Modeling Airline Overbooking Using Monte-Carlo Simulation

Ashish Neupane, Minh Nguyen, Zian Jiang
Math - 87 Final Project

05/08/2017

Abstract

Airline industry contains a broad class of challenges that involve allocation of resources, maximization of passenger satisfaction and of return on investment. Overbooking is one such task, in which as many tickets as possible have to be sold and bumped passengers have to be compensated appropriately to retain the airline's reputation. This project introduces 2 different approaches in which Monte Carlo simulation is used to simulate the number of passengers who will show up to the flight, and the number of passengers who are willing to be bumped, based on past data. Then, the number of tickets to oversell is decided accordingly based on the ticket price and compensation plan. Two different algorithms are used in the two approaches: a naive algorithm that only simulates one flight, and a more complex algorithm that simulates all flights over the course of one year, taking rescheduling into account for bumped passengers. The route in consideration is Boston Logan International Airport (BOS) to Ronald Reagan Washington National Airport (DCA) by American Airlines.

Introduction

Our awareness and interest in airline overbooking stem from the recent United Airlines overbooking incident [6], in which a passenger was violently assaulted and dragged off an overbooked flight, the video clip of which we found very disturbing and unpleasant. A cursory search for the airline overbooking practices led us to the realization that airlines business is perhaps one of the most difficult business to stay profitable; it requires a balance of emphasis on revenue and satisfaction of customers. The 2002 MCM problem [8] on airline overbooking shows that the importance of the issue has not gone unnoticed. The availability of flight related data from the Bureau of Transportation Statistics [1] also played a part in our decision to model this problem.

In the flight overbooking problem, the number of tickets to sell directly influences revenue; selling too many tickets may result in an overbooked flight, and some passengers may be denied boarding, which impacts the airline's reputation. On the other hand, not overbooking leaves empty seats because some tickets will be canceled, which reduces the revenue. Thus, airlines may ask for volunteers to give away their seats or refuse boarding to certain passengers in exchange for a compensation that may include an additional free ticket, an upgrade in a later flight, or cash.

Airline companies thus oversell tickets to ensure that 100% of available supply will be used to result in maximum return on investment. In essence, the objective is to maximize revenue from ticket sale while satisfying as many passengers as possible and minimizing the total compensation. This decision is based on a number of parameters, including the size of the plane, ticket price, show-up rate, involuntarily bumped rate, and flight schedule.

The goal of this model is to reflect the randomness and unpredictability of the show-up rate on a certain flight and of the ratio between passengers who are and are not willing to give up their seats, while maintaining some basic procedures when addressing passengers. This randomness is a binomial distribution that is approximated to a normal distribution. This probabilistic modeling is handled using Monte Carlo simulation.

Assumptions and Limitations

1. The average ratio of number of involuntarily bumped passengers to total number of bumped passengers is kept constant throughout the whole year. This is because the data on bumped passengers is very scarce.

2. All tickets that are up for sale will be bought; there will be no leftover tickets.

3. The flight schedule is the same over the year. This information is available on the internet [3].

4. Every bumped passenger will accept the compensation and agree to give up their seat. The compensation plans for each approach will be fully described in Strategy of One Flight Model and Strategy of Full Year Model.

5. Aircraft for this route is the same over the year, which is the Airbus A319 [7].

6. Ticket price is the same for every customer and is the average price of first class, business class, and economy class.

7. Ticket price is constant for every quarter of the year. This is because that the data on average price of this route that can be found is only specific to quarters.

8. The results are not restricted to integers.

9. The ratio of the number of passengers who show up to the total number of tickets on sale is not available, because we do not have access to the financial information of American Airlines; we used the load factor, which is the ratio of show ups to seating capacity, to replace that ratio. However, airlines usually do not overbook by a large amount and usually tickets sell out. Thus, using load factor should not significantly affect the accuracy of the model.

Model

Overall description of model:

This project seeks to model daily flights of American Airlines from the Boston Logan International Airport (BOS) to the Ronald Reagan Washington National Airport (DCA) in different overbooking scenarios. There are 126 seats [7] on each aircraft, and the price for each individual seat varies by time of the year around $175 [1]. The show-up rate of passengers and the ratio of the number of involuntarily bumped passengers to the number of voluntarily bumped passengers are derived from data collected from the Bureau of Transportation Statistics [1]. To find the optimal overbooking strategy, we introduce 2 different approaches:

- One Flight Model: Simulate one single flight.
- Full Year Model: Simulate all flights in a year based on the schedule posted on the American Airlines website [3].

In both model, we use Monte Carlo simulation to calculate the following values:

- The number of passengers who show up, based on the number of tickets sold and the expected show-up rate.
- The number of passengers that are voluntarily bumped and involuntarily bumped in case the plane is overbooked, using the number of bumped passengers and the expected ratio between the two categories of passengers.
- The amount of compensation paid for bumped passengers, determined by the strategies described in the respective sections.

The compensation strategies of the two models are different.

Mathematical methods and concepts used in model:

- Since the sample size of passengers who show-up is constant and the average show up rate is the probability of success in the Bernoulli random variable, we can use the binomial distribution to approximate the number of passengers who show up for the flight. The number of involuntarily bumped passengers with every bumped passengers can be calculated in the same way.
- In the model, we use the *de Moivre-Laplace theorem*, which states that:
  As n grows large, for k in the neighborhood of np we can approximate:

$$\binom{n}{k} p^k q^{n-k} \approx \frac{1}{\sqrt{2\pi npq}} exp(-\frac{(k-np)^2}{2npq}), \quad p+q=1, \quad p,q>0$$

In short, the normal distribution may be used as an approximation to the binomial distribution with large n.

- Given the expected number of passengers who show up and ratio of the number of involuntarily bumped passengers to the number of voluntarily bumped passengers, we calculate the real value by representing the probability of each possible value in a binomial distribution, and approximate the distribution using the *de Moivre-Laplace theorem*.

- Monte Carlo simulations are used to account for the randomness and unpredictability of various variables in the model, such as the show-up rate for a single flight. This method relies on repeated random sampling to acquire numerical results of problems that are deterministic in nature. By the Central Limit Theorem, the sample mean converges to the theoretical mean as the number of samples goes to infinity.

Strategy of One Flight Model

This model simulates one single flight from the Boston Logan International Airport (BOS) to the Ronald Reagan Washington National Airport (DCA). If the number of passengers who show up for the flight is greater than the capacity of the aircraft, we use the following compensation strategy for bumped passengers:

- Have 2 levels of compensation: 2 times the ticket price and 4 times the ticket price.
- First, ask for volunteers to change flight with the lower compensation.
- Then, if the plane is still overbooked, randomly bump passengers until the plane is not overbooked anymore and pay every bumped passenger (voluntarily and involuntarily) the higher compensation.

This model does not account for flight rescheduling for bumped passengers, and is implemented in the MATLAB (see Appendix::codes 1, 2, 3).

Results and Discussion of One Flight Model

S.No.	Extra Tickets	Revenue ($)	Ratio of Bumps	Ratio of Involuntary Bumps
1	0	17520	0	0
2	10	18893	0	0
3	20	20271	2.44×10^{-4}	1.73×10^{-5}
4	30	21116	0.0074	3.92×10^{-4}
5	50	19923	0.0427	0.0024

Table 1. Results of One-Flight Model at Critical Points

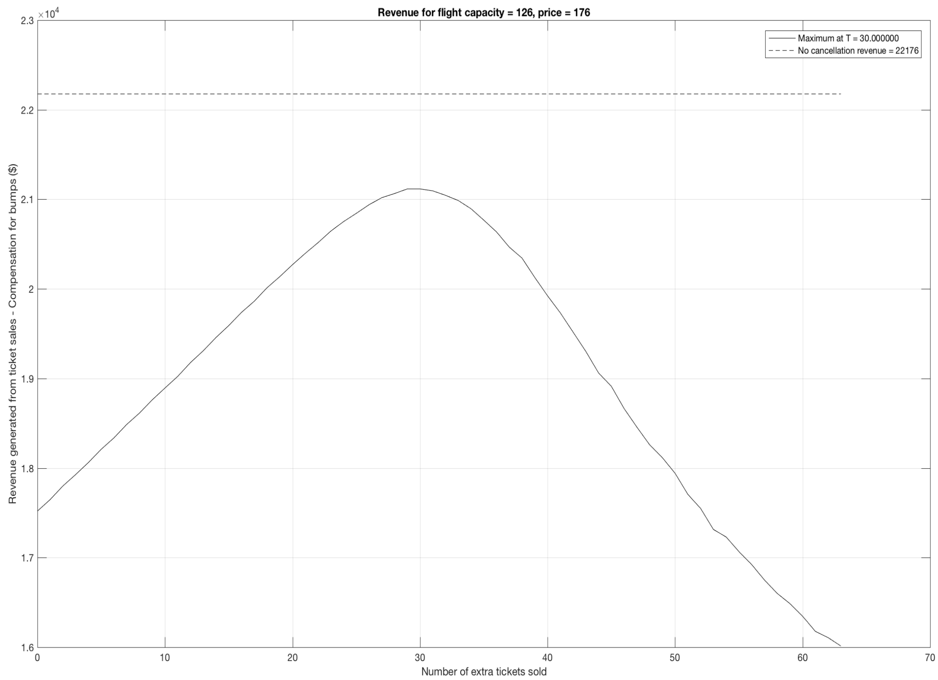

Figure 1. Revenue generated in the One Flight model

As shown in Figure 1, there is a local maximum revenue generated by the airlines from ticket sales after giving out compensation when the flight is overbooked by 30 seats. This maximum feasible revenue is around $1000 less than the airlines would make if every ticket was sold and none was canceled. If there is no overbooking, they fall short by around $4600. A difference of $3000 might not sound like a lot, but the loss adds up when you consider the number of flights each airlines makes. This is why airlines overbook flights, to get as close as possible to the maximum feasible revenue without upsetting too many customers. This is beneficial to the customers too because the ticket price would be much higher than they are now if there were no overbooking.

