Meritorious 2007

Lee Rosenberg, Lydie Van Holland, Derlin Campbell

Print This Page Close This Window

Team Control Number

1993

For office use only
T1 _____
T2 _____
T3 _____
T4 _____

Problem Chosen

B

For office use only
F1 _____
F2 _____
F3 _____
F4 _____

2007_ Mathematical Contest in Modeling (MCM) Summary Sheet
(Attach a copy of this page to each copy of your solution paper.)

Type a summary of your results on this page. Do not include the name of your school, advisor, or team members on this page.

Airlines are faced with a paradoxical situation with regard to the boarding of aircraft. On one margin, they are necessitated to form a more functional system of loading and unloading of the aircraft, yet on another, they risk the addition of change that could possibly mar the very operation of their trade. The problem given to us is that of modeling and optimizing this process in a quantitative manner.

Developing a basic model of human behavior was the most excruciating portion of the endeavor. The basic model involved the use of random numbers and the concept of a queue to simulate human behavior. Having a basic model provided a means of comparison to existing research and a guide to perfecting the existing techniques.

Perfecting the model was the absolute intent of our work. We introduced the idea of a scalar to represent the expedience of a given passenger, as well as a stepwise function for the stowage of carry-on baggage in the overhead bins. A discrete function involving distance, each passenger's boarding time, and other factors hindering the procession to a seat was utilized to simulate the abstract model of different airplanes.

We devised a new method that airlines could implement that takes into account the many different variables that affect human behavior, such as age and air travel experience. This method uses the behavior scalar and sorts the passengers both by row number (rearmost to foremost row) and behavior scalar values.

We devised a complex model for Southwest's random boarding, as well as the outside-in and rear to front methods. These simulations performed both to our specification and that of real world statistics. With respect to the new method that we formulated, we were able to take from 2.13 minutes up to 11.83 minutes off of the standard boarding times for a Boeing 737-300 aircraft.

Airplane Boarding Model

Table of Contents

Introduction

Airplanes have become a way of life; where we used to drive, no matter the distance, the trip has been replaced by a small aircraft that can land at any given airport. One major issue that has recently surfaced is that of airplane boarding, which can easily diverge into a complete disaster. So which method would be the best way of loading and unloading an airplane?

Airlines have tried many different methods of boarding and have found that some are much better than others. Some airlines, like Southwest, let the passengers choose their own seats, formally known as the random technique. Other airlines use the "Back to Front" method, allowing the first class passengers and those with special needs to board first. The rest of the plane is then boarded, beginning with the zone in the back of the plane and ending with the foremost zone. All studies, and many airlines, agree that this is by far the worst technique and, in fact, many airlines are moving away from its use.

The next model we investigated was the "Outside-In" model. This plan still boarded the first class and the passengers with special needs first. The rest of the plane was then allowed to board, first by the passengers seating next to a window, then the center seat, and finally, the aisle seat. Some of the models we examined divided the plane further into different zones and started boarding with the rearmost zone.

The article in the New York Times, which was referenced by the problem, was useful in finding a starting point for our model. We chose to create a program in C++ to simulate the time that it took to board the plane using the time mentioned in the Times article with respect to Southwest's technique. We determined a range of times that, when given to our program, simulated very close to 25 minutes. We based our model off of a Boeing 737-300 similar to the planes flown by Southwest. We used a random number to assign where the people would sit in the plane and how long it would take to get to their seats. From here we were able to look at different models, like the "back to front" and "outside-in" and compare the results to other airline boarding times.

After creating a working model that reflected the data we were able to obtain, we looked at the different factors that would affect boarding time. The different types of passengers, such as children, passengers in wheelchairs, frequent travelers and first time travelers were taken into consideration. Each was assigned a time value (the behavior scalar) that would slow them down from our average time value. This allowed us to look at a broader range of situations than just the quintessential travelers that are able to board and sit down quickly.

Using the results from our simulations, we were able to determine which method was superior and come up with our own ideas on how to effectively board an airplane. We focus mainly on the time to board rather than the unloading time because it will have more of an impact on the turn around time of an airplane.

1. Forming a Basic Model

This model was a preliminary simulation of the time it would take for passengers to board given different boarding methods and average times for passengers to board. This model uses random generators to try to capture the most optimistic boarding times reported by airlines.

Variables

T = Total time to board plane
t_i = time for individual person to board plane
t_a = median time for each passenger to board
P = passengers

For this model, the number of passengers equals the number of seats on the plane.

Assumptions

There were many simplifying assumptions made in this model. This model assumes that:

- The average time every passenger takes to board the plane is with in a certain neighborhood of the mean amount of time calculated (not taking into account any delays a person might face while boarding a plane).
- The plane is totally filled.
- The time it takes the next passenger to board is independent of the passenger that boarded before him/her.
- Passengers board randomly within sections allowed to board.
- More than one passenger may board at a time.
- The only delay for one passenger's boarding is due to the previous passenger's boarding slower.
- All the rows had the same number of seats.

The Model

In this model, our goal was to simulate passengers boarding a plane and to calculate the time for boarding. Our model randomly selected different times from a neighborhood of the calculated mean for each person boarding the plane and then summed the times. To calculate the average time it took a passenger to board, we used the data reported by airlines and divided the total time by the total number of passengers. The neighborhood of the value was chosen based off of judgment of reasonable values for the average passenger boarding the plane. The sum of all individual passengers' boarding times is defined as the total boarding time of the plane.

We ran simulations of different boarding methods and different sizes of aircraft. For the simulations in this model, we assumed that the floor plan for the plane was composed of only coach-class seating. To change the size of the aircraft for our simulations, we needed only to adjust the number of rows and seats, which we based off the seat plans for a Boeing 737, 757 and 777 based upon data from Continental Airlines. For the larger plane, we devised a floor plan that would give the desired number of seats required to meet the large aircraft mentioned in the problem specifications.

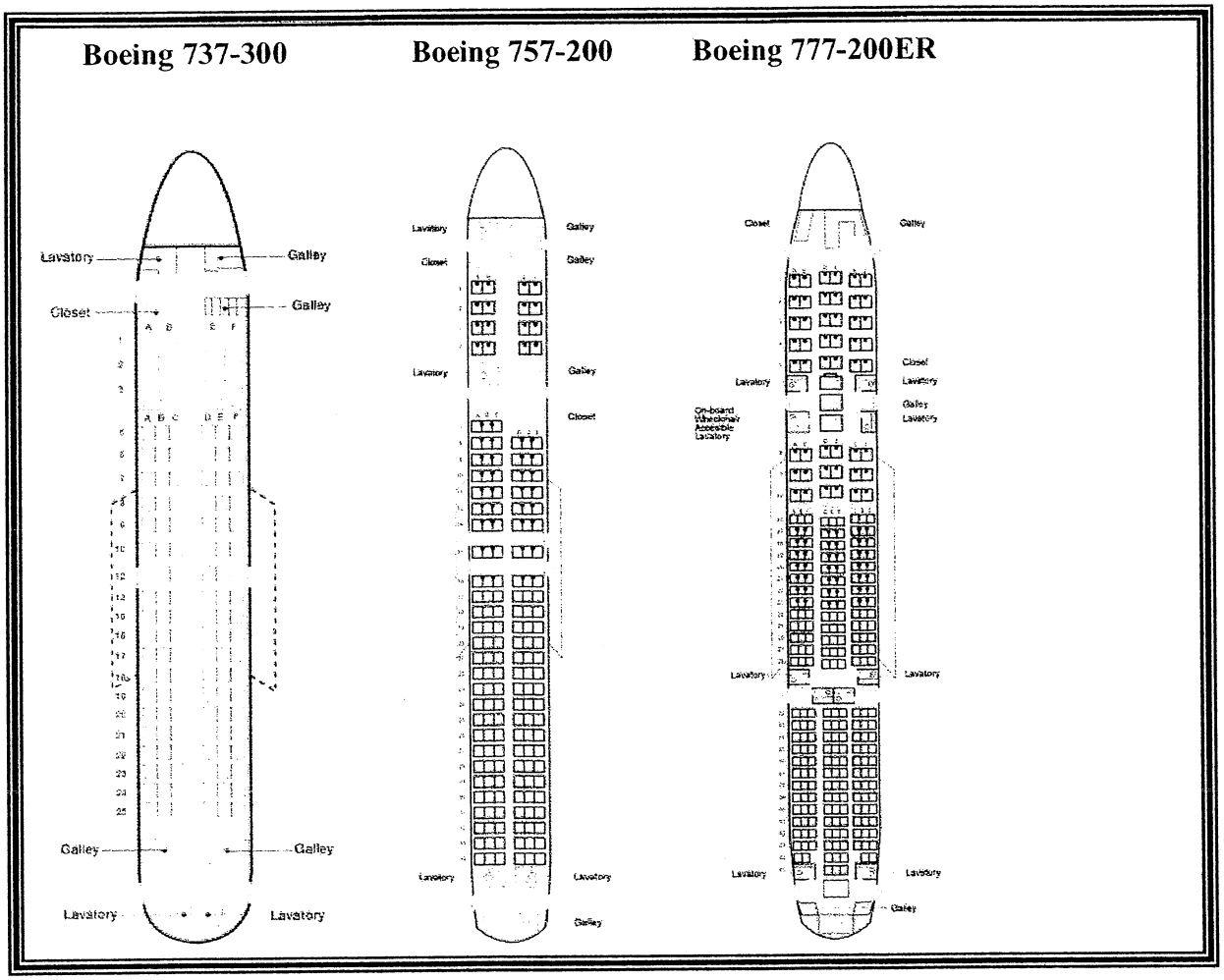


Figure 1.1 – Configurations of Boeing aircraft used in this project

For our first approach, we attempted to simulate the random boarding approach that Southwest Airlines practices without any form of assigned seating. To accomplish this, we randomly assigned seating and randomly assigned boarding times to passengers until the plane was completely filled. We calculated the average time based upon the data reported by Southwest Airlines by taking the shortest reported time for boarding and divided that by the number of passengers on the plane.

$$t_a = \frac{T}{P}$$

Equation 1.1 – Time per person

When our simulation was run, the results were compared to the data given by Southwest Airlines. The total time was reasonably comparable to the lowest reported time by Southwest. An average of five runs for this simulation resulted in the following graph.

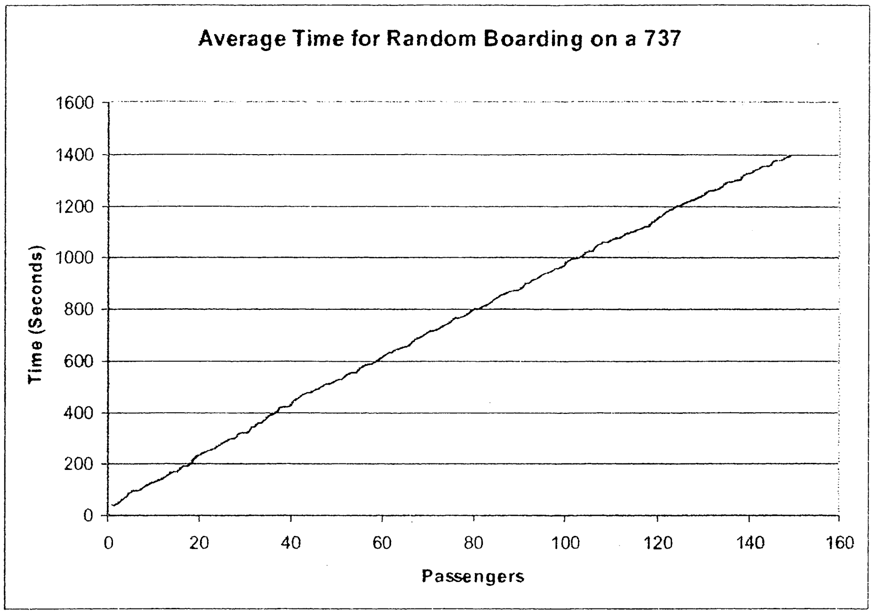


Figure 1.2 – graph of the accumulation of boarding time for Southwest Airlines' random boarding technique

Our second simulation was designed to simulate the "Back-to-Front" loading technique. To do this, we randomly generated the row assignments for all passengers and grouped the plane into four to five sections by row. To validate our simulation, we compared our results to the lower end of the times (highlighted in grayish-yellow in the table below) reported by a study by van den Briel from Arizona State University. This simulation was an average of five different runs of the program that we created.

