2012

Discrete Event Simulation of Elevator Systems

Sasi Bharath Desai
Claremont McKenna College

CLAREMONT McKENNA COLLEGE

DISCRETE EVENT SIMULATION OF ELEVATOR SYSTEMS

SUBMITTED TO

PROFESSOR MARK HUBER

AND

DEAN GREGORY HESS

BY

SASI BHARATH DESAI

FOR

SENIOR THESIS

SPRING 2012
23 APRIL 2012

Discrete Event Simulations of Elevator Systems

Sasi Desai

April 22, 2012

Abstract

The intent of this paper is to present the reader with a simple comparison of two systems of vertical transportation. Vertical transportation is a relatively new field and is the subject of much interest in today's world. As buildings get taller and real estate becomes more expensive, the need to find a quick, efficient system with a small footprint becomes important. By performing a simulation and subjecting the two systems under study to similar traffic conditions, one can determine the effectiveness of one system relative to the other. Additionally, we look at the effects of changing various system attributes to gain a better understanding of the primary drivers of average travel time.

1 Introduction

1.1 Past Work

Much work has been done in the field of vertical transportation in the last decade and a majority of this work has centered on control of elevators and simulation of elevator systems. "Control of Traffic Systems in Buildings" (Markon, Kise, Kita and Batrz-Beielstein), "Elevator Traffic Handbook" (Barney) and "The Vertical Transportation Handbook" (Strakosch and Caporale) are all comprehensive efforts in this direction.

In today's world where real estate in cities is becoming more expensive, the need to optimize elevator systems over a given area is of extreme economic significance. Several publications from the private sector address the problem of optimal car dispatching and passenger flow simulation [1]. While most research has tended to focus on systems with a single car per elevator shaft, some work has been done in simulating systems with multiple cars per shaft [2] [3].

Research has also been done in the field of optimizing elevator scheduling with advance information or using reinforced learning [4]. Most simulations however, have been conducted by using randomly generated data; the reason being the impracticality of using historical passenger data. Passenger data is difficult to collect and is influenced by the nature of the elevator system in use.

Research so far has tended to classify the passenger traffic into uppeak traffic, two-way traffic and downpeak traffic. The focus of the research has traditionally been on uppeak traffic for the sake of simplicity [5] [6]. Elevator control mechanisms used in these cases have been relatively straightforward; calls in the same direction as current car movement are answered before calls in the opposite direction [7]. While elevator systems in real life are a little more complicated, they rarely perform much better and as research has noted, the algorithms tend to be ad hoc and heuristic [6]. This paper uses a slightly modified version the basic elevator algorithm for a multi car system. Unlike most other papers, it addresses all three forms of traffic while maintaining its analytical focus on the results of the morning rush hour.

1.2 Problem

1.2.1 System 1

We are comparing two different kinds of systems that are functionally the same but are set up very differently. The first system should be familiar to most that have used elevators in high rise buildings while the second system is an innovative concept that this paper is trying to evaluate.

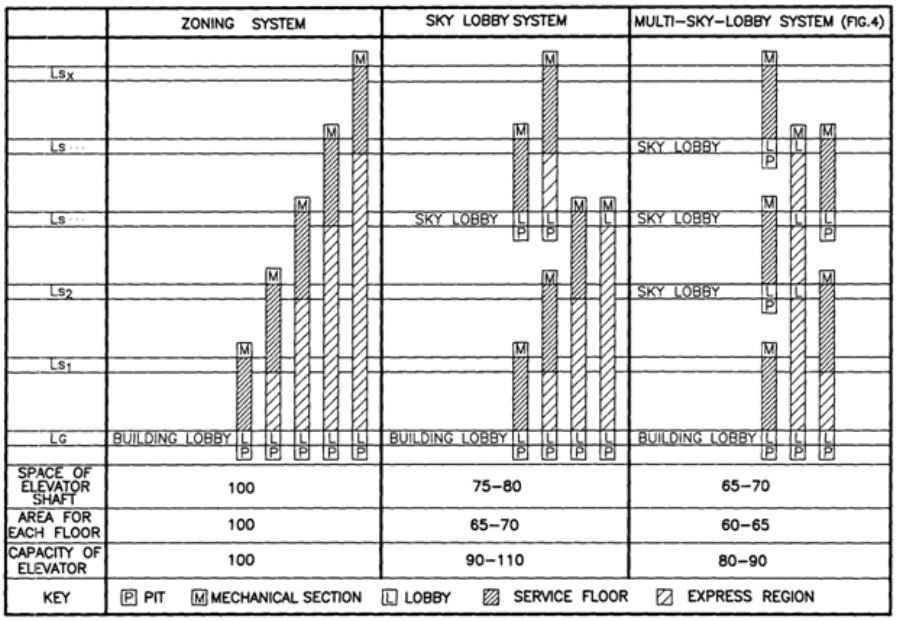


Figure 1: United States Patent 7198136; Elevator systems used by The Otis Company

Figure 1 depicts three commonly used systems by The Otis Elevator Company. System 1 (simulated in this paper) is the 'Zoning System' depicted on the left. This is a frequently used system in mid-high rise buildings. Passengers enter the zone that services their floor and if their destination is on a different zone, then they transfer between zones on landing floors.

In simulating this system, the job is made easier by the fact that every zone is completely independent of other zones. Passengers travelling between two zones can be treated as distinct passengers on each zone. Therefore, modelling the building's traffic boils down to modelling the individual zones.

1.2.2 System 2

Figure 2.1 depicts an existing system used for vertical transportation of cargo using a spiral escalator. The escalator has multiple landing floors at which cargo can be dropped off as required. Figures 2.1 and 2.2 show existing design ideas for a similar system to carry human beings. The basic idea behind System 2 is that by providing passengers with a spiral escalator that drops them off at various landing floors, the express region of each zone in System 1 can be eliminated. The new elevator system (shown in Figure 2.4 (compare to Figure 1: 'Zoning system')), is now more space efficient as the express regions have been eliminated and the elevators serving each zone have been 'stacked' in the same shaft.

The helical escalator will gradually accelerate passengers to speeds of over 1 m/s in the vertical direction (Figure 2.3). This escalator would wrap around the elevator shafts and service the landing floors where passengers can be dropped off so that they can access the elevators that service their zones.

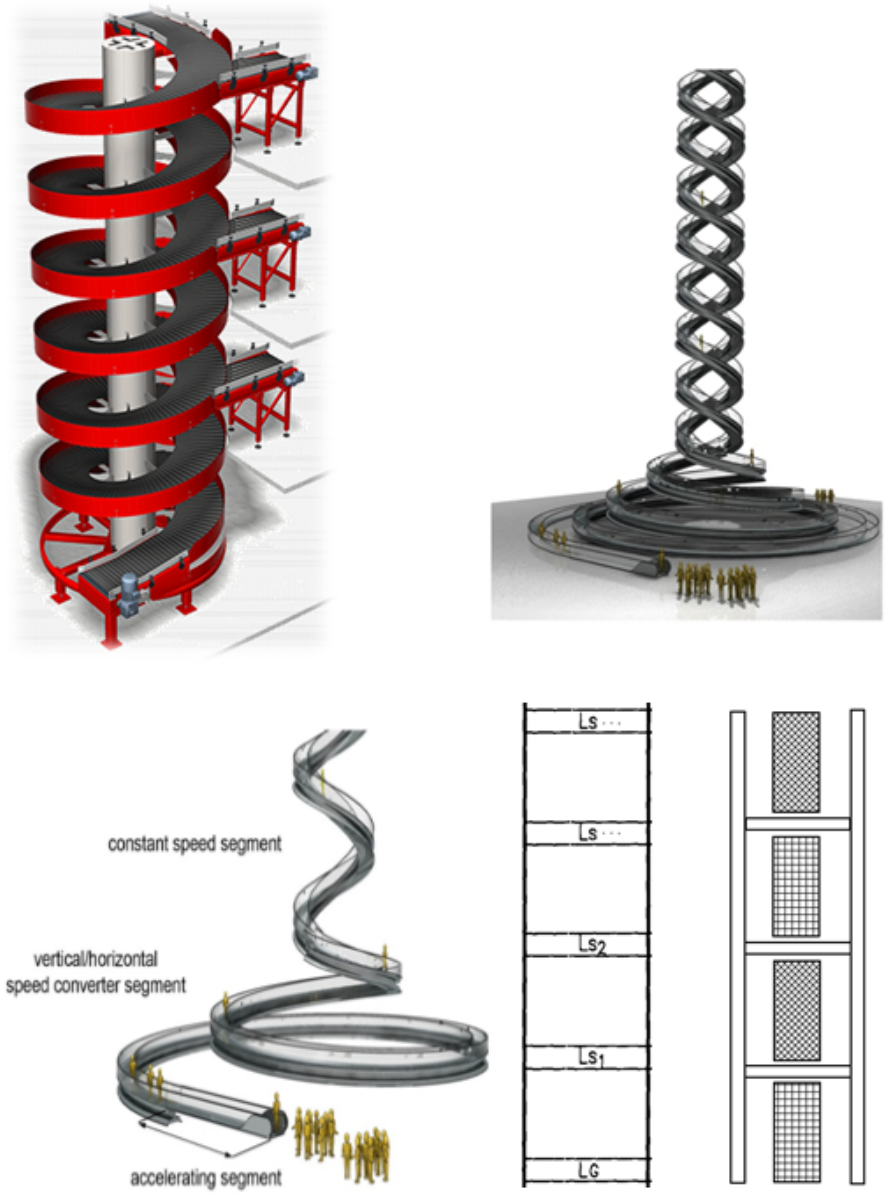


Figure 2: 2.1: Ryson Multi Entry Spiral, 2.2: Helixator - Artist Rendition, 2.3: Helixator - Concept, 2.4: System 2: All elevator zones are in the same shaft, surrounded by a Multi Entry Helixator (not depicted in 2.4)

1.3 Motivation

By providing an alternative manner in which to get to the landing floors of each zone, the 'express region' used in System 1 to take passengers up to the elevators' primary zone of operation is now freed up. The exact design and a study of the stability of such a system are beyond the scope of this paper. The economic gains of freeing up the real estate offer a financial incentive for adopting such a system if it proves to be effective.

Another potential advantage of using such a system is that unlike an elevator, an escalator has the ability to continuously service passengers. While the time to service each passenger may be slower (this can be overcome to a certain extent by the use of accelrating segments - as shown in Figure 2.3), it is very feasible that the gains obtained by eliminating wait time will more than compensate for the loss in speed. The rest of the paper will seek to quantitatively study the difference in total passenger travel time between the two systems.

2 Discrete Event Simulation (DES) Model

The DES model for an elevator system can be built by modelling the arrival of occupants as a Poisson process. The Poisson process assumes independence in arrivals which emulates reality well. The elevators are the servers in this case and the decision to assign particular servers to queue members is made at run time.

2.1 Model Description

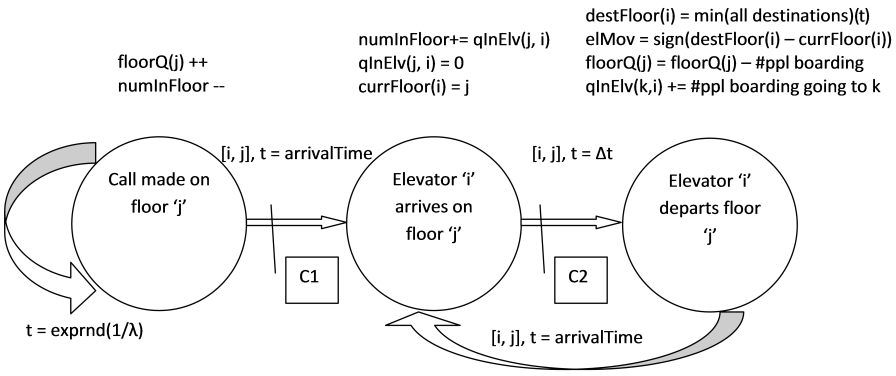


Figure 3: DES graph depicting a multi elevator system
C1 : Call the closest elevator that is stopped/moving in the same direction as the call as long as that call does not interrupt a current trip. Designate this elevator 'i'
C2 : If there is an existing call on elevator 'i' or if there are people waiting on floor 'j'

Figure 3 depicts the core of the DES this paper implements. There are three main types of events that the simulation processes;

- Passenger arrival: Arrivals are treated as a Poisson process with a rate parameter λ. Every arrival event seeds another arrival event. Once an arrival event is encountered, each floor's current population is used as the probability 'weight' for determining which floor the arrival occurs on. The higher the number of people on floor 'j', the more likely that the call originates on that floor.

- Elevator Arrival: Once a call has been made, an elevator arrives to service the passenger. The time at which the elevator arrives is determined by the formula;
  $arrivalTime =$
  $+ Start\ time$
  $+ numFloors\ to\ travel * Time\ per\ floor$
  $+ numViaStops * Time\ per\ stop$
  An arrival event may also be scheduled by a passenger seeking to go to a particular floor. Once an arrival event is processed, it releases any existing passengers scheduled to get off on that floor.

- Elevator Departure: After an arrival, if an elevator is scheduled to arrive elsewhere, a departure event is created and processed. In this step, any passenger queue on the floor 'j' is picked up and corresponding arrival events are scheduled. The arrival times are determined using a similar formula to the one used in the previous step.

