Improving Parking Garage Efficiency using Reservation Optimization Techniques

By

ARJUN RAO

A thesis submitted to the

Graduate School-New Brunswick

Rutgers, The State University of New Jersey

in partial fulfillment of the requirements

for the degree of

Master of Science

Graduate Program in Electrical and Computer Engineering

written under the direction of

Professor Ivan Marsic

and approved by

____

____

____

____

New Brunswick, New Jersey

October 2011

ABSTRACT OF THE THESIS

Improving Parking Garage Efficiency using Reservation Optimization Techniques

By Arjun Rao

Thesis Director:

Professor Ivan Marsic

This thesis describes and evaluates techniques that can be implemented by parking garages to augment parking garage efficiency. The issues studied in this thesis were i) Real-time tracking of car position ii) maximizing the number of reservations made for the parking garage by re-arrangement of existing reservations (Reservation Defragmentation) and iii) maximizing revenue for the parking garage through increased occupancy (Revenue Management).

For the tracking problem, in order to be able to track the real-time position of the vehicle inside the parking garage, we have proposed two techniques. We simulated various conditions of sensor failure rate and determined our metric to be number of tracked points as a percentage of the path to the destination. For the reservation defragmentation problem, we looked at increasing occupancy efficiency for i) Next day reservations and ii) Current day reservations. For increased revenue management, we suggested the application of two techniques: Booking limits and Overbooking.

We obtained the following results for the algorithms implemented. In case of the tracking algorithm, as the sensor failure rate increased, the inaccuracy of the two proposed algorithms also increased. For 2% failure rate, we track 0.4% of the incoming cars inaccurately (given that a tracking is marked as correct if 75% or less of all sensors along the path of the car fail). In case of reservation defragmentation, we obtained best results for Recursive First-Fit algorithm. For next day reservation defragmentation, using a mean of 15% cancellation of reservations resulted in 14.6% decrease in occupied parking spots. For current day reservations, we were able to increase maximum occupancy of the parking garage by 5.5% using Recursive First Fit algorithm.

For Booking Limits, we evaluated Poisson arrival distribution and Binomial distribution. We evaluated overbooking for several combinations of No-show rates, mean and standard deviation values and the highest amount of overbooking we obtained was 1.93 times maximum garage capacity and this implies that permitting this number of reservations for the parking garage would minimize the number of parking spots being under-utilized and increase the revenue of the parking garage operator due to effective use of parking spots.

Acknowledgements

First and foremost, I would like to thank Dr. Ivan Marsic for providing me the opportunity to work on an extremely interesting and novel problem. His guidance and his ability to think of tangential solutions to a problem helped me tackle several issues in unique ways. I would also like to thank my Dad for his advice on maintaining discipline during the course of the thesis. My Mom gave me all her positive vibes which lent me the conviction to solve my problems with ease and without whose help I would not be where I am. I acknowledge the words of encouragement offered by my Sisters, Priyanka and Nandini, who helped me proceed through the thesis with the vigor required. I would like to thank my friends at Rutgers University who were always there for me in times of need.

Contents

List of Illustrations

List of Tables

Chapter 1: Introduction

1.1 About Parking

Parking facilities are a major expense to society and parking conflicts are among the most common problems facing infrastructure planners. These problems can be most often described either in terms of supply or in terms of management. Parking management describes the process of optimizing the use of parking policies while making use of policies and programs that are applicable to parking. A well-thought out parking strategy often helps reduce the number of parking spots required in a particular situation and provides a variety of socio-economical and environmental benefits. When all factors are taken into consideration, improved management is often the best solution to parking problems. Management solutions tend to be significantly more optimum than increasing supply as they tend to support more strategic objectives. Some of these objectives are listed below.

• Improved user options and quality of service

• Facility cost savings.

• If the strategies are decided properly, there can be significant revenue generation that could help finance other facilities and improve transportation infrastructure.

1.2 Problems with the Parking Industry

1.2.1 Lack of Use of Technology for Parking Guidance

Parking guidance is an optimization control problem which provides driving route suggestion and slot status by using computer technology, mechanics of communication, and control technique for the purpose of guiding drivers to the expected parking place. The result of such methods is to guide the customer to the expected parking place by driving on planned route (Jun Y., 2010) [25]. The disadvantage of current smart parking or parking guidance systems is that they only obtain the availability information of parking spots from deployed sensor networks and that they just broadcast the parking information directly to drivers. Since these systems do not actually direct a driver to the designated parking spot they sometimes make the situation worse and are hence deemed not smart enough. It is, therefore, strongly desired to provide an effective strategy to address these concerns (Section 3.2.2)

1.2.2 Environmental Concerns

Hunting for a vacant parking spot in a metropolitan/suburban area is a daily source of anxiety for most drivers and it is time-consuming. It generally results more traffic congestion and air pollution by constantly cruising in certain area only for an available parking space. For instance, a recent survey (White, 2007) [51], shows that during rush hours in most big cities, the traffic generated by cars searching for parking spots takes up to 40% of the total traffic and a correspondingly high proportion of CO_2 emissions. Motor vehicle accidents and other situations cause high number of fatalities, injuries, and economic distress resulting from emergency and health care services as well as property damage. Parking is a major part of overall mobility as every vehicle trip finally concludes in parking the car somewhere at the destination. Over the course of a year, vehicles looking for parking in one small business district of Los Angeles burned 47,000 gallons of gasoline that totals 945,000 extra miles traveled or two round trips to the moon and produced 730 tons of carbon dioxide (Shoup, 2007) [42]. To deal with aimless wandering caused by the search for parking, we have incorporated a reservation system for parking. In this, the user needs to make a reservation and a spot is allocated to him along with directions to that spot. We have provided algorithms that increase the efficiency of this reservation system (Section 3.2.3)

1.2.3 Parking Space Inefficiency

Often, people complain of lack of parking spots when actual counts show that only 60 to 75 percent of spots are occupied (Tumlin et al., 2004) [48]. It is very important to deal with perceptions of parking shortages. Shoup points out in his study that the most appropriate way for cities to address parking shortages is to price the spots. According to Shoup, that would result in 14 percent of spots being made available (Shoup, 2007) [42]. We have provided a 2-class parking strategy involving booking limits where we have a differential pricing of parking spots in order to increase revenue (Section 3.2.2 Part A).

1.2.4 Lack of Revenue Management

A typical automobile is parked 23 hours each day, and uses several parking spots each week (Local Motion, 2006) [34]. The US Parking Lots and Garages industry provides parking and valet service for nearly 250 million motor vehicles on an hourly, daily or monthly basis (Andrews, 2011)[1].

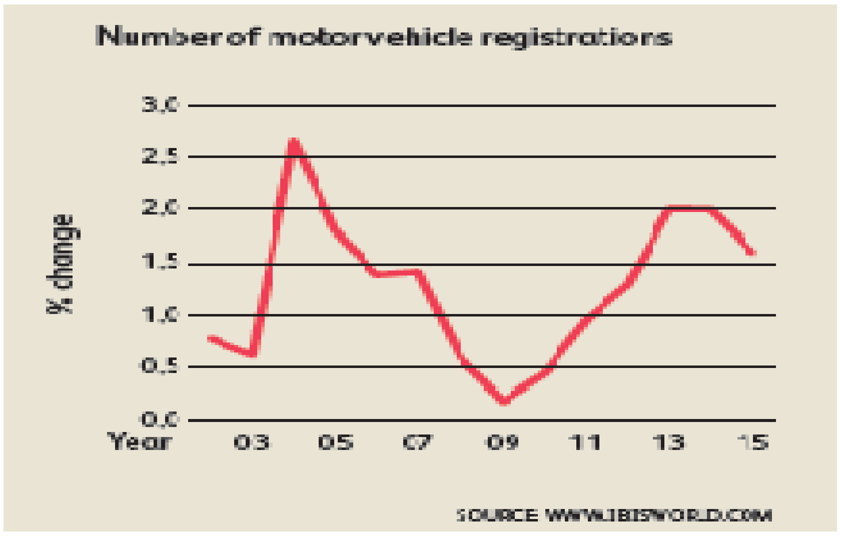


Figure 1: Number of motor vehicle registrations (Data recorded as of February 2011)

Source: www.ibisworld.com

As can be seen from Figure 1, the percentage change in number of motor vehicle registrations is always positive and seems to be growing in the coming few years. This is indicative of the fact that there will be more necessity for parking spots for all these new cars and facilities will have to be upgraded / added in order to deal with this increase in cars. The cost of developing parking is about $50,000 per parking space, depending on the location and can decide the project's financial viability (Macht, 2007) [35]. In order to obtain a profit after such expenses, it is imperative that the parking garage has high occupancy most of the time. We present methodologies to enable this (Section 3.2.3).