	BF6	BF5	BF4	BF3	OI6	OI5	OI4	OI3
Avg Seat Interferences	[illegible]	[illegible]	[illegible]	[illegible]	2.94	2.94	2.94	26.0[illegible]
Avg Aisle Interferences	52.27	52.74	53.36	[illegible]	42.64	42.92	42.02	[illegible]
Avg Total Interferences	124.49	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	44.96	[illegible]
Avg Boarding Time (sec)	1493.68	[illegible]	[illegible]	1436.76	[illegible]	[illegible]	[illegible]	1412.79

Figure 1.3 – Table of values used to check our model (van den Briel et. al)

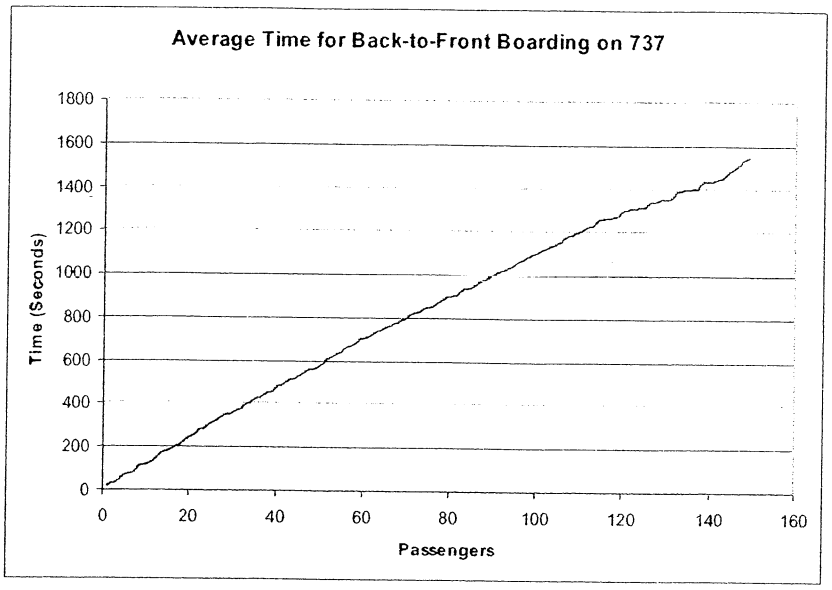


Figure 1.4 – Graph for time accumulation of Back-to-Front boarding Technique

The last simulation we created was that of the outside to inside boarding method. This boarding method has passengers who are sitting by the windows board first, followed by those sitting in the center seats, and finally those that are sitting in aisle seats. This simulation divided the plane into $\frac{N_r}{2}$ (where N_r is the number of seats in a row) sections by columns and then used a probabilistic function to determine which passenger from the group would board. The average of five runs of this simulation yielded the following graph of total time. Figure 1.6 shows a comparison of the three models previously discussed with "Back to Front" being the slowest and random the fastest.

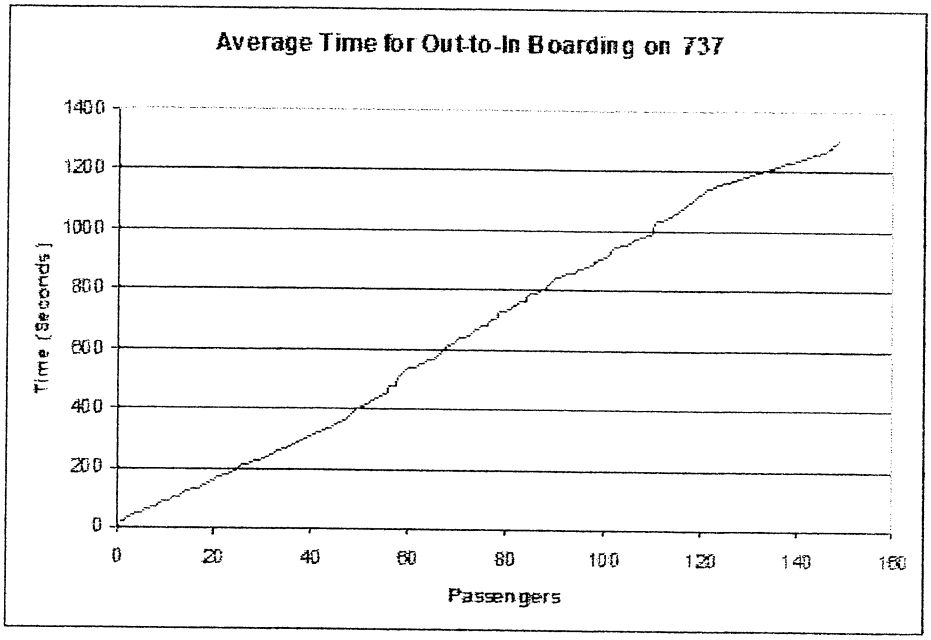


Figure 1.5 – Time accumulation graph of Window-to-Aisle

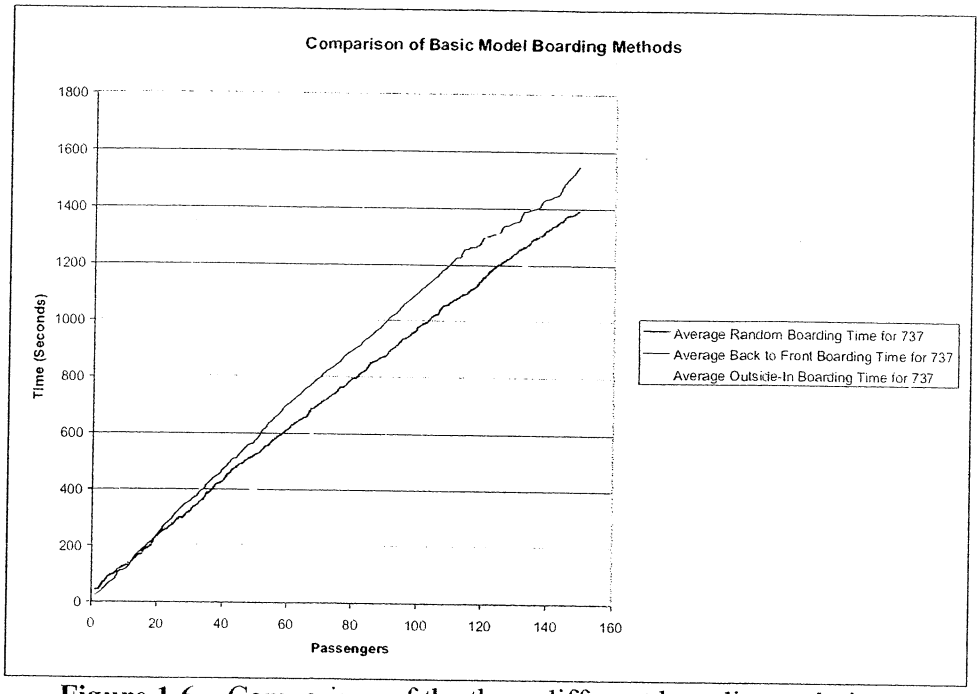


Figure 1.6 – Comparison of the three different boarding techniques. Note: "Back-to Front" is the slowest and the random is the most efficient

2. Forming a New Model

There are many different models available that have addressed the boarding time issue. The one model we found that represents the time it actually takes rather than the delay time was written by Pieric Ferrari and Kai Nagel. They used an equation that would calculate the number of bags the person has and how many times the passenger vacates their seat to allow another passenger through. We will look at both models and will explain the equations and how they were used.

Time to Sit Down

t_s = Total time for seating
t_p = Time getting up or down from seat and into aisle
n_s = Number of seats with people already sitting that the passenger has to wait on before sitting.

$$t_s = t_p + 2 * t_p * n_s = t_p(1 + 2n_s)$$

Equation 2.1 – Researched time for an average passenger to actually sit

We used this formula in our model along with an equation of the time it takes to load the bags into the overhead bins.

Time for loading overhead bins

T_b = Total time for loading bags into overheads
N_{bin} = Number of bags already in the bins
N_i = Number of bags the passenger is loading

$$t_b = 2 + \frac{n_{bin} + n_i}{2} * n_i$$

Equation 2.2 – Researched equation for overhead stowage time

The authors point out that as the overheads bins reach maximum capacity, the time becomes very large (they assumed every passenger would take at least one bag). For example, on a Boeing 757, if you are the 201st passenger and each passenger had one bag, the time would be just over 3 minutes. If each had two bags, the time would be over 6½ minutes. We felt this was not realistic, so we used their ideas to create our own model of the time it takes to fill the overhead bins.

We based our model on the volume and the number of bags that a passenger might carry. The simulation program randomly assigned the number of bags and how big each was up to the maximum size allowed for a carry-on.

T_b = Time it takes to out way the bags
N_i = Number of bags
V_u = Fraction of used volume in the overhead bins

$$t_b = \begin{cases} 2 + 1n_i : 0 \le V_u \le \frac{1}{4} \\ 2 + 2n_i : \frac{1}{4} < V_u \le \frac{1}{2} \\ 2 + 4n_i : \frac{1}{2} < V_u \le \frac{3}{4} \\ 2 + 6n_i : \frac{3}{4} < V_u \le 1 \end{cases}$$

Equation 2.3 – Our version of the baggage stowage equation

This equation resulted in more realistic times when the overhead bins were reaching full capacity. When the used volume is greater than ¾ of the total, the total time it takes the person to put away two bags would be 14 seconds. When a passenger does not have any bags, the time is still two seconds due to behavioral reasons.

The next part to our equation for finding the boarding time considered the fact that some people take longer than others to board. For example, a frequent flier will take less time than someone who is flying for the first time, and someone with special needs will require even longer. We broke each group down and assigned a scalar (the Behavior Scalar or f_p) that would increase the overall boarding time. These values may be under or overstated, and may be an area for future research and data collection.

	Frequent Traveler	First Time Traveler
Children	0.50	1.00
Senior Citizens	0.75	0.90
Vacationers	0.25	0.35
Business Travelers	0.10	0.40
Special Needs	1.00	1.32

Figure 2.1 – Different behavior scalars

Total Boarding Time

The total boarding time is going to be the sum of each of the time it takes to put the bags into the overhead bins and the time it takes to sit down. The passenger fits into one of the categories in the table, and the time is then multiplied by that factor.

$$t = (t_b + t_s) f_p$$

Equation 2.4 – Total boarding time including the behavior scalar

3. Simulation of the Boarding Process of a 737-300: A New Model

In essence, we needed to find a way to accurately generate passengers with varying demographics and subsequently simulate those passengers' journey in boarding the airplane. There were a few variables that we felt were essential for the modeling of such an event, such as dimensions of carry-on luggage and the age of the individual. Figure 3.1 shows all of the variables used herein.

Variable Name	Units	Sample Value	Description
myBagLength	Inches	30.11	These three variables were used to represent the dimensions of carry-on luggage for a given passenger
myBagWidth	Inches	6.10	
myBagHeight	Inches	8.77	
myBagVolume	Cubic Inches	1613.74	Representation of the volume of a passenger's bag. This is used to determine boarding time based upon the necessity to look for an empty bin
myAge	Years	26.5	This is used to calculate various values
myNumChildren	Quantitative	2	This is used by the behavior scalar (discussed later) to add an arbitrary amount of time to the boarding process.
myAverageChildAge	Years	4.08	This is also used by the behavior scalar and is used to judge the experience of the traveling child.
myPersonFactor	Seconds/ Person	1.15	This is the root of our calculation method for the time that it takes for the passenger to board. This basically scales the time based upon experience and the type of traveler.
myRowNumber	Arbitrary	18	This is the number of the passenger's row
mySeatLetter	Arbitrary	0	This uses the convention that 0-6 = A-F

Figure 3.1 – Variables used in the simulation program

Simulation

The basis of the algorithm is quite complex; it uses the basis of human behavior to model the time that it takes to perform the various tasks involved in boarding an airplane. As most have probably experienced, children and elderly individuals generally take longer to board than those who have flown for a long time. The confusion involved in finding the seat, finding a location for overhead baggage, and coordinating the movement of those blocking one's seat is usually enough to mar the progress of the plane for a decent amount of time; in our model this is all taken into account.

The actual simulation of boarding is carried out using an iterative function based upon the blockage of the aisle and rows, along with the time that each individual takes to make their way to the seat.

In the process of simulation, all passengers are put into a long, linked list based queue where they wait to board. In the respective boarding methods, care is taken to randomize the order in which people board (for example, in the "Back to Front" model, passengers are randomized within their respective 'zones'). Passengers are then taken from a given index of the linked list and begin to board. The program then 'walks' the passenger to their respective row and places them there until they are able to sit. Meanwhile, the next passenger is popped off of the linked list and is given the opportunity to board. The program keeps track of the nearest row to the front in which the aisle is occupied, and if the next passenger is past that boundary, the following formula decides the time in which they are expected to wait until the aisle clears. Equation 3.1 shows one of the formulas used in the simulation program.

t_A=The time that Passenger A takes to board

t_B=The time that Passenger B takes to board

N_A=Passenger A's row number

N_B=Passenger B's row number

$$t_{added} = (t_A - t_B)\left(\frac{N_A}{\begin{Bmatrix} N_B = 0:1 \\ N_B = 1:N_B \end{Bmatrix}}\right) \text{ such that } t_A > t_B \text{. If not, then } t_{added} = 0$$

Equation 3.1 – Time that is added for a person to vacate the aisle when blocking another passenger

The decision as to how much time is added for the shuffling to get *into* the row is given by the following:

L_A=The seat "Letter" of passenger A

L_B=The seat "Letter" of passenger B

t_M=Any arbitrary *random* number between zero and ten

$$|L_A - L_B| \cdot (t_M)$$

Equation 3.2 – Time to add for an already seated passenger to allow another passenger to their seat

The boarding time of passenger B is only used if he/she blocks the next passenger from boarding, as multiple passengers will not contribute to the total boarding time if they both are concurrently moving to their seats. After the individual time is added to the total, the minimum row variable is set to the passenger that just boarded and the loop resumes with the next passenger in the linked list.

In the process, a graphical representation of the seating pattern is created. Figure 3.2 shows a sample sequence for completely random boarding using our models of the methods (that will be discussed later) as the time that a given passenger takes to seat themselves.