2.2 System Variables

- t : current time
- nF : number of floors in the elevator zone
- nE : number of elevators servicing the zone
- N : number of residents in the elevator zone
- λ : arrival rate parameter
- elCap : capacity of elevator car
- CapNumInFloor : capacity of each floor to hold occupants

2.3 State Variables

- qInElv : A [nF x nE] matrix holding information about the number of people in elevator 'i' going to floor 'j'
- currFloor : A [2 x nE] matrix holding information about the most recent floor on which elevator 'i' made a stop and the time at which the stop was made
- destFloor : A [2 x nE] matrix holding information about the immediate destination of elevator 'i' and the time at which the stop is expected to be made
- elMov : A [1 x nE] matrix holding information about the current state (moving up/down/stopped) of elevator 'i'
- floorQ : A [nF x 2] matrix holding information about the number of people on floor 'j' waiting to go up/down
- numInFloor : A [nF x 1] matrix holding information about the current population of each floor

3 Simulation Code

3.1 Implementation of DES

The flowchart in Figure 4 should illustrate the code in Appendix B. It is designed to simulate the DES pictured in Figure 3. A couple of things stand out in this implementation. First, the program is designed to run for infinite time without termination. In the interest of analyzing results, we modify the code to stop adding passenger arrivals after the morning rush hour. Secondly, even though the code is generic, and does not distinguish between lobby traffic and inter-floor traffic or uppeak traffic and non-uppeak traffic, we can gain some insight into the efficiency of the systems by focusing on these particular kinds of traffic.

Both the systems we are interested in can be simulated with the same basic idea depicted in Figure 4. The only difference will arise in the determination of total wait times and arrival times for elevators which will change based upon the zone of service. The actual code makes some important assumptions that are worth noting here;

- Elevators travel with a constant velocity: While this is not true in real life (elevators accelerate through express regions to reach their zones), this assumption holds up well in the actual zones of operation. By using different velocities in zones of operation and the express regions, this acceleration component is captured and the simulation reflects reality surprisingly well.
- The zones are independent of each other: This has been discussed before, but it is worth repeating that all traffic can be captured by treating each zone independently. Inter-zone traffic is treated as distinct traffic in each zone of operation.
- Systems rules are obeyed: This would involve an umbrella of conditions. The most important ones are that elevator capacity rules are observed and standard elevator etiquette is practiced. An additional assumption is that nobody accidentally enters cars that they are not supposed to.
- Passenger traffic is a Poisson process with rate parameter λ. The probability of a person queuing up on floor 'j' is directly proportional to the current population of floor 'j' and the probability that a person's destination is floor 'j' is directly proportional to the remaining population capacity of floor 'j'.

3.2 Machine Characteristics

The simulation code (Appendix B) was primarily developed and tested on a HP dv5t-1000 64-bit Windows 7 machine with 4.00 GB RAM and an Intel Core 2 Duo CPU clocked at 2.40GHz. It was written in MATLAB v7.8 (R2009a Student Version) and should be compatible with more recent releases of the software package.

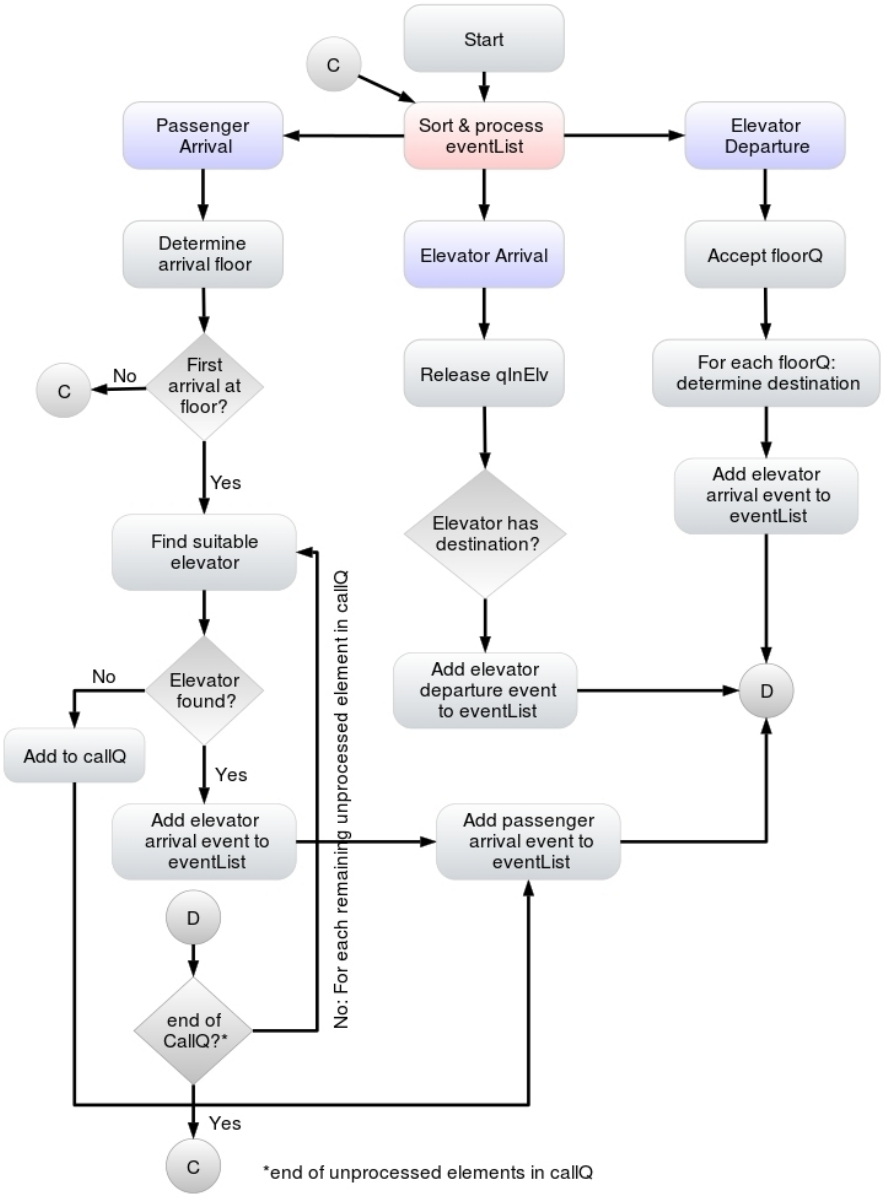


Figure 4: Flowchart depicting simulation code

4 Results

This paper's results section is primarily built around simulation of the elevator system of the KPMG tower (formerly the IBM tower) of the Wells Fargo Center on Bunker Hill in downtown Los Angeles. Located at 355 South Grand Avenue, this high rise building is the 16th tallest building in Los Angeles. It has 45 floors and is serviced by 24 elevators. For the purposes of our simulation, we will be dividing the building into three zones of fifteen floors and eight elevators each. As a preview to the monte carlo simulations performed on this system, the results of a single simulation on zone three are shown in Appendix A. The focus in the appendix is on Floor 34 and Elevator 1 as they are representative of the system in general.

We now compare the two systems on several different parameters and interpret the results after running the simulation several times. We start off by comparing the total travel time (wait + transit) for each zone under the two systems. To gain a better understanding of the system, we will be focusing on the traffic flow from 8:00 AM to 9:00 AM. The base parameters used are;

- Time the elevator takes to travel 1 floor = 1 sec
- Time the elevator takes to travel 1 floor in the express region = 0.5 sec
- Time the escalator takes to travel one floor in the express region = 1 sec
- Capacity of an elevator = 20 people
- Dwell time at each floor = 10 sec
- Number of passengers in the building = 3000 per zone * 3 zones
- Number of floors in the building = 15 per zone * 3 zones
- Number of elevators in the building = 8 per zone * 3 zones
- Capacity of each floor = 200 people
- Rate of arrival of passengers = Number of passengers in zone/3600 sec

Using these parameters, we run the elevator simulation 1000 times to get a distribution of the average total travel time (wait + transit) for a passenger under each system in each zone. Note that we are primarily interested in comparing total travel time given that we use a certain amount of real estate. In the elevator-only system, where we have 24 elevators, the total real estate used is 16 full length shafts (8*(1/3)*shafts for zone 1 + 8*(2/3)*shafts for zone 2 + 8 shafts for zone 3 - see Figure 1, 'Zoning System'). Thus in elevator-escalator system, even if we dedicate 2 full length shafts to escalator space, we have room for 14 more shafts. Since in the elevator-escalator system, we have 3 cars per shaft (see Figure 2.4), we can have up to 42 cars servicing this system. However, in this simulation we reduce the number of elevators in the escalator-elevator system to 18. This way, the 18 cars occupy 6 shafts taking the total real estate footprint of this system to 8 shafts - half the footprint of the elevator-only system. The rest of the results that follow are based on these assumptions. While interpreting results, it is important to note that elevator-escalator system is utilizing only half the real estate that the elevator-only system is.

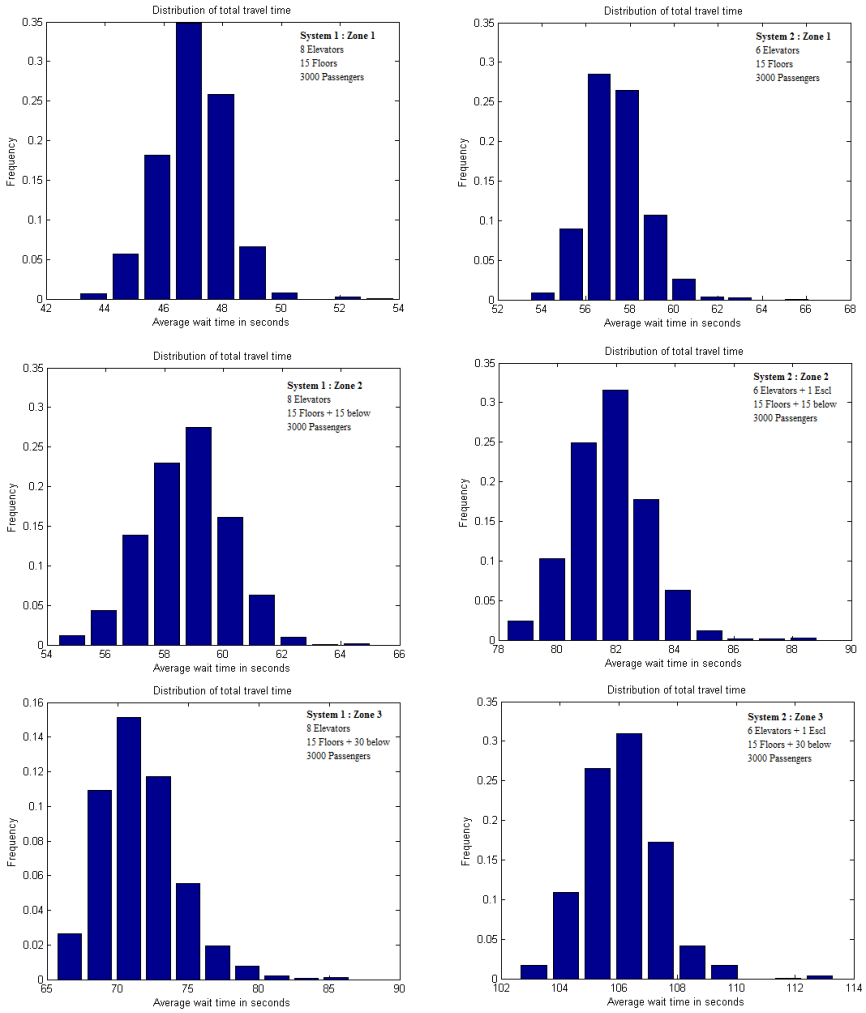
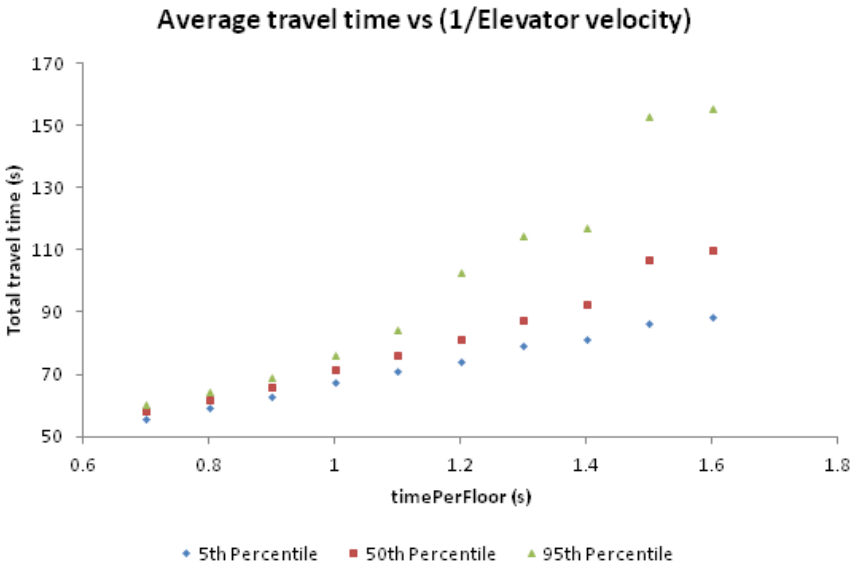
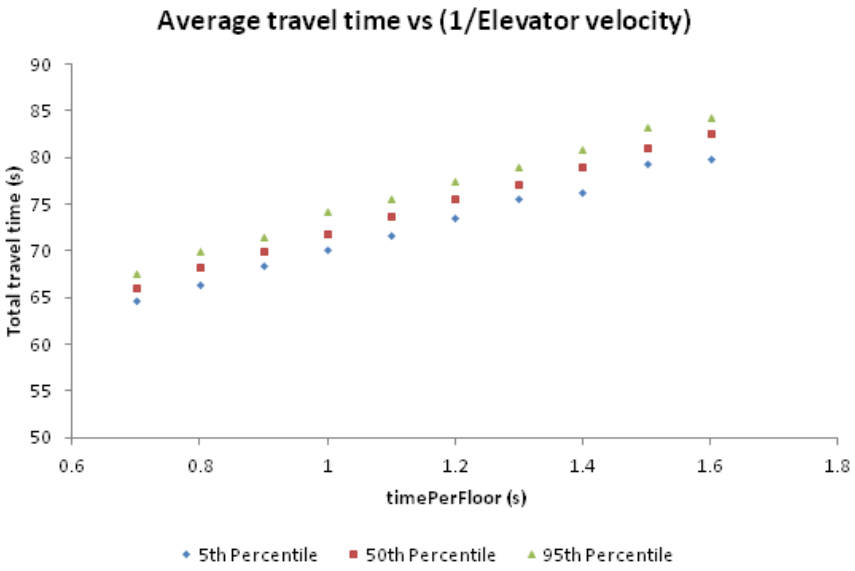