1.3 Research Questions

Studies on the parking situation today are divided into two broad categories

a. Technology to facilitate search for free parking spots

b. Optimizing parking for increased revenue generation and efficiency of operations

While there have been many studies based on (a) and which are reviewed in Section 2.2, the research in (b) is still in its nascent stages. This is the category of research that we seek to undertake in this thesis. When it comes to optimization techniques to deal with increasing efficiency of parking garages, there are very few techniques that have actually been studied (Read Section 2.4.3). Most of the strategies are those that are applied to the airline and hotel industry and there has been no significant porting over of these techniques to the parking industry.

The research conducted for the thesis is broadly divided into three segments:

a. Tracking car position for real-time monitoring
b. Reservation optimization techniques
c. Revenue/Yield Management for parking garages

In (a), we try to answer the following questions:

i. What hardware devices and software algorithms can be used to track the position of a car in the parking lot?
ii. Why do we need to track car position in the parking garage?
iii. What is the cost and accuracy of maintaining and developing such a system?
iv. Which method is the optimum one in terms of trade-off between monetary cost and accuracy?
v. What is the scalability of this algorithm?
vi. What metrics were developed to discuss the efficiency of the system?

In (b), we try to answer the following questions:

i. How can we pack more number of reservations for a given parking spot?

ii. Determining optimum solution based on monetary cost and algorithmic complexity (performance).

iii. What is the scalability of these reservation optimization techniques?

iv. What is the efficiency of algorithms for reservation optimization?

v. Which algorithm provided the best results?

In (c), we try to answer the following questions:

i. What techniques can be used to increase the yield of parking garages?

ii. How are these techniques affected by the rate of users entering the parking garage?

iii. What metrics were developed for revenue management and were they directly ported from other industries?

In this thesis, we present algorithms that answer these questions. These algorithms are improvements over existing methodologies or in some cases unique in that such algorithms have not been put forward in the past. We also present the analysis of the results of these algorithms that in some cases are the reflective of their utility and showcase the benefits of using these algorithms for implementation in parking garages.

1.4 Thesis Organization

This thesis is organized as follows: Chapter 2 identifies the most important studies related to the problems that we are attempting to solve. Chapter 3 describes the motivation for this research as well as gives an overview of the methods used to tackle the intended problems. Chapter 4 describes in detail the algorithms used as part of the thesis. Chapter 5 presents the results obtained from running various simulations under a variety of conditions to increase parking garage efficiency in the three categories mentioned above. Chapter 6 presents an analysis of the results obtained with an aim to understand the impact of the implemented algorithms. Chapter 7 puts forward the work that can be done in the future with regard to the implemented algorithms. Chapter 8 consists of a list of the literature that was used in order to better understand the problem we are dealing with.

Chapter 2: Literature Review

2.1 Introduction

The Parking Lots and Garages industry is an $8.2bn industry in the United States, with a predicted annual growth of 4.2% over the next 5 years (Andrews, 2011). Techniques to improve parking garage efficiency primarily focus on very simplistic methods dealing with parking lot construction and other physical attributes heuristics. The usage of software to simulate various parking garages conditions and study their behavior is not yet being used widely. While improving reservation efficiency is a well-documented topic for the airline industry and hotel industry, despite the similarities the parking industry shares with the aforementioned market sectors, not much research has been conducted for the latter. This chapter seeks to compare state-of- the art technologies in three of the key areas where technology can be applied in order to improve effectiveness of a parking garage.

2.2 Tracking

2.2.1 Tracking Free Spots

In order to provide quick parking solutions to the patrons, having the ability to determine which parking spots are empty and can be used by the incoming patron can be immensely helpful. This problem has been studied in a number of ways by different researchers. One approach uses prediction of number of free parking spots in the parking lot modeled by a continuous time Markov chain (Klappenecker et. al, 2010) [27]. In this method the parking lot regularly communicates the number of occupied spots, capacity, arrival and parking rate through a vehicular ad-hoc network. The navigation system in the car will compute the probability of getting a free space using all this data. There is one method in which a user is apprised of the existence of parking spots while on the move (Delot et al., 2009) [13]. In this a vehicular ad-hoc network is used, in which drivers can receive information from a central server about the empty spots while driving. There have also been systems where drivers (not individually) are navigated through the parking lot with lit up arrows, indicating the presence of vacant spots in that direction as implemented in the Baltimore International airport (Charette, 2007) [11]. When the spots get filled up, the number of vacant spots is updated, and if a particular section gets filled up completely, there will be no more arrows directing drivers to that section. A parking lot system based on wireless sensor networks has been studied (Tang et al, 2006) [46]. It finds a closest spot and guides a car to that spot. The sensors employed by the system are expensive thereby increasing costs.

2.2.2 Tracking Car Position

Tracking of car positions within the parking lot is a relatively new field and still is in the nascent form of research. Some of the research that has been carried out in this field has been listed. A method of networked parking spots with architecture and applications is studied (Basu et.al, 2004) [4], in which a multi-hop wireless parking meter network is coupled with a GPS receiver to allow a user to locate and navigate to an empty parking space. The method uses wireless radio frequency transceivers and auxiliary hardware and software. Another scheme employs parking lot Road Side Units [RSU's] to survey and manage the whole parking lot (Lu, 2010). It is enabled by communication between vehicles and RSU's. After the vehicles are equipped with the wireless units, the RSU's communicate with them and provide the drivers with real time parking navigation service. Moreover the system is also used to provide antitheft protection and parking information dissemination using the concept of VANET's. Another navigational method has several info-stations are set up across the parking lot and whenever users with mobile devices/PDA's come in the vicinity of the Info-Station, it will receive information from that Info-station about the availability of parking spots there (Ganchev et.al. , 2008) [17]. If the mobile device is a smart-phone, then the user will also receive a graphical representation of the layout of the parking lot that will navigate the user to the parking space. A parking guidance system based on Wireless Sensor networks was suggested in which a driver is guided to an available parking lot (Yoo et.al. 2008) [52]. The system consists of a WSN based vehicle detection sub-system [VDS] and a management sub-system. The VDS collects information of how many free spots are available in the parking lot and the management sub-system processes this information and then uses it to guide the driver to the parking lot using a Variable Messaging System [VMS].

2.3 Reservation Optimization

2.3.1 Parking Reservation Systems

In order to study the process of optimizing reservations to obtain better 'packing', it is prudent to study the different methods of reservation systems that exist today. An intelligent parking system was proposed in which the system is designed to be compatible with aspects that allow drivers to reserve a parking spot through the internet when the space is available (Inaba et.al, 2001) [22]. The system also uses a smart card for payment which provides recognition and payment services. It uses time share and real time reservation services for allocating reservations.

2.3.2 Memory/Process Optimization Techniques

One method for defragmenting memory occurs after the usage of the buddy algorithm (Defoe et al, 2005) [12]. Knuth's buddy system is an attractive algorithm for managing storage allocation, and it can be made to operate in real-time. The paper investigates the issue of defragmentation for heaps that are managed by the buddy system and presents tight bounds for the amount of storage necessary to avoid defragmentation. It also presents an algorithm for defragmenting buddy heaps. The problem of adjacent resource scheduling has also been looked into (Duin et. al., 2006) [14]. The problem of airport check-ins is treated taking into account lines of passengers and service at multiple desks with the intention of reducing the number of desks i.e. reducing the resources available to get maximum output. Genetic algorithms are another method of dealing with Job Shop scheduling problems (Goncalves et al, 2002) [20]. The schedules are constructed using a priority rule in which the priority is defined by the genetic algorithm. In order to obtain the schedule for the operations on the machines the precedence constraints minimizing the makespan (finish time of the last operation) are taken into account. A method to minimize number of machines for scheduling jobs with equal processing times has also been studied (Brucker et al, 2009) [9]. A polynomial time algorithm is suggested which aims to determine the minimum time required for a job to be processed and then suggest a feasible schedule for the jobs to be executed.

2.3.3 Parking Space Optimization Techniques

Methods to improve parking space optimization techniques are similar to the ones described in the section memory/process optimizations. However, to my knowledge, there are no direct instances of research into the field of reservation optimization for parking reservation systems.

2.4 Revenue Management

2.4.1 Introduction

Revenue Management is an economic discipline approach applicable to many industries in which “market segment pricing” is combined with statistical analysis to expand the market for the service and increase the revenue “revenue” per unit of available capacity (Anon., 2005)[2]. Revenue Management (RM) techniques provide an effective way of increasing revenue, creating new business opportunities for companies in a wide variety of business areas. RM emerged in the 1980s in the airline industry. Today, for many airlines RM is the difference between profit and loss. For each flight, airlines determine in detail how many seats will be offered to the customers at a given moment and price (Belobaba, 1987) [5]. The stupendous success of RM has generated interest in other business areas, such as hotels and car rental companies. In order to increase revenue in a parking garage, it is essential to know what resources are available at the time the parking spot has been requested. Given below are the criteria which show the similarity between the parking industry and the other industries in which RM techniques have already been applied.