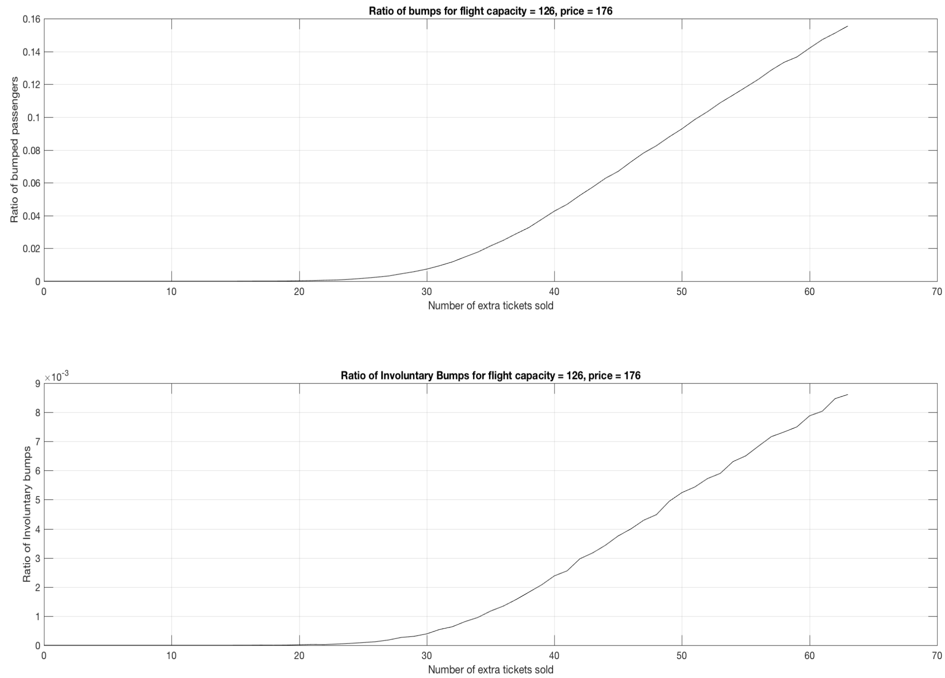

Figure 2. Passenger statistics in the One Flight model

From the graph in Figure 2, the number of bumps seems to stay reasonably low until T = 20 before rising steeply. This makes the ratio of bumped passengers 0.74% in the case of maximum feasible revenue. From a greedy airline's point of view, the maximum revenue might be the optimal point but bumping 740 out of every 100000 customers is a bit too many bumps to stay out of trouble, especially when on average 40 of those 740 bumped passengers are bumped involuntarily. So, two points until the ratio of bumps stays pretty small (T = 10, and T = 20) have been included in Table 1 for discussion. Overbooking by only 10 seats seems to keep the airlines out of trouble related with bumps but in the long run, it generates a revenue of only 89% of the maximum revenue. On the other hand, overbooking by 20 increases that revenue to 96% of the maximum but 50 out of every 200000 customers are bumped and of those 50, only 4 are involuntarily bumped. So, the optimal region is somewhere between 10 and 20 extra tickets. This model can not be used to draw a tighter bound on the estimated optimal number of extra tickets to sell. Another problem with this model is that it does not account for the fact that bumped passengers need to be put in a future flight and that the compensation depends on the time by which the passenger is delayed in arriving at the destination.

Strategy of Full Year Model

This model simulates all flights from the Boston Logan International Airport (BOS) to the Ronald Reagan Washington National Airport (DCA) over the course of a year. The model keeps a queue of bumped passengers for rescheduling. The compensation strategy of this model pertains to the rules of overbooking set forth by the federal Department of Transportation [2]. For each flight:

- If the number of passengers who show up for the flight is greater than the capacity of the plane, the model puts the bumped passengers at the end of the queue, and also keeps track of their original schedule.

- If the number of passengers who show up for the flight is smaller than the capacity of the plane, the model fills the available seats with passengers at the front of the waiting queue. For each passenger, the model calculates the difference between the arrival time of that passenger's original schedule and new schedule. If the new arrival time is within 1 hour of the original schedule, the passenger receives no compensation. If the new arrival time is within 1 to 2 hours of the original schedule, the passenger receives a compensation equivalent to 2 times the ticket price. Else, if the new arrival time is within more than 2 hours of the original schedule, the passenger receives a compensation equivalent to 4 times the ticket price.

The model keeps track of the waiting queue to make sure that every passenger will only have to wait in the queue for at most 3 days to be rescheduled. Since the flight schedule for every day is the same, if a passenger is likely to be rescheduled to a different flight, it is likely to happen within 1 day of the original schedule. If a passenger stays on the queue for more than 3 days, it is likely that every plane will be overbooked and the that passenger will not be rescheduled at all. The model will stop its execution in this case. This model is also implemented in MATLAB(see Appendix::codes 3,4,5,6,7).

Results and Discussion of Full Year Model

S.No.	Type	Extra-T	Revenue($)	Show-ups	Bumps	Involuntary	Queue/night
1	No overbooking	0	107.5 mil.	612920	0	0	0
2	No Bumps	6	112.7 mil.	642100	0.639	0.0382	0
3	No Involuntary	10	116.1 mil.	661570	15.047	0.8054	0.0026
4	1^{st} quartile	13	118.7 mil.	676160	102.870	5.661	0.0179
5	Mid-point	16	121.1 mil.	690760	501.717	27.635	0.0921
6	3^{rd} quartile	19	123.3 mil.	705350	1806	99.821	0.3801
7	Max. Revenue	22	124.4 mil.	719930	4973.1	274.907	1.475
8	Max. Delay	24	123.8 mil.	729670	8614.4	475.447	4.158

Table 2. Results of Full Year Model at Critical Points

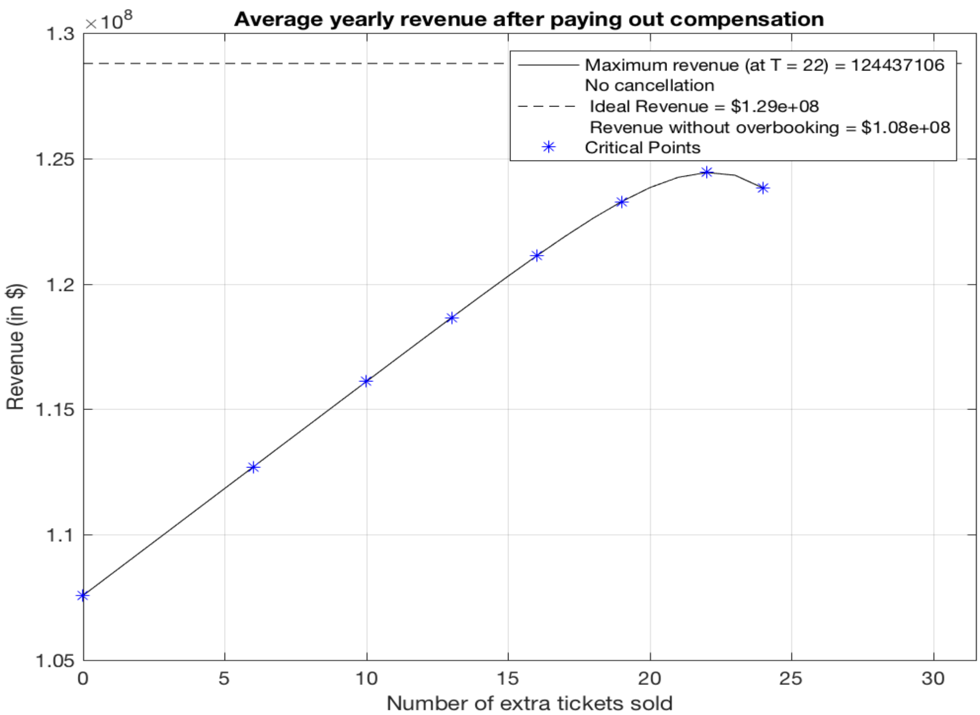

Figure 3. Revenue Generated for the Full Year Model

The first noticeable thing in Figure 3 is that on average, airlines do not make even close to the money they would make in an ideal world where every ticket is sold and not canceled. Almost all airlines lose a lot of money because of canceled tickets. American Airlines falls around 22 million USD per year short of the revenue generated in an ideal world even for a two hour flight from Boston to DC. Extend that to thousands of more domestic and international flights, overbooking might make a difference between an airline being able to offer their services and going out of business. As with any business, airlines want to maximize their revenue from ticket sales without making the customers unhappy. Note that there is a maximum revenue when the airline overbooks by around 17% (see Figure 3). At that point, however, the average fraction of bumped passengers is about 0.69% (see Table 2), which can be disastrous to their stock value and reputation. For this reason, other critical points based on the number of bumps and number of involuntary bumps need to be studied to get as close to the maximum revenue as possible. On the other side of the maximum revenue, the graph is cutoff at the point where some passengers had to wait for more than three days before getting onto another flight. Anything greater than the number of extra tickets at maximum revenue (T = 22) can be ruled out because the revenue decreases and the number of bumps increases.

