R., Milekh, R., Zaben, N. (2017). Reconciling the value of the elevator round trip time between calculation and simulation. Simulation, 93(8), 707–722.

[3] Al-Sharif L. (2010). The effect of multiple entrances on the elevator round trip time under up-peak traffic. Mathematical and Computer Modelling, 52(3-4), 545–555.

[4] Al-Sharif, L. (2014). Calculating the Elevator Round Trip Time for the Most Basic of Cases (METE II). Lift Report, 40, 18–30.

[5] Al-Sharif, L., Algzawi, H. S., Hammodeh, A. T. (2013). The use of the Markov Chain Monte Carlo method in deriving the elevator round trip time under incoming traffic conditions and a single entrance. AMO–Advanced Modeling and Optimization, 15(3), 689–695.

[6] Al-Sharif, L., Alqumsan, A. M. A., Hammoudeh, A. T. (2014). Analytical, Numerical and Simulation: Six Methods for Evaluating the Elevator Round Trip Time. Conference Paper, Elevcon 2014, 62–73.

[7] Brand, M., Nikovski, D. (2004). Optimal parking in group elevator control. Proceedings of IEEE International Conference on Robotics and Automation 2004. ICRA'04. Vol. 1, 1002–1008.

[8] Crites, R., Barto, A. (1995). Improving elevator performance using reinforcement learning. NIPS'95: Proceedings of the 8th International Conference on Neural Information Processing Systems, 1017-–1023

[9] Das, T.K., Gosavi, A., Mahadevan, S., Marchalleck, N. (1999). Solving semi-Markov decision problems using average reward reinforcement learning. Management Science, 45(4), 560–574.

[10] Feng, Z., Redner, S. (2021). When will an elevator arrive?. Journal of Statistical Mechanics: Theory and Experiment. 2021, 043403, 23 pp.

[11] Hakonen, H., Siikonen, M. L. (2008). Elevator traffic simulation procedure. Elevator Technology, 17, 131–141.

[12] Jones, B. (1923). The probable number of stops made by an elevator. General Electric Review, 26(8) 583–587.

[13] S.P. Ladany and M. Hersh (1979). The design of an efficient elevator operating system. European Journal of Operational Research, 3(3), 216–221.

[14] Li, Z., Tan, H.-Z., Zhang, Y.-N., Mao, Z.-Y. (2007). Dynamic optimization of elevator group control based on artificial immune algorithm for inter-floor peak traffic during lunch-time. Control Theory and Applications, 24(2), 177–182.

[15] Mitric, S.. (1975). Elevator systems for tall buildings, Part I: Single mode elevator systems. Transportation Sci., 9, 54–73.

[16] Mitric, S. (1975). Elevator systems for tall buildings, Part II: Mixed mode elevator systems. Transportation Sci., 9, 74–85.

[17] Mulvaney, D., White, J., Hamdi, M. (2010). Elevator dispatching using heuristic search. Intelligent Automation and Soft Computing, 16(1), 77–87.

[18] Nikovski, D., Brand, M. (2004). Exact calculation of expected waiting times for group elevator control. IEEE transactions on automatic control, 49(10), 1820–1823.

[19] Patiño-Forero, A.A., Muñoz, D.M., de Carvalho, G.C., Llanos, C.H. (2012). Modeling of an elevator group control system using programmable logic control and destination control system. Proc. ABCM Symp. Ser. Mechatron, (Vol. 4), 433–441.

[20] Pepyne, D. L., Cassandras, C. G. (1998). Design and implementation of an adaptive dispatching controller for elevator systems during uppeak traffic. IEEE Transactions on Control Systems Technology, 6(5), 635–650.

[21] Powell, B.A. (1975). Elevator banking for high rise buildings. Transportation Sci., 9, 200–210.

[22] Siikonen, M.L. (1993). Elevator traffic simulation. Simulation, 61(4), 257–267.

[23] Sweet, A.L. and Duket, S.D. (1976). A simulation study of energy consumption by elevators in tall buildings. Computers and Industrial Engineering, 1, 3–11.

[24] Sun, J., Zhao, Q.C., Luh, P.B. (2009). Optimization of group elevator scheduling with advance information. IEEE transactions on automation science and engineering, 7(2), 352–363.

[25] Tanida, S. (2022). The synchronization of elevators when not all passengers will ride the first-arriving elevator. Physica A: Statistical Mechanics and its Applications, 605, 127957, 8 pp.

[26] Utgoff,P .E. and Connell, M.E. (2012). Real-time combinatorial optimization for elevator group dispatching. IEEE Transactions on Systems, Man, and Cybernetics Part A: Systems and Humans, 42(1), 130–146.

[27] Zhang, J., Tsiligkaridis, A., Taguchi, H., Raghunathan, A., Nikovski, D. (2022). Transformer Networks for Predictive Group Elevator Control. 2022 European Control Conference (ECC), 1429–1435.

Vita Auctoris

Naveed Arafat (he/him) was born in 1995 in Dhaka, Bangladesh. He received his undergraduate degree in Applied Statistics from East West University. He is completing the requirements for his MSc in Statistics in July, 2023 and formally receives his degree in October, 2023.