R.M. Criterion	**Airline**	**Hotel**	**Car Rental**	***Car Parking***
Unit	Flight seat	Hotel room	Car	*Parking spot*
Pre- Booking	Yes	Yes	Yes	*Yes*
No. of possible prices/unit	Many	Few	Many	*Few*
Duration of use	Fixed	Variable	Variable	*Variable*
Management	Central	Central/local	Central/regional/ local	*Central/region/ local*

Table 1: Revenue management criteria for various industries

[last column has been added by us to provide meaningful comparison between the different industries and the parking industry]

Source: The Basics of Revenue Management, Ideas, 2005.

2.4.2 Booking Limits

A. Description

In a system where parking spots are allotted based on fare charged to the customer, ***booking limits*** are defined as the number of parking spots that are allotted to customers who are charged the lower fare. Airline seat inventory control involves selling the right seats to the right people at the right time (Belobaba, 1987) [6]. If an airline would sell tickets on a first-come first-serve basis, its capacity would get filled up early with leisure travelers, who would be eager to reserve a seat the moment they see it. In this scenario, if we have late bookers, generally business travelers who are willing to pay a higher fare then they will find that all seats have been reserved and these important sales will be lost as selling seats to them will result in higher revenue. By imposing booking limits on the lower fare classes this can be avoided.

Booking limits are determined at the start of the booking process based on forecasted demand and are then dynamically updated multiple times during the booking period, although it is practically inefficient to recalculate them after every booking request. For profitability, the available capacity should be offered at different prices (Van der Mei et.al, 2009) [50]. In the airline industry, the airline determines what fraction of the (remaining) seats is to be offered at what price. The fundamental requirement is different groups of users who are willing to pay different prices. These customer groups must also be distinguishable by the time of reservation and their choice of additional features (such as providing options for cancellation) (Van der Mei et. al., 2009) [50].

B. Booking Limits in Other Industries

One of the known examples that categorize its users into such segments is the airlines industry in which exist different classes, such as first, business, and economy. Using this approach, the airline can offer users from different classes with various fares and restrictions, based on their flexibility, price sensitivity, and time of bookings prior to departure times. Delta Airlines has estimated that selling only one seat per flight at full rather than at discount rate can add over $50 million to its annual revenue (Beckman., et al., 1958) [7]. The idea of marginal revenue using Booking Limits was revolutionary in Revenue Management (Littlewood, 1972) [33]. In 1989, Belobaba extended Littlewoods rule to multiple nested fare classes and introduced the term expected marginal seat revenue (EMSR) (Belobaba, 1989).

C. Booking Limits in Parking Industry

To my knowledge, booking limits have not been studied in that much detail for the parking industry. One of the most significant adaptations of the booking limits usage is the one used by Park n Fly Parking Management Company (Eijnden F., 2009) [15]. In this they use Littlewoods two class model with Belobaba's nested class modification (EMSR Model) for 'n' classes.

2.4.3 Overbooking

A. Description

Overbooking has its inception in the airline industry. In this the airline books a plane beyond its actual capacity by a certain extent. As a result, even if cancellations/no-shows occurs, there are sufficient people who will board the plane so as to avoid any spoilage costs. As long as the spoilage costs and denied boarding costs are balanced in such a way that there is no revenue loss for the airline, overbooking is a good revenue management solution.

B. Overbooking in Other Industries

In a study done on American Airlines (Smith et al., 1992) [45], 50% of the bookings were resulted in cancellations or no-shows. Moreover, the report found that 15% of the flight seats would be unused, if bookings were only limited to the capacity of a plane. Overbooking analysis is performed to determine the extent to which a future flight should be overbooked so as to minimize the sum of the lost revenues associated with empty seats and the costs of denying boarding to passengers with confirmed reservations (Rothstein 1985) [41]. Overbooking models were introduced to address the problem in unanticipated cancellations and no-shows, by several researchers in the airlines industry. The overbooking policies were also studied and applied to several industries, such as hotel (Hadjinicola et.al., 1997) [21], and car rentals (Geraghty et. al., 1997) [18]. One model uses a simple overbooking model in which the surviving bookings are modeled as a binomial process. This allows setting a booking limit on the number of reservations based on empirical data on yields, variable costs, costs per denied boarding and no-show probabilities (Klophaus R., 2007) [29]. There is some literature available on techniques of overbooking for clinics. A method suggested for clinics provides insights into rules that perform well to increase provider productivity while balancing the increased waiting time and overtime costs of overbooked schedules. (Laganga L, Lawrence S., 2009) [30]. This method models overbooked appointment schedules with deterministic service times and clinic capacity fixed as the total number of patients that can be served within the normal operating time of a clinic session. For hotels, it has been demonstrated that even if the hotel is assured of payment and there are penalties for over sales, the property has an incentive to overbook (Arenberg Y., 1991) [3].

C. Overbooking in Parking Industry

To my knowledge, there are no direct studies of related to overbooking for the parking industry. Overbooking is a practice implemented for the airline, hotel and healthcare industries. Application of overbooking to the parking industry has been discussed in the future chapters.

Chapter 3: Thesis Overview

3.1 Motivation

Research in the parking industry is focused primarily on increasing the ability to search for a parking spot in as efficient a manner as possible. While multiple methods are suggested in order to make this happen, it is also equally important to make sure that the efficiency of overall parking is increased.

This thesis was started with the aim of improving the efficiency of parking garages and coming up with ideas that would benefit the management of the parking garage. As a result, the research began in a direction to be able to track the position of the car inside the parking garage and determine whether or not the car parked in the right parking spot or not. The aim was to come up with reasonable results that would study this strategy for the parking garage.

However, while these studies were being undertaken, several other aspects of parking garages came into light that were in some way associated with increasing the productivity of the parking garages in general. Of these two were studied further in detail. These were reservation defragmentation using bitmap techniques and revenue management techniques applicable to the parking industry.

This thesis was focused on developing methods to be able to determine where the particular incoming customer parked his/her car and based on this information, we could then carry out reservation defragmentation due to the knowledge of position of car and duration of its stay in the parking garage. Increasing revenue for the parking lot by implementing revenue management techniques is another method of increasing efficacy of parking garages. The work done for the thesis is divided into three categories:

1. Improving customer experience as well as enhancing knowledge about the parking garage, by enabling car tracking and providing real-time information to the database system about car position which in turn can be used for the benefit of the driver.
2. Improving reservation efficiency by using techniques of defragmentation which allow more number of users to use the parking garage at any given time.
3. Improving revenue management by applying booking techniques and observing probability distributions.

Since the basis of our thesis is having a reservation based system, one question that springs to mind is if it is natural to expect people to know how long they will use the parking garage for. In other words, are we expecting too much from customers, when we ask them to decide how long they will use the parking spot before they actually park. The answer to the question lies in the fact that today there are street parking meters where people put in coins (quarters) for a certain amount of time before they actually conduct the task they actually came for. Similarly, it is not unnatural to expect users to decide the duration of parking in a reservation based system because the logic of estimating time of parking is similar to using quarters to estimate time of street parking.

3.2 Overview of Techniques Used

3.2.1 Description of System

Customers will register at the company website before using the parking garage services and all this information will be stored in the Parking Lot Server (PLS) for future use. The PLS is the central computing system for the parking garage that maintains the accounts of all the users registered with the parking lot as well as tracks usage of the parking lot by the customers. Once registered, the customer will be able to look for parking space availability for a desired date and time interval with the help of a client device such as Web browser or a smart phone application. If the system returns availability of spots, the customer will be able to make the parking reservation. Assumptions about the system have been mentioned in Section 4.1.