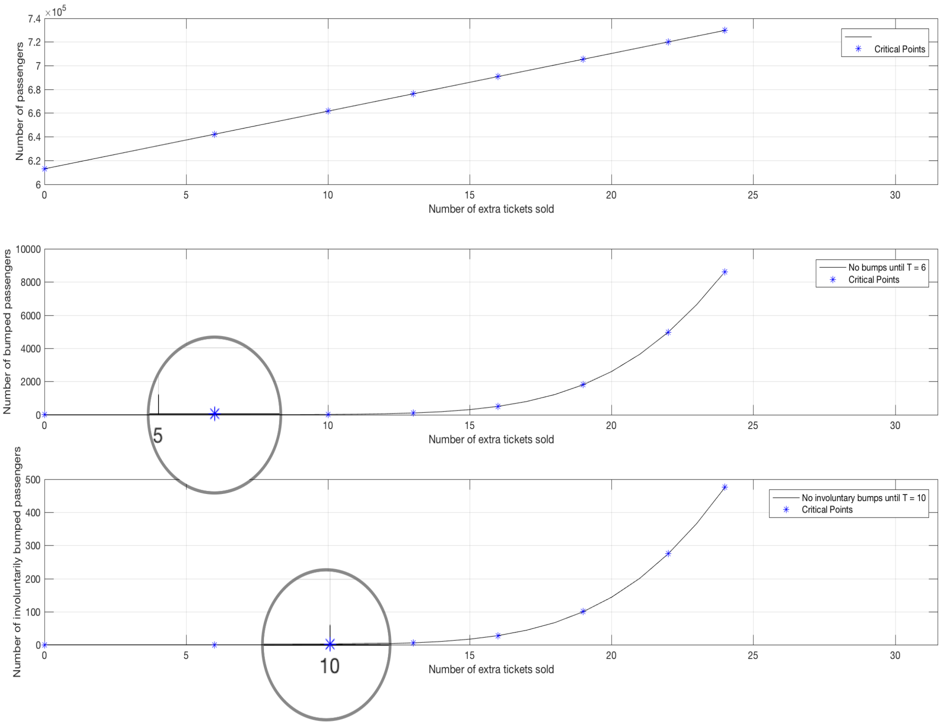

Figure 4. Passenger statistics for the Full Year Model

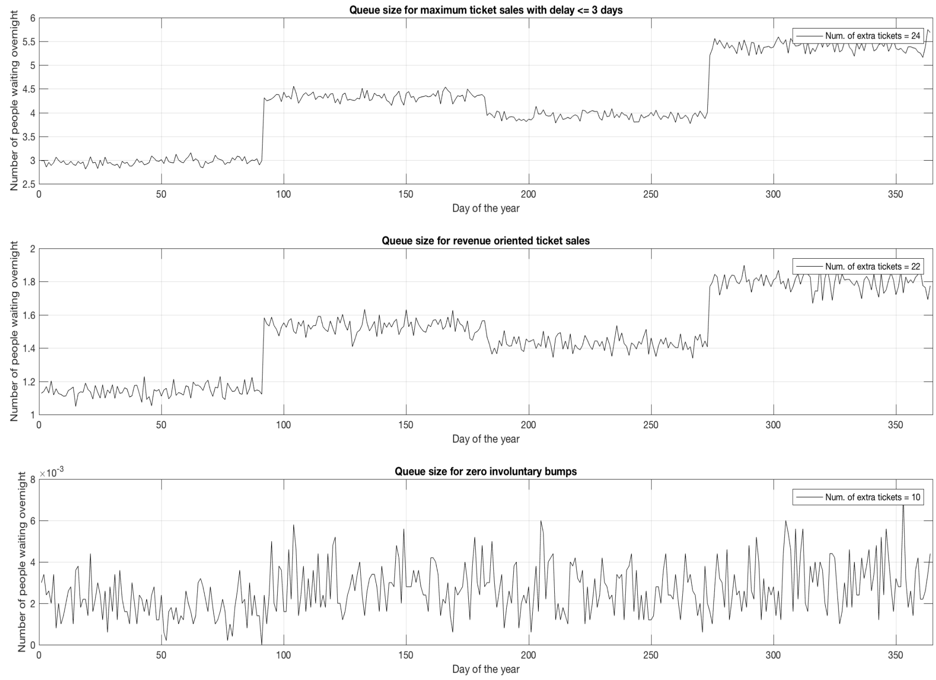

Figure 5. Queue size at the boundaries of the optimal region

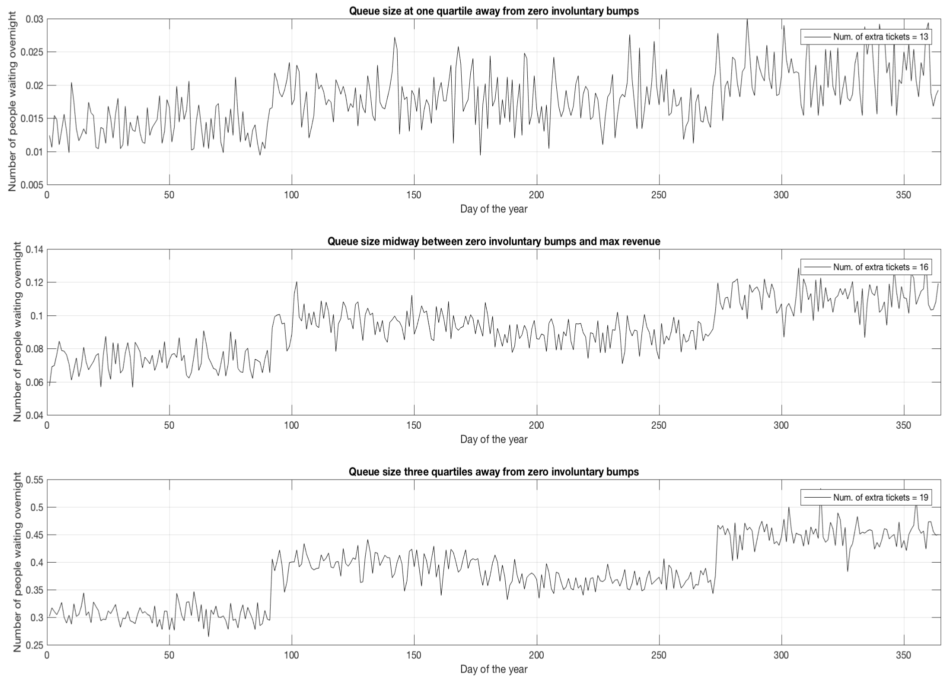

Figure 6. Queue size at quartiles inside the optimal region

A couple of interesting points on Figure 4 are the cases when there are effectively zero voluntary bumps and zero involuntary bumps. The second critical point highlighted in the graphs represents the number of extra tickets sold when zero passengers were bumped on average. One could argue that this is the optimal point from an ethical point of view but some people like getting bumped to later flights for compensation [5]. In addition, the revenue generated when there are no bumps is only 90% of the feasible maximum revenue compared to the 93% generated when there are a few bumps but no involuntary bumps. For this reason, the number of extra tickets for which effectively no passenger is bumped involuntarily seems to be the more important critical point.

The feasible optimal region in the graph lies under the revenue curve and between 10 and 22 extra tickets. This region can be divided into four segments separated by three quartiles between the boundaries of the feasible optimal region. The revenues generated for these three points seem to increase linearly with number of extra tickets. From the plot of the average number of people who waited overnight (see Figure 5), effectively zero people wait overnight when the airline is overbooked by 10 seats, and on average 1.5 people wait overnight everyday i.e. around 45 people wait overnight in a month when the airline is overbooked by 22 seats to maximize revenue. That number is a bit too high for comfort, and on top of that, around 10 more people wait overnight on average in the months October to December. Holiday season is a bad time to make people wait long for flights. When 13 extra tickets are sold, every 5 months, an average of 3 passengers have to wait overnight with that number rising up to as high as 4 in the final quarter of the year. Midway between the boundaries of the optimal feasible region ($T = 16$) , 3 customers per month are expected to wait overnight for available seats, and for the critical point generating 99% of the maximum revenue ($T = 19$), the expected number of customers who have to wait overnight is 12 per month with that number rising as high as 15 customers per month.

This brings us to a very difficult question of how many tickets the airlines should oversell. An exact number is probably not the right way to answer this question because after all, this is a probabilistic model and the answer depends on the point of view. From a greedy economic point of view, the optimal point is probably close to the max revenue ($T = 22$). From an ethical point of view, the optimal point may be close to the case of zero involuntary bumps ($T = 10$) because only people who want to be bumped are bumped in that scenario. The zero involuntary point seems to be a win-win situation for both the customers and the airlines. The increase in the average number of people who wait overnight per day seems to grow steeper with the increase in number of extra tickets sold. 0, 0.01, 0.1, 0.4, and 1.5 people are expected to wait overnight every day when the flight is overbooked by 10, 13, 16, 19, and 22 respectively. The revenue generated for these cases are 93%, 95%, 97%, 99%, and 100% of the maximum feasible revenue respectively. Judging from the numbers, selling 10 to 16 extra tickets might strike a balance between the greedy analysis and the ethical analysis, but the model needs to be improved further for a more precise estimate.