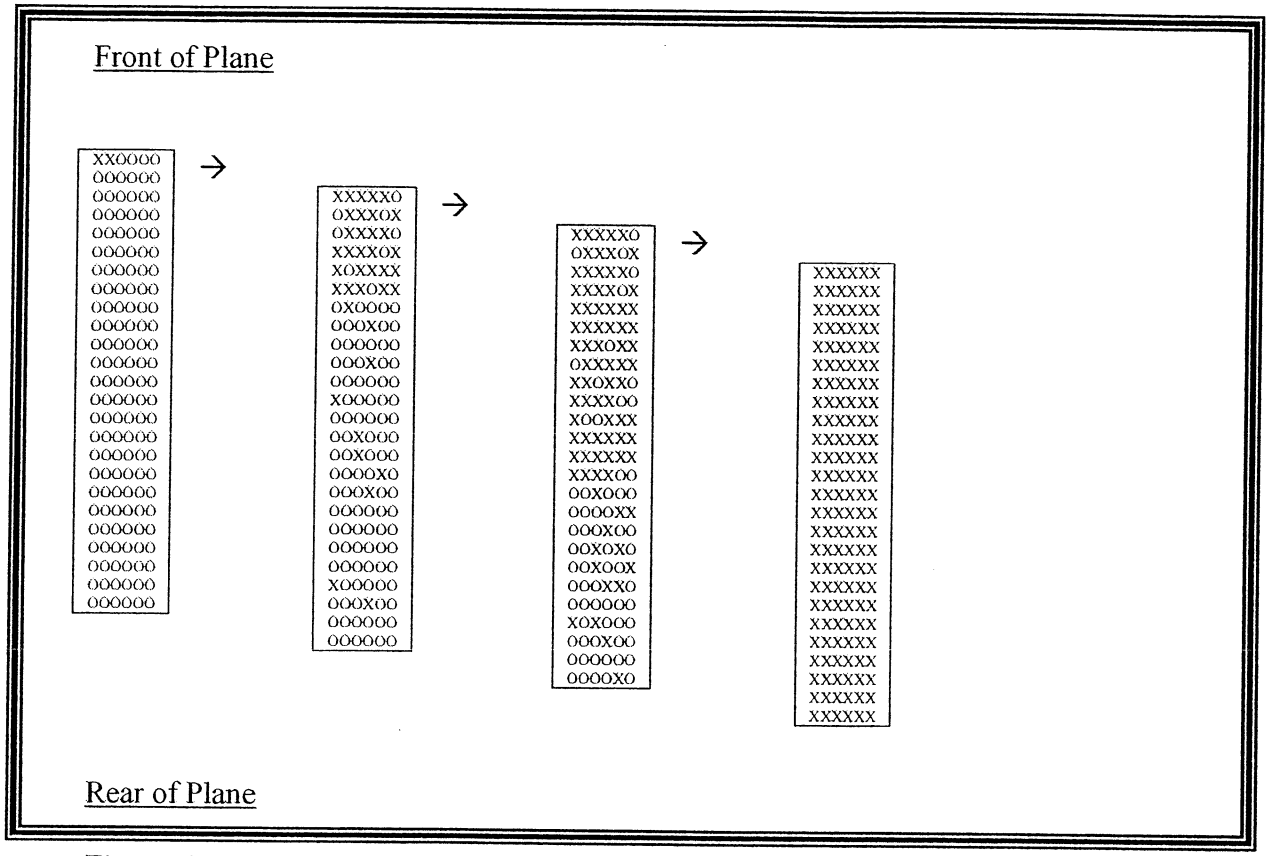


Figure 3.2 – The boarding sequence of a 737-300 using Southwest® Airlines' random seating system (Note: X's are filled seats)

Coding

The following code implements the simulation algorithm and iterates through however many passengers will fit in the plane. The number of passengers that fit in the plane is defined by two different class variables, ROWSINPLANE and SEATSINROW. These two variables can be changed to any given size; even those that do not presently exist. Figure 3.3 shows a code snippet from the simulation program.

```
cPassenger cp=getLoc(myBoardingQueue,randomness[0])->getData();
minimumRow=cp.getRowNumber();
myPlane[cp.getRowNumber()][cp.getSeatLetter()]='X';
totalTime+=cp.getBoardingTime();
for(int i=1;i<ROWSINPLANE*SEATSINROW;i++)
{

        cPassenger cp2=getLoc(myBoardingQueue,randomness[i])->getData();
        myPlane[cp2.getRowNumber()][cp2.getSeatLetter()]='X';
        int currrow=cp2.getRowNumber();
        if(currrow>=minimumRow)
        {
                totalTime+=(cp.getBoardingTime()- \ cp2.getBoardingTime()*(cp.getRowNumber()/(cp2.getRowNumber() ==
                0 ? 1 : cp2.getRowNumber())))>=0 ? (cp.getBoardingTime()-
                cp2.getBoardingTime()*(cp.getRowNumber()/(cp2.getRowNumber() == 0 ? 1 : \ cp2.getRowNumber()))) :
                0;
        if(cp2.getSeatLetter() > cp.getSeatLetter())
        {
                totalTime+=abs(cp2.getSeatLetter() - \ cp.getSeatLetter())*(double)rand()/(double)RAND_MAX*10;
        }
        minimumRow=cp2.getRowNumber();
        cp=cp2;
```

Figure 3.3 – Code snippet of the simulator

4. Simulation Results With the New Model

Overall, the new model simulated time more realistically, capturing the higher end of boarding times reported. Many interesting results have surfaced using our new algorithm such as the divergence of the "Outside-In" boarding system in the worst case scenarios. It seems as if the completely random boarding method utilized by Southwest Airlines takes on a linear representation, which is interesting in the sense that utter chaos would intuitively seem to be nonlinar. The "Back to Front" method appeared to take a logistic shape as passengers filled the aircraft. Futhermore, the "Outside-In" technique took on a graph that was similar to a logistic, but much more complex.

Random Boarding

Although intuitively inferior, the random boarding technique proved to be more efficient. When given an extreme situation, random performs better overall than the other methods for the Boeing 737, however due to linear nature of this method we anticipate that this method would take longer as the plane size increases. In figure 4.1, a representation of the demographics used in the simulation of the worst case scenario is given in pie-chart form. Similarly, our best case scenario was represented by 80 percent expedient passengers.

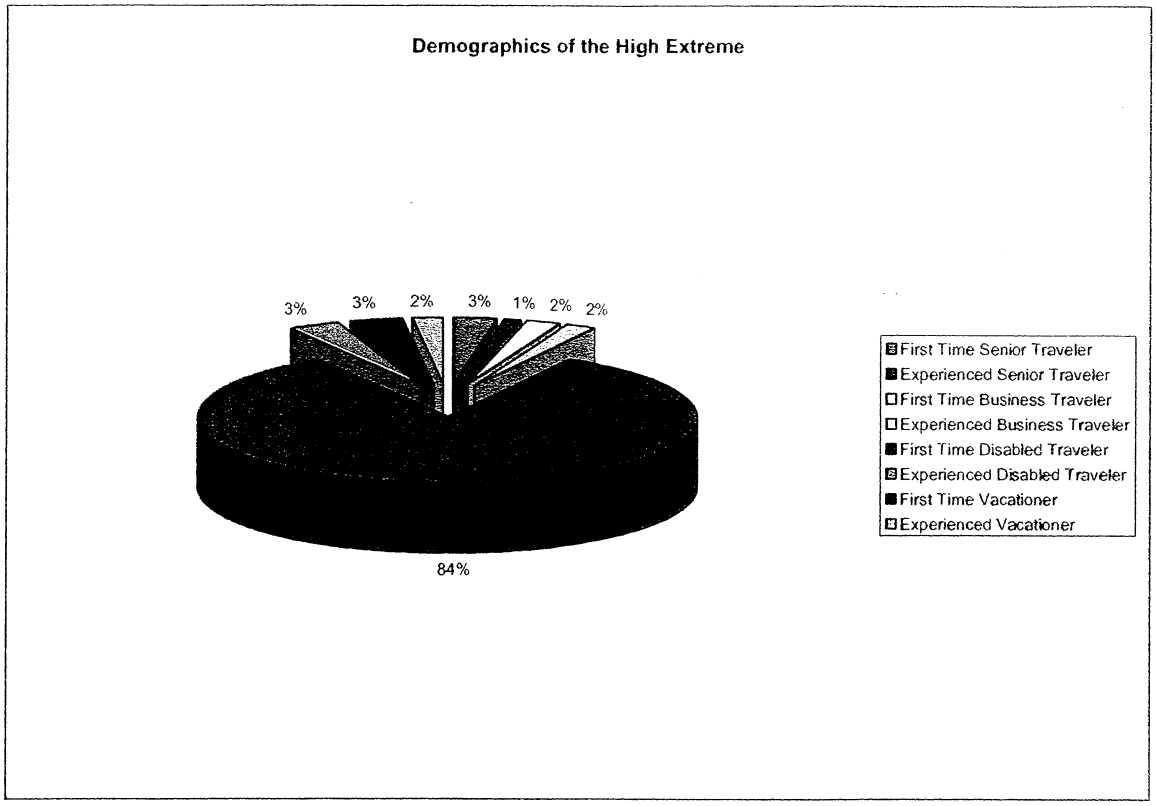


Figure 4.1 – Demographic distribution of the worst-case scenario with random boarding

We have formulated a few observations and theories as to why the random boarding varies such a great deal.

The Random Boarding Results

- Seemed to follow some sort of exponential or linear function in this situation as opposed to the others that seemed to taper as passengers boarded.
- Takes the shortest amount of time when compared to the other two with respect to having fewer passengers. Once the plane begins to fill, the line seems to cross and exceed the "Outsize-In" and "Back to Front" models.

We hypothesize that the variance in random boarding is caused by the expedience of seating more of the special needs passengers *where they wish* rather than in the less desirable seats.

The very root of random boarding is that there is no set way to board the plane, however, with our program we were able to determine an average time and a good range of how long it will actually take to board.

The extremes of the random boarding technique are widely spread; the best case scenario differs from the worst by approximately *fourteen minutes*. Figure 4.2 demonstrates this.

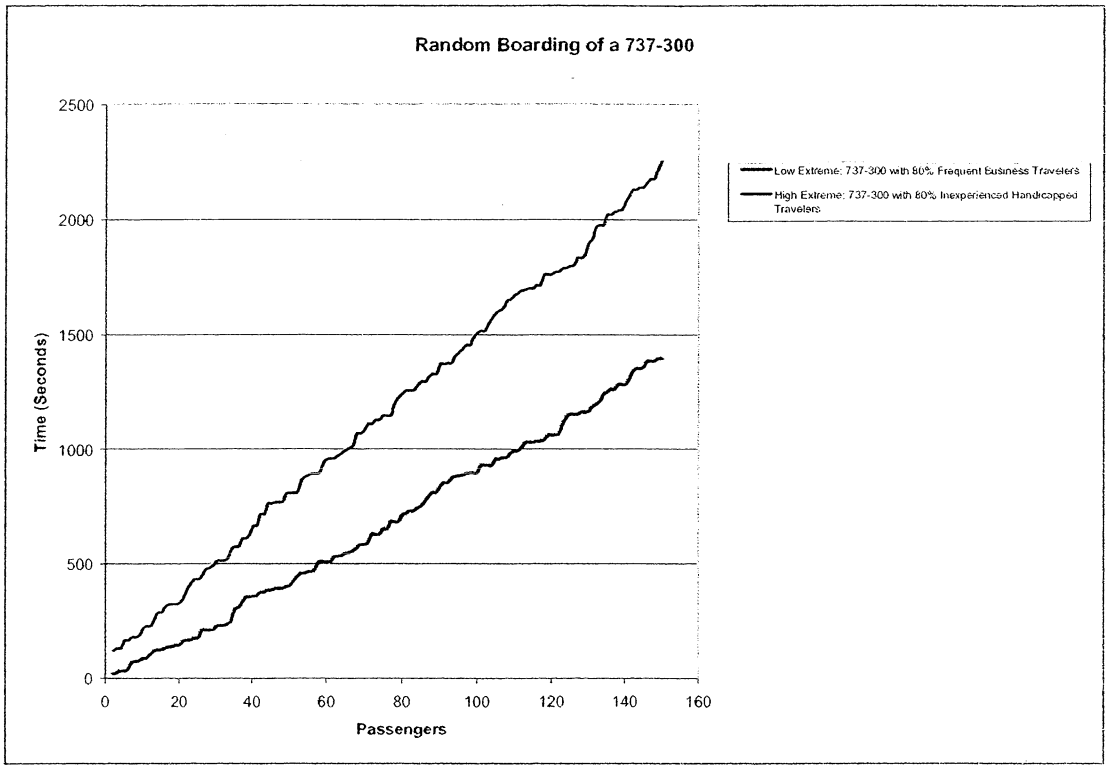


Figure 4.2 – The two extremes for the random boarding technique.

Obviously, we are not going to have a plane full of special needs passengers, nor are we going to have an airplane full of seasoned travelers that take little time to board; this is where the randomization of the demographics comes into play. The following (Figure 4.3) shows the demographics of one such randomly-boarding flight in our simulation.