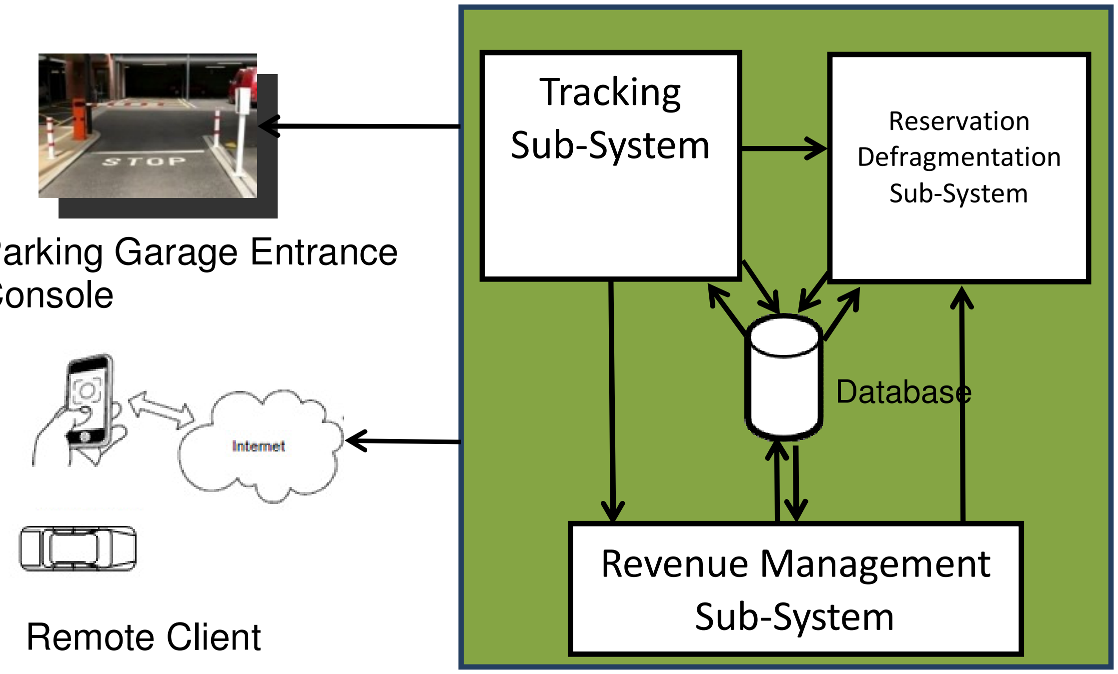

Tracking Sub-System
Reservation Defragmentation Sub-System
Database
Revenue Management Sub-System
Parking Garage Entrance Console
STOP
Internet
Remote Client

Figure 2: Parking garage technology infrastructure

The problems that have been addressed in this thesis are described below:

3.2.2 Improving Customer Experience using Vehicle Tracking

The use of reservation systems is to primarily increase the yield/revenue of the parking garage while the use of guidance is primarily to enhance customer experience in the parking garage thereby providing an incentive to the driver to return to the garage which in turn causes a reliable source of revenue. Assumptions about the system have been mentioned in Section 4.1.

A. Implemented Methods

The approach chosen by us for the thesis involves knowing the position of the car in the parking lot for two prime reasons:

1. To know the exact location of the car and where the car is likely to be headed and to track each car by its unique ID which will help us know where a particular car is at what time.
2. To be able to provide the driver the necessary instructions to proceed in the direction of the reserved spot.

There are several more reasons as to why we would want to track car position in the parking garage and these have been listed in Section 3.2.2 part B. In order to enable tracking, sensors will be needed which will enable us to determine which part of the garage the car is in. For this purpose, we propose the use of cheap ultrasonic sensors. The cost analysis of these sensors is done in Appendix 2 Part A2.1. Ultrasonic sensors are placed in the garage in strategic positions, primarily in branching points where the vehicle could take a turn that would change the direction of its progress. It is our aim to establish where the car is actually moving to determine whether it is going on the correct path or the wrong path and to know the exact location of a car by its ID. If there are very basic sign boards indicating the sensor number (which can corroborate to the spot number for easy identification), then we can have an in-garage navigation mechanism. Based on the sensor layout for the parking garage, two algorithms have been developed for the purpose of efficient tracking of the car. These two algorithms have been developed for parking layouts with two different optimizations:

a. Higher Accuracy (Higher cost) [Algorithm T1] [Section 4.3.1 Part A]
b. Lower Cost (Lower accuracy) [Algorithm T2] [Section 4.3.1 Part B]

B. Reason for choosing the Tracking Problem

1. <u>Navigation System</u>: Ability to track real-time position of vehicle in the parking garage and provide navigation to the final parking spot.
2. <u>Reduce accidents in the parking garages</u>: Since the PLS knows positions of all the cars in the Parking Lot, if there are two/more cars that are headed in the same direction with very close proximity, it can send a warning to the involved parties to take caution about an impending accident.
3. <u>Prevention of theft of car</u>: This feature can be integrated only if the departing car is being tracked as well. However, since the premise is the same, this feature can be discussed. If a car is being driven away from the parking lot, before the scheduled time, the PLS can validate this with the actual owner (by contacting on smart-phone)

and only if authorization is given, can the car be departed otherwise an alarm can be raised.

4. Speed Check: Based on the rate, at which the driver is crossing the ultrasonic sensors, the PLS can calculate the speed of the car and if the car is travelling above the speed limit, he/she can be sent a cautionary message to reduce speed.
5. Maintaining list of parked spots: Since the PLS is tracking the car position until the parking space that the car finally parks in (Algorithm T2), the PLS knows exactly as to who has parked where. As a result, any wrong parker can be made note of and fined since he/she parked in the wrong spot which could probably have been allotted to someone else.
6. Provide new spot to user in real-time: Assume that there is a user B who entered the parking lot and he was assigned a certain parking space. Suppose there was a user A, who entered the parking lot just before this user B. If A and B are both allotted spots and instead A parks in B's spot while he is in transit to this spot, then the PLS can immediately allot a new spot to B (after it records A's wrong parking) based on the position where B currently is. This will result in fuel saving and the needless travelling by B to the earlier parking spot when he would realize that his space is now occupied. Since the new spot given to B is based on B's position (obtained from the real-time sensor path of B), this is a very efficient allotment method.

3.2.3 Improving Reservation Efficiency using Defragmentation

This section introduces the discussion about the problem of being able to maximize the parking reservations that are made in a day. Our approach is to keep the complete information for all reservations in a central database and then create a *Reservation*

Bitmap (we will call this 'bitmap' in short) of all parking spots where each cell indicates whether the corresponding spot is reserved or available for that particular time period. A bitmap is a 2 dimensional array made up of 1's and 0's (Figure 3). Each 1 represents an occupied status for that spot for 30 minutes (Mathew, 2009) [23] while each 0 represents vacant status for that spot for 30 minutes. The choice of 30 minutes is based on the following two reasons:

i. Smaller the parking reservation slots (lesser than 30 minutes), higher are the possibilities of moving reservations into free spaces during the defragmentation process. However, there is greater amount of external defragmentation (larger number of reservations to defragment to obtain a defragmented reservation schedule). This increases the time complexity of the algorithm and decreases performance of the algorithm.

ii. Larger the parking reservation slot (greater than 30 minutes), lesser is the external fragmentation (i.e. fewer reservations to defragment to obtain a defragmented reservation schedule) thereby reducing time complexity of the algorithm. However, the larger the reservation slot, the lesser are the possibilities of moving reservations into free spaces during the defragmentation process.

Thus, a bitmap is constructed for each day and it has N rows for time and M columns for parking spot identifiers. Assuming 30 minute increments, there will be $N = 2 * 24$ (hours of the day) = 48 rows. A garage with 500 spots can be considered as large, so $M = 500$. Then the bitmap has $N * M = 48 * 500 = 24{,}000$ cells. Since we need only 1 bit per cell (reserved/available), the bitmap size for each day is 24,000/ 8 = 3 kB. This is easily manageable for desktop computers or servers available today.

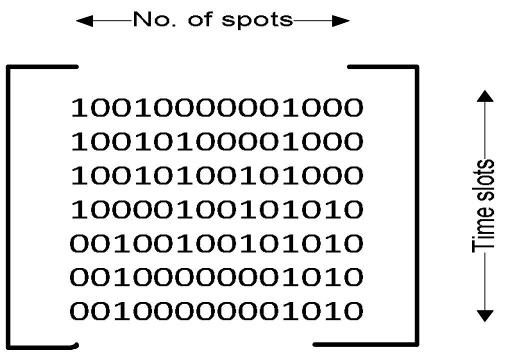
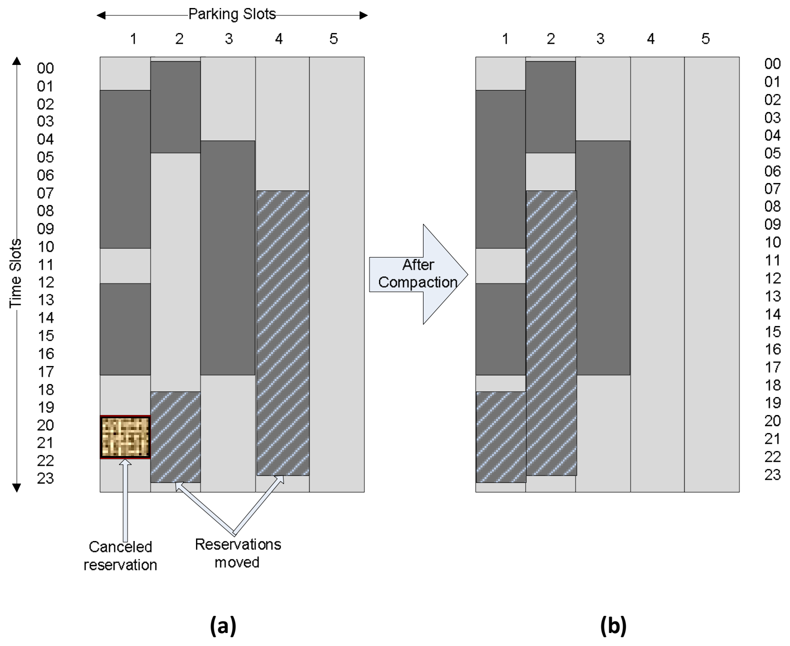