Further Research/Improvements

The goal of this project was only to maximize the revenue while minimizing the number of bumped passengers, while the potential impact on other areas, such as stock performance, is yet to be explored. This impact is expected to be significant. As shown in the United Airlines incident, its shares dropped $1.4 billion, or 4%, only one day after the passenger-removal controversy [4].

In this model, it is assumed that every ticket will be bought so that there will be no leftover ticket (Assumption 2). However, that assumption can be relaxed, and the number of tickets that will be sold can also be simulated, using Monte Carlo simulation. This would require access to the financial report of American Airlines.

Also, the result is a number of tickets to oversell that optimizes the revenue and the number of involuntary bumps. In essence, it is a static optimization problem. This can be improved by making a dynamic simulation and figuring out the extra number of tickets to oversell accordingly before every flight. This would require high synchronization between data. For example, the show-up rate of the flight has to be updated immediately after every flight, so that the number of tickets to oversell can be determined accordingly based on the latest data.

Last but not least, the model will be better if the interval of data can be smaller. In this model, the interval is quarterly. Access to more data can be used to decrease the size of the interval and hence, improve the model.

Conclusion

Overbooking is a necessity for most airlines. The losses incurred due to canceled tickets will increase the tickets prices if overbooking is avoided altogether, which is bad for the economy of airlines industry. However, overbooking by the right amount can increase the revenue while keeping all customers happy. American Airlines flights from Boston to Washington D.C. using Airbus A319 with a seating capacity of 126 should overbook by around 8% to 12% when the average show-up rate is around 85%. Considering the recent events related to flight overloads, keeping the overbooking rate close to 8% rather than 12% can help the airlines maintain a good reputation.

References

[1] Bureau of Transportation Statistics *Airlines and Airports*
http://www.rita.dot.gov/bts/sites/rita.dot.gov.bts/files/subject_areas/airline_information/index.html

[2] Department of Transportation *Fly Rights - A Consumer Guide to Air Travel*
http://www.transportation.gov/airconsumer/fly-rights#Overbooking

[3] American Airlines *American Airlines Daily Schedule from Boston (BOS) to Washington DC (DCA)*
http://fortune.com/2017/04/11/united-airlines-stock-drop

[4] Lucinda Shen *United Airlines Stock Drops $1.4 Billion After Passenger-Removal Controversy*
http://www.aa.com/travelInformation/flights/schedule

[5] Yuki Noguchi *Flight Overbooked? Perfect: These Frequent Flyers Want To Get Bumped*
http://www.npr.org/2017/04/27/525415187/flight-overbooked-perfect-these-frequent-flyers-want-to-get-bumped

[6] Holly Yan, Christina Zdanowicz and Emanuella Grinberg *Backlash erupts after United passenger gets yanked off overbooked flight* CNN ,Wed April 12, 2017
http://www.cnn.com/2017/04/11/travel/united-customer-dragged-off-overbooked-flight/

[7] Seat Guru *Aircraft seat maps, flight shopping and flight information*
https://www.seatguru.com

[8] Bill Fox and Rich West *Airline Overbooking, Mathematical Contest in Modeling*
http://www.comap.com/undergraduate/contests/mcm/contests/2002/problems/mcm.php

APPENDIX
MATLAB CODE

1 Script: optimizeTickets.m

```
%%   Optimizer Script
%    Defines a vector for the number of extra tickets sold
%    Defines the constants price per seat and seating capacity
%    Get the results
%    Optimize the number of tickets sold


%% Define Constants

% R_s : Average show up rate
R_s = (83 + [0.2, 0.6, 0.5, 0.8])/100;
% R_i : ratio of involuntary bumps to voluntary bumps
R_i = 0.0552;
% S : Seating Capacity
S = 126;
% C : Price of flight per seat
C = 176;

% numSim: Number of simulations
numSim = 10000;
% Range of extra tickets we can sell
extraTickets = (0:1:S/2)';

% Ideal case : no cancellation
idealRevenue = C * S;

%% Run Simulations for all number of extra tickets
% Preallocating for speed
revenue = zeros(size(extraTickets));
ratioBumped = zeros(size(extraTickets));
ratioInv = zeros(size(extraTickets));
for i = 1 : length(extraTickets)
    [revenue(i), ratioBumped(i), ratioInv(i)] =  ...
                        simulateFlights(numSim, extraTickets(i), C, S);
end

%% Print out data for the critical points
```

```matlab
critical = [find(extraTickets == 0,1), find(extraTickets == 10,1), ...
            find(extraTickets == 20, 1), find(revenue == max(revenue), 1),...
            find(extraTickets == 40, 1)];
disp([{'No.', '# Extra', 'Revenue','Ratio Bumps','Ratio Involuntary'}; ...
      num2cell((1:5)'), ...
      num2cell(extraTickets(critical)), ...
      num2cell(revenue(critical)), ...
      num2cell(ratioBumped(critical)), ...
      num2cell(ratioInv(critical))])

%% Graphical Analysis

% Plot the revenue as a function of number of extra tickets sold
figure (1)
plot(extraTickets, revenue, '-k', extraTickets, ...
                   idealRevenue * ones(size(extraTickets)), '--k')
grid on
legend(sprintf('Maximum at T = %f',extraTickets(revenue == max(revenue))),...
       sprintf('No cancellation revenue = %.0f', idealRevenue))
xlabel('Number of extra tickets sold')
ylabel('Revenue generated from ticket sales - Compensation for bumps ($)')
title(sprintf('Revenue for flight capacity = %.0f, price = %.0f', S, C))
saveas(gcf, 'RevenueNaive.png')

% Plot the normalized revenue and the ratio of involuntary bumps
figure (2)
subplot 212
plot(extraTickets, ratioInv, '-k')
grid on
ylabel('Ratio of Involuntary bumps')
xlabel('Number of extra tickets sold')
title(sprintf('Ratio of Involuntary Bumps for flight capacity = %.0f, price = %.0f', S, C))

```

```
subplot 211
plot(extraTickets,ratioBumped, '-k')
grid on
ylabel('Ratio of bumped passengers')
xlabel('Number of extra tickets sold')
title(sprintf('Ratio of bumps for flight capacity = %.0f, price = %.0f'
    , S, C))
saveas(gcf, 'BumpNaive.png')
```

2 Function: simulateFlights.m

```
function [R, bRatio, invbRatio] = simulateFlights(numSim, T, C, S, printon)
    % simulateFlight: To perform a Monte Carlo simulation and calculate the
    % parameters listed below for a given flight using a constant average show
    % up rate and average rate of involuntarily bumped passengers.
    %
    % Inputs:
    % ` numSim: Number of simulations to run
    %   T: Number of extra tickets sold
    %   C: Flight price per seat
    %   S: Plane's seating  capacity
    %
    % Output:
    %   R: Total revenue from selling ticket - amount of compensation paid
    %   bRatio: The ratio of bump passengers / total capacity
    %   invbRation: The ratio of involuntarily bump passengers / total capacity

assert(nargin == 5 || nargin == 4, 'Invalid number of inputs');

% revenue from selling ticket
% revenue = (T + S) * C;

% Overbook rate of airlines, number of extra tickets / capacity
% Not taken from reliable source, if we do simulation with the company's
% data in hand, this will be more reliable
ovbRate = 0.05;

% load factor of airlines, taken from United report
loadFactor = 0.829;

% show up rate of passengers with ticket
r_s = loadFactor/(1 + ovbRate);
```

```
% Ratio of involuntary bump / total bump (taken from United Airlines
    2016)
r_i = 0.0552;

% Average compensation, number of bumps, and involuntary bumps
avgComp = 0;
avgBmp = 0;
avgInvb = 0;
avgShowup = 0;
avgRevenue = 0;

% 2 levels of compensation
level1 = C*2;
level2 = C*4;

for i = 1:numSim
    [N, numInvbump, numVBump] = flightInfo(T, S, r_s, r_i);
    numBump = numInvbump + numVBump;
    if (numInvbump > 0)
        compensation = numBump*level1;
    else
        compensation = numBump*level2;
    end

    avgRevenue = (avgRevenue * (i - 1) + N * C) / i;
    avgShowup = (avgShowup*(i - 1) + N)/i;
    avgComp = (avgComp*(i - 1) + compensation)/i;
    avgBmp = (avgBmp*(i - 1) + numBump)/i;
    avgInvb = (avgInvb*(i - 1) + numInvbump)/i;

    if nargin == 5 && printon == 1
    fprintf('\n Current average compensation %.0f,', avgComp)
    fprintf('\n Current average number of passengers volunteered to be
        bumped %.0f,', avgBmp)
    fprintf('\n Current average number of passengers forcefully bumped
        %.0f,', avgInvb)
    end

```

```
end

% To return
R = avgRevenue - avgComp;
bRatio = avgBmp/avgShowup;
invbRatio = avgInvb/avgShowup;

end
```