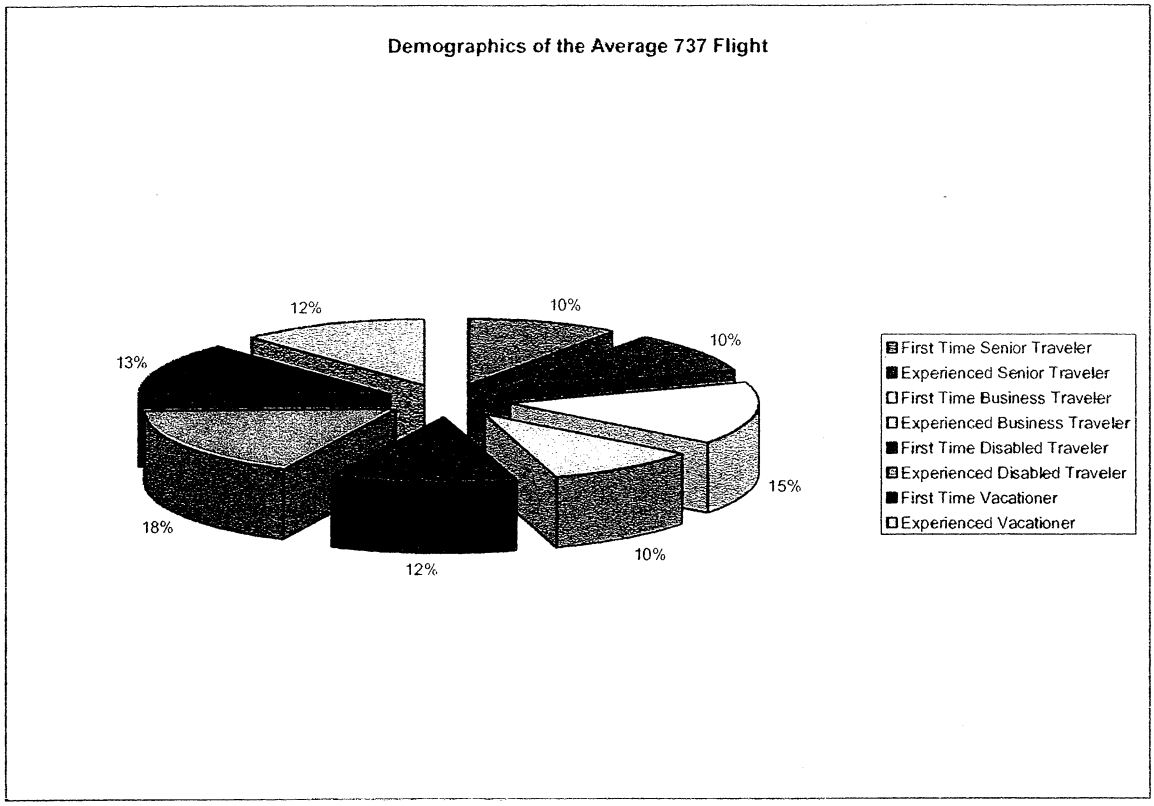


Figure 4.3 – Representation of the average modern day flight.

Still, we are not ever going to see a flight where *twenty percent* of the travelers represent the special needs group. In our research, we found that only 1.2% of most flights' passengers are handicapped; a far cry from the twenty percent shown in the above model. With any model, there are going to be those who do not accurately represent their behavior scalar. This goes to say that any given special needs person can take less time than a seasoned business traveler; there is no real set way to represent human behavior. Because of this, the overrepresentation of certain groups will cancel out the under-representation of others. This is one avenue of improvement that could possibly be followed in further research.

Outside-to-Inside Boarding

For most of our simulations, this was the most efficient manner to board an aircraft; however in some simulations, as the number of passengers increased, the "Outside-In" boarding technique quickly diverged. Another interesting observation that we have noted is that of the inflated rate of time increase in the middle of the boarding process. The following figure (Figure 4.4) shows the odd shape of the curve in the middle of the boarding process that eventually returns to the norm.

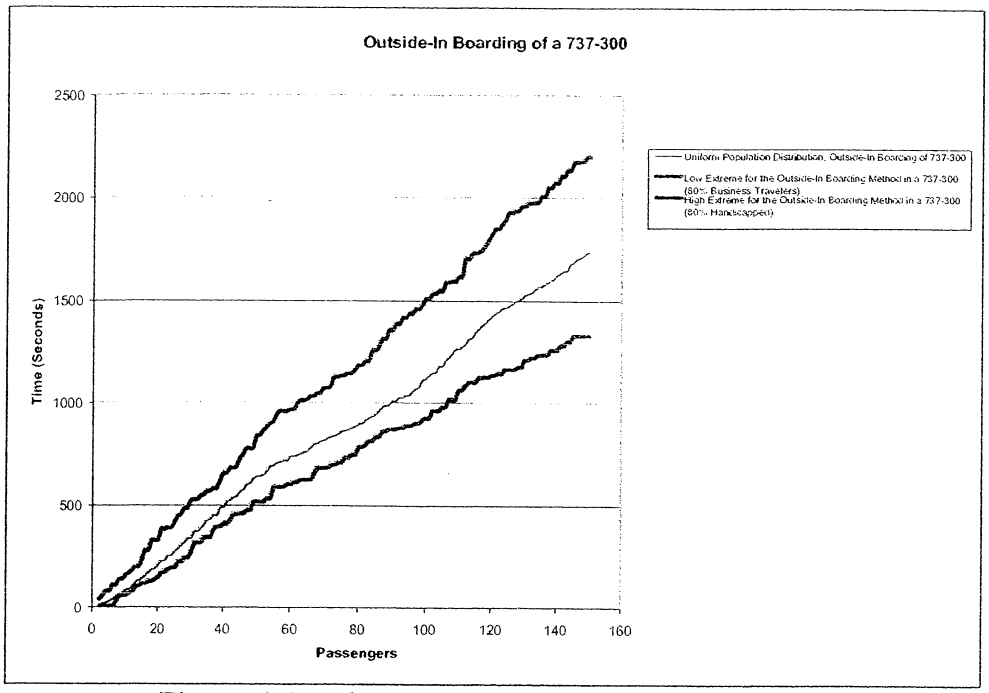


Figure 4.4 – Three runs of the outside-in boarding technique. Note the jump at the arrival of the fortieth passenger and the 110th passenger in the top two curves

Outside-In – A Deeper Look

This, oddly enough, is the most time consuming of the three models when given a large population of handicapped or special needs passengers. We hypothesize that this occurred because of the efforts that are necessary for moving a wheelchair-bound or slower individual to a middle or window seat. This is only exacerbated by the fact that only a few individuals are able to expedite the process (and the fact that when a wheelchair is blocking the aisle, there is absolutely no way around).

To pose an example, think of the situation where there are campers with varying walking speeds that need to cross a bridge meant for two people. They only have one lantern, so one of the campers needs to cross back over with the lantern in order to give it to the others. In this situation, the order in which the campers cross and bring back the lantern is essential to crossing in the time period that is required (one hour in the case of the bridge problem). The situation quickly diverges if the campers have to wait for a long period of time for the others at an extreme.

It is precisely the same situation with airplane boarding; if all of the passengers at the beginning take a very long time, the time for the rest of the boarding process will increase. However, if all of the slower passengers board at the end, there will be a significantly greater number of passengers onboard, therefore creating a situation involving the movement of seated passengers to allow others to their seats.

Back-to-Front Method

By far, this is the worst of the simple methods that we tested. The time that it took to board the plane by this method was greater for most of the process due to the fact that the passengers are trying to sit in the same zone at the same time and quickly congest the aisle. The time then tapered off at the end due to passengers finally settling down into their seats. The variance within the zones adds to the effect, along with the fact that not everyone boards in their own zone. To take a look at a modern airports, one can easily see that some people decide to board when it is not their assigned time; someone in zone six will more readily board in zone four or five because it will get them there sooner. Figure 4.5 shows the graphs of the different runs of the Back-to-Front simulations.

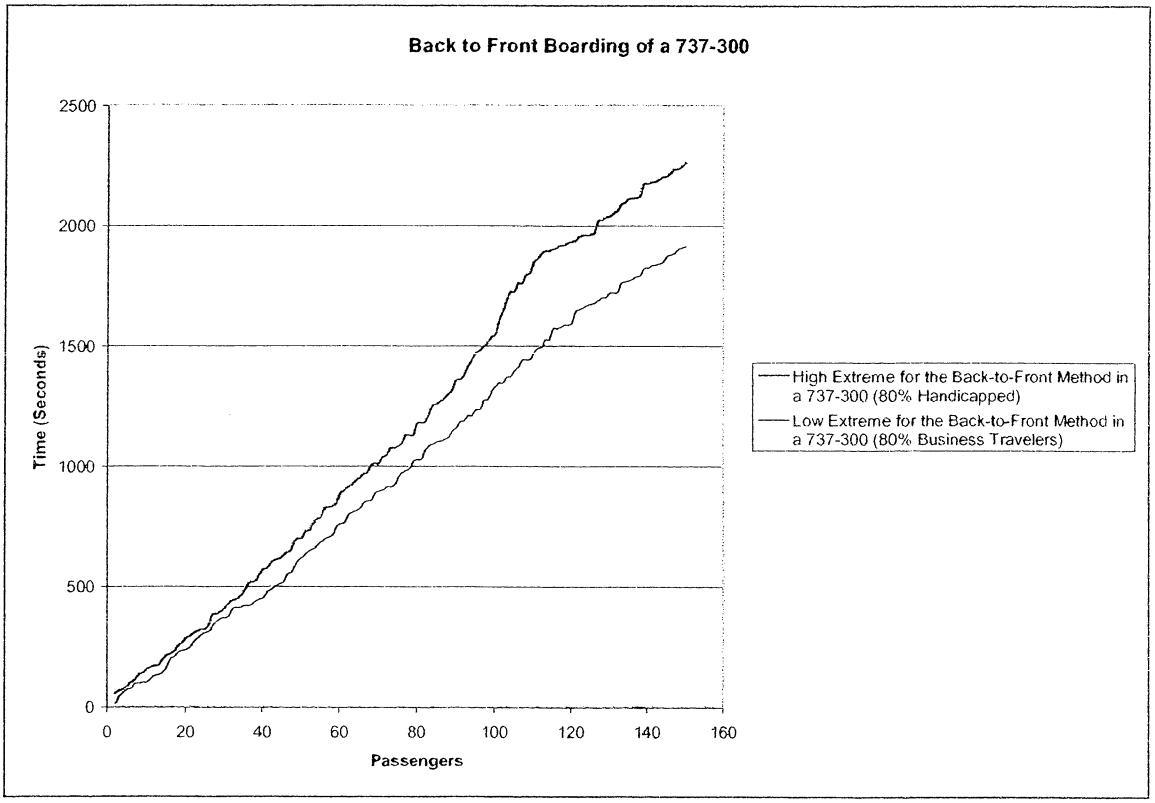


Figure 4.5 – Two extreme cases for the Back-to-Front model

Note that the difference between the two times is not large; the time it takes to board a plane with passengers that take longer will not be much longer the time that it takes to board quicker passengers.

Back to Front: In Depth

We hypothesize that the Back-to-Front model is the worst due to the fact that there still is no real structure to the boarding process. Given that the 'zones' are fairly structured has no real effect on the time that it takes for an individual to board. The fact that people do not obey the

zone rules further exacerbates the situation, along with the fact that filling of the rear overhead bins forces the passengers in the back to block the foremost aisles.

Larger Aircraft

So far we have only examined the relationships with simulations of boarding a Boeing 737-300; thus far we have not looked at any of the larger planes. As described before, the models for the different boarding techniques showed that:

- Random boarding processes gave a linear relationship between the amounts of time to board the plane and the number of passengers boarding.
- "Back-to-Front" technique seemed to show a logistic relationship between number of passengers and boarding time.
- "Outside-In" method showed a peculiar relationship between number of passengers and boarding time.

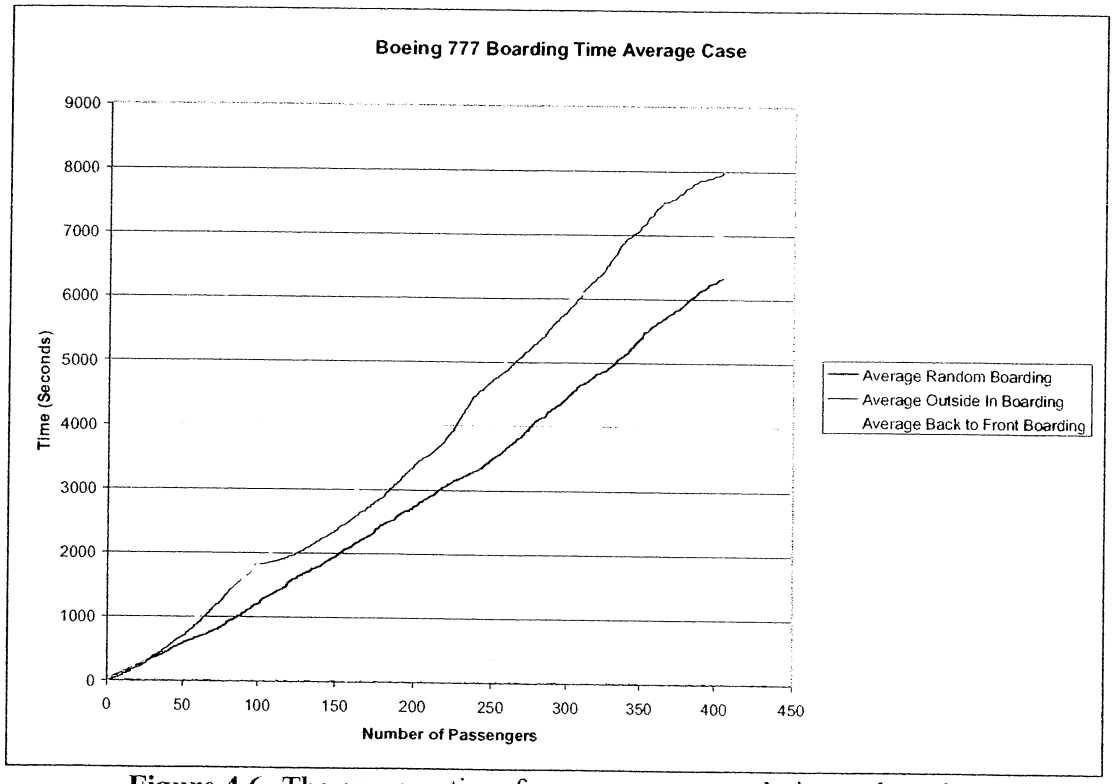


Figure 4.6- The average time for an average population to board a Boeing 777. This graph compares all three different boarding methods.