Figure 5: Monte Carlo Experiment to determine distribution of total travel times

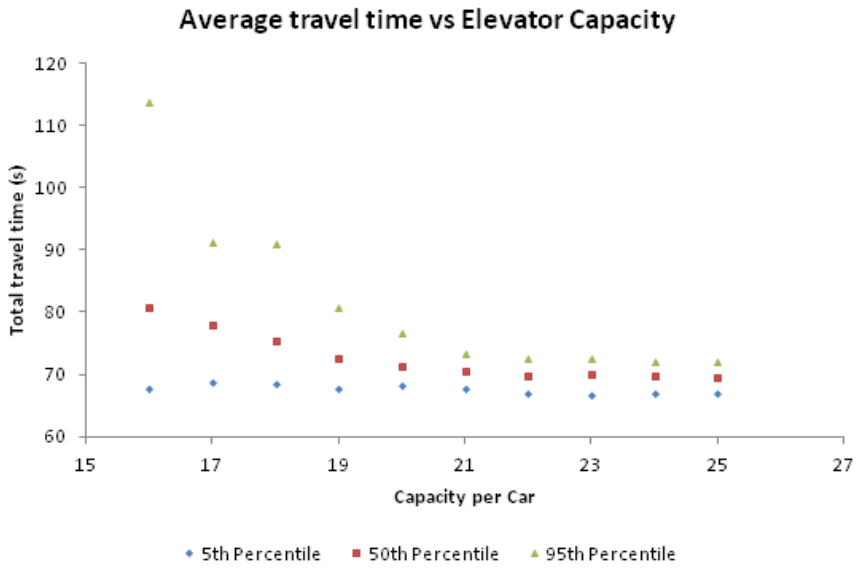
We also run simulations of each of these systems by changing individual parameters (from the base case) to see how the average travel time changes. As in the previous case, simulations are carried out a multiple times and the 95th, 50th and 5th percentiles are plotted.



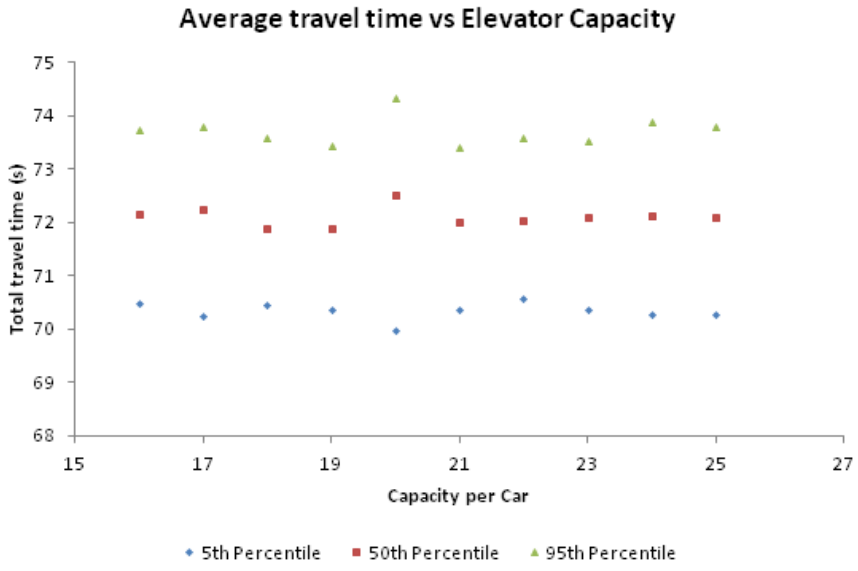
Elevator-Only system's performance against Elevator velocity



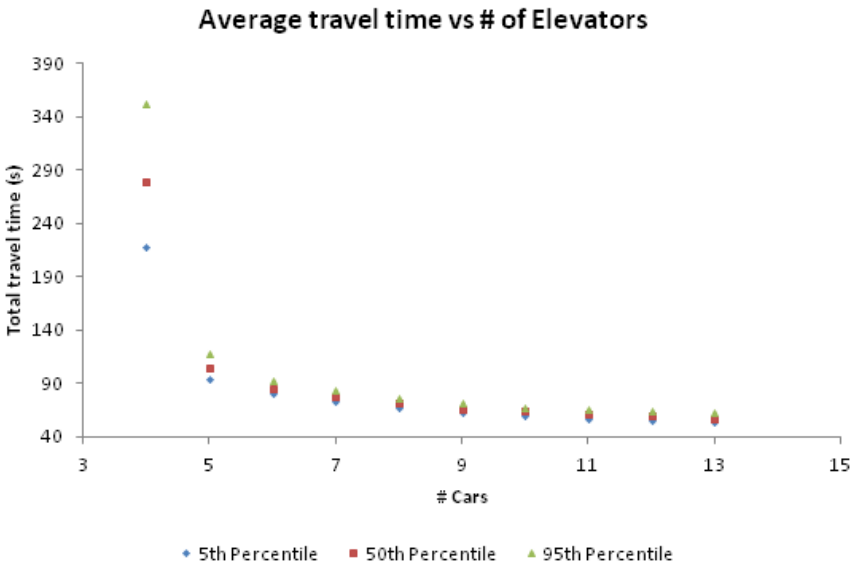
Elevator-Escalator system's performance against Elevator velocity



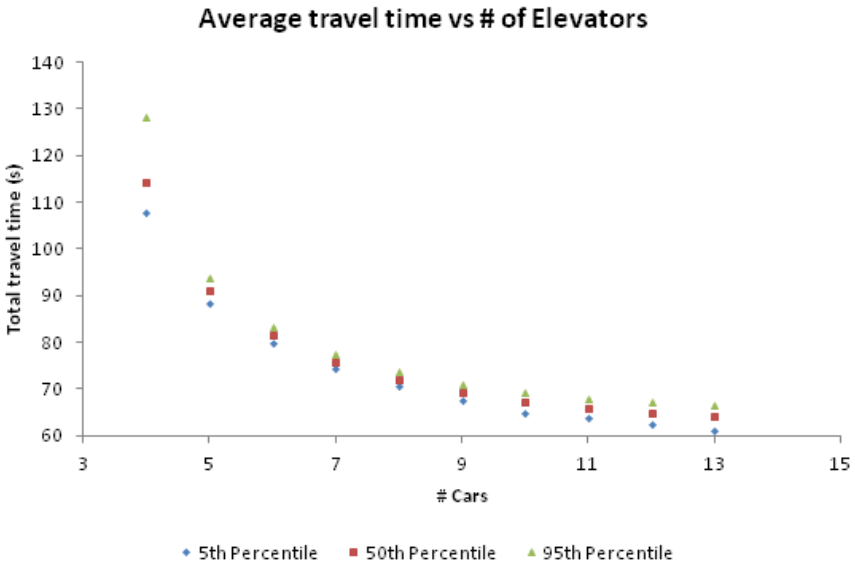
Elevator-Only system's performance against Elevator capacity



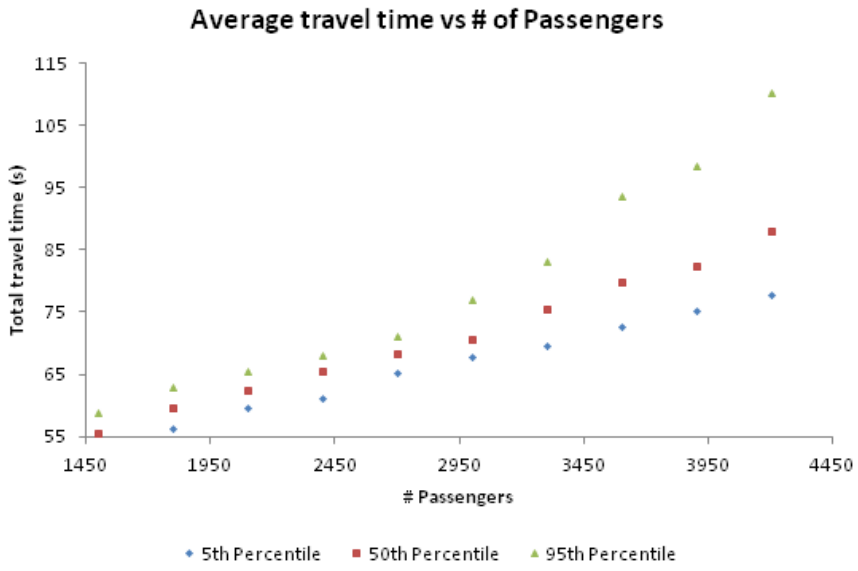
Elevator-Escalator system's performance against Elevator capacity



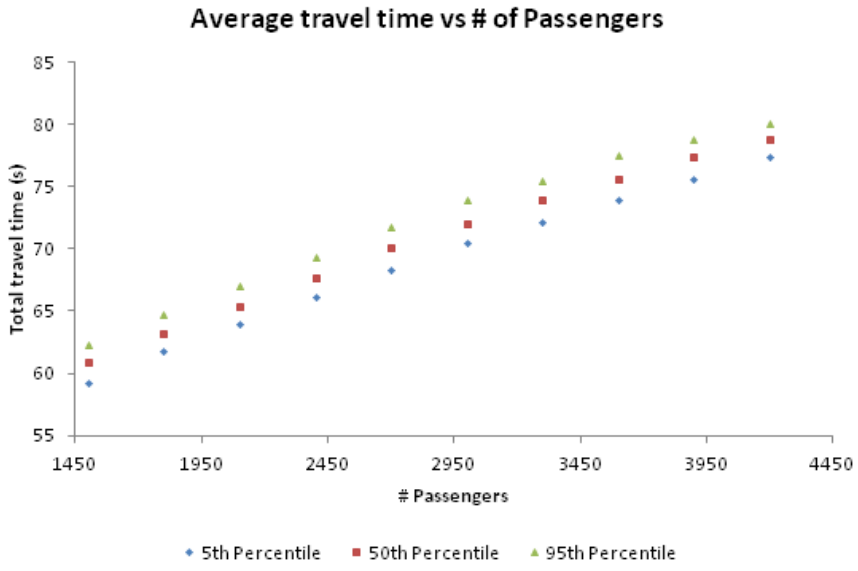
Elevator-Only system's performance against number of cars



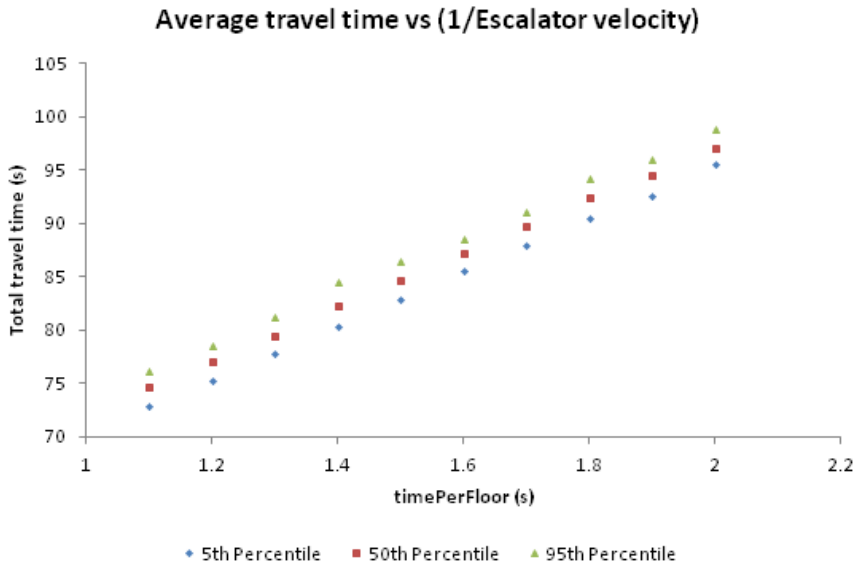
Elevator-Escalator system's performance against number of cars



Elevator-Only system's performance against number of passengers



Elevator-Escalator system's performance against number of passengers



Elevator-Escalator system's performance against escalator velocity

5 Conclusion

The results of the simulation are surprising, and yet in retrospect, intuitive and obvious. While one might have expected the elevator-only system to completely outperform the elevator-escalator system based purely upon the amount of real estate utilized, it appears that the elevator-escalator system performs only marginally worse than the elevator-only system. What's more, the former system actually performs better under stress. When the systems are exposed to extreme conditions such as increased traffic flow, or stalled cars, the escalator tends to have a moderating effect on the system. This is because the escalator is a continuous service system and the elevator-only system does not have access to this luxury. The extreme conditions imposed affect the elevator-only system not only in the operating zone, but also in the express region. These effects add up resulting in exponentially increasing travel times. Looking at the charts, other insights can be drawn. In particular, the following results are interesting;

- There is a critical elevator speed that must be maintained/achieved without which travel times increase exponentially in the elevator-only system.
- There is a critical elevator capacity that must be maintained/achieved without which travel times increase exponentially in the elevator-only system. On the flipside, beyond the critical elevator capacity, additional capacity does not seem to have an impact on the travel times.
- The number of functioning elevators is a critical parameter.
  - In both systems, if a couple of elevators break down, travel times increase rapidly. This problem seems particularly acute in the elevator-only system.
  - On the flip-side, there seems to be a sweet spot for the number of elevators beyond which the addition of elevators does not seem to reduce travel times. In the case of the KPMG tower, our simulations tell us that this number is eight which happens to be the number of elevators servicing each zone in the actual tower.
  - This number is particularly important because deviations from it are of extreme economic significance;
    - A lower number can increase travel times exponentially
    - A higher number means wasted real estate and lesser rents
- In line with the general trend, it appears that the number of passengers affects the elevator-only system more than the elevator-escalator system. The problem is particularly acute at the extreme cases (even as the median travel time tends to increase linearly, the 95th percentile increases exponentially).