3 Function: flightInfo.m

```
function [N, I, V] = flightInfo(T, S, R_S, R_I, print)
    % function [N, I, V] = flightInfo(T, S, R_S, R_I, print)
    % Simulates a single flight for the given parameter using
    % DeMoivre-Laplace Transform to approximate the binomial distribution
    % of the number of people who show up and the number of people who
    % don't volunteer to get bumped.
    % Inputs:
    %       T : Number of extra tickets sold
    %       S : Seating Capacity of the flight
    %       R_S : Average show up rate of the flight, defined as the ratio
    %             of customers who show up to the total number of tickets
    %             sold
    %       R_I : Average ratio of involuntarily bumped passengers, defined
    %             as the ratio of customers who are involuntarily bumped to
    %             the ratio of customers who are bumped
    %       print(optional) : prints the flight info if passed in as '1'
    % Outputs:
    %       N : Number of passengers who showed up for the flight
    %       I : Number of involuntarily bumped passengers
    %       V : Number of voluntarily bumped passengers

assert(nargin == 5 || nargin == 4, 'Invalid number of inputs');

% totalTickets: The total number of tickets sold
totalTickets = S + T;

% Mean and standard deviation of the number of passengers who show up
mu_s = totalTickets * R_S;
sigma_s = sqrt(totalTickets * R_S * (1 - R_S));

% Generate and random number from the normal distribution to simulate
```

```
    the
% number of show ups
N = round(sigma_s * randn(1) + mu_s);

% bumps : Number of passengers that need to be bumped
bumps = N - S;

% Initialize voluntary and involuntary bumps before simulation
I = 0;
V = 0;
if bumps > 0
    % Generate a random number between 0 and 1 that represents the cdf
    % of the binomial distribution and use that to simulate the number
        of
    % involuntary bumps
    I = binoinv(rand(1), bumps, R_I);
    I(I < 0) = 0;
    V = bumps - I;
end

if nargin == 5 && print == 1
    fprintf('\n %.0f passengers showed up,', N)
    fprintf('\n %.0f had to be forcefully bumped \n', I)
    fprintf('\n %.0f volunteered to be bumped,', V)
end

return
```

Outputs of One Flight Model

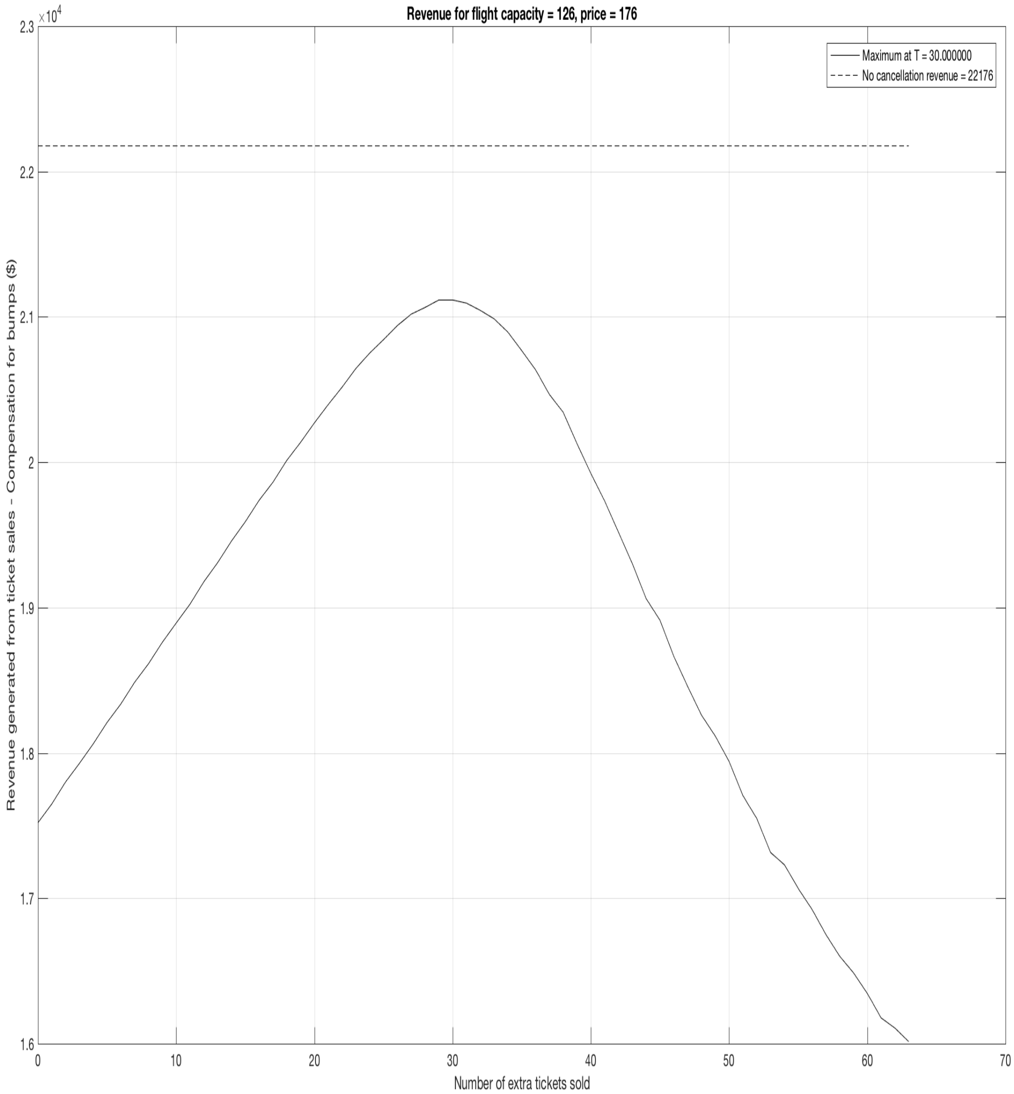

Figure 1. Revenue generated in the One Flight model

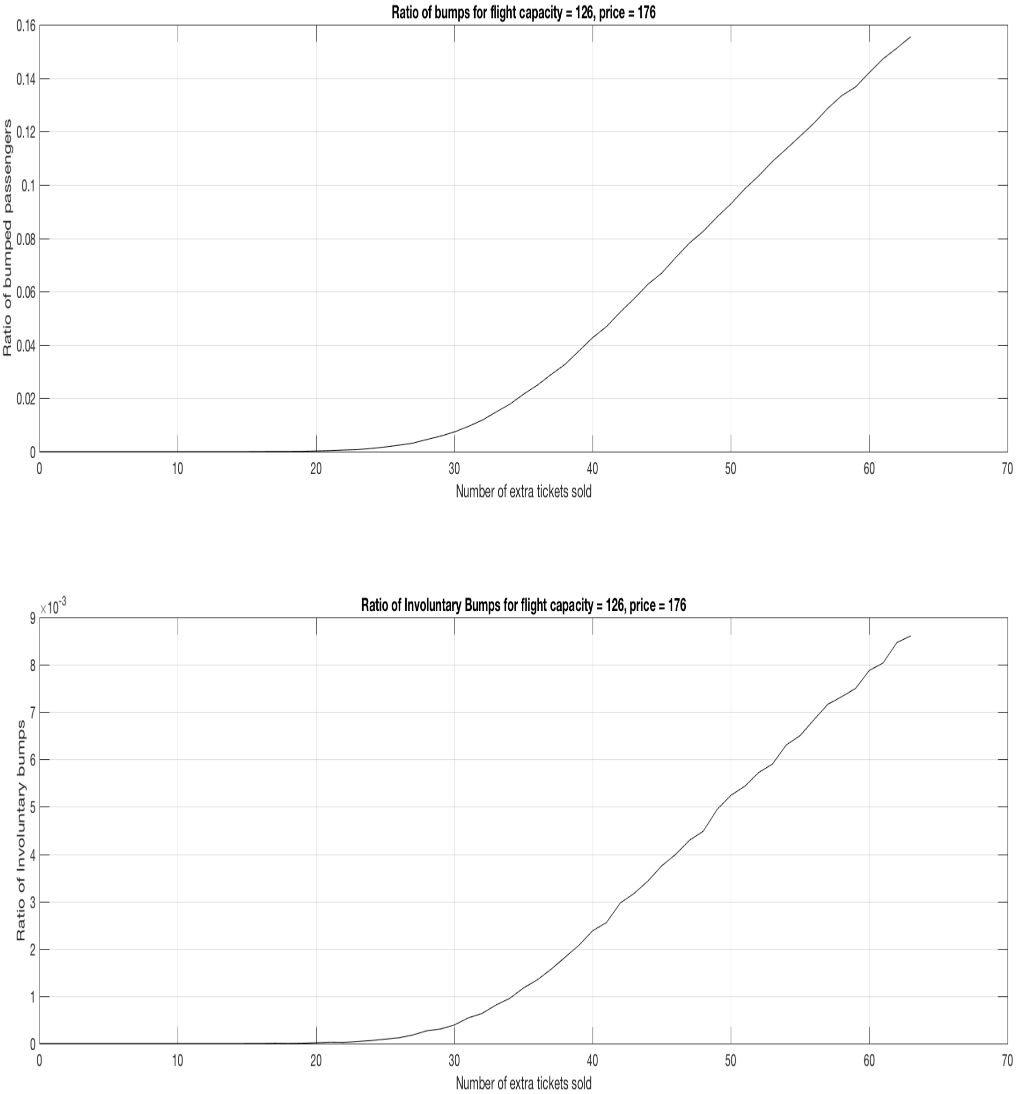

Figure 2. Passenger statistics in the One Flight model

4 Script: optimizeOverbooking.m

```
clear
h = waitbar(0, 'Preparing simulation ...');
%% Data
R_s = (83 + [0.2, 0.6, 0.5, 0.8])/100;
R_i = 0.0552;
C = [175, 178, 166, 183];
S = 126;
departures = 60 * (6:1:21);
arrivals  = 60 * (7:1:22) + [35, 49, 42, 46, 40, 41, 37, ...
                             34, 41, 38, 43, 48, 46, 49, 40, 46];
schedule = struct('departure', num2cell(departures), ...
                  'arrival', num2cell(arrivals));
% The ideal scenario where no ticket sold is refunded/cancelled
idealRev = sum(C * 91 * length(schedule) * S);

%% Simulation parameters
numSims = 5000;
maxT = S/4;
T = (0 : 1 : maxT)';

rev = zeros(length(T), 4);
N = zeros(length(T), 4);
I = zeros(length(T), 4);
V = zeros(length(T), 4);
queue = zeros(length(T), 364);

%% Monte-Carlo Simulation
for k = 1 : length(T)
    waitbar(k/length(T),h,sprintf('Running simulation for T = %.0f',T(k)));
    [rev(k,:), N(k,:), I(k,:), V(k,:), queue(k,:)] = ...
                        simulateYear(numSims, R_s, R_i, C, schedule, T(k), S);
end

waitbar(1, h, 'Generating plots ...')