Figure 4.6 shows that the processes and data the simulation of the boarding process of a Boeing 737 are conserved when boarding a 777. With respect to the models, we found that the technique of boarding Back-to-Front is more time conservative than the Outside-In method. As shown in figure 4.7, the Random boarding technique is still the quickest boarding method of all.

We hypothesize that the random boarding technique is faster for the larger planes as well as the smaller planes because this method allows people to sit where they would like. Most

people, when boarding a plane, are a bit xenophobic; they will not sit by people they do not know and will spread out throughout the plane, therefore lowering the amount of congestion in the aisle.

The model for "Back-to-front" technique seemed to be logistic; it tapers off to a constant later in the process, which means it may be a better method for boarding (as the other techniques continue to increase in time). For the Boeing 777, the "Back-to-Front" method did not prove to be any faster than the random method, but it did perform better than the "Outside-In" method.

5. Our Solution to the Problem

Despite the cacophony of ideas in the current models, we have devised a method that will take significantly less time than the other three that were modeled in this paper.

Preface

We realized that in boarding a plane, the time it takes for a person to board is the ultimate deciding factor in how long the flight takes to board. For this, we have devised a specialized scalar called the 'Behavior Scalar'. This behavior scalar scales the amount of time that the average person takes to board the plane, whether up or down, and allows us to accurately represent the time that a given *specialized* individual will take. We broke this down into categories such as vacationers, businessmen, senior citizens, children, etc. and assigned each of these categories a modifier for "Frequent" and "First Time" flying status. For example, the behavior scalar of a special needs passenger that is flying for the first time would be 1.32 while the factor for a seasoned business traveler would be 0.1; this allows us to develop highly specialized models for the type of passenger.

The Method

We propose to take the behavior scalar into account while making the decision as to which zone queue the passenger will be placed. For example, a seasoned business traveler would need to be placed closer to the end of the zone queue than someone with many children because business travelers will clear the aisle significantly faster for the next zone to board.

Our idea is to take the "Back-to-Front" model that is already in place and add the behavior scalar into each of the zones. The plane will be initially broken into zones, first sorted by behavior scalar such that the faster passengers are in the rear of the zone queue. These zone *queues* will contain passengers sorted first by behavior scalar, and then by the row number. Very inexperienced passengers and special needs individuals will board first from the front of the zone queue. This will free the aisles in the front significantly, therefore allowing faster individuals such as regular vacationers to board. In essence, each passenger is assigned a boarding number after most tickets are sold and the passengers are boarded with respect to this boarding number. Figure 5.1 shows a sample boarding process for our method, along with figure 5.2 that represents a random boarding of the aircraft to compare.

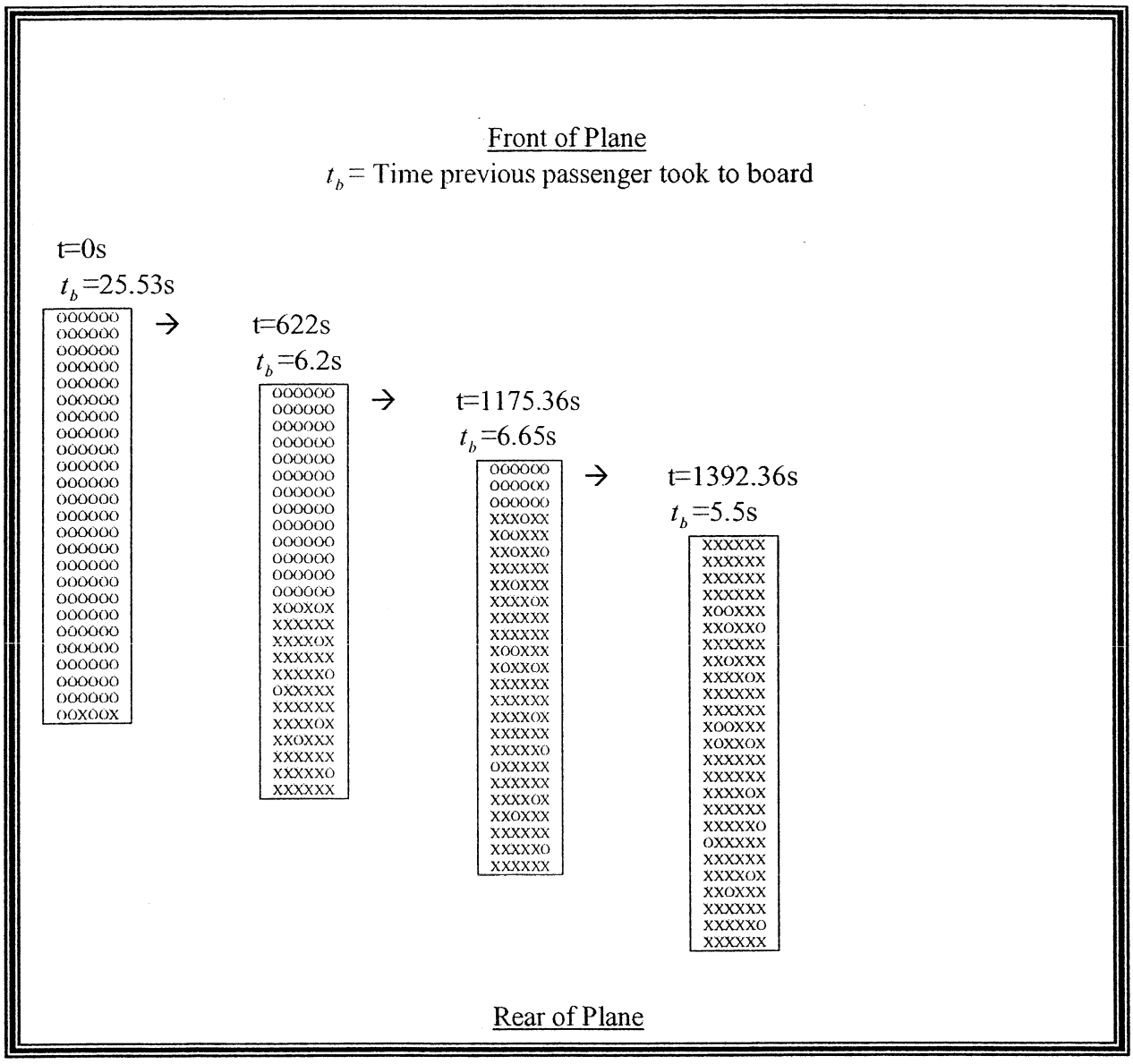


Figure 5.1 – Our method for boarding the aircraft

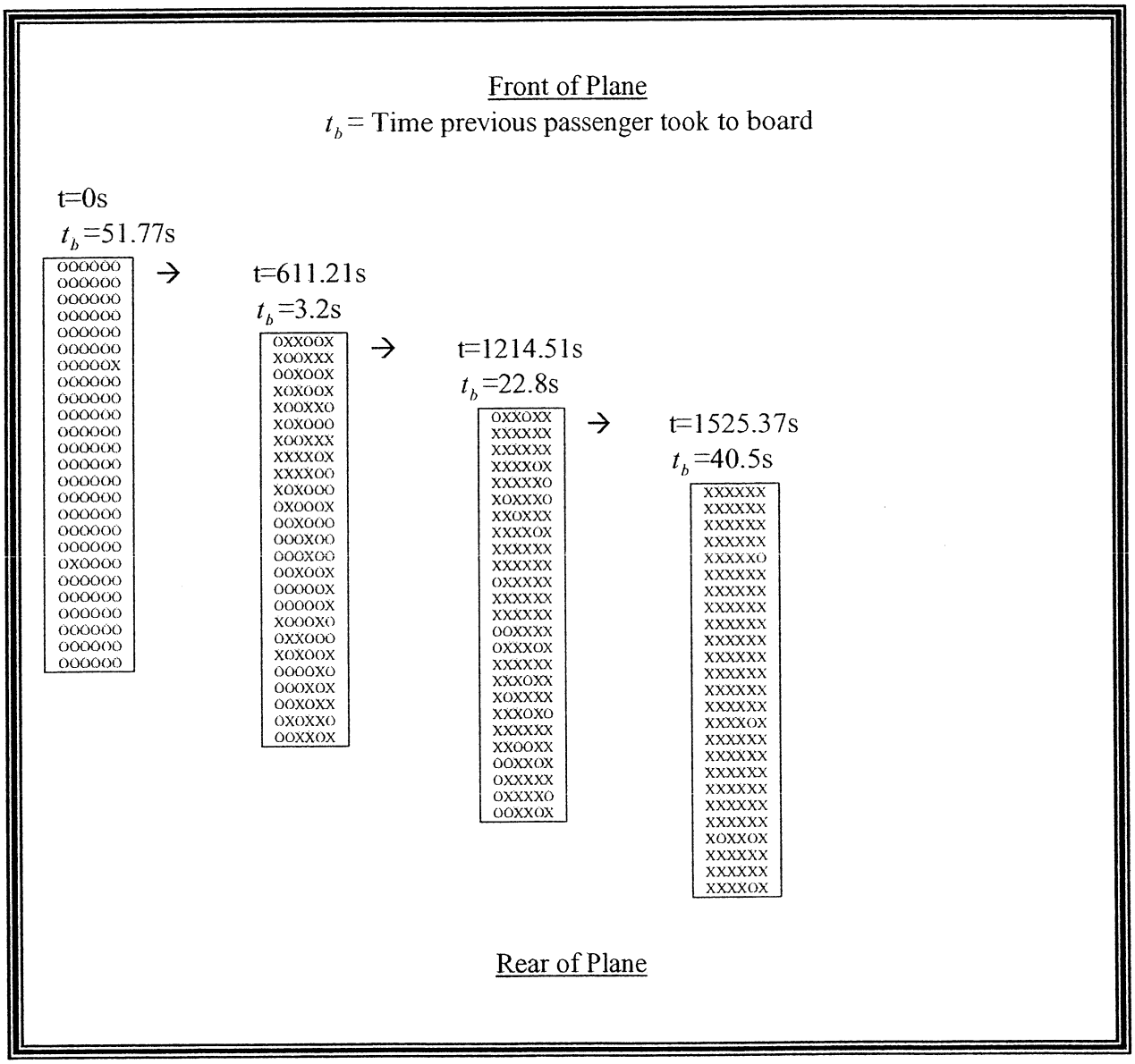


Figure 5.2 – Simulation of the same demographics with the random boarding Technique.

Results

The end results were astonishing; our method for boarding a Boeing 737-300 took 155.12 seconds less time than the random boarding technique. Even more amazingly, our method shaved 128.04 seconds off of the most *efficient* method, the Outside-In method, and took a hallowing 710.3 seconds off of the time to board using the Back-to-Front system. That translates

to a maximum of 11.83 ***minutes*** being shaved off of the current airline techniques. Figure 5.3 shows a graphical comparison of the average runtimes of the other methods.

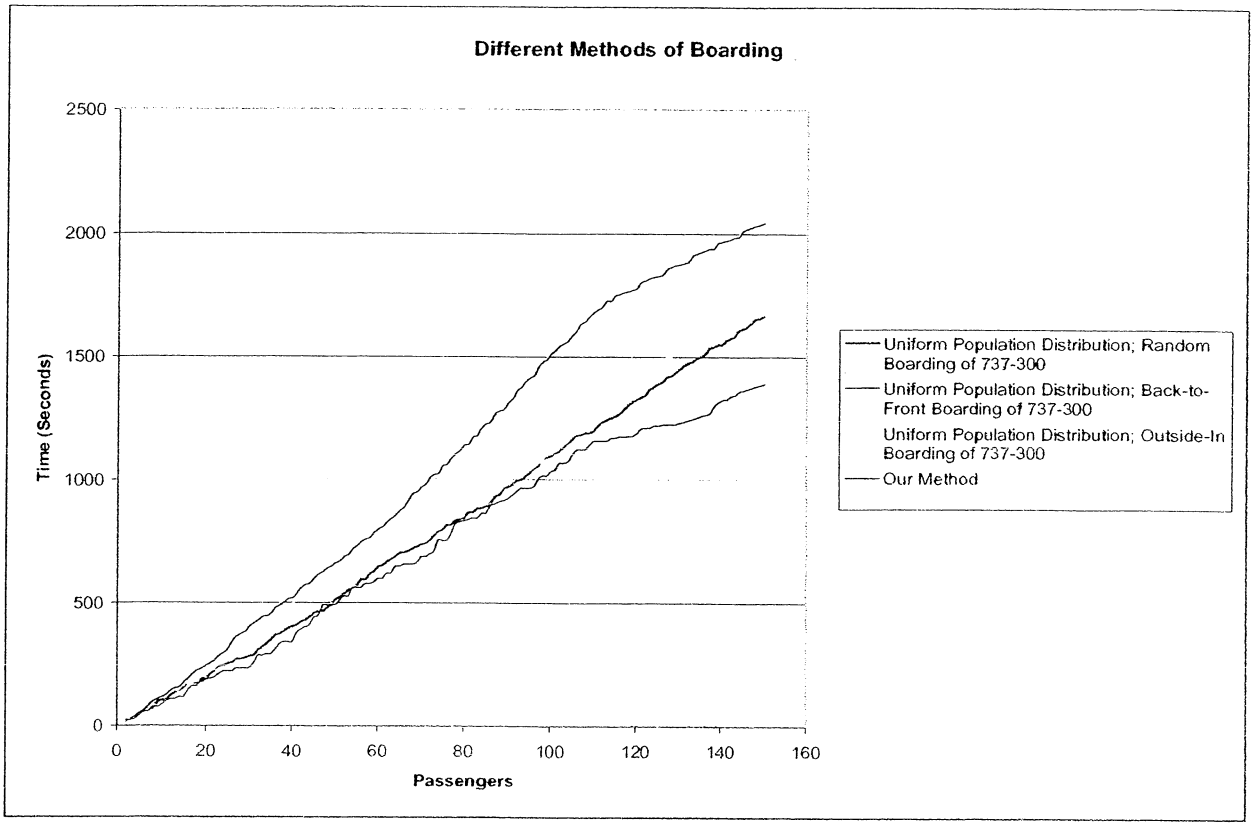


Figure 5.3 – Comparison of the four methods used to board a Boeing 737-300

Note the extravagant difference in the times. At the beginning of the boarding process, the times are relatively closely related, but once approximately one-hundred passengers are on the plane, our method levels off far below the other curves.