In the end, it boils down finding the balance between rent and convenience that the building owners seek to obtain that would drive the decision to choose one system over the other.

6 Acknowledgements

I am grateful to several individuals for helping me complete this thesis. Most of all, I would like to thank my thesis reader, Professor Mark Huber for his guidance, help, support and encouragement during this entire process.

I would also like to thank my academic advisor, Professor James Higdon, whose constant counsel guided me through my four years of college. I will cherish my conversations with him for years to come.

I am grateful to all of my professors at Claremont McKenna College. This thesis is a culmination of four years of undergraduate education in which every single one of my professors played an extremely important role. In particular, I would like to thank Professor Mike ONeill and Professor Yaron Raviv for instilling in me a passion for their respective fields of study.

Finally, I would like to thank my family and friends for their constant support. I would like to especially thank my father for believing in me through my ups and downs, and my grandparents for their constant words of encouragement and concern for my well being.

My friends have always been a fantastic sounding board for my ideas and I would like to thank Aditya Bindal, Joshua Buss, Rajat Gupta, Matt Morton, Vishnu Narasimhan and George Posner for being excellent listeners and for helping me polish my ideas. My friends and neighbors on Auen 8th have been particularly kind to me and tolerant of my tendency to bore them with talk of elevator algorithms. They have been the perfect neighbors and fantastic friends. Finally, I would like to thank my best friend and confidant, Chandana Prakash, for her humor, support and encouragement.

References

[1] Marja-Liisa Siikonen, *Passenger Traffic Flow Simulation in Tall Buildings*, IFHS, International Conference on Multi-Purpose High-Rise Towers and Tall Buildings

[2] Sandor Markon, Hiromichi Suzuki, Kokolo Ikeda, Hajime Kita, *Direct Control of Multi-Car Elevators with Real-Time GA*

[3] Sara Pfenninger, *Simulation of a New Lift System*

[4] Jin Sun, Qian-Chuan Zhao, Peter B. Luh, *Optimization of Group Elevator Scheduling With Advance Information*, IEEE Transactions on Automation Science and Engineering, Vol 7, No 2, April 2010

[5] Richard Braun, *Need a Lift? An Elevator Queueing Problem*, UTRC, August 2003

[6] David L. Pepyne, Christos G. Cassandras, *Optimal Dispatching Control for Elevator Systems During Uppeak Traffic*, IEEE Transactions on Control Systems Technology, Vol 5, No 6, November 1997

[7] Marja-Liisa Siilkonen, *Elevator Traffic Simulation*, Society for Modelling and Simulation International (SCS)

A Simulation Results

Appendix A.1 and Appendix A.2 are extracted from a single run of the simulation code shown in Appendix B. They provide insight into events occurring on Floor 34 and in Elevator 1 respectively. The tables shown are subsets of the larger set of events occurring in the entire building. By focusing on these individual components of the system, we can better understand (and if needed, debug) the code behind the simulations.

In order to understand the tables better, the following terms need to be clarified;

- ArrivalReleases : This indicates that the elevator has arrived on the floor in question (in the case of A.1, Floor 34) and has released x number of passengers (where x is the number following *ArrivalReleases*)
- DepartureTakes : This indicates that the elevator has left the floor in question and has taken x number of passengers with it (where x is the number following *DepartureTakes*)
- FloorHas : Number of occupants of the current floor
- QUp : This indicates the number of people currently waiting to go up from the floor in question
- QDown : This indicates the number of people currently waiting to go down from the floor in question
- PassengerQueues : A single occupant from the current floor has decided to travel to another floor and has entered the queue
- NewCall : Usually follows a *PassengerQueues* event. This indicates that the passenger entering the queue is the first person to do so and an elevator has to be called (*InformElev*)
- StatElev : Usually follows a *NewCall* event. This means that a stationary elevator has been found that can take this call. The elevator number follows in the next column
- MovElev : Usually follows a *NewCall* event. This means that a moving elevator has been found that can take this call. The elevator number follows in the next column
- InFloor/JustInFloor : The most recent floor on which the *StatElev*/*MovElev* was located
- XY_ServiceCall : This is to indicate that a call has been made to the elevator in question. XY changes depending on the nature of the called elevator (*MovElev* = MV, *StatElev* on same floor = SF, *StatElev* on different floor = DF)
- Qgoto : This is followed by a *DepartureTakes* event. *Qgoto* determines the destination floor of a single passenger who has embarked on the elevator in question to go to another floor.

A.1 Floor 34

t	=	28839	:	Elevator	2	:	ArrivalReleases	1	:	FloorHas	1							
t	=	28849	:	Elevator	2	:	DepartureTakes	0	:	Floorhas	1	:	QUp	0	:	QDown	0	
t	=	28961	:	Elevator	2	:	ArrivalReleases	2	:	FloorHas	3							
t	=	28971	:	Elevator	2	:	DepartureTakes	0	:	Floorhas	3	:	QUp	0	:	QDown	0	
t	=	29004	:	Elevator	1	:	ArrivalReleases	4	:	FloorHas	7							
t	=	29014	:	Elevator	1	:	DepartureTakes	0	:	Floorhas	7	:	QUp	0	:	QDown	0	
t	=	29064	:	Elevator	5	:	ArrivalReleases	3	:	FloorHas	10							
t	=	29074	:	Elevator	5	:	DepartureTakes	0	:	Floorhas	10	:	QUp	0	:	QDown	0	
t	=	29096	:	Elevator	4	:	ArrivalReleases	3	:	FloorHas	13							
t	=	29106	:	Elevator	4	:	DepartureTakes	0	:	Floorhas	13	:	QUp	0	:	QDown	0	
t	=	29148	:	Elevator	2	:	ArrivalReleases	3	:	FloorHas	16							
t	=	29158	:	Elevator	2	:	DepartureTakes	0	:	Floorhas	16	:	QUp	0	:	QDown	0	
t	=	29177	:	Elevator	1	:	ArrivalReleases	2	:	FloorHas	18							
t	=	29187	:	Elevator	1	:	DepartureTakes	0	:	Floorhas	18	:	QUp	0	:	QDown	0	
t	=	29226	:	Elevator	3	:	ArrivalReleases	2	:	FloorHas	20							
t	=	29236	:	Elevator	3	:	DepartureTakes	0	:	Floorhas	20	:	QUp	0	:	QDown	0	
t	=	29264	:	Elevator	4	:	ArrivalReleases	1	:	FloorHas	21							
t	=	29274	:	Elevator	4	:	DepartureTakes	0	:	Floorhas	21	:	QUp	0	:	QDown	0	
t	=	29291	:	Elevator	5	:	ArrivalReleases	1	:	FloorHas	22							
t	=	29291	:				PassengerQueues			FloorHas	21	:	QUp	1	:	QDown	0	
t	=	29291	:				NewCall			InformElev								
t	=	29291	:	StatElev	1	:	InFloor	45	:	DF_ServiceCall								
t	=	29301	:	Elevator	5	:	Qgoto	35	:	dir	1							
t	=	29301	:	Elevator	5	:	DepartureTakes	1	:	Floorhas	21	:	QUp	0	:	QDown	0	
t	=	29302	:	Elevator	1	:	ArrivalReleases	0	:	FloorHas	21							
t	=	29374	:	Elevator	6	:	ArrivalReleases	1	:	FloorHas	22							
t	=	29384	:	Elevator	6	:	DepartureTakes	0	:	Floorhas	22	:	QUp	0	:	QDown	0	
t	=	29441	:	Elevator	1	:	ArrivalReleases	4	:	FloorHas	26							
t	=	29451	:	Elevator	1	:	DepartureTakes	0	:	Floorhas	26	:	QUp	0	:	QDown	0	
t	=	29480	:	Elevator	3	:	ArrivalReleases	2	:	FloorHas	28							
t	=	29490	:	Elevator	3	:	DepartureTakes	0	:	Floorhas	28	:	QUp	0	:	QDown	0	
t	=	29512	:				PassengerQueues			FloorHas	27	:	QUp	1	:	QDown	0	
t	=	29512	:				NewCall			InformElev								
t	=	29512	:	MovElev	4	:	JustInFloor	32	:	MV_ServiceCall								
t	=	29523	:				PassengerQueues			FloorHas	26	:	QUp	2	:	QDown	0	
t	=	29533	:	Elevator	4	:	ArrivalReleases	0	:	FloorHas	26							
t	=	29540	:	Elevator	6	:	ArrivalReleases	5	:	FloorHas	31							
t	=	29543	:	Elevator	4	:	Qgoto	36	:	dir	1							
t	=	29543	:	Elevator	4	:	Qgoto	35	:	dir	1							
t	=	29543	:	Elevator	4	:	DepartureTakes	2	:	Floorhas	31	:	QUp	0	:	QDown	0	
t	=	29550	:	Elevator	6	:	DepartureTakes	0	:	Floorhas	31	:	QUp	0	:	QDown	0	
t	=	29573	:				PassengerQueues			FloorHas	30	:	QUp	1	:	QDown	0	
t	=	29573	:				NewCall			InformElev								
t	=	29573	:	MovElev	5	:	JustInFloor	1	:	MV_ServiceCall								
t	=	29597	:	Elevator	5	:	ArrivalReleases	0	:	FloorHas	30							
t	=	29601	:	Elevator	7	:	ArrivalReleases	1	:	FloorHas	31							

t	=	29605	:	Elevator	1	:	ArrivalReleases	1	:	FloorHas	32								
t	=	29607	:	Elevator	5	:	Qgoto	37	:	dir	1								
t	=	29607	:	Elevator	5	:	DepartureTakes	1	:	Floorhas	32	:	QUp	0	:	QDown	0		
t	=	29611	:	Elevator	7	:	DepartureTakes	0	:	Floorhas	32	:	QUp	0	:	QDown	0		
t	=	29613	:				PassengerQueues			FloorHas	31	:	QUp	1	:	QDown	0		
t	=	29613	:				NewCall			InformElev									
t	=	29613	:	StatElev	2	:	InFloor	45	:	DF_ServiceCall									
t	=	29615	:	Elevator	1	:	Qgoto	45	:	dir	1								
t	=	29615	:	Elevator	1	:	DepartureTakes	1	:	Floorhas	31	:	QUp	0	:	QDown	0		
t	=	29624	:	Elevator	2	:	ArrivalReleases	0	:	FloorHas	31								
t	=	29656	:	Elevator	3	:	ArrivalReleases	1	:	FloorHas	32								
t	=	29666	:	Elevator	3	:	DepartureTakes	0	:	Floorhas	32	:	QUp	0	:	QDown	0		
t	=	29701	:	Elevator	4	:	ArrivalReleases	2	:	FloorHas	34								
t	=	29711	:	Elevator	4	:	DepartureTakes	0	:	Floorhas	34	:	QUp	0	:	QDown	0		
t	=	29740	:	Elevator	1	:	ArrivalReleases	3	:	FloorHas	37								
t	=	29750	:	Elevator	1	:	DepartureTakes	0	:	Floorhas	37	:	QUp	0	:	QDown	0		
t	=	29766	:	Elevator	2	:	ArrivalReleases	4	:	FloorHas	41								
t	=	29776	:	Elevator	2	:	DepartureTakes	0	:	Floorhas	41	:	QUp	0	:	QDown	0		
t	=	29802	:	Elevator	5	:	ArrivalReleases	2	:	FloorHas	43								
t	=	29812	:	Elevator	5	:	DepartureTakes	0	:	Floorhas	43	:	QUp	0	:	QDown	0		
t	=	29812	:				PassengerQueues			FloorHas	42	:	QUp	1	:	QDown	0		
t	=	29812	:				NewCall			InformElev									
t	=	29812	:	MovElev	3	:	JustInFloor	1	:	MV_ServiceCall									
t	=	29849	:	Elevator	3	:	ArrivalReleases	0	:	FloorHas	42								
t	=	29859	:	Elevator	3	:	Qgoto	45	:	dir	1								
t	=	29859	:	Elevator	3	:	DepartureTakes	1	:	Floorhas	42	:	QUp	0	:	QDown	0		
t	=	29924	:	Elevator	1	:	ArrivalReleases	2	:	FloorHas	44								
t	=	29934	:	Elevator	1	:	DepartureTakes	0	:	Floorhas	44	:	QUp	0	:	QDown	0		
t	=	29953	:	Elevator	8	:	ArrivalReleases	2	:	FloorHas	46								
t	=	29963	:	Elevator	8	:	DepartureTakes	0	:	Floorhas	46	:	QUp	0	:	QDown	0		
t	=	29972	:				PassengerQueues			FloorHas	45	:	QUp	1	:	QDown	0		
t	=	29972	:				NewCall			InformElev									
t	=	29972	:	MovElev	2	:	JustInFloor	1	:	MV_ServiceCall									
t	=	30005	:	Elevator	2	:	ArrivalReleases	3	:	FloorHas	48								
t	=	30015	:	Elevator	2	:	Qgoto	39	:	dir	1								
t	=	30015	:	Elevator	2	:	DepartureTakes	1	:	Floorhas	48	:	QUp	0	:	QDown	0		