```

```
totalRev = sum(rev,2);
totalN = sum(N, 2);
totalI = sum(I,2);
totalV = sum(V,2);

%% Critical points
tooManyTix = find(isnan(totalRev), 1) - 1;
revenueOriented = find(totalRev == max(totalRev), 1);
zeroBumps = find(totalI + totalV < 1, 1, 'last');
zeroInvBumps = find(totalI < 1, 1, 'last');
midPoint = round((zeroInvBumps + revenueOriented)/2);
firstQuartile = round((zeroInvBumps + T(midPoint))/2);
thirdQuartile = round((T(midPoint) + revenueOriented)/2);
% vector of all critical points
critical = [1, zeroBumps, zeroInvBumps, firstQuartile, ...
            midPoint, thirdQuartile, revenueOriented, tooManyTix];


%% Print out data for the critical points
disp([{'No.', '# Extra', 'Revenue', 'Show-ups', 'Bumps', ...
        'Involuntary', 'QueueSize/Night'}; ...
      num2cell((1:8)'), ...
      num2cell(T(critical)), ...
      num2cell(totalRev(critical)), ...
      num2cell(totalN(critical)), ...
      num2cell(totalI(critical) + totalV(critical)), ...
      num2cell(totalI(critical)), ...
      num2cell(mean(queue(critical, :), 2))])

%% Tickets sold vs. Revenue
figure(1)
plot(T, totalRev, '-k', T, idealRev * ones(size(T)), '--k', ...
     T(critical), totalRev(critical), '*b')
grid on
xlim([0,maxT])
xlabel('Number of extra tickets sold')
ylabel('Revenue (in $)')
legend(sprintf('Maximum revenue (at T = %.0f) = %.0f', ...
               T(revenueOriented), totalRev(revenueOriented)),...
```

```
        sprintf(['No cancellation \n Ideal Revenue = $%1.2e', ...
                 '\n Revenue without overbooking = $%1.2e'], ...
                 idealRev, totalRev(1)), 'Critical Points')
title('Average yearly revenue after paying out compensation')
saveas(gcf, 'revenue.png')

%% Passenger stats vs. extra tickets sold
figure(2)
subplot 311
plot(T, totalN, '-k', T(critical), totalN(critical), '*b')
grid on
xlim([0,maxT])
xlabel('Number of extra tickets sold')
ylabel('Number of passengers')
legend('', 'Critical Points')

subplot 312
plot(T, totalI+totalV, '-k', T(critical), ...
        totalI(critical) + totalV(critical), '*b')
grid on
xlim([0,maxT])
xlabel('Number of extra tickets sold')
ylabel('Number of bumped passengers')
legend(sprintf('No bumps until T = %.0f', T(zeroBumps)), 'Critical Points')


subplot 313
plot(T, totalI, '-k', T(critical), totalI(critical), '*b')
grid on
xlim([0,maxT])
xlabel('Number of extra tickets sold')
ylabel('Number of involuntarily bumped passengers')
legend(sprintf('No involuntary bumps until T = %.0f', T(zeroInvBumps)), ...
        'Critical Points')
saveas(gcf, 'passengers.png')

%% Waiting line for some critical points
```

```
figure(3)
subplot 311
plot(queue(tooManyTix, :), '-k')
xlim([0 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(tooManyTix)))
title('Queue size for maximum ticket sales with delay <= 3 days')

subplot 312
plot(queue(revenueOriented, :), '-k')
xlim([0 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(revenueOriented)))
title('Queue size for revenue oriented ticket sales')

subplot 313
plot(queue(zeroInvBumps, :), '-k')
xlim([0 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(zeroInvBumps)))
title('Queue size for zero involuntary bumps')

saveas(gcf, 'queueLessImp.png')

figure(4)
subplot 311
plot(queue(firstQuartile, :), '-k')
xlim([0, 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(firstQuartile)))
title('Queue size at one quartile away from zero involuntary bumps')
```

```
subplot 312
plot(queue(midPoint, :), '-k')
xlim([0, 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(midPoint)))
title('Queue size midway between zero involuntary bumps and max revenue
    ')

subplot 313
plot(queue(thirdQuartile, :), '-k')
xlim([0, 365])
grid on
xlabel('Day of the year')
ylabel('Number of people waiting overnight')
legend(sprintf('Num. of extra tickets = %.0f', T(thirdQuartile)))
title('Queue size three quartiles away from zero involuntary bumps')

saveas(gcf, 'queueMoreImp.png')

delete(h);
```

5 Function: simulateYear.m

```
function [revenue, N, I, V, waits] = ...
            simulateYear(numSims, R_s, R_i, C, sched, T, S)
    % function [revenue, N, I, V, queue] = simulateYear(R_s, R_i, C, Sched, T, S)
    % Runs the overbooking simulation for a year
    % Inputs:
    %       R_s: Vector of average show up rates every quarter
    %       R_i: average ratio of involuntary to voluntary bumps
    %       C: Vector of average ticket prices every quarter
    %       sched: Schedule of departure and arrival times of a day
    %               Is an array of structs with fields arrivalTime and
    %               departureTime in minutes of the day
    %       T: Number of extra tickets to sell
    %       S: Seating capacity
    % Outputs:
    %       revenue: total revenue generated from ticket sales after giving
    %                out compensations
    %       N: Total number of passenger show ups in a year
    %       I: number of involuntary bumps
    %       V: number of voluntary bumps
    %       waits: Vector of number of people in queue at the end of every
    %               day

% numQuarters: number of quarters to run the simulation for
numQuarters = 4;
% numDays: number of days in every quarter
numDays = 91;
% minsInDay: number of minutes in a day
minsInDay = 24 * 60;

% time : time right now in the simulation
% Itoday : Involuntary bumps today
% Vtoday : voluntary bumps today
% Ntoday : Number of show ups today
% CompToday : total compensation paid today
```

```
% Initialize outputs
N = zeros(numSims, 4);
I = zeros(numSims, 4);
V = zeros(numSims, 4);
compensation = zeros(numSims,4);
waits = zeros(numSims, 364);
revenue = zeros(numSims, 4);

for n = 1 : numSims
% overNightWaits: array of number of people waiting at the end of each
    day

queue = struct('numPassengers', {}, 'arrivalTime', {});

 for quarter = 1 : numQuarters
     for day = 1 : numDays
         time = struct('quarter', quarter, 'day', day, 'minutes', 0);
         [queue, Itoday, Vtoday, Ntoday, compToday] = simulateDay( ...
                  R_s(quarter), R_i, C(quarter), T, S, sched, time,
                     queue);

          N(n, quarter) = N(n, quarter) + Ntoday;
          I(n, quarter) = I(n, quarter) + Itoday;
          V(n, quarter) = V(n, quarter) + Vtoday;
          compensation(n, quarter) = compensation(n, quarter) +
             compToday;
          waitingToday = 0;
          for i = 1 : length(queue)
            waitingToday = waitingToday + queue(i).numPassengers;
          end
          waits(n, (quarter - 1) * numDays + day) = waitingToday;
          if isnan(Ntoday)
              N = NaN * ones(1,4);
              I = NaN * ones(1,4);
              V = NaN * ones(1, 4);
              waits = NaN * ones(1,364);
              revenue = NaN * ones(1, 4);
              return;
```

```
                end
            end
    end
revenue(n, :) = C .* N(n, :) - compensation(n, :);
end

N = mean(N);
I = mean(I);
V = mean(V);
waits = mean(waits);
revenue = mean(revenue);
return
```