Figure 3: Reservation bitmap

Better reservation arrangement will be required in the following two conditions:

Case 1: Cancellation of reservations:



▨ Reservations moved due to defragmentation

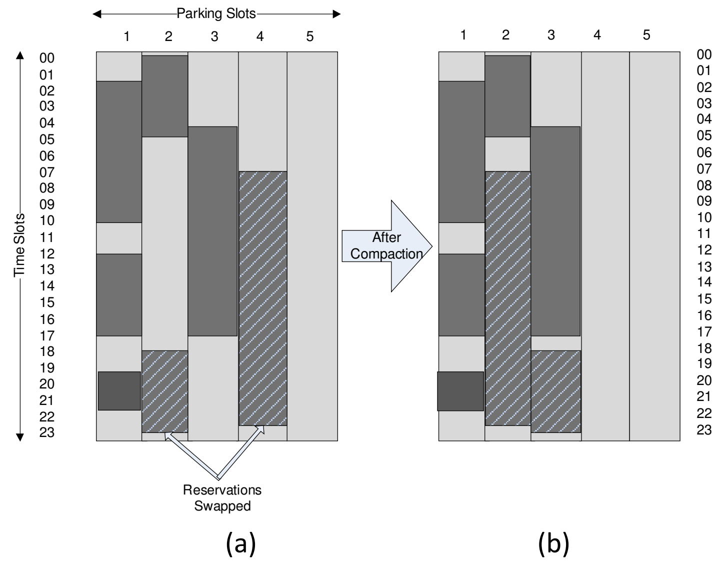
■ Reservations not moved even after defragmentation

▦ Reservations cancelled

Figure 4: Reservation defragmentation in case of cancelation

We need to consider the problem of inefficient use of parking spots. This problem may arise particularly if some of the existing reservations are canceled and random gaps are left in the reservations bitmap. See Figure 4. Figure 4 explains the process of defragmentation as it takes place when there is cancellation of reservations. The horizontal axis indicates the parking spot index while the vertical index indicates the time slots. Figure 4(a) indicates the reservations before defragmentation is applied and the Figure 4(b) indicates the reservations in the parking garage after the defragmentation is applied. This scenario considers the cancellation of a reservation which helps bring about defragmentation. As can be seen in 4(a), the reservations seem to be scattered with a lot of free time slots in between the reservations. When the reservation gets cancelled (indicated by horizontal and vertical stripes), there is availability of space in order to bring about re-arrangement of reservations. On the other hand, in figure 4(b) we can see that the reservations are much closer to each other with free space being reduced due to defragmentation.

Case 2: Re-arrangement of reservations to accommodate new reservation



▨ Reservations moved due to defragmentation

■ Reservations not moved even after defragmentation

Figure 5: Reservation defragmentation in case of swapping of reservations

Figure 5 illustrates how a free parking spot could be found by rearranging the existing reservations.

Figure 5 explains the process of defragmentation as it takes place when there is inefficient first fit allocation (See Section 4.3.2 Part b). The horizontal axis indicates the parking spot index while the vertical index indicates the time slots. Figure 5(a) indicates the reservations before defragmentation is applied and the Figure 5(b) indicates the reservations in the parking garage after the defragmentation is applied. This scenario considers the inefficient allocation of reservations due to the first fit algorithm (See Section 4.3.2 Part b). As can be seen in 5(a), the reservations seem to be scattered with a lot of free time slots in between the reservations. However, since first fit algorithm considers the order in which the reservations arrive, this causes inefficiencies when all the reservations are considered. Hence, there is availability of space in order to bring about re-arrangement of reservations. In figure 5(b) we can see that the reservations are much closer to each other with free space being reduced due to defragmentation.

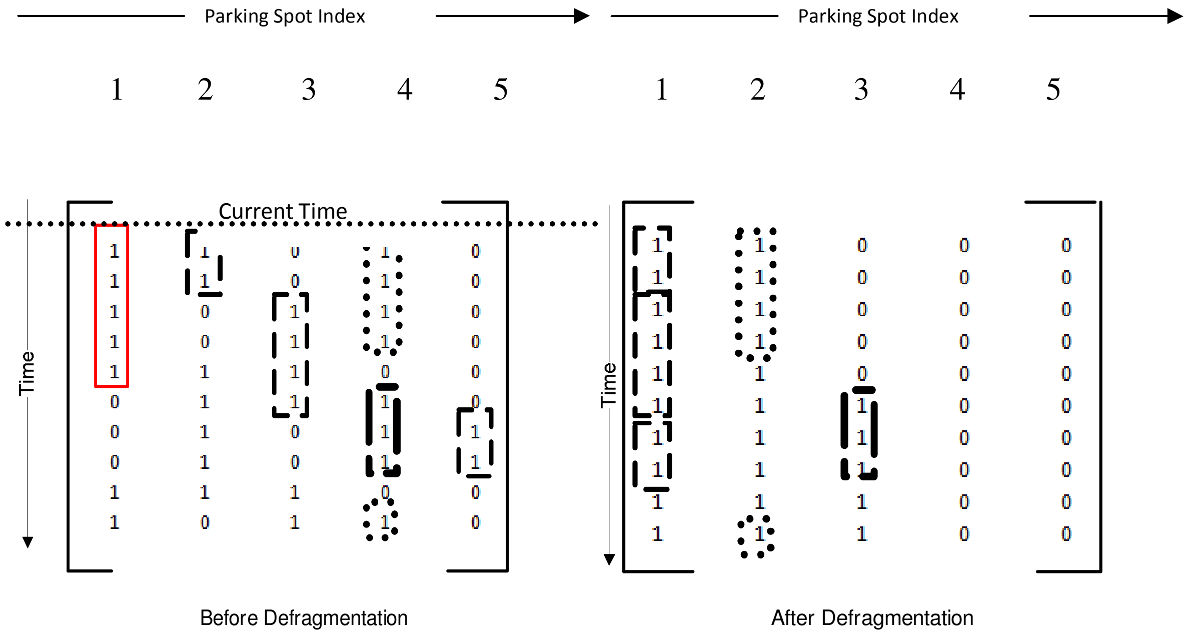
There is an addendum to (2) which can be dealt with in the same manner as the case (2). Consider that there is a parked customer 'A' who does not depart even after his reservation is complete. Consider there is another reservation for customer 'B' starting immediately after the expected end of reservation 'A'. If we do not switch reservation of 'B' to a new parking spot, it will create an overlap which will not be possible since 'A' and 'B' both cannot park at the same parking spot at the same time. Hence, to avoid this situation, we allocate the next available free spot to the customer 'B' when he is about to enter into the parking garage since at this point we know that 'A' has not vacated his spot which was meant for 'B'. In this manner, the above described problem is avoided and defragmentation is carried out. The above two cases (1) and (2) create a lot of inefficiency because of the inconsistency in the filling up of the garage In addition to this, no-shows and cancellations by customers lends to even more inter-reservation free-space leading to more space inefficiencies.. Hence, longer reservations cannot be made because of the pockets of smaller reservations that are scattered all over the reservation bitmap. Hence, we need to coagulate all reservations as much as possible to free more space. We have labeled this process as '*Reservation Defragmentation*' since our aim is to bring about defragmentation of existing reservations in order to free up as many spots as possible. The "*Reservation Defragmentation*" could be run as a daemon process during idle periods or periods of low activity. Assumptions about the system have been mentioned in Section 4.1.