Our model, when simulated with a larger aircraft, still performed much better in the end. Figures 5.4 and 5.5 show the simulation run both on a 757 and a 777.

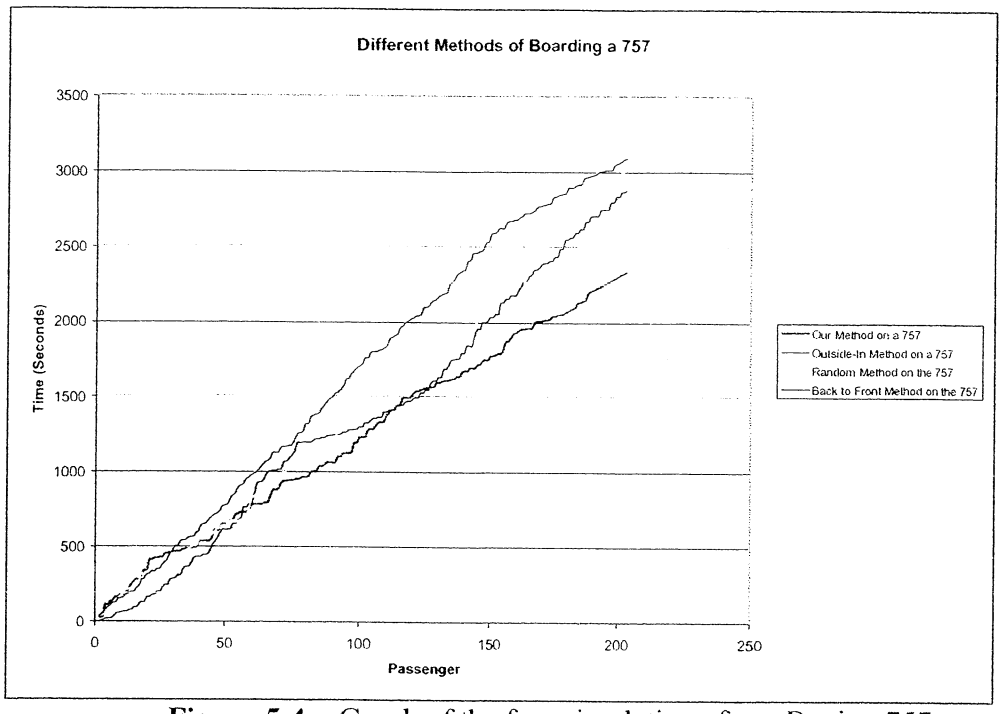


Figure 5.4 – Graph of the four simulations for a Boeing 757

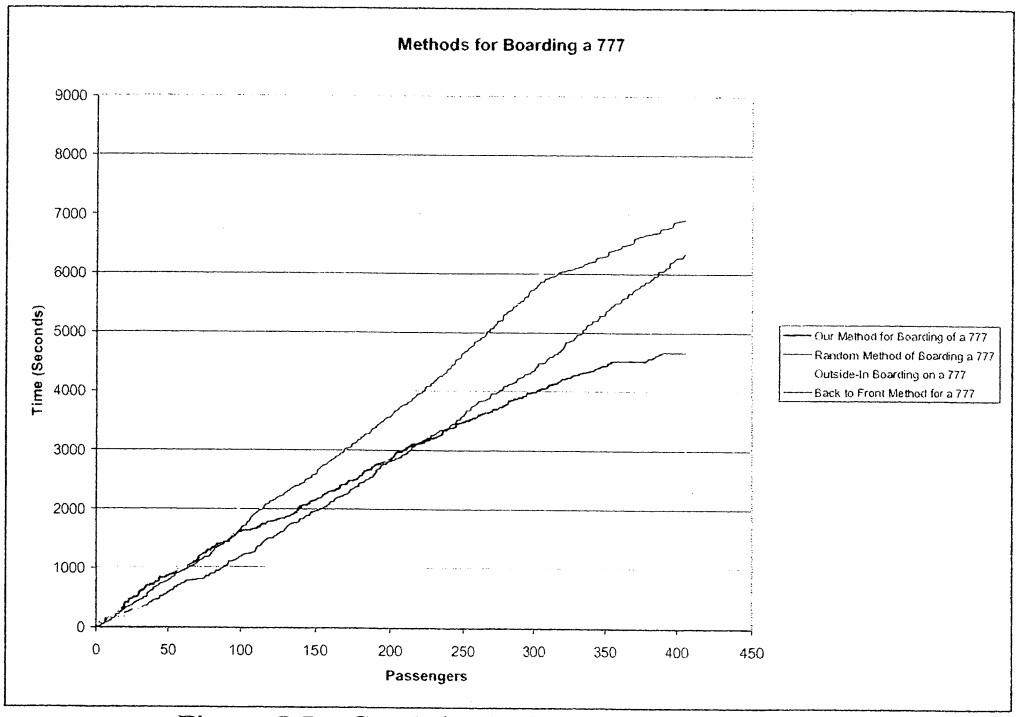


Figure 5.5 – Graph for the four methods on a Boeing 777

Figure 5.6 shows a collection of all of the data used in this project. We include both the researched data and the data created with our simulations in this table to create a detailed comparison of all methods used herein.

Type of Boarding System	Researched	Basic Model on 737-300	Improved 737-300	Improved 757	Improved 777
Random Boarding	25 min	23.18 min	27.75 min	50.34 min	105 min
Outside-In	23.54 min	21.73 min	34.10 min	48.0 min	127 min
Back to Front	23.94 min	25.84 min	28.97 min	51.5 min	112.3 min
Behavior Scalar Sorted Back to Front (Our Proposed Model)	N/A	N/A	23.21 min	38.91 min	77.65 min

Figure 5.6 – Collection of final data

Implementation

The real problem lies in the implementation of a better aircraft boarding method. Our method would be relatively difficult to implement in the real world at first, but the improvements would be incredible. The easiest way that we propose to implement the behavior scalar sorted rear-boarding method would be at check-in or at the time of ticket purchase.

Many contemporary passengers, if not all, purchase their tickets online. If some sort of survey comprised of four or five short queries into such avenues as the passenger's age, flight experience, etc, this can easily be quantified into some sort of discrete equation that will generate a behavior scalar for the passenger.

This brings an interesting roadblock of compliance, which in turn can easily be overcome with data that the airlines already have. Frequent Flier programs (to which most travelers inevitably subscribe) can be utilized to determine the experience of the passenger (number of miles), the passenger's age, number of family members, etc... It may be slightly ethically questionable to obtain this information but this method would be no different than the passenger's local retailer asking for his/her zip code and phone number at the time of transaction.

6. Strengths and Weaknesses

Basic Model

The strength of our basic model was the fact that it was totally random; much like randomness of natural populations. The population of passengers boarding the airplane is going to be random for most flights; therefore, using an entirely random simulation will capture this aspect of passenger demographics better. The model used random times that better simulate the unpredictable occurrences that can delay boarding.

The basic model assumed that all passengers would take around the same amount of time to board, which underestimates the time required for some passengers. We have discussed how some passengers, more specifically those with special needs, take more time to board a plane: an aspect of nature that is overlooked by the basic model. Due to the fact that this model assumes that every passenger takes about the same amount of time to board the plane, we are assuming that the population of travelers is fairly uniform. Another weakness of this model is that it does not take into account first-class seating.

Improved Model

The improved model took into account different types of people and how this would affect the amount of time it took for that person to board the plane. This model broke passengers down into different categories and than assigned the categories different behavior scalars based off whether they would take more or less time to board. We used our judgment and comparison of simulated times to reported times to assign the behavior scalars to each category. The improved model also took into account how long it would take for someone to stow their carry-on luggage. The time for carry-on luggage took into account the number of bags a passenger had to put away as well as the proportion of space that was filled.

As mentioned before, we assigned behavior scalars based on our judgment and comparison of total times to data, but there is no concrete data to which we can compare these values. Finding a concrete method to quantify human psychology is a very difficult, if not impossible task, giving some validity to the random approach of quantifying human nature. The improved model was not as random as the original, since we assigned a different behavior scalar to each category of passenger type. Due to the limited variety in this model, it does not simulate natural populations of passengers as well as it could. We also did not take into account the boarding or the existence of a first class section.

7. Areas for Future Research and Improvement

There are many areas that we decided that need more research in order to get a more accurate model. We feel that if the times used in the simulation were expressed as a range of times covering someone who is slow to a fast person, the simulation would be able to give more accurate results. We also feel that if the data was taken from people boarding planes rather than simulation data, it would prove more useful in simulating the real world. These time ranges include:

- The range of times it takes someone to walk to their aircraft
- The range of times it takes someone to put their baggage in the overhead bins
- The range of times it takes someone to get in and out of their seat

Other factors that we included in our later simulations, the behavior scalar and percent of each group on a flight, might also be an area for future research in the quest to improve the simulation. The numbers that we used were purely estimations from experience and may or may not actually reflect the real world. Theses areas are:

- Percent of passengers that are children
- Percent of passengers that are senior citizens
- How much longer each group takes to board on average

Using a range of values, the simulation can take a random number between the high and low times and percentages. This would allow the program to consider the movement of different people in different situations. The data may or may not be the same for each aircraft, so each

would have to be investigated. Another area that might be researched is the setup of the plane to see if a different layout would change the boarding time. Different boarding strategies might work better if the layout was changed.

If more time was allowed, other boarding techniques may be of interest to simulate. An example of this may include a revised reverse pyramid. This method would split the plane into zones, and board "Outside-In" within theses zones. If the passengers were given a number and then told to board according to the number, it would cut down on the congestion. These numbers could be arranged so every passenger would have room to move around each other while boarding.

One other area of future research that should be explored is the trade off from implanting a new boarding strategy. Our final simulation, for example, may have a trade off that may hurt the airlines rather then improve. If the people were lined up and boarded according to speed, it may anger many passengers and cause them to take their business elsewhere. If that happened, the improvements would be far from beneficial for the company. All boarding plans have a trade off whether it is financial investments, making customers angry, or losing seats in a new layout. Airlines should explore these tradeoffs before changing their boarding techniques.

Our model also does not address the issue of de-boarding time, which should also be a major topic of study. If the de-boarding process can be minimized, this will help save time for airlines to generate higher revenue. This process would need to be modeled and have a specialized technique developed.

8. Conclusion

Without a doubt, the air travel industry has been plagued with a major problem with their internal operation. With a slight bit of help, aircraft can be boarded with ease using miniscule variations of the different existing methods. The task before us was to accurately model humans boarding an aircraft and to develop a better method of doing so.

Our data show what we believe to be an accurate representation of the real life boarding of an aircraft. Our basic model was enough to steer our research and allow us to improve to a more realistic model. On average, from our research, it will take any given aircraft anywhere from twenty minutes to over an hour to board.

Our proposed method was to use the behavior scalar that we devised as a means of sorting passengers and assigning them a boarding number and zone. We have simulated this and have determined that our method did significantly decrease the amount of time taken to board a Boeing 737-300, 757, and 777. This could potentially decrease the turnaround time for many flights and allow more aircraft to be in the air at any given time. This would increase the number of potential flights for the airlines per day and vastly increase their yearly revenue.

9. References

Airline Route Maps. 2007. 8 Feb. 2007 <www.airlineroutemaps.com>.

Bachmat, Eitan, et al. "Analysis of Airplane Boarding Times." *Ben-Gurion University Department of Computer Science* (2007): 1-24. *Home Page of Dr. Eitan Bachmat.* 11 Feb. 2007 <www.cs.bgu.ac.il/~ebachmat/managesubmit.pdf>.

"Continental Airlines - Aircraft." *Continental Airlines - Airline Tickets, Vacations Packages, Travel Deals, and Company Information.* 2007. 10 Feb. 2007 <http://www.continental.com/web/en-US/content/travel/inflight/aircraft/default.aspx?SID=2823788289AD49289529DBC878F3E575>.

"FAA Carry On Restrictions." *FAA Carry On Restrictions.* 2007. Suitcase.com. 8 Feb. 2007 <http://www.suitcase.com/FAA-Carry-on-Restrictions.html>.

Ferrari, Pieric, and Kai Nagel. "Robustness of Efficient Passenger Boarding in Airplanes." *ILS Verkehrssystemplanung und* (2004): 23 pp. *ILS Verkehrssystemplanung und -telematik.* 10 Feb. 2007 <www.vsp.tu-berlin.de/publications/airplane_boarding/15nov04.pdf>.

"Haseltine Systems." *Haseltine Systems.* 2007. 11 Feb. 2007 <http://www.haseltine.com/data.html>.

"The Mathematics of Aircraft Boarding." 2007. 10 Feb. 2007 <http://www.maa.org/devlin/devlin_05_06.html>.

Menkes. "Airplane Boarding." Ed. Menkes. 2007. 11 Feb. 2007 <http://www.public.asu.edu/~dbvan1/projects/boarding/boarding.htm>.

TransStats HomePage. 2006. USDOT. 8 Feb. 2007 <http://www.transtats.bts.gov/>.