6 Function: simulateDay.m

```
function [q, invBump, vBump, showUp, compensation] = ...
                        simulateDay(R_s, R_i, C, T, S, schedule, today, q)
% simulateDay: To perform a Monte Carlo simulation and calculate the
% parameters listed below for a given flight using a constant average show
% up rate and average rate of involuntarily bumped passengers.
% Variables:
% R_s : scalar - show up rate
% R_i : ratio between involuntary bump/voluntary bump
% C: scalar - avg. price that day
% T, S : extra tickets, seating capacity
% schedule: struct with two fields arrival and departure each of which is
%           a time struct with fields quarter, day, minutes
% today: Time struct with today?s quarter and day
% q: Current waiting queue
%
% Output:
% q: Updated waiting queue
% invBump: Total number of involuntary bumps in the day
% vBump: Total number of involuntary bumps in the day
% showUp: Total number of passengers who show up during the day
% compensation: Total amount of compensation paid that day
%

maxDay = 91;


showUp = NaN;
invBump = NaN;
vBump = NaN;
compensation = NaN;
% If a passenger has been waiting for too long then the queue likely will
% not decrease in size
if length(q) > 0
```

```
    if (calculateTimeDiff(q(1).arrivalTime,today)) > 24 * 3
        return;
    end
end



% Re-initialization of variables
showUp = 0;
invBump = 0;
vBump = 0;
compensation = 0;
numFlights = length(schedule);

% Simulate each flight in the schedule
for f = 1:numFlights
    today.minutes = schedule(f).departure;
    [N, I, V] = flightInfo(T, S, R_s, R_i);
    showUp = showUp + N;

     % set the arrival time of this lot of passengers
     arrivalTime = struct('quarter', today.quarter, 'day', today.day
        ,...
                         'minutes', schedule(f).arrival);

     % Overnight flight
     if schedule(f).arrival <= schedule(f).departure
         arrivalTime.quarter = arrivalTime.quarter + ...
                                            floor(arrivalTime.day /  maxDay);
        arrivalTime.day = mod(arrivalTime.day, maxDay) + 1;
     end


    % Flight is overbooked, put all bumped passengers in a queue
    if N > S
        invBump = invBump + I;
        vBump = vBump + V;
        q = [q, struct('numPassengers', I+V, 'arrivalTime', arrivalTime
           )];
```

```
        else
            if N < S
                capacity = S - N;
                while capacity > 0 && length(q) > 0
                    timeWait = calculateTimeDiff(q(1).arrivalTime,...
                                                                arrivalTime);
                    % We have a lot of avalable seats
                    if capacity >= q(1).numPassengers
                        capacity = capacity - q(1).numPassengers;

                        % Calculate the amount of compensation based on the
                        % amount of waiting time
                        if (timeWait > 1 && timeWait <= 2)
                            compensation = compensation + 2*C*q(1).numPassengers;
                         else
                            if timeWait > 2
                                compensation = compensation + 4*C*q(1).numPassengers;
                            end
                        end
                        q(1) = [];
                    % The capacity only allows a fraction of passengers
                    else
                         if (timeWait > 1 && timeWait <= 2)
                            compensation = compensation + 2 * C * capacity;
                         else
                            if timeWait > 2
                                compensation = compensation + 4 * C * capacity;
                            end
                         end
                         q(1).numPassengers = q(1).numPassengers - capacity;
                    end
                end
            end
        end
end
return
```

7 Function: calculateTimeDiff.m

```
function diff = calculateTimeDiff(start, endt)
% calculateTimeDiff: Given 2 time objects, which contain the quarter,
    day,
% and hour, calculate the difference in hours of the 2 time point
% Input:
% - start: The starting point, a time object
% - endt: The ending point, a time object
% Output:
% - diff: The difference between the 2 points in hours

startQ = start.quarter;
startD = start.day;
startH = start.minutes / 60;

endQ = endt.quarter;
endD = endt.day;
endH = endt.minutes / 60;

diffQ = endQ - startQ;
diffD = endD - startD;
diffH = endH - startH;

diff = (diffQ*91 + diffD)*24 + diffH;
end
```