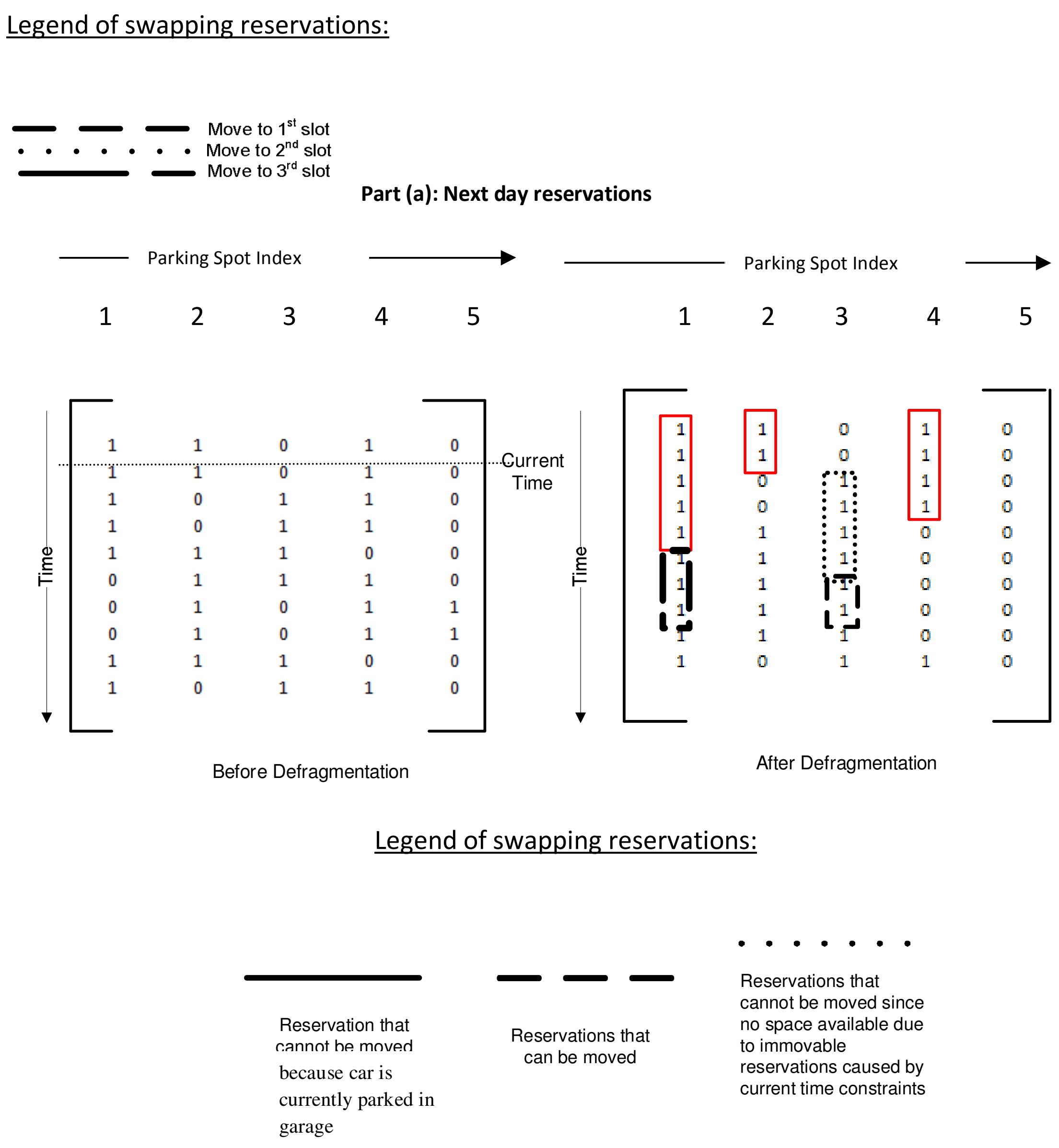
A. Implemented Methods

a. Initial allotment of new reservation: When a reservation is first made, it is saved in the central database. We then make the equivalent reservation in the Reservation Bitmap. The First Fit method is used for entering the reservation into the bitmap based on the simplicity of the algorithm as well as the speed of execution (Robson, 1977) [39].

b. Post-cancellation defragmentation: After some reservations get cancelled, there are arbitrary free spots created in the reservation time slots. In order to reduce this and bring about more packing, we run our defragmentation algorithms [Section 4.3.2].

Example of defragmentation:





Part (a): Next day reservations

Part (b):Current Day reservations

Figure 6: Example of defragmentation using sample bitmap

B. *Explanation of Figure 6 [Part (a) and Part (b)]*

When we calculate *effectiveness* of the algorithms, we consider

i. Number of free parking spots created

ii. Reduction in free time slots (by ensuring occupancy for that time slot)

iii. Reduction in *Mean length of contiguous free time slots*

iv. Increase in maximum occupancy of garage over 24 hour period

a) Consider Figure 6 Part (a). This bitmap indicates next day reservations. Let us calculate the parameter values for the above example (Figure 6 (a)) of a 10 x 5 bitmap i.e. 10 time slots and 5 parking spots:

i. Before defragmentation, there are no spots that are entirely free. After we carry out defragmentation, we can see creation of two free spots i.e. parking spot 4 and 5 (See Figure 6 Part a).

ii. Before defragmentation, up to the last reservation, there are eighteen 30 minute slots that are free, i.e. 18 0's (zeros). After defragmentation, we see only five 30 minute slots free, i.e. 5 0's (zeros), giving a reduction of 13 free time slots.

iii. When we see the mean free space, we are tracking the contiguity of free spaces. Our aim is to reduce the contiguity and yet not cause too much fragmentation if avoidable. The free space vector can be obtained by looking at Figure 6 Part (a). Before defragmentation, unto the last reservation, the free spaces vector can be given as [1,3,2,2,2,1,3,2,3,2,4,1,4,1,5,6,5,2] where each even numbered index (starting from '0') in the vector is the parking spot index and each odd numbered index (starting from '1') is the number of 30-minute time slots that are empty in between reservations. After defragmentation, the free spaces vector is [1,0,2,0,3,5,4,10,5,10] where each even numbered index (starting from ‘0’) in the vector is the parking spot index and each odd numbered index (starting from ‘1’) is the number of 30-minute time slots that are empty in between reservations. If we plot a graph of number of free time slots versus parking spot index, we will observe greater fluctuation in free time slots before defragmentation and lesser fluctuation in free time slots after defragmentation indicating more closely packed reservations post-defragmentation.

b) Consider Figure 6 (b). This bitmap indicates is applicable for defragmentation using current day reservations. The dotted line through the ‘Before defragmentation’ matrix, indicates the current time. All the reservations ‘1’s that the dotted line is passing through are currently parked in the garage and cannot be moved during the defragmentation process. In the ‘After defragmentation’ bitmap, the reservations surrounded by the solid line are the ones that could not be moved due to current time constraints because these cars are currently parked in the garage. The reservations surrounded by the dotted lines cannot be moved since there is not space available for such a movement. The cause for this is the immovable current occupancies (solid lines). The reservations surrounded by the dotted lines are the ones that can be freely moved around during defragmentation.

In order to fix the position of the current time reservations, we made the use of a *‘move flag’* which is a part of the reservation defragmentation arraylist. Details about the move flag and the arraylist are provided in Section 4.3.2 Part C (b).

C. Explanation of Figure 7

Figure 7 is the general flowchart for the reservation defragmentation algorithms. Since the algorithms consist of the initial first fit algorithm followed by the reservation defragmentation algorithms themselves, as indicated in the flowchart, once a new reservation is made it is allocated a parking spot using the first fit algorithm (See Section 4.3.2 Part B). If space is not available, then the reservation is rejected. Once the reservations have been made, the reservation algorithms (either R1 (See Section 4.3.2 Part C) or R2 (See Section 4.3.2 Part D) or R3 (See Section 4.3.2 Part E)) are executed in order to carry out reservation defragmentation.