United Airlines - Airline Ticket Reservations, Vacation packages and deals to Domestic.... 2007. 8 Feb. 2007 <www.united.com>.

van den Briel, Menkes H.L., J. Rene Villalobos, and Gary Hogg. "The Aircraft Boarding Problem." *Department of Industrial Engineering* (2005): 6 pp. *Arizona State University.* 10 Feb. 2007 <http://www.public.asu.edu/~dbvan1/papers/IERC2003MvandenBriel.pdf>.

Welcome to Hartsfield-Jackson International Airport. 8 Feb. 2007 <www.atlanta-airport.com>.

Executive Summary

Approximately 7,962,579 passengers passed through Atlanta's Hartsfield-Jackson International Airport in July of 2006; it is clear that the loading and unloading of aircraft in an efficient manner is the backbone of the airline industry. If the turn-around time between shorter flights was decreased in a large enough degree, airlines could possibly schedule more flights in the average day, theoretically skyrocketing yearly revenue. The parts that waste most of this turn around time, sadly, are the components of these crucial processes involved in boarding and deplaning of passenger jets.

We propose a new method of boarding which airlines could realistically implement that takes into account complex human behavior. This method would use an adjusted "Back-to-Front" technique that allows people who need more time to board first in each of the five or so zones. People are placed in these zones to board the plane based upon how long their estimated boarding time will be; in essence, each passenger is given an index that will assist in creating a zone. When compared to methods already in place in contemporary airlines, the model we have designed showed that airlines could save as much as eleven minutes of critical boarding time.

Our model calculates the amount of time for all passengers to board the plane by taking into account different types of passengers, some who need more time than others, and the time it ptakes for people to stow carry-on luggage. The model tries to capture the unpredictable nature of human beings without the use of overly-complicated mathematics. A number, which we coined the "Behavior Scalar", allows us to scale the time that someone takes based upon the ratio of time that the average passenger takes to board.

We used this model as basis for our program that simulates the time needed to board passengers using four different methods (including our own). To capture the capricious nature of different passengers on a given flight, the types of passengers, as well as other demographics, were chosen randomly. Boarding was then simulated using the four different methods. The program calculated the time for each passenger and generated a value for the total boarding time of the aircraft.

The three existing models that we simulated were the random boarding technique set by Southwest Airlines, the "Back-to-Front" method already in place with most airlines, and the "Outside-In" process that few airlines utilize. Out of these three, the Outside-In proved to be the most efficient of the existing airline systems on smaller aircraft. The random method ended up to be the most efficient in larger planes such as a Boeing 777, and was a close contender to the Outside-In method on a 737-300.

After creating a working model that reflected the data, both researched and generated, we began to look at the different factors that would affect total boarding time. The different types of passengers, such as children, passengers in wheelchairs as well as a subcategory of frequent travelers and first time travelers were taken into consideration. Each was assigned a time ratio, the aforementioned behavior scalar, that would scale

them with respect to our average passenger's boarding time. This allowed us to look at a broader range of situations than just the quintessential traveler that is able to board and sit down quickly.

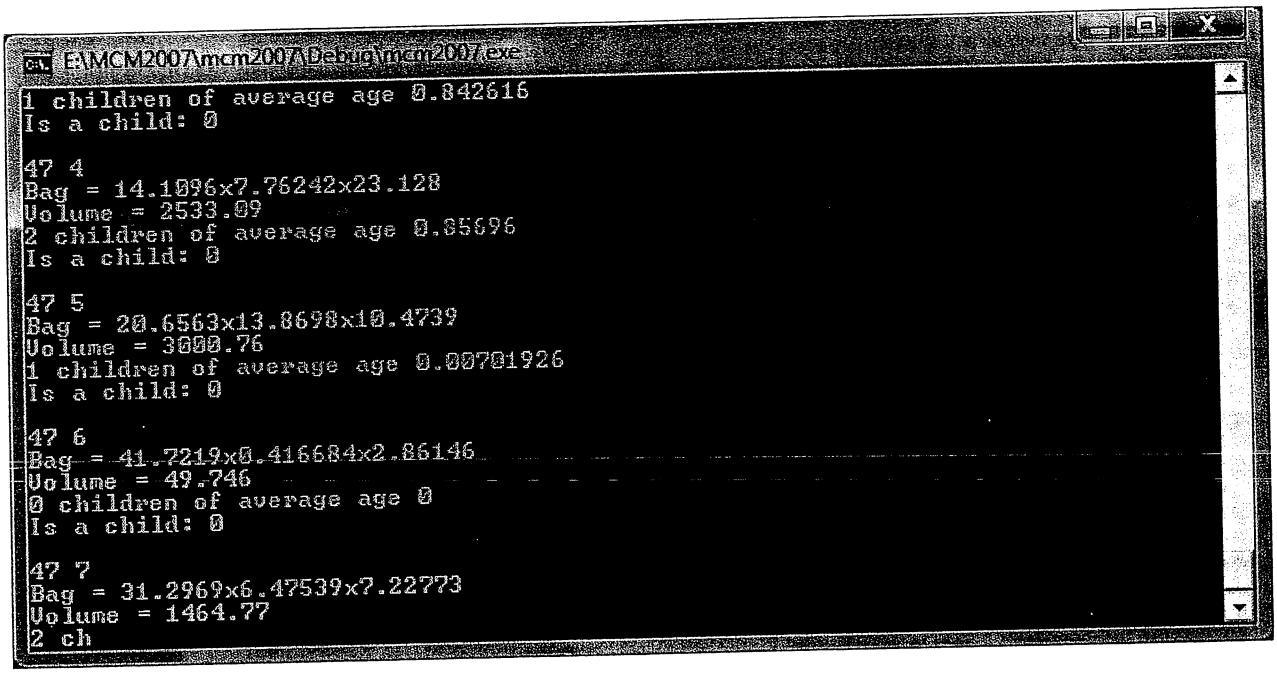
Using the results from our simulations, we were able to determine which method was superior and were able to come up with our own ideas on how to effectively board an airplane. We focus mainly on the time to board rather than the unloading time because it would have more of an impact on the turn around time of an airplane.

Our models, when tested against existing data that were collected by major airlines, proved to be very accurate. Using diverse demographics, we were anywhere from a matter of seconds to approximately seven minutes different from these real life times, which can be attributed to human nature.

The fact is that major airlines such as United are necessitated to rectify this issue of aircraft boarding. As mentioned before, with a superior method for boarding their aircraft, their yearly revenue could skyrocket and possibly bring the air travel industry out of the deep rut into which they have recently been placed. The future of the airline industry is inevitably in the hands of the intellectuals and mathematicians interspersed within our general population.

Appendix A – Code for Simulator Program

To quickly mention the different classes and the structure of the program, the simulation program was written in C++ and involved a driver code file *mcm.cpp*, a class called cNode for the implementation of the linked list in two files named *cNode.tem* and *cNode.h*. The actual simulation was broken into three files; *cPassenger.h* houses the cPassenger class with all of the necessary data for the passenger, *cSimulator.h* and *cSimulator.cpp* are the interfaces and implementation (respectively) for the cSimulator (simulation core) class.



cSimulator.cpp

```
#include "cSimulator.h"
#include "cPassenger.h"
#include <fstream>
#include <time.h>
#include <cmath>

using namespace std;

std::ostream &operator<<(std::ostream &os, cPassenger cp)
{
        os  << cp.myRowNumber << " " << cp.mySeatLetter << endl
                << "Bag = " << cp.myBagLength << "x" << cp.myBagWidth << "x" << cp.myBagHeight <<
endl
                << "Volume = " << cp.myBagVolume << endl
                << cp.myNumChildren << " children of average age " << cp.myAverageChildAge << endl
                << "Is a child: " << cp.isChild << endl;
        return os;
}

void cSimulator::randomSimulationStep()
{
        ofstream os("datarnd.txt");
        ofstream simuldata("simuldatarnd.txt");

        for(int i=0;i<ROWSINPLANE;i++)
```

```
                        {
                                os << myPlane[i][j];
                        }
                        os << endl;
                }
                os << endl;


        }
        int sum=0;
                for(int i=0;i<8;i++)
                        sum+=demographics[i];
        ofstream demo("demographicsrnd.txt");
        demo << (double)demographics[0]/(double)sum << " " << (double)demographics[1]/(double)sum <<
" " << (double)demographics[2]/(double)sum << " " << (double)demographics[3]/(double)sum << " " <<
(double)demographics[4]/(double)sum << " " << (double)demographics[5]/(double)sum << " " <<
(double)demographics[6]/(double)sum << " " << (double)demographics[7]/(double)sum << " " << endl;
}


void cSimulator::o2iSimulationStep()
{
        srand(time(0));
        ofstream os("datao2i.txt");
        ofstream simuldata("simuldatao2i.txt");
        for(int i=0;i<ROWSINPLANE;i++)
                for(int j=0;j<SEATSINROW;j++)
                        myPlane[i][j]='O';
        double stepsize=0.1; //0.1 seconds
        double currentTime=0.0;
        double totalTime=0.0;
        int minimumRow=0; //This keeps track of who is holding up the line
        int *randomness=new int[ROWSINPLANE*SEATSINROW+1000];
        int ctr22=0;
        int randsave[1000];
        for(int i=0;i<1000;i++)
                randsave[i]=0;
        int randctr=0;
        int *randomness2=new int[ROWSINPLANE*SEATSINROW];
        for(int i=0;i<ROWSINPLANE*SEATSINROW;i++)
                randomness2[i]=i;
        for(int i=0;i<10;i++)
        {
                int idx=rand()%(ROWSINPLANE*SEATSINROW);
                int idxswap=rand()%(ROWSINPLANE*SEATSINROW);
                int temp=randomness2[idx];
                randomness2[idx]=randomness2[idxswap];
                randomness2[idxswap]=temp;
        }
        for(int i=0;i<(ROWSINPLANE*SEATSINROW);i++)
        {
                if(i%SEATSINROW==0)
                {
                        if(rand()%2==0)
                        {
                                randomness[ctr22]=i;

                                if(rand()%3==0)
                                {
                                        randomness[++ctr22]=i+SEATSINROW-1;
                                        randsave[randctr++]=i+SEATSINROW*2-1;
                                }
                                else
                                {
                                        randomness[++ctr22]=i+SEATSINROW*2-1;
                                        randsave[randctr++]=i+SEATSINROW-1;
                                }
                        }
                        else
                        {
                                if(rand()%5==0)
```

```
                        break;
                if(checker==ctr22)
                        randomness[ctr22++]=randsave[r]; //put back the ones that weren't chosen
        }

        //cout << ctr22 << " " << ROWSINPLANE*SEATSINROW << endl;

        /*for(int i=0;i<(ROWSINPLANE*SEATSINROW)+500;i++)
        {
                if(i%6==3)
                {
                        if(rand()%2==0)
                        {
                                randomness[ctr22]=i;
                                if(rand()%3==0)
                                        randomness[++ctr22]=i+4;
                                else
                                        randomness[++ctr22]=i+5;
                        }
                        else
                        {
                                if(rand()%5==0)
                                        randomness[ctr22]=i+4;
                                else
                                        randomness[ctr22]=i+5;
                                randomness[++ctr22]=i;
                        }
                        ctr22++;
                }
        }*/
        randctr=0;
        /*for(int i=0;i<(ROWSINPLANE*SEATSINROW);i++)
        {
                if(i%6==3)
                {
                        if(rand()%3==0)
                        {
                                randomness[ctr22++]=i;

                                if(rand()%3==0)
                                {
                                        randomness[++ctr22]=i+4;
                                        randsave[randctr++]=i+5;
                                }
                                else
                                {
                                        randomness[++ctr22]=i+5;
                                        randsave[randctr++]=i+4;
                                }
                                cout << randomness[ctr22] << endl;
                        }
                        else
                        {
                                if(rand()%5==0)
                                {
                                        randomness[ctr22]=i+4;
                                        randsave[randctr++]=i+5;
                                }
                                else
                                {
                                        randomness[ctr22]=i+5;
                                        randsave[randctr++]=i+4;
                                }
                                randomness[++ctr22]=i;
                        }
                        //ctr22++;
                }
        }
        for(int r=0;r<=randctr;r++)
        {
                int checker=0;
```

```
                if(getLoc(myBoardingQueue,randomness[i])!=NULL)
                        cp2=getLoc(myBoardingQueue,randomness[i])->getData();
                myPlane[cp2.getRowNumber()][cp2.getSeatLetter()]='X';
                int currrow=cp2.getRowNumber();
                if(currrow>=minimumRow)
                {
                        //The following line does the following:
                        //Passenger A walks to their seat and holds up passenger B from getting to
theirs.
                        //One minus the preportion of time it takes for passenger B to get to their
seat from passenger A's seat
                        //is what is going to be added extra on to the time it takes for passenger B
to get to their seat, since
                        //Passenger B is being held up only from the time it takes them to get to
Passenger A's seat. If Passenger
                        //A takes a shorter time to sit down than passenger B takes to get to his
seat, NO EXTRA TIME WILL BE ADDED.
                        //totalTime+=(cp.getBoardingTime()-cp2.getBoardingTime());
                        totalTime+=(cp.getBoardingTime()-
cp2.getBoardingTime()*(cp.getRowNumber()/(cp2.getRowNumber() == 0 ? 1 : cp2.getRowNumber()))>=0 ?
(cp.getBoardingTime()-cp2.getBoardingTime()*(cp.getRowNumber()/(cp2.getRowNumber() == 0 ? 1 :
cp2.getRowNumber())) : 0;
                        //totalTime+=cp2.getBoardingTime()*(cp.getRowNumber()/(cp2.getRowNumber() ==
0 ? 1 : cp2.getRowNumber())); //Let him walk the rest of the way
                }
                /*else
                        totalTime+=cp2.getBoardingTime();*/
                if(cp2.getSeatLetter() > cp.getSeatLetter())
                {
                        totalTime+=abs(cp2.getSeatLetter() -
cp.getSeatLetter())*(double)rand()/(double)RAND_MAX*1;
                }
                minimumRow=cp2.getRowNumber();
                cp=cp2;
                //cout << "Total time: " << totalTime << " seconds with passenger " << randomness[i]
<< " boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds
to board alone." << endl;
                os << "Total time: " << totalTime << " seconds with passenger " << randomness[i] << "
boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds to
board alone." << endl;
                simuldata << totalTime << " " << randomness[i] << " " << cp2.getRowNumber() << " " <<
cp2.getBoardingTime() << endl;
                for(int i=0;i<ROWSINPLANE;i++)
                {
                        for(int j=0;j<SEATSINROW;j++)
                        {
                                os << myPlane[i][j];
                        }
                        os << endl;
                }
                os << endl;


        }
        int sum=0;
                for(int i=0;i<8;i++)
                        sum+=demographics[i];
        ofstream demo("demographicso2i.txt");
        demo << (double)demographics[0]/(double)sum << " " << (double)demographics[1]/(double)sum <<
" " << (double)demographics[2]/(double)sum << " " << (double)demographics[3]/(double)sum << " " <<
(double)demographics[4]/(double)sum << " " << (double)demographics[5]/(double)sum << " " <<
(double)demographics[6]/(double)sum << " " << (double)demographics[7]/(double)sum << " " << endl;
}


void cSimulator::btfSimulationStep()
{
        ofstream os("databtf.txt");
        ofstream simuldata("simuldatabtf.txt");
        for(int i=0;i<ROWSINPLANE;i++)
                for(int j=0;j<SEATSINROW;j++)
```

```
                minimumRow=cp2.getRowNumber();
                cp=cp2;
                //cout << "Total time: " << totalTime << " seconds with passenger " << randomness[i]
<< " boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds
to board alone." << endl;
                os << "Total time: " << totalTime << " seconds with passenger " << randomness[i] << "
boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds to
board alone." << endl;
                simuldata << totalTime << "-" << randomness[i] << " " << cp2.getRowNumber() << " " <<
cp2.getBoardingTime() << endl;
                for(int i=0;i<ROWSINPLANE;i++)
                {
                        for(int j=0;j<SEATSINROW;j++)
                        {
                                os << myPlane[i][j];
                        }
                        os << endl;
                }
                os << endl;


        }
        int sum=0;
                for(int i=0;i<8;i++)
                        sum+=demographics[i];
        ofstream demo("demographicsbtf.txt");
        demo << (double)demographics[0]/(double)sum << " " << (double)demographics[1]/(double)sum <<
" " << (double)demographics[2]/(double)sum << " " << (double)demographics[3]/(double)sum << " " <<
(double)demographics[4]/(double)sum << " " << (double)demographics[5]/(double)sum << " " <<
(double)demographics[6]/(double)sum << " " << (double)demographics[7]/(double)sum << " " << endl;
}

void cSimulator::idealSimulationStep()
{
        ofstream os("dataideal.txt");
        ofstream simuldata("simuldataideal.txt");

        for(int i=0;i<ROWSINPLANE;i++)
                for(int j=0;j<SEATSINROW;j++)
                        myPlane[i][j]='O';
        double stepsize=0.1; //0.1 seconds
        double currentTime=0.0;
        double totalTime=0.0;
        int minimumRow=0; //This keeps track of who is holding up the line
        int *randomness=new int[ROWSINPLANE*SEATSINROW+100];
        for(int i=0;i<ROWSINPLANE*SEATSINROW;i++)
                randomness[i]=0;
        double persfact=100.;
        double persfactprev=0;
        int ctrpf=0;
        double peoplet[10]={FREQBUSINESS, FTBUSINESS, FREQVACATION, FTVACATION, FREQSENIOR,
FTSENIOR, FREQDISABLED, FTDISABLED};

        for(int i=0;i<8;i++)
        {
                for(int j=0;j<ROWSINPLANE*SEATSINROW;j++)
                {
                        cPassenger cp88;
                        if(getLoc(myBoardingQueue,j)!=NULL)
                                cp88=getLoc(myBoardingQueue,j)->getData();

                        if(cp88.getPersonFactor()==peoplet[i])
                        {
                                //cout << sqrt(pow((cp88.getPersonFactor()-peoplet[i]),2.0)) << " " <<
peoplet[i] << endl;
                                randomness[++ctrpf]=j;
                        }
                }
        }
        for(int j=0;j<ROWSINPLANE*SEATSINROW;j++)
        {
```

```
                }
                /*else
                        totalTime+=cp2.getBoardingTime();*/
                if(cp2.getSeatLetter() > cp.getSeatLetter())
                {
                        totalTime+=abs(cp2.getSeatLetter() -
cp.getSeatLetter())*(double)rand()/(double)RAND_MAX*10;
                }
                minimumRow=cp2.getRowNumber();
                cp=cp2;
                //cout << "Total time: " << totalTime << " seconds with passenger " << randomness[i]
<< " boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds
to board alone." << endl;
                os << "Total time: " << totalTime << " seconds with passenger " << randomness[i] << "
boarding now in row " << cp2.getRowNumber() << " taking " << cp2.getBoardingTime() << " seconds to
board alone." << endl;

                //demo << "FTSenior FREQSenior FTBusiness FREQBusiness FTDisabled FREQDisabled
FTVacation FREQVacation" << endl;

                simuldata << totalTime << " " << randomness[i] << " " << cp2.getRowNumber() << " " <<
cp2.getBoardingTime() << endl;
                for(int i=0;i<ROWSINPLANE;i++)
                {
                        for(int j=0;j<SEATSINROW;j++)
                        {
                                os << myPlane[i][j];
                        }
                        os << endl;
                }
                os << endl;


        }
        int sum=0.00001;
                for(int i=0;i<8;i++)
                        sum+=demographics[i];
        ofstream demo("demographicsrnd.txt");
        demo << (double)demographics[0]/(double)sum << " " << (double)demographics[1]/(double)sum <<
" " << (double)demographics[2]/(double)sum << " " << (double)demographics[3]/(double)sum << " " <<
(double)demographics[4]/(double)sum << " " << (double)demographics[5]/(double)sum << " " <<
(double)demographics[6]/(double)sum << " " << (double)demographics[7]/(double)sum << " " << endl;
}
```