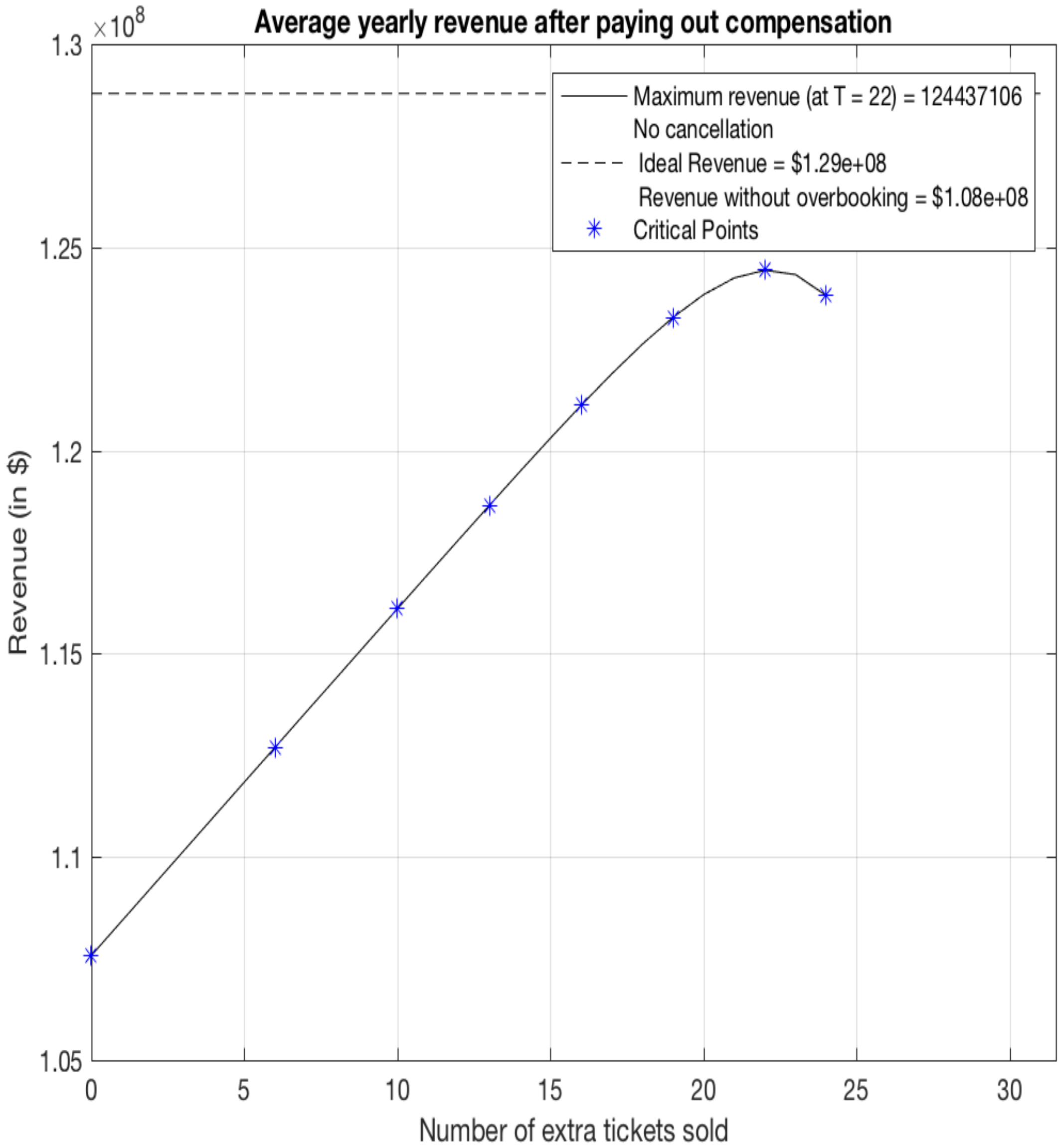

Figure 3. Revenue Generated for the Full Year Model

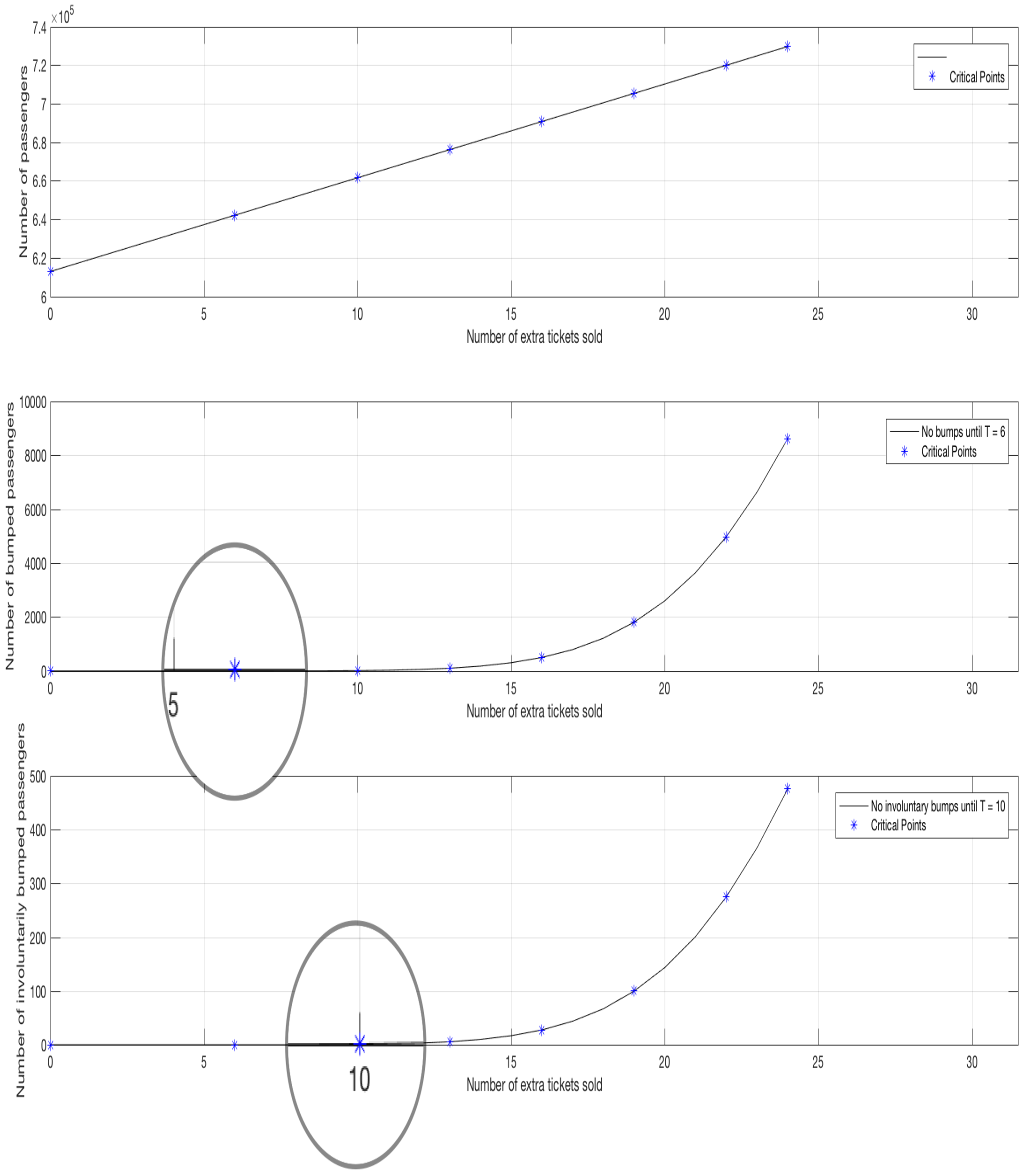

Figure 4. Passenger statistics for the Full Year Model

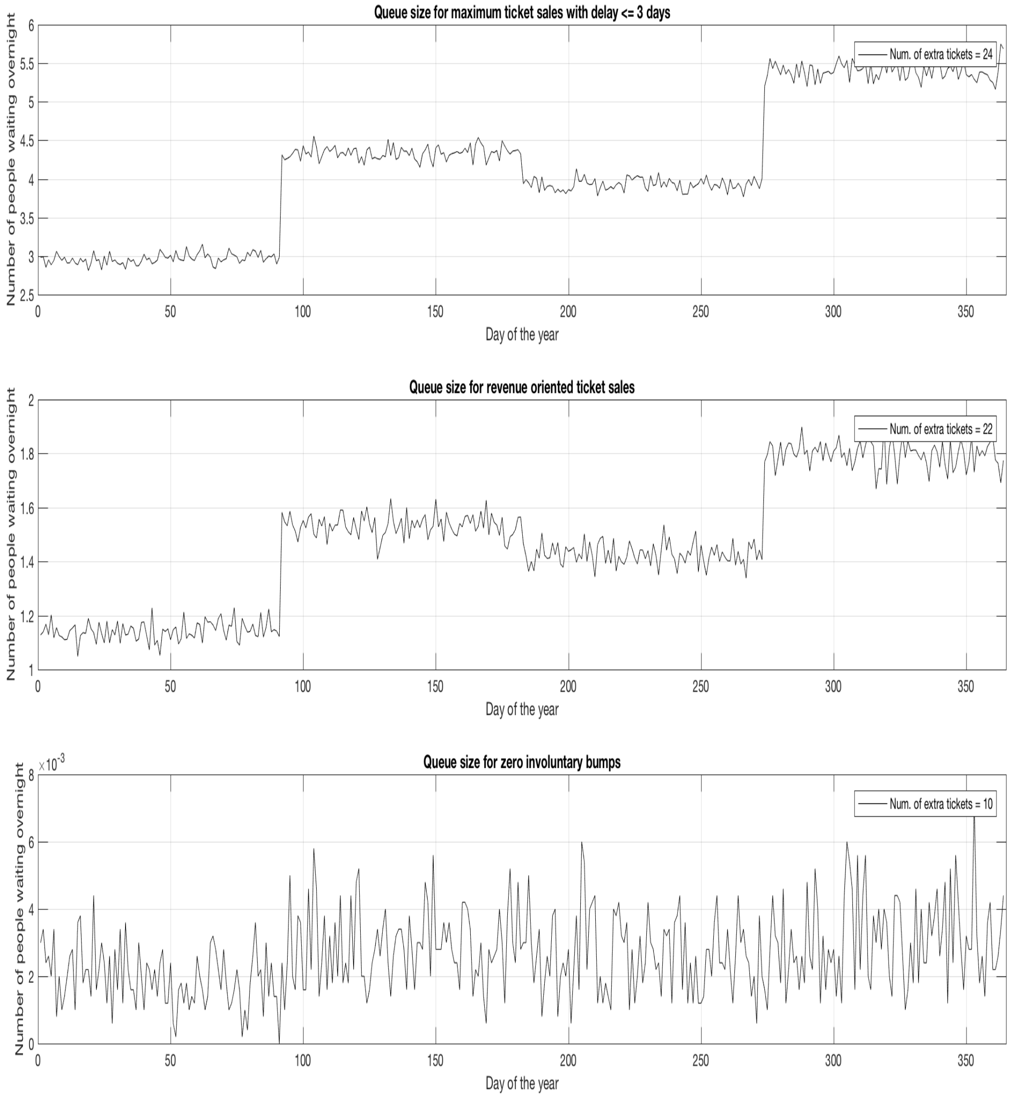

Figure 5. Queue size at the boundaries of the optimal region

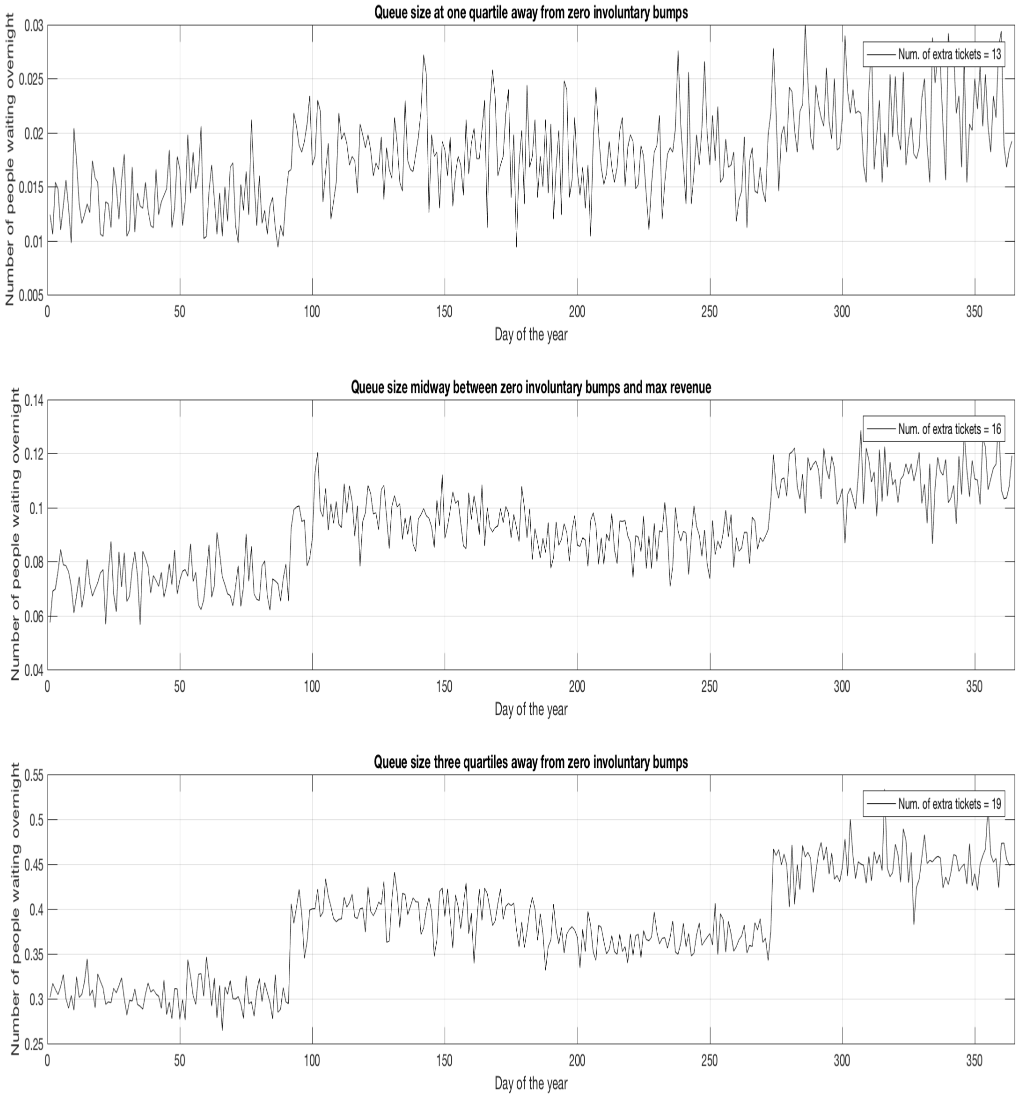

Figure 6. Queue size at quartiles inside the optimal region