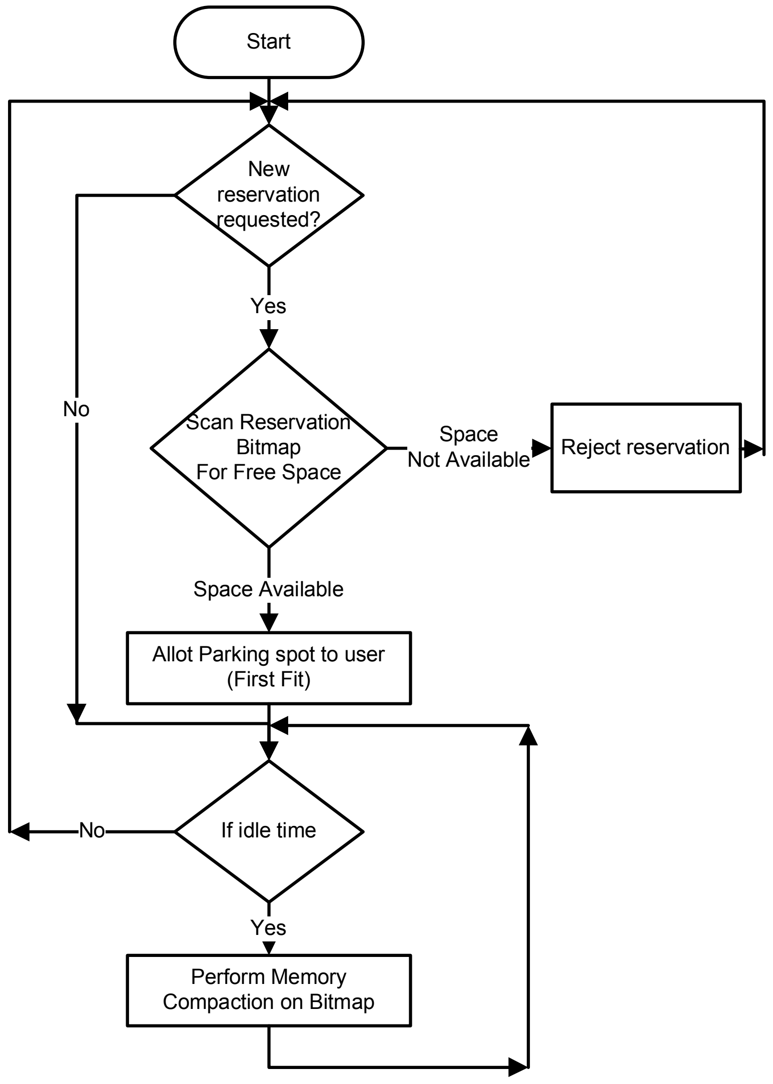

Start
New reservation requested?
Yes
No
Scan Reservation Bitmap For Free Space
Space Not Available
Reject reservation
Space Available
Allot Parking spot to user (First Fit)
If idle time
No
Yes
Perform Memory Compaction on Bitmap

Figure 7: Flowchart of defragmentation algorithm

3.2.4 Inter-dependence of Tracking Algorithms and Reservation Defragmentation Algorithms

The tracking algorithms help us determine the position of the car in the parking garage and using this we can determine if the car finally parks in the correct spot or wrong spot. The reservation defragmentation algorithms maintain a bitmap called the ***reservation bitmap*** which maintains a record of which reservations are made at what time and for what parking spot (See Section 3.2.3). It is clear from the description of the two algorithms above that the output of the tracking algorithm could be provided to the reservation defragmentation algorithms. In other words, the final position of the cars will help us determine which parking spot will be occupied and for how long (since reservations are made by individual users prior to coming to the garage). This will help us populate the reservation bitmap with the users who are currently in the parking garage. Hence, even if a car ‘X’ parks wrongly, the tracking algorithm will know this as a wrong parking and the ***reservation bitmap*** will be populated with the reservation of car ‘X’ at the parking spot where car ‘X’ actually parked and defragmentation can be carried out on the ***reservation bitmap***. On the other hand, if a car parks wrongly and the tracking algorithm is unable to determine this due to sensor failure (See Section 5.1.1, 5.1.2 and 5.1.3), then the *reservation bitmap* will be populated wrongly and defragmentation will occur incorrectly since the ***reservation bitmap*** has cars positioned at the wrong parking spots. In other words, in order to facilitate an inter-connectivity of the tracking and the reservation defragmentation algorithms, it is absolutely necessary that the tracking algorithm is able to tell accurately where a car has been positioned (even if it is parked wrongly), otherwise defragmentation will be carried out on the wrong reservation bitmap. All assumptions associated with the system including the types of sensor failure are noted in Section 4.1.3 Part A.

3.2.5 Inter-dependence of Reservation Bitmap and Overbooking Algorithm

While the reservation defragmentation algorithms have been implemented using the reservation bitmap, overbooking algorithms do not use the reservation bitmap for calculating amount of overbooking needed. The reason for not using reservation bitmaps for overbooking the same way as reservation defragmentation is the manner in which reservation bitmaps hold information. Each time slot in the reservation booking indicates whether the parking spot for that time slot is occupied by any one vehicle. However, overbooking involves allocating the same spot to multiple vehicles (before their actual arrival) in order to account for no-shows (See Section 4.1.5 Part B). But multiple bookings for the same time slot and same parking spot cannot be shown in the reservation bitmap. However, there is a way to use the bitmap in a different way for overbooking. Now instead of 1's and 0's in the bitmap, each time slot of each parking spot could hold the number of vehicles who have been designated for that parking spot and that particular time slot. For eg., if 2 cars have overlapped occupancy times for parking spot '5' between 8:00am and 8:30am, then the time slot of parking spot '5' between 8am and 8:30am will have '2'. By referring to the ***summary vector*** (See Section 4.3.2 Part C), we could see which vehicles ID's actually occupy that parking spot. While this has not been implemented, this concept could be utilized in future work.

3.2.6 Improvement of Revenue Management for Parking Garages

A. Techniques Used

The cornerstone of enabling revenue management for the parking garage is the usage of a reservation system. As we will have the data of who will come at what time and for how long, we can manipulate these numbers in order to ensure maximum occupancy for the parking garage. We have chosen two strategies for the increase in revenue and maximizing garage occupancy. These are

a. Booking Limits

The premise of booking limits [Appendix 1] is having multiple categories of bookings for different demographics of people. The difference in the categories comes about by establishing different prices for varied quality of service. In this thesis, we describe a model in which there exist two classes of parking spot reservation. These are:

1. Leisure Class: It consists of spots that are reserved by the common people for daily usage. This could include hourly/daily/monthly reservations. Reservation cost is comparatively lower. The number of spots reserved for the leisure class is known as ***booking limits.***
2. Corporate Class: It consists of spots that are reserved by corporations for their employees. Reservation cost is comparatively higher. The number of spots reserved for the corporate class in the parking garage is known as ***protection level.***

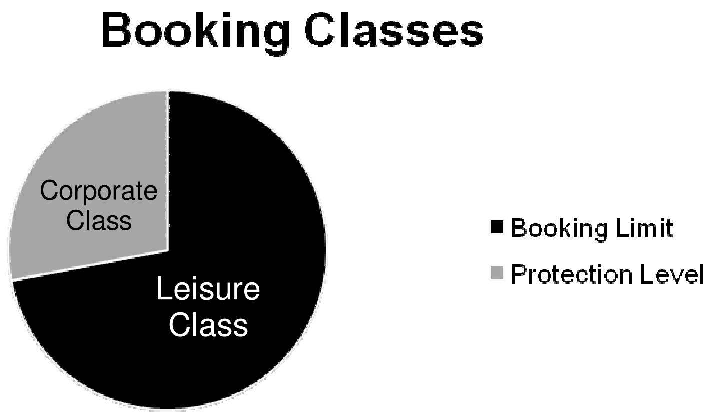


Figure 8: Categories of booking classes

The difference between the two classes can be based on several factors such as guaranteed reservations (i.e. no overbooking for corporate class), increased security (video camera surveillance) and valet parking (for Corporate Class). The booking limit will constrain the number of parking spots that these customers get (Figure 8).

b. Overbooking

Overbooking has been modified for the parking industry from its application in the airline industry (Belobaba, 1987) [5], in the following manner. In other words, there will be a percentage of people who will not show up (i.e. no show rate (NSR) will be greater than 0) despite making reservations. Thus, there will be a spot that might be empty when the reservation time starts which could have been avoided if there had been someone else to occupy the spot during that time period. In order to ensure guaranteed revenue, we will implement a practice called overbooking for the parking garage. In this the parking garage management will book a garage beyond its actual capacity by a certain extent. As a result, even if cancellations / no-shows do occur, there are sufficient people who will turn up for the parking reservation so as to avoid any spoilage costs [Appendix 1]. As long as the spoilage costs and denied parking cost [Appendix 1] are balanced in such a way that there is no revenue loss for the parking garage, overbooking is a good revenue management solution.

The practice of overbooking is not well suited to the parking industry. In the airline industry, for the airline of say capacity 100, you can fix the overbooked capacity to a higher value, say 160, because the flight durations for all passengers is the same. Hence, the revenue will be maximized if 100 people show up. In the parking industry, for a parking garage of say 500 spots, we cannot fix this number to be a fixed number, say 550, because duration of reservation is variable. Hence, in order to apply overbooking to the parking industry, we need to overbook the number of hours that the garage is used for and not the number of spots. For example, if ideally we need the parking garage to be occupied for 500 x 24 = 12000 hours. Hence, overbooking will tell us how many extra hours do we need to book, such that the parking garage will remain full even with no-show occurring.

It should be noted that hotel overbooking while in some ways is similar to the airline industry, is also similar to the parking garage industry. Hotels tend to use 'Cost Overbooking' solutions [60]. In this method, the option of loss of revenue due to a no-show (spoilage cost) is compared with the costs that the hotel might have to pay due to denying accommodation to a person who had a prior reservation. For hotels, this latter cost is significant because they do not want to lose a customer in the long run and they end up providing a lot of free amenities to compensate for the loss of good-will. As a part of the algorithm, hotels overbook rooms on a per-night basis and do so on the basis that someone will cancel at the last minute. Other methods of overbooking include 'Central limit theorem method' which also overbook rooms on a per-day basis (Toh, et al, 2002) [61].