A.2 Elevator 1

t	=	28801	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2999	:	ElevHas	0										
t	=	28811	:	Floor	1	:	Qgoto	37	:															
t	=	28811	:	Floor	1	:	Qgoto	36	:															
t	=	28811	:	Floor	1	:	Qgoto	42	:															
t	=	28811	:	Floor	1	:	Qgoto	38	:															
t	=	28811	:	Floor	1	:	Qgoto	39	:															
t	=	28811	:	Floor	1	:	Qgoto	39	:															
t	=	28811	:	Floor	1	:	Qgoto	40	:															
t	=	28811	:	Floor	1	:	DepartureTakes	7	:	Floorhas	2993	:	ElevHas	7	:	QUp	0	:	QDown	0				
t	=	28831	:	Floor	36	:	ArrivalReleases	1	:	FloorHas	1	:	ElevHas	6										
t	=	28841	:	Floor	36	:	DepartureTakes	0	:	Floorhas	1	:	ElevHas	6	:	QUp	0	:	QDown	0				
t	=	28842	:	Floor	37	:	ArrivalReleases	1	:	FloorHas	1	:	ElevHas	5										
t	=	28852	:	Floor	37	:	DepartureTakes	0	:	Floorhas	1	:	ElevHas	5	:	QUp	0	:	QDown	0				
t	=	28853	:	Floor	38	:	ArrivalReleases	1	:	FloorHas	1	:	ElevHas	4										
t	=	28863	:	Floor	38	:	DepartureTakes	0	:	Floorhas	1	:	ElevHas	4	:	QUp	0	:	QDown	0				
t	=	28864	:	Floor	39	:	ArrivalReleases	2	:	FloorHas	3	:	ElevHas	2										
t	=	28874	:	Floor	39	:	DepartureTakes	0	:	Floorhas	3	:	ElevHas	2	:	QUp	0	:	QDown	0				
t	=	28875	:	Floor	40	:	ArrivalReleases	1	:	FloorHas	2	:	ElevHas	1										
t	=	28885	:	Floor	40	:	DepartureTakes	0	:	Floorhas	2	:	ElevHas	1	:	QUp	0	:	QDown	0				
t	=	28887	:	Floor	42	:	ArrivalReleases	1	:	FloorHas	2	:	ElevHas	0										
t	=	28919	:	Floor	39	:	ArrivalReleases	0	:	FloorHas	8	:	ElevHas	0										
t	=	28956	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2913	:	ElevHas	0										
t	=	28966	:	Floor	1	:	Qgoto	34	:															
t	=	28966	:	Floor	1	:	Qgoto	37	:															
t	=	28966	:	Floor	1	:	Qgoto	34	:															
t	=	28966	:	Floor	1	:	Qgoto	39	:															
t	=	28966	:	Floor	1	:	Qgoto	34	:															
t	=	28966	:	Floor	1	:	Qgoto	33	:															
t	=	28966	:	Floor	1	:	Qgoto	39	:															
t	=	28966	:	Floor	1	:	Qgoto	34	:															
t	=	28966	:	Floor	1	:	Qgoto	33	:															
t	=	28966	:	Floor	1	:	Qgoto	37	:															
t	=	28966	:	Floor	1	:	Qgoto	41	:															
t	=	28966	:	Floor	1	:	Qgoto	43	:															
t	=	28966	:	Floor	1	:	Qgoto	39	:															
t	=	28966	:	Floor	1	:	Qgoto	37	:															
t	=	28966	:	Floor	1	:	Qgoto	38	:															
t	=	28966	:	Floor	1	:	Qgoto	32	:															
t	=	28966	:	Floor	1	:	Qgoto	42	:															
t	=	28966	:	Floor	1	:	Qgoto	45	:															
t	=	28966	:	Floor	1	:	Qgoto	32	:															
t	=	28966	:	Floor	1	:	DepartureTakes	20	:	Floorhas	2909	:	ElevHas	20	:	QUp	1	:	QDown	0				
t	=	28982	:	Floor	32	:	ArrivalReleases	2	:	FloorHas	6	:	ElevHas	18										
t	=	28992	:	Floor	32	:	DepartureTakes	0	:	Floorhas	6	:	ElevHas	18	:	QUp	0	:	QDown	0				
t	=	28993	:	Floor	33	:	ArrivalReleases	2	:	FloorHas	6	:	ElevHas	16										

t	=	29003	:	Floor	33	:	DepartureTakes	0	:	Floorhas	6	:	ElevHas	16	:	QUp	0	:	QDown	0
t	=	29004	:	Floor	34	:	ArrivalReleases	4	:	FloorHas	7	:	ElevHas	12						
t	=	29014	:	Floor	34	:	DepartureTakes	0	:	Floorhas	7	:	ElevHas	12	:	QUp	0	:	QDown	0
t	=	29017	:	Floor	37	:	ArrivalReleases	4	:	FloorHas	12	:	ElevHas	8						
t	=	29027	:	Floor	37	:	DepartureTakes	0	:	Floorhas	12	:	ElevHas	8	:	QUp	0	:	QDown	0
t	=	29028	:	Floor	38	:	ArrivalReleases	1	:	FloorHas	4	:	ElevHas	7						
t	=	29038	:	Floor	38	:	DepartureTakes	0	:	Floorhas	4	:	ElevHas	7	:	QUp	0	:	QDown	0
t	=	29039	:	Floor	39	:	ArrivalReleases	3	:	FloorHas	13	:	ElevHas	4						
t	=	29049	:	Floor	39	:	DepartureTakes	0	:	Floorhas	13	:	ElevHas	4	:	QUp	0	:	QDown	0
t	=	29051	:	Floor	41	:	ArrivalReleases	1	:	FloorHas	7	:	ElevHas	3						
t	=	29061	:	Floor	41	:	DepartureTakes	0	:	Floorhas	7	:	ElevHas	3	:	QUp	0	:	QDown	0
t	=	29062	:	Floor	42	:	ArrivalReleases	1	:	FloorHas	4	:	ElevHas	2						
t	=	29072	:	Floor	42	:	DepartureTakes	0	:	Floorhas	4	:	ElevHas	2	:	QUp	0	:	QDown	0
t	=	29073	:	Floor	43	:	ArrivalReleases	1	:	FloorHas	4	:	ElevHas	1						
t	=	29083	:	Floor	43	:	DepartureTakes	0	:	Floorhas	4	:	ElevHas	1	:	QUp	0	:	QDown	0
t	=	29085	:	Floor	45	:	ArrivalReleases	1	:	FloorHas	5	:	ElevHas	0						
t	=	29139	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2824	:	ElevHas	0						
t	=	29149	:	Floor	1	:	Qgoto	44	:											
t	=	29149	:	Floor	1	:	Qgoto	36	:											
t	=	29149	:	Floor	1	:	Qgoto	42	:											
t	=	29149	:	Floor	1	:	Qgoto	32	:											
t	=	29149	:	Floor	1	:	Qgoto	44	:											
t	=	29149	:	Floor	1	:	Qgoto	38	:											
t	=	29149	:	Floor	1	:	Qgoto	39	:											
t	=	29149	:	Floor	1	:	Qgoto	35	:											
t	=	29149	:	Floor	1	:	Qgoto	37	:											
t	=	29149	:	Floor	1	:	Qgoto	41	:											
t	=	29149	:	Floor	1	:	Qgoto	35	:											
t	=	29149	:	Floor	1	:	Qgoto	34	:											
t	=	29149	:	Floor	1	:	Qgoto	39	:											
t	=	29149	:	Floor	1	:	Qgoto	44	:											
t	=	29149	:	Floor	1	:	Qgoto	34	:											
t	=	29149	:	Floor	1	:	Qgoto	45	:											
t	=	29149	:	Floor	1	:	Qgoto	42	:											
t	=	29149	:	Floor	1	:	Qgoto	37	:											
t	=	29149	:	Floor	1	:	Qgoto	35	:											
t	=	29149	:	Floor	1	:	DepartureTakes	19	:	Floorhas	2820	:	ElevHas	19	:	QUp	0	:	QDown	0
t	=	29165	:	Floor	32	:	ArrivalReleases	1	:	FloorHas	14	:	ElevHas	18						
t	=	29175	:	Floor	32	:	DepartureTakes	0	:	Floorhas	14	:	ElevHas	18	:	QUp	0	:	QDown	0
t	=	29177	:	Floor	34	:	ArrivalReleases	2	:	FloorHas	18	:	ElevHas	16						
t	=	29187	:	Floor	34	:	DepartureTakes	0	:	Floorhas	18	:	ElevHas	16	:	QUp	0	:	QDown	0
t	=	29188	:	Floor	35	:	ArrivalReleases	3	:	FloorHas	14	:	ElevHas	13						
t	=	29198	:	Floor	35	:	DepartureTakes	0	:	Floorhas	14	:	ElevHas	13	:	QUp	0	:	QDown	0
t	=	29199	:	Floor	36	:	ArrivalReleases	1	:	FloorHas	10	:	ElevHas	12						
t	=	29209	:	Floor	36	:	DepartureTakes	0	:	Floorhas	10	:	ElevHas	12	:	QUp	0	:	QDown	0
t	=	29210	:	Floor	37	:	ArrivalReleases	2	:	FloorHas	21	:	ElevHas	10						
t	=	29220	:	Floor	37	:	Qgoto	39	:											
t	=	29220	:	Floor	37	:	DepartureTakes	1	:	Floorhas	20	:	ElevHas	11	:	QUp	0	:	QDown	0
t	=	29221	:	Floor	38	:	ArrivalReleases	1	:	FloorHas	7	:	ElevHas	10						