cSimulator.h

```
#ifndef _CSIMULATOR_H
#define _CSIMULATOR_H

#include "cNode.h"
#include "cPassenger.h"
#include <cstdlib>
#include <iostream>

#define RANDBREADTH 100
#define SIMUPOPULATIONDIST 0.00 //0.00 tells it to go completely random
#define SIMUPOPULATIONTYPE FTBUSINESS

//#define



using namespace std;



class cSimulator
{
private:
```

```
                                        double hmult2=(1.0-lmult2)-wmult2;
                                        double baglen2=lmult2*45.0;
                                        double bagheight2=hmult2*45.0;
                                        double bagwidth2=wmult2*45.0;
                                        //cout << ctr-SEATSINROW*ROWSINPLANE << endl;
                                        int numchildr2=0;
                                        double childage2=0;

pass[ctr]=cPassenger(randomage2,baglen2,bagwidth2,bagheight2,j,k,numchildr2,childage2,true);
                                        ctr++;
                                        ctr2++;
                                        j++;
                                        k++;
                                        k%=SEATSINROW;
                                }*/
                                //ctr++;
                        }

                }
                for(int i=0;i<SEATSINROW*ROWSINPLANE;i++)
                        cout << pass[i] << endl;

                myBoardingQueue=new cNode<cPassenger>();
                myBoardingQueue->setData(pass[0]);
                myBoardingQueue->setNext(NULL);
                for(int i=1;i<SEATSINROW*ROWSINPLANE;i++)
                {
                        cNode<cPassenger> *cn=new cNode<cPassenger>();
                        cn->setData(pass[i]);
                        cn->setNext(NULL);
                        insert(myBoardingQueue,cn);
                }
                //printList(myBoardingQueue);
                //      myBoardingQueue=insert(myBoardingQueue,new cNode<cPassenger>());
                //SEATSINROW=3;
                //ROWSINPLANE=20;
                myPlane=new char*[ROWSINPLANE+50];
                for(int i=0;i<ROWSINPLANE;i++)
                        myPlane[i]=new char[SEATSINROW+10];
        }

        void randomSimulationStep();
        void btfSimulationStep();
        void o2iSimulationStep();
        void idealSimulationStep();




};

#endif
```

mcm.cpp

```
#include <iostream>
#include "cNode.h"
#include "cPassenger.h"
#include "cSimulator.h"
#include <time.h>
#include <cstdlib>
int main()
{
        srand(time(0));
        cSimulator cs;
        cSimulator cs1;
        cSimulator cs2;
```

```
        //The above will represent 18A as the passenger's seat
        double mySeatingIndex;  //To be calculated later once we have a model...this will allow for
the seating assignment
                                                        //of passengers in an efficient manner.

public:

        inline cPassenger(){
                myBagLength=1.;
                myBagWidth=1.;
                myBagHeight=1.;
                myBagVolume=1.;
                mySeatLetter=0;
                myRowNumber=1;
                mySeatingIndex=0.0;
                myNumChildren=0;
                myAge=25.;
                myAverageChildAge=0.0;
                isChild=false;
        }

        inline cPassenger(double age, double bl, double bw, double bh, double rn, char sletter,
double numch, double cage, bool child)
        {
                //myTimeToBoard=12;
                myTimeToBoard=8+(double)rand()/(double)RAND_MAX*5;
                //cout << myTimeToBoard << endl;
                myBagLength=bl;
                myBagWidth=bw;
                myBagHeight=bh;
                myBagVolume=bl*bw*bh;
                mySeatLetter=sletter;
                myRowNumber=rn;
                mySeatingIndex=0.0;
                myNumChildren=numch;
                myAge=age;
                myAverageChildAge=cage;
                isChild=child;
        }

        inline cPassenger(double age, double bl, double bw, double bh, double rn, char sletter,
double numch, double cage, double personfa, int numbags, int exper)
        {
                //myTimeToBoard=12;

                myPersonFactor=personfa; //Set the experience level scalar for the passenger
                myNumBags=numbags;
                myBagLength=bl;
                myBagWidth=bw;
                myBagHeight=bh;
                myBagVolume=bl*bw*bh;
                mySeatLetter=sletter;
                myRowNumber=rn;
                mySeatingIndex=0.0;
                myNumChildren=numch;
                myAge=age;
                myAverageChildAge=cage;
                if(myNumChildren == 0)
                        myAverageChildAge=0.0;
                isChild=false;
                if(myPersonFactor <= 0)
                {
                        if(myAge >= 65 && myAge < 85)
                                myPersonFactor=((myTimesInTransit >= 10) ? FREQSENIOR : FTSENIOR);
                        else if(myAge >= 85)
                                myPersonFactor=((myTimesInTransit >= 10) ? FREQDISABLED : FTDISABLED);
                        if(myTimesInTransit > 60 && myAge >= 27 && myAge <65)
                                myPersonFactor=((myTimesInTransit > 60 && myTimesInTransit < 80) ?
FTBUSINESS : FREQBUSINESS);

                }
```

```
                myBagVolume=l*w*h;
        }

        inline void setRowNumber(int n)
        {
                myRowNumber=n;
        }

        /*inline double getSeatingIndex()
        {
                return mySeatingIndex;
        }*/

        inline void setAverageChildAge(double age)
        {
                myAverageChildAge=age;
        }

        inline void setBoardingTime(double n)
        {
                myTimeToBoard=n;
        }

        inline void setNumChildren(int n)
        {
                myNumChildren=n;
        }

        inline double getPersonFactor()
        {
                return myPersonFactor;
        }
        friend std::ostream &operator<<(std::ostream &os, cPassenger cp);
};

#endif
```

cNode.h

```
#ifndef _CNODE_H
#define _CNODE_H


template<typename T> class cNode
{
protected:
        T myData;
        cNode *next;
public:
        //Precondition: data is valid
        //Postcondition: myData is set
        void setData(T data) { myData=data; };

        //Precondition: None
        //Postcondition: Data is returned
        T getData() { return myData; };

        //Precondition: nx is a valid pointer
        //Postcondition: link is set
        void setNext(cNode *nx) { next=nx; };

        //Precondition: next is a valid pointer
        //Postcondition: link pointer is returned
        cNode<T> *getNext() { return next; };

        //Precondition: None
        //Postcondition: Node is copied without chain
        inline cNode<T> *copy()
        {
```

```
                return NULL;
        if(pos==0)
                return head;
        return getLoc(head->getNext(),pos-1);
}

template<typename T> cNode<T> *copyList(cNode<T> *head, cNode<T> *nhead)
{
        if(head==NULL)
        return head;
        if(head->getNext()==NULL)
        {
                cNode<T> *nd=new cNode<T>();
                nd->setData(head->getData());
                nd->setNext(new cNode<T>);
                nhead->setNext(nd);
                return nd;
        }
        cNode<T> *nd=new cNode<T>();
        nd->setData(head->getData());
        nd->setNext(head->getNext());
        nd->getNext()->setNext(new cNode<T>());
        if(nhead->getNext()!=NULL)
                return copyList(head->getNext(),nhead->getNext()->getNext());
}

template<typename T> void printList(cNode<T> *head)
{
        if(head->getNext()==NULL)
                return;
        cout << head->getData() << endl;
        printList(head->getNext());
}
```