t	=	29231	:	Floor	38	:	DepartureTakes	0	:	Floorhas	7	:	ElevHas	10	:	QUp	0	:	QDown	0
t	=	29232	:	Floor	39	:	ArrivalReleases	3	:	FloorHas	20	:	ElevHas	7						
t	=	29242	:	Floor	39	:	Qgoto	42	:											
t	=	29242	:	Floor	39	:	DepartureTakes	1	:	Floorhas	20	:	ElevHas	8	:	QUp	0	:	QDown	0
t	=	29244	:	Floor	41	:	ArrivalReleases	1	:	FloorHas	12	:	ElevHas	7						
t	=	29254	:	Floor	41	:	DepartureTakes	0	:	Floorhas	12	:	ElevHas	7	:	QUp	0	:	QDown	0
t	=	29255	:	Floor	42	:	ArrivalReleases	3	:	FloorHas	11	:	ElevHas	4						
t	=	29265	:	Floor	42	:	DepartureTakes	0	:	Floorhas	11	:	ElevHas	4	:	QUp	0	:	QDown	0
t	=	29267	:	Floor	44	:	ArrivalReleases	3	:	FloorHas	9	:	ElevHas	1						
t	=	29277	:	Floor	44	:	DepartureTakes	0	:	Floorhas	9	:	ElevHas	1	:	QUp	0	:	QDown	0
t	=	29278	:	Floor	45	:	ArrivalReleases	1	:	FloorHas	14	:	ElevHas	0						
t	=	29302	:	Floor	34	:	ArrivalReleases	0	:	FloorHas	21	:	ElevHas	0						
t	=	29333	:	Floor	39	:	ArrivalReleases	0	:	FloorHas	24	:	ElevHas	0						
t	=	29393	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2682	:	ElevHas	0						
t	=	29403	:	Floor	1	:	Qgoto	37	:											
t	=	29403	:	Floor	1	:	Qgoto	38	:											
t	=	29403	:	Floor	1	:	Qgoto	34	:											
t	=	29403	:	Floor	1	:	Qgoto	32	:											
t	=	29403	:	Floor	1	:	Qgoto	41	:											
t	=	29403	:	Floor	1	:	Qgoto	43	:											
t	=	29403	:	Floor	1	:	Qgoto	37	:											
t	=	29403	:	Floor	1	:	Qgoto	34	:											
t	=	29403	:	Floor	1	:	Qgoto	43	:											
t	=	29403	:	Floor	1	:	Qgoto	34	:											
t	=	29403	:	Floor	1	:	Qgoto	40	:											
t	=	29403	:	Floor	1	:	Qgoto	43	:											
t	=	29403	:	Floor	1	:	Qgoto	39	:											
t	=	29403	:	Floor	1	:	Qgoto	32	:											
t	=	29403	:	Floor	1	:	Qgoto	42	:											
t	=	29403	:	Floor	1	:	Qgoto	34	:											
t	=	29403	:	Floor	1	:	Qgoto	33	:											
t	=	29403	:	Floor	1	:	Qgoto	42	:											
t	=	29403	:	Floor	1	:	Qgoto	36	:											
t	=	29403	:	Floor	1	:	Qgoto	41	:											
t	=	29403	:	Floor	1	:	DepartureTakes	20	:	Floorhas	2676	:	ElevHas	20	:	QUp	4	:	QDown	0
t	=	29419	:	Floor	32	:	ArrivalReleases	2	:	FloorHas	27	:	ElevHas	18						
t	=	29429	:	Floor	32	:	DepartureTakes	0	:	Floorhas	27	:	ElevHas	18	:	QUp	0	:	QDown	0
t	=	29430	:	Floor	33	:	ArrivalReleases	1	:	FloorHas	21	:	ElevHas	17						
t	=	29440	:	Floor	33	:	DepartureTakes	0	:	Floorhas	21	:	ElevHas	17	:	QUp	0	:	QDown	0
t	=	29441	:	Floor	34	:	ArrivalReleases	4	:	FloorHas	26	:	ElevHas	13						
t	=	29451	:	Floor	34	:	DepartureTakes	0	:	Floorhas	26	:	ElevHas	13	:	QUp	0	:	QDown	0
t	=	29453	:	Floor	36	:	ArrivalReleases	1	:	FloorHas	21	:	ElevHas	12						
t	=	29463	:	Floor	36	:	DepartureTakes	0	:	Floorhas	21	:	ElevHas	12	:	QUp	0	:	QDown	0
t	=	29464	:	Floor	37	:	ArrivalReleases	2	:	FloorHas	23	:	ElevHas	10						
t	=	29474	:	Floor	37	:	DepartureTakes	0	:	Floorhas	23	:	ElevHas	10	:	QUp	0	:	QDown	0
t	=	29475	:	Floor	38	:	ArrivalReleases	1	:	FloorHas	15	:	ElevHas	9						
t	=	29485	:	Floor	38	:	DepartureTakes	0	:	Floorhas	15	:	ElevHas	9	:	QUp	0	:	QDown	0
t	=	29486	:	Floor	39	:	ArrivalReleases	1	:	FloorHas	28	:	ElevHas	8						
t	=	29496	:	Floor	39	:	DepartureTakes	0	:	Floorhas	28	:	ElevHas	8	:	QUp	0	:	QDown	0

t	=	29497	:	Floor	40	:	ArrivalReleases	1	:	FloorHas	26	:	ElevHas	7										
t	=	29507	:	Floor	40	:	DepartureTakes	0	:	Floorhas	26	:	ElevHas	7	:	QUp	0	:	QDown	0				
t	=	29508	:	Floor	41	:	ArrivalReleases	2	:	FloorHas	20	:	ElevHas	5										
t	=	29518	:	Floor	41	:	DepartureTakes	0	:	Floorhas	20	:	ElevHas	5	:	QUp	0	:	QDown	0				
t	=	29519	:	Floor	42	:	ArrivalReleases	2	:	FloorHas	23	:	ElevHas	3										
t	=	29529	:	Floor	42	:	DepartureTakes	0	:	Floorhas	23	:	ElevHas	3	:	QUp	0	:	QDown	0				
t	=	29530	:	Floor	43	:	ArrivalReleases	3	:	FloorHas	21	:	ElevHas	0										
t	=	29567	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2601	:	ElevHas	0										
t	=	29577	:	Floor	1	:	Qgoto	35	:															
t	=	29577	:	Floor	1	:	Qgoto	33	:															
t	=	29577	:	Floor	1	:	Qgoto	34	:															
t	=	29577	:	Floor	1	:	DepartureTakes	3	:	Floorhas	2597	:	ElevHas	3	:	QUp	0	:	QDown	0				
t	=	29594	:	Floor	33	:	ArrivalReleases	1	:	FloorHas	31	:	ElevHas	2										
t	=	29604	:	Floor	33	:	DepartureTakes	0	:	Floorhas	31	:	ElevHas	2	:	QUp	0	:	QDown	0				
t	=	29605	:	Floor	34	:	ArrivalReleases	1	:	FloorHas	32	:	ElevHas	1										
t	=	29615	:	Floor	34	:	Qgoto	45	:															
t	=	29615	:	Floor	34	:	DepartureTakes	1	:	Floorhas	31	:	ElevHas	2	:	QUp	0	:	QDown	0				
t	=	29616	:	Floor	35	:	ArrivalReleases	1	:	FloorHas	31	:	ElevHas	1										
t	=	29626	:	Floor	35	:	DepartureTakes	0	:	Floorhas	31	:	ElevHas	1	:	QUp	0	:	QDown	0				
t	=	29633	:	Floor	42	:	ArrivalReleases	0	:	FloorHas	27	:	ElevHas	1										
t	=	29643	:	Floor	42	:	DepartureTakes	0	:	Floorhas	28	:	ElevHas	1	:	QUp	0	:	QDown	0				
t	=	29646	:	Floor	45	:	ArrivalReleases	1	:	FloorHas	37	:	ElevHas	0										
t	=	29692	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2541	:	ElevHas	0										
t	=	29702	:	Floor	1	:	Qgoto	41	:															
t	=	29702	:	Floor	1	:	Qgoto	34	:															
t	=	29702	:	Floor	1	:	Qgoto	43	:															
t	=	29702	:	Floor	1	:	Qgoto	33	:															
t	=	29702	:	Floor	1	:	Qgoto	40	:															
t	=	29702	:	Floor	1	:	Qgoto	40	:															
t	=	29702	:	Floor	1	:	Qgoto	36	:															
t	=	29702	:	Floor	1	:	Qgoto	42	:															
t	=	29702	:	Floor	1	:	Qgoto	42	:															
t	=	29702	:	Floor	1	:	Qgoto	42	:															
t	=	29702	:	Floor	1	:	Qgoto	35	:															
t	=	29702	:	Floor	1	:	Qgoto	34	:															
t	=	29702	:	Floor	1	:	Qgoto	40	:															
t	=	29702	:	Floor	1	:	Qgoto	39	:															
t	=	29702	:	Floor	1	:	Qgoto	40	:															
t	=	29702	:	Floor	1	:	Qgoto	33	:															
t	=	29702	:	Floor	1	:	Qgoto	34	:															
t	=	29702	:	Floor	1	:	Qgoto	32	:															
t	=	29702	:	Floor	1	:	Qgoto	36	:															
t	=	29702	:	Floor	1	:	Qgoto	35	:															
t	=	29702	:	Floor	1	:	DepartureTakes	20	:	Floorhas	2538	:	ElevHas	20	:	QUp	1	:	QDown	0				
t	=	29718	:	Floor	32	:	ArrivalReleases	1	:	FloorHas	36	:	ElevHas	19										
t	=	29728	:	Floor	32	:	DepartureTakes	0	:	Floorhas	36	:	ElevHas	19	:	QUp	0	:	QDown	0				
t	=	29729	:	Floor	33	:	ArrivalReleases	2	:	FloorHas	40	:	ElevHas	17										
t	=	29739	:	Floor	33	:	DepartureTakes	0	:	Floorhas	40	:	ElevHas	17	:	QUp	0	:	QDown	0				
t	=	29740	:	Floor	34	:	ArrivalReleases	3	:	FloorHas	37	:	ElevHas	14										

t	=	29750	:	Floor	34	:	DepartureTakes	0	:	Floorhas	37	:	ElevHas	14	:	QUp	0	:	QDown	0
t	=	29751	:	Floor	35	:	ArrivalReleases	2	:	FloorHas	35	:	ElevHas	12						
t	=	29761	:	Floor	35	:	DepartureTakes	0	:	Floorhas	35	:	ElevHas	12	:	QUp	0	:	QDown	0
t	=	29762	:	Floor	36	:	ArrivalReleases	2	:	FloorHas	32	:	ElevHas	10						
t	=	29772	:	Floor	36	:	DepartureTakes	0	:	Floorhas	32	:	ElevHas	10	:	QUp	0	:	QDown	0
t	=	29775	:	Floor	39	:	ArrivalReleases	1	:	FloorHas	26	:	ElevHas	9						
t	=	29785	:	Floor	39	:	DepartureTakes	0	:	Floorhas	26	:	ElevHas	9	:	QUp	0	:	QDown	0
t	=	29786	:	Floor	40	:	ArrivalReleases	4	:	FloorHas	34	:	ElevHas	5						
t	=	29796	:	Floor	40	:	DepartureTakes	0	:	Floorhas	34	:	ElevHas	5	:	QUp	0	:	QDown	0
t	=	29797	:	Floor	41	:	ArrivalReleases	1	:	FloorHas	33	:	ElevHas	4						
t	=	29807	:	Floor	41	:	DepartureTakes	0	:	Floorhas	33	:	ElevHas	4	:	QUp	0	:	QDown	0
t	=	29808	:	Floor	42	:	ArrivalReleases	3	:	FloorHas	35	:	ElevHas	1						
t	=	29818	:	Floor	42	:	DepartureTakes	0	:	Floorhas	35	:	ElevHas	1	:	QUp	0	:	QDown	0
t	=	29819	:	Floor	43	:	ArrivalReleases	1	:	FloorHas	30	:	ElevHas	0						
t	=	29876	:	Floor	1	:	ArrivalReleases	0	:	FloorHas	2440	:	ElevHas	0						
t	=	29886	:	Floor	1	:	Qgoto	40	:											
t	=	29886	:	Floor	1	:	Qgoto	39	:											
t	=	29886	:	Floor	1	:	Qgoto	35	:											
t	=	29886	:	Floor	1	:	Qgoto	45	:											
t	=	29886	:	Floor	1	:	Qgoto	36	:											
t	=	29886	:	Floor	1	:	Qgoto	45	:											
t	=	29886	:	Floor	1	:	Qgoto	34	:											
t	=	29886	:	Floor	1	:	Qgoto	44	:											
t	=	29886	:	Floor	1	:	Qgoto	32	:											
t	=	29886	:	Floor	1	:	Qgoto	32	:											
t	=	29886	:	Floor	1	:	Qgoto	33	:											
t	=	29886	:	Floor	1	:	Qgoto	38	:											
t	=	29886	:	Floor	1	:	Qgoto	38	:											
t	=	29886	:	Floor	1	:	Qgoto	45	:											
t	=	29886	:	Floor	1	:	Qgoto	34	:											
t	=	29886	:	Floor	1	:	Qgoto	44	:											
t	=	29886	:	Floor	1	:	Qgoto	40	:											
t	=	29886	:	Floor	1	:	Qgoto	38	:											
t	=	29886	:	Floor	1	:	Qgoto	42	:											
t	=	29886	:	Floor	1	:	Qgoto	41	:											
t	=	29886	:	Floor	1	:	DepartureTakes	20	:	Floorhas	2436	:	ElevHas	20	:	QUp	3	:	QDown	0
t	=	29902	:	Floor	32	:	ArrivalReleases	2	:	FloorHas	40	:	ElevHas	18						
t	=	29912	:	Floor	32	:	DepartureTakes	0	:	Floorhas	40	:	ElevHas	18	:	QUp	0	:	QDown	0
t	=	29913	:	Floor	33	:	ArrivalReleases	1	:	FloorHas	44	:	ElevHas	17						
t	=	29923	:	Floor	33	:	DepartureTakes	0	:	Floorhas	44	:	ElevHas	17	:	QUp	0	:	QDown	0
t	=	29924	:	Floor	34	:	ArrivalReleases	2	:	FloorHas	44	:	ElevHas	15						
t	=	29934	:	Floor	34	:	DepartureTakes	0	:	Floorhas	44	:	ElevHas	15	:	QUp	0	:	QDown	0
t	=	29935	:	Floor	35	:	ArrivalReleases	1	:	FloorHas	40	:	ElevHas	14						
t	=	29945	:	Floor	35	:	Qgoto	38	:											
t	=	29945	:	Floor	35	:	DepartureTakes	1	:	Floorhas	40	:	ElevHas	15	:	QUp	0	:	QDown	0
t	=	29946	:	Floor	36	:	ArrivalReleases	1	:	FloorHas	38	:	ElevHas	14						
t	=	29956	:	Floor	36	:	Qgoto	43	:											
t	=	29956	:	Floor	36	:	DepartureTakes	1	:	Floorhas	38	:	ElevHas	15	:	QUp	0	:	QDown	0
t	=	29958	:	Floor	38	:	ArrivalReleases	4	:	FloorHas	32	:	ElevHas	11						

t	=	29968	:	Floor	38	:	DepartureTakes	0	:	Floorhas	32	:	ElevHas	11	:	QUp	0	:	QDown	0
t	=	29969	:	Floor	39	:	ArrivalReleases	1	:	FloorHas	33	:	ElevHas	10						
t	=	29979	:	Floor	39	:	Qgoto	43	:											
t	=	29979	:	Floor	39	:	DepartureTakes	1	:	Floorhas	33	:	ElevHas	11	:	QUp	0	:	QDown	0
t	=	29980	:	Floor	40	:	ArrivalReleases	2	:	FloorHas	40	:	ElevHas	9						
t	=	29990	:	Floor	40	:	DepartureTakes	0	:	Floorhas	40	:	ElevHas	9	:	QUp	0	:	QDown	0
t	=	29991	:	Floor	41	:	ArrivalReleases	1	:	FloorHas	42	:	ElevHas	8						
t	=	30001	:	Floor	41	:	DepartureTakes	0	:	Floorhas	42	:	ElevHas	8	:	QUp	0	:	QDown	0
t	=	30002	:	Floor	42	:	ArrivalReleases	1	:	FloorHas	41	:	ElevHas	7						
t	=	30012	:	Floor	42	:	DepartureTakes	0	:	Floorhas	41	:	ElevHas	7	:	QUp	0	:	QDown	0
t	=	30013	:	Floor	43	:	ArrivalReleases	2	:	FloorHas	38	:	ElevHas	5						
t	=	30023	:	Floor	43	:	DepartureTakes	0	:	Floorhas	38	:	ElevHas	5	:	QUp	0	:	QDown	0
t	=	30024	:	Floor	44	:	ArrivalReleases	2	:	FloorHas	36	:	ElevHas	3						
t	=	30034	:	Floor	44	:	DepartureTakes	0	:	Floorhas	36	:	ElevHas	3	:	QUp	0	:	QDown	0
t	=	30035	:	Floor	45	:	ArrivalReleases	3	:	FloorHas	52	:	ElevHas	0						

B Simulation Code

```
function workingElevatorOnlySystem(nF, nE, nP, skip)
%% Simulates an elevator system for a specified period of time
% Function/Parameter Description nF = # of floors in the system nE = # of
% elevators in the system N  = # of people who work in that system skip = #
% of floors below that system

%%  Declaring State Variables

global t;
global totalWait;
global N;
global jump;
global timeAtFloor;
global timePerFloor;
global elCap;
global qInElv;
global currFloor;
global destFloor;
global elMov;
global floorQ;
global numInFloor;
global CapNumInFloor;
global eventList;
global event;
global callQ;


t = 28800;
% t = current time

totalWait = 0;
% This keeps track of the total time that all passengers have spent waiting

N = nP;
jump = skip;

timeAtFloor = 10;
% Every stop is 10 seconds. This is a bit rigid, but will serve well for
% the purposes of this simulation

timePerFloor = 1;
% While in motion, it travels at a rate of 1 floor/1 seconds

elCap = 20;
% Elevator capacity

qInElv = zeros(nF, nE);
% qInElv holds information about the number of people in elevator 'i' (nE)
% going to floor 'j' (nF). At the beginning, it is a zero matrix.

currFloor = ones(2, nE);
% currFloor holds information about the current floor 'j' that elevator 'i'
```

```
% is on. At the beginning, it is a zero matrix. The 2nd row keeps track of
% the last time spent at the current floor
currFloor(2,:) = t;

destFloor = ones(2, nE);
% destFloor keps information about the current destnation of elevator 'i'
% the 2nd row keeps information about the anticipated arrival time at
% destination floor
destFloor(2,:) = t;

elMov = zeros(1, nE);
% elMov keeps track of whether elavator 'i' is moving up(1), down(-1), or
% is still(0)

floorQ = zeros(nF, 2);
% floorQ holds information on the number of people on floor 'j' waiting to
% go up or down (column 1 = going up, column 2 - going down). At the
% beginning, it is a zero matrix.

numInFloor = zeros(nF, 1);
% numInFloor holds information about the number of people on floor 'j'.
% This information will be used to calculate the probability distribution
% of people leaving/arriving at floor 'j'
numInFloor(1) = N; % At the start of the day, they are all on floor 1.
% capacity of number of people per floor
CapNumInFloor = zeros(nF, 1);
CapNumInFloor(:) = N/(nF-1);
CapNumInFloor(1) = N;


eventList = [0, 0, 0, 0];
event = [0, 0, 0, 0];
% eventList & event are maintained in the format 'time eventType floor
% elevator' eventType = 0 (simulation starts) eventType = 1 (passenger
% enters queue/makes call) eventType = 2 (elevator arrives) (on floor 'j')
% eventType = 3 (elevator leaves) (floor 'j')

numCalls = 0;
% keeps track of number of people processed

callQ = [0, 0];
% callQ maintains a list of calls that have not been resolved (translated
% into arrival events) it should be maintained in the format (buttonPress,
% dir)

%%  Main Function
while ~isempty(eventList)

    %fprintf('\nLoop Iteration\n') %debug%

    eventList = sortrows(eventList,1);
    event = eventList(1,:);
    eventList(1,:) = [];
```

```
    %% Process simulation start event
    if (event(1,2) == 0) %i.e If the simulation has started
        fprintf('\n\nThe simulation has started!\n');
        enterQueue(t);
    end


    %% Process call event
    if (event(1,2) == 1) %i.e If a call has been made

        %% Determine what floor the call has been made on
        %This is done using a distribution with the # of people on each
        %floor as the weights
        t = event(1,1);
        buttonPress = unifrnd(0,1); %will hold information about what floor
the call was made on
        cumE = cumsum(numInFloor)/sum(numInFloor);
        for i = 1:nF
            if buttonPress < cumE(i)
                buttonPress = i; %buttonPress now holds info about which
floor the call originated on
                break;
            end
        end

        numInFloor(buttonPress) = numInFloor(buttonPress) - 1;

        %% Determine which direction the passenger is headed
        %We do this using a bernouli variable w/ the parameter equal to the
        %ratio of the number of people below him vs above him
        if buttonPress == 1 %if you are 1st floor
            dir = 1; %go up
        elseif buttonPress == nF
            dir = -1; %go down
        else
            nDown = cumsum(numInFloor(1:(buttonPress-1)));
            %nDown is the number of people in the floors below
            nUp = N - nDown - numInFloor(buttonPress);
            p = nDown/(nUp+ nDown);
            dir = unifrnd(0,1);
            if dir<p
                dir = -1;
            else
                dir = 1;
            end
        end

        floorQ(buttonPress, (dir==-1)*3 + dir) = floorQ(buttonPress, (dir==-
1)*3 + dir) + 1;
        fprintf('t = %d : Floor %d : Passenger Queues : FloorHas %d : No
ElInfo : : : : QUpHas %d : QDwnHas %d \n', t, buttonPress +
(buttonPress~=1)*jump, numInFloor(buttonPress, 1), floorQ(buttonPress, 1),
floorQ(buttonPress, 2)); %debug%
```

```
        %bookkeeping for total wait time
        totalWait = totalWait - t;
        numCalls = numCalls + 1;

        %% Make a call to an elevator
        if floorQ(buttonPress, (dir==-1)*3 + dir) == 1  %i.e if the button
has been pressed for the first time
            fprintf('t = %d : Floor %d : New call : Inform Elevator\n', t,
buttonPress + (buttonPress~=1)*jump); %debug%
            processCall(buttonPress, dir);
        end
        %We have now successfully made a call to an elevator

        if t < 30000
            enterQueue(t);
        end

    end

    %% Process Elevator arrival
    if (event(1,2) == 2) %i.e If eventType is an elevator arrival

        %% Arrival event bookkeeping
        t = event(1,1);
        nowOnFloor = event(1,3);
        nowInEle = event(1,4);
        aaa = 0;

        currFloor(1, nowInEle) = nowOnFloor;
        currFloor(2, nowInEle) = t;

        %% We release everyone whose destination is the current floor
        numInFloor(nowOnFloor, 1) = numInFloor(nowOnFloor, 1) +
qInElv(nowOnFloor, nowInEle);
        totalWait = totalWait + t*qInElv(nowOnFloor, nowInEle); %bookkeeping
for total wait time
        aaa = qInElv(nowOnFloor, nowInEle); %debug% Used in later %debug%
        qInElv(nowOnFloor, nowInEle) = 0;

        %% Elevator bookkeeping
        dir = 0;
        dir = (elMov(nowInEle)==1)*1 + (elMov(nowInEle)==-1)*2;
        if (sum(qInElv(:,nowInEle)) == 0) && (dir~=0) && (floorQ(nowOnFloor,
dir) == 0) && isempty(eventList(:, 4) == nowInEle)
            elMov(nowInEle) = 0; % Not going to continue in the same
direction (it may reverse)
            %fprintf('here\n'); %debug%
        end
        %Again.
        dir = 0;
        dir = (elMov(nowInEle)==1)*1 + (elMov(nowInEle)==-1)*2;

        if dir==0 && (sum(floorQ(nowOnFloor,:)) > 0)
            elMov(nowInEle) = (floorQ(nowOnFloor,1)>0)*1 +
(floorQ(nowOnFloor,2)>0)*-1;
```

```
            dir = (elMov(nowInEle)==1)*1 + (elMov(nowInEle)==-1)*2;
        end


        fprintf('t = %d : Floor %d : Elevator %d : ArrivalReleases %d :
FloorHas %d : ElevHas %d\n', t, nowOnFloor + (nowOnFloor~=1)*jump, nowInEle,
aaa, numInFloor(nowOnFloor, 1), sum(qInElv(:, nowInEle))); %debug%
        %%%fprintf('t = %d : Elevator %d : Floor %d : ArrivalReleases %d :
        %%%NowHas %d \n', t, nowInEle, nowOnFloor, aaa, sum(qInElv(:,
        %%%nowInEle))); %debug%

        %% Add departure event if elevator is not stationary
        if dir~=0
            eventList = [eventList; t+timeAtFloor, 3, nowOnFloor, nowInEle];
            eventList = sortrows(eventList,1);
        end

    end


    %% Process Elevator departure

    if (event(1,2) == 3) %i.e If eventType is an elevator departure

        %% Departure event bookkeeping
        t = event(1,1);
        nowOnFloor = event(1,3);
        nowInEle = event(1,4);

        dir = 0;
        if nowOnFloor == 1
            dir = 1;
        elseif nowOnFloor == nF
            dir =2;
        else
            dir = (elMov(nowInEle)==1)*1 + (elMov(nowInEle)==-1)*2;
        end
        numPasIn = 0;
        %Note: dir will never be zero because if it was, then the arrival
        %event would not have generated a departure event


        %If there are no passengers to pick up and there is nobody in your
        %elevator waiting to go somewhere, then do nothing This new check
        %is necessary because a different elevator could have scooped up
        %some passengers between the arrival event check and the departure
        %event check.
        if (sum(qInElv(:, nowInEle)) == 0) && (dir~=0) &&(floorQ(nowOnFloor,
dir) == 0) && isempty(find(eventList(:,4) == nowInEle))
            elMov(nowInEle) = 0;
            fprintf('t = %d : Elevator %d : Floor %d : Now Idle\n', t,
nowInEle, nowOnFloor + (nowOnFloor~=1)*jump);
        else
            if dir ~= 0 % Process departure only if not stationary.
Basically, if it is a stationary elevator, then you don't depart
```

```
                numPasIn = min(floorQ(nowOnFloor, dir), (elCap -
sum(qInElv(:,nowInEle)))*((elCap - sum(qInElv(:,nowInEle))) >-1) );
                floorQ(nowOnFloor, dir) = floorQ(nowOnFloor, dir) - numPasIn;
                %decrease floorQ only at the departure event. That way, you
                %can account for last minute arrivals (and last minute
                %arrivals won't make new elevator calls)

                if dir == 1
                    flrRng = (nowOnFloor+1):nF;
                elseif dir == 2
                    flrRng = 1:(nowOnFloor-1);
                else
                    flrRng = 1:nF;
                end


                %We then 'push' destination floor buttons for each
                %passenger waiting on the current floor.
                for i = 1:numPasIn
                    %Determine destination floor This is done using a
                    %distribution with the # of people on each floor as (1
                    %- the weights)
                    buttonPress = unifrnd(0,1); %will hold information about
the destination floor
                    tempNumInFloor = CapNumInFloor - numInFloor;
                    tempNumInFloor(nowOnFloor) = 0; %set weight of current
floor = 0;
                    cumE =
cumsum(tempNumInFloor(flrRng))/sum(tempNumInFloor(flrRng));
                    for j = 1:length(flrRng)
                        if buttonPress < cumE(j)
                            buttonPress = j + (dir==1)*nowOnFloor;
%buttonPress now holds info about which floor the passenger wants to go to
                            break;
                        end
                    end
                    qInElv(buttonPress, nowInEle) = qInElv(buttonPress,
nowInEle) + 1;
                    fprintf('t = %d : Floor %d : Elevator %d : Qgoto %d : dir
%d\n', t, nowOnFloor + (nowOnFloor~=1)*jump, nowInEle, buttonPress +
(buttonPress~=1)*jump, dir); %debug%

                    %add arrival event at floor if it doesn't already exist
                    addEvent = find(eventList(:,2) == 2 & eventList(:,3) ==
buttonPress & eventList(:,4) == nowInEle);
                    if isempty(addEvent)
                        %eventlist bookkeeping arrivalTime = start of
                        %journey time + # floors elevator travel*time/floor
                        %+ # of stops along the way*time/stop
                        arrivalTime = t + abs(currFloor(1,nowInEle) -
buttonPress)*timePerFloor + (buttonPress==1 | nowOnFloor ==
1)*jump*timePerFloor/2 + timeAtFloor*(length(find( (eventList(:,2) == 2) &
(eventList(:,4) == nowInEle) & (eventList(:,3) > buttonPress)
))*(sign(currFloor(1,nowInEle) - buttonPress)==1) + length(find(
```

```
((eventList(:,2) == 2) & (eventList(:,4) == nowInEle) &eventList(:,3) <
buttonPress)))*(sign(currFloor(1,nowInEle) - buttonPress)==-1) ) ;
                        %increase the scheduled time of all
                        %stops/departures that occur after this stop by
                        %time/stop*1
                        eventList(find( (eventList(:,2) == 2) &
(eventList(:,4) == nowInEle & (eventList(:,1) > arrivalTime)) ), 1) =
eventList(find( (eventList(:,2) == 2) & (eventList(:,4) == nowInEle &
(eventList(:,1) > arrivalTime)) ), 1) + timeAtFloor;
                        eventList(find( (eventList(:,2) == 3) &
(eventList(:,4) == nowInEle & (eventList(:,1) > arrivalTime)) ), 1) =
eventList(find( (eventList(:,2) == 3) & (eventList(:,4) == nowInEle &
(eventList(:,1) > arrivalTime)) ), 1) + timeAtFloor;
                        %add arrival event to eventlist
                        eventList = [eventList; arrivalTime, 2, buttonPress,
nowInEle];
                        eventList = sortrows(eventList,1);
                    end
                end
                %change destFloor of current elevator Display for
                %confirmation%%fprintf('t = %d : Elevator %d has left floor
                %%d with %d new passengers\n', t, nowInEle, nowOnFloor,
                %%numPasIn); %debug% Display for confirmation%%fprintf('t =
                %%%d : Elevator %d now has %d passengers\n', t, nowInEle,
                %%%sum(qInElv(:, nowInEle))); %debug%
                %eventList(((eventList(:,2) == 2) & (eventList(:,4) ==
                %nowInEle)), :) %debug% find(((eventList(:,2) == 2) &
                %(eventList(:,4) == nowInEle))) %debug%
                %min(find((eventList(:, 2) == 2) & (eventList(:, 4) ==
                %nowInEle))) %debug%
                if ~isempty(find((eventList(:, 2) == 2) & (eventList(:, 4) ==
nowInEle)))
                    destFloor(1, nowInEle) = eventList(min(find((eventList(:,
2) == 2) & (eventList(:, 4) == nowInEle))), 3);
                    destFloor(2, nowInEle) = eventList(min(find((eventList(:,
2) == 2) & (eventList(:, 4) == nowInEle))), 1);
                    elMov(nowInEle) = sign(destFloor(1, nowInEle) -
currFloor(1, nowInEle));
                    fprintf('t = %d : Floor %d : Elevator %d : DepartureTakes
%d : Floorhas %d : ElevHas %d : QUpHas %d : QDwnHas %d\n', t, nowOnFloor +
(nowOnFloor~=1)*jump, nowInEle, numPasIn, numInFloor(nowOnFloor, 1),
sum(qInElv(:, nowInEle)), floorQ(nowOnFloor, 1), floorQ(nowOnFloor, 2));
%debug%
                    %%%fprintf('t = %d : Elevator %d : Floor %d :
                    %%%DepartureTakes %d : NowHas %d\n', t, nowInEle,
                    %%%nowOnFloor, numPasIn, sum(qInElv(:, nowInEle)));
                    %%%%debug%
                end

                if floorQ(nowOnFloor, dir) > 0
                    processCall(nowOnFloor, (dir==1)*1+(dir==2)*-1);
                end
            end
        end
    end
```

```
    %% Check if callQ can be resolved
    if length(callQ(:,1)) > 1
        l = length(callQ(:,1));
        for k = 2:l
            buttonPress = callQ(2,1);
            dir = callQ(2,2);
            if floorQ(buttonPress, (dir==1)*1 + (dir==-1)*2) > 0
                processCall(buttonPress, dir);
            end
            callQ(2,:) = [];
        end

    end


end

sum(floorQ)
avgWait = totalWait/numCalls

end




%%  Helper functions
function[l] = rateTravel(t) %represents rate at which people decide to use
the elevator
global N;
l = 0; %represents the rate of arrival

l = (t< 28800)*0                 +...   %(before 8:00 AM, no one uses the
elevator)
    (t>=28800 && t<32400)*N/3600 +...   %(between 8:00AM and 9:00 AM, people
use it regularly)
    (t>=32400 && t<43200)*eps    +...   %(after 9:00 AM before Lunch, people
use it rarely)
    (t>=43200 && t<50400)*N/3600 +...   %(LunchTime! 12:00 - 14:00. Heavy
elevator usage)
    (t>=50400 && t<61200)*eps    +...   %(after 14:00 before 17:00, Post
lunch - light use)
    (t>=61200 && t<64800)*N/3600 +...   %(Closing time 17:00 - 18:00. Heavy
use)
    (t>=64800)*0;                       %(After hours (post 18:00) - No use)

end

function enterQueue(t)      %adds elevator call event to the eventlist
global eventList;
tA = ceil(exprnd(1/rateTravel(t)));       %Determine time of event
eventList = [eventList; t+tA, 1, 0, 0];   %Add call event to eventlist
eventList = sortrows(eventList,1);
end
```

```
function processCall(buttonPress, dir)

global t;
global totalWait;
global N;
global jump;
global timeAtFloor;
global timePerFloor;
global elCap;
global qInElv;
global currFloor;
global destFloor;
global elMov;
global floorQ;
global numInFloor;
global CapNumInFloor;
global eventList;
global event;
global callQ;



%Make an elevator call This process involves calling the nearest elevator
%that is moving towards you (unless there is a stopped elevator closer to
%you). Ensure that you are not altering current elevator trip

%debug% Display X, Y, Z
X = (elMov(1, :) == dir) .* ( ( (destFloor(1,:)>buttonPress) &
(currFloor(1,:)>buttonPress) ).*(dir==-1) | ( (destFloor(1,:)<buttonPress) &
(currFloor(1,:)<buttonPress) ).*(dir==1) );
%The 1st boolean evaluates to 1 for all elevators that are moving towards
%the call The 2nd boolean evaluates to 1 if the bottonPress is not
%inturrupting a current elevator trip So X = 1 implies that the elevator is
%suitable for the trip
Y = abs(destFloor(1,:) - buttonPress).*X; %distances of eligible moving
elevators from the call floor
Z = abs(destFloor(1,:) - buttonPress).*(elMov == 0); %distances of eligible
stopped elevators from the call floor %Doesn't work if stopped elevator is on
the same floor as buttonPress



%If there is a stopped elevator on this floor, we use that elevator
callE =   min(find( (currFloor(1, :) == destFloor(1,:)) .*  (currFloor(1, :)
== buttonPress) )); %debug% Display callE
if  sum(currFloor(1, currFloor(1, :) == destFloor(1,:) ) == buttonPress) > 0
&&  ~isempty(callE) && elMov(callE) == 0
    %currFloor(1,:); %debug% Display destFloor(1,:); %debug% Display
    callE =   min(find( (currFloor(1, :) == destFloor(1,:)) .*  (currFloor(1,
:) == buttonPress) )); %debug% Display callE
    fprintf('t = %d : Floor %d : StatElev %d : SF_ServiceCall Accepted\n', t,
buttonPress + (buttonPress~=1)*jump, callE); %debug%
    elMov(callE) = dir;
    arrivalTime = t;
```

```
    %add arrival&departure events to eventlist
    eventList = [eventList; arrivalTime, 2, buttonPress, callE];
    eventList = sortrows(eventList,1);
    %fprintf('We are adding an arrival event\n') %debug%

    %If there is a moving elevator that is suitable for this trip we assign
    %the call to the elevator whose current destination is closest to the
    %caller's floor unless there is a stopped elevator closer to the caller
elseif ~isempty(find(X==1))

    if isempty(find(Z~=0)) | min(Y(Y~=0)) < min(Z(Z~=0)) %if moving elevator
satisfies the criteria
        callE = min(find(X==1)); %debug% Display
        fprintf('t = %d : Floor %d : MovElev %d : JustInFloor %d :
MV_ServiceCall Accepted\n', t, buttonPress + (buttonPress~=1)*jump, callE,
currFloor(1, callE) + (currFloor(1, callE)~=1)*jump); %debug%

        %we add an arrival event to the eventlist only if the current
        %(moving) elevator already does not have the buttonPress floor as a
        %destination
        addEvent = find(eventList(:,2) == 2 & eventList(:,3) == buttonPress &
eventList(:,4) == callE);
        if isempty(addEvent)
            %eventList Bookeeping arrivalTime = start of journey time + #
            %floors elevator travel*time/floor + # of stops along the
            %way*time/stop
            arrivalTime = currFloor(2,callE) + timeAtFloor +
abs(currFloor(1,callE) - buttonPress)*timePerFloor + (buttonPress==1 |
currFloor(1, callE) == 1)*jump*timePerFloor/2 + timeAtFloor*(length(find(
(eventList(:,2) == 2) & (eventList(:,4) == callE) & (eventList(:,3) >
buttonPress) ))*(sign(currFloor(1,callE) - buttonPress)==1) + length(find(
((eventList(:,2) == 2) & (eventList(:,4) == callE) &eventList(:,3) <
buttonPress)))*(sign(currFloor(1,callE) - buttonPress)==-1) ) ;
            %fprintf('Elevator most recently on floor: %d, Stops in
            %between: %d', currFloor(1,callE), length(find( (eventList(:,2)
            %== 2) & (eventList(:,4) == callE) & (eventList(:,3) >
            %buttonPress) ))*(sign(currFloor(1,callE) - buttonPress)==1) +
            %length(find( ((eventList(:,2) == 2) & (eventList(:,4) ==
            %callE) &eventList(:,3) <
            %buttonPress)))*(sign(currFloor(1,callE) - buttonPress)==-1) );
            %%debug%

            %increase the scheduled time of all stops/departures that occur
            %after this stop by time/stop*1
            eventList(find( (eventList(:,2) == 2) & (eventList(:,4) == callE
& (eventList(:,1) > arrivalTime)) ), 1) = eventList(find( (eventList(:,2) ==
2) & (eventList(:,4) == callE & (eventList(:,1) > arrivalTime)) ), 1) +
timeAtFloor;
            eventList(find( (eventList(:,2) == 3) & (eventList(:,4) == callE
& (eventList(:,1) > arrivalTime)) ), 1) = eventList(find( (eventList(:,2) ==
3) & (eventList(:,4) == callE & (eventList(:,1) > arrivalTime)) ), 1) +
timeAtFloor;
            %add arrival event to eventlist
            eventList = [eventList; arrivalTime, 2, buttonPress, callE];
            eventList = sortrows(eventList,1);
            %fprintf('We are adding an arrival event\n') %debug%
```

```
        else
            %fprintf('But we are NOT adding a arrival event cuz it already
            %exists\n') %debug%
        end

    else %if stationary elevator satisfies the criteria
        callE = min(find(Z~=0)); %debug% Display
        fprintf('t = %d : Floor %d : StatElev %d : InFloor %d :
DF*_ServiceCall Accepted\n', t, buttonPress+ (buttonPress~=1)*jump, callE,
currFloor(1, callE)+ (currFloor(1, callE)~=1)*jump); %debug%
        elMov(callE) = dir;
        %eventList Bookeeping arrival time = current time + # floors
        %elevator travel*time/floor
        arrivalTime = t + abs(currFloor(1,callE) - buttonPress)*timePerFloor
+ (buttonPress==1 | currFloor(1, callE) == 1)*jump*timePerFloor/2 ;
        %add events to eventList
        eventList = [eventList; arrivalTime, 2, buttonPress, callE];
        eventList = sortrows(eventList,1);
        %fprintf('We are adding an arrival event\n') %debug%
    end

elseif ~isempty(find(Z~=0)) %If we are using a stationary elevator
    callE = min(find(Z~=0)); %debug% Display
    fprintf('t = %d : Floor %d : StatElev %d : InFloor %d : DF_ServiceCall
Accepted\n', t, buttonPress+ (buttonPress~=1)*jump, callE, currFloor(1,
callE) + (currFloor(1, callE)~=1)*jump); %debug%
    elMov(callE) = dir;
    %eventList Bookeeping arrival time = current time + # floors elevator
    %travel*time/floor
    arrivalTime = t + abs(currFloor(1,callE) - buttonPress)*timePerFloor +
(buttonPress==1 | currFloor(1, callE) == 1)*jump*timePerFloor/2 ;
    %add events to eventList
    eventList = [eventList; arrivalTime, 2, buttonPress, callE];
    eventList = sortrows(eventList,1);
    %fprintf('We are adding an arrival event\n') %debug%

    %We assign the call to the elevator that is furthest away from the
    %caller's floor
else
    fprintf('t = %d : Floor %d : NoEle Found : addTo callQ\n', t,
buttonPress+ (buttonPress~=1)*jump); %debug%
    callQ = [callQ; buttonPress, dir];
end

end
```