The main goal of overbooking is to find OR>1 such that

$$AU = CAP * OR$$

Where AU is total overbooked capacity (in hours) and CAP is total capacity of the parking garage (in hours) and OR is the overbooked ratio.

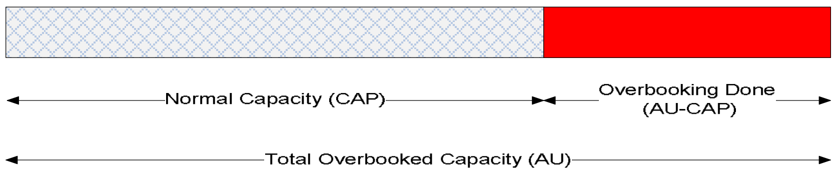


Figure 9: Implementation of Overbooking (Garage capacity is measured in hours)

1. <u>Implemented Methods</u>

In order to consider the effect of overbooking, the following algorithm has been implemented. The actual mathematical formulae used for implementation have been discussed in Chapter 4. However, an outline of the algorithm is given below:

i. Probabilistic/Risk model [Algorithm OB1]

It incorporates the uncertainty about No Show Rate (NSR) for future reservations. The aim is to find AU (Overbooked capacity) that will keep DP (Denied parking) = 0, while assuming that level of confidence is 95% (standard assumption for Gaussian distribution). Knowing the standard deviation and the show up rate, we can calculate the overbooked capacity (AU) for the parking lot with capacity CAP.

Chapter 4: Proposed Approach

4.1 Assumptions

4.1.1 Parking Lot Structure

a. The parking garage is assumed to have a capacity of 500. (Fawley Bryant Architects, 2010) (Urban Parking Concepts, 2008) [49].

b. The parking garage has 5 floors, each having 100 parking spots (Figure 10).

c. The floor of the parking garage has size 200ft x 202 ft. It has parking spots each of which is 18ft x 9ft (Temecula Municipal Code, 2010) [47].

d. The design of the parking garage is such that the up and down ramps (indicated by arrows in Figure 10) go in clockwise direction.

e. Once the user drives to the floor, he has 3 options: 1. Go up 2. Go down 3. Proceed through the floor and then go either up/down. The user should not cruise back and forth on the same floor as motion in only one direction is permitted. This is to maintain consistency for sensor tracking (Indicated by arrows on each floor in Figure 10). Dealing with violations of this rule is part of the future work and has been mentioned in Section 7.1.

f. The parking garage has one entrance and one exit (Figure 10).

g. At the entrance, there is a checkpoint, where the user information is verified and the spot allocation is provided.

h. At the exit, there is a checkpoint, where the user information is verified and the car is permitted to leave the premises.

i. All sensors are placed on the ceiling. For sensor details see Appendix 2 Section A2.1.

l. The tracking sensors as well as the occupancy sensors are ultrasonic sensors.

j. Each parking spot has an ultrasonic sensor indicating if the parking spot has been occupied or not.

k. In algorithm T1 [Section 4.3.1 Part A], there are sensors placed 9 feet apart at equidistance.

l. To prevent the drivers from entering the upper decks via the exit driveway, there will be a one-way barrier installed at the endpoint of the exit driveway.

m. All the sensors are linked together with the help of a 3-wire bus for sensor networking and power supply (Carlo Gavazzi Automation components) [10].

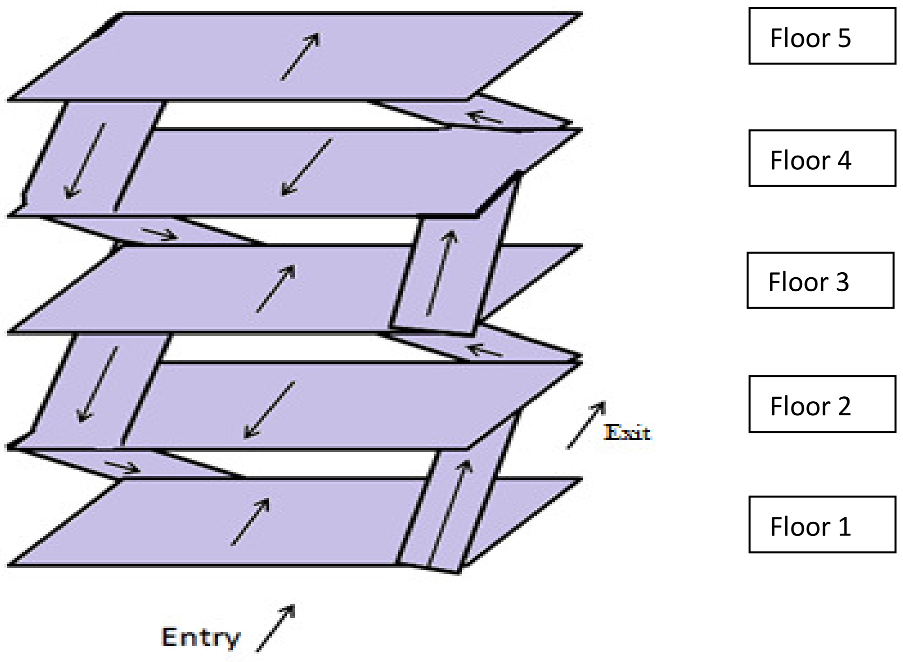


Figure 10: Prototype of parking garage design

4.1.2 Reservation System

a. Customers will register at the company website in advance of using the parking garage. At the registration time, the customer will provide demographic information and a valid email and his or her credit card number.

b. Smart-phones within the parking garage are the mode of communication between the Parking Lot Server (PLS) and the customer. The user will get his designated parking spot on the smart phone (for tracking purposes) as well as on the display board at the entrance.

c. The user is expected to park at his designated parking spot, otherwise a fine will be levied to discourage such activity. See Appendix A2.7 for how the algorithm treats violations of this assumption.

4.1.3 Position Tracking

A. Ultrasonic Sensor

a. The response time of the sensor is 70ms.i.e change of state from 1 to 0 or 0 to 1 takes 70 ms (Lee et.al., 2008) [32].

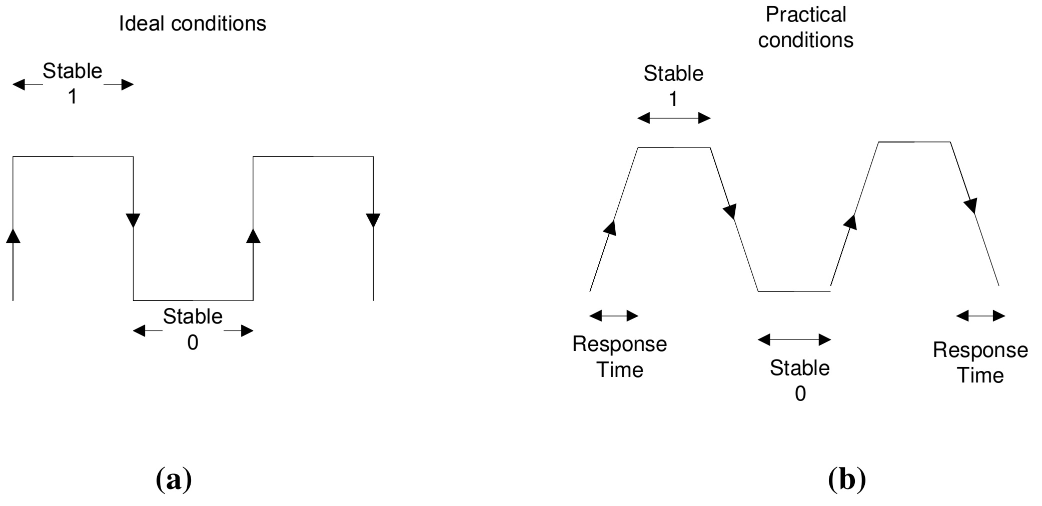


Figure 11: Response time characteristics for ultrasonic sensors

The customer will obey the maximum speed limit of 30.51 mph that has been established in Appendix 2 for the parking lot of this layout.

b. Sensor failure could either mean unable to detect once (either due to car speeding or faulty sensor) or complete failure for which maintenance would be required. The failure rate (i.e. either the sensor stops working or incorrect ‘0’ to ‘1’ or incorrect ‘1’ to ‘0’ transitions) of the sensor can vary from 2% to 50% (Goble, 2002) [19]. Sensor failure rate is independent of distance of parking spot from garage entrance.

c. The real-time position of the car is said to be lost when 25 % (or more) of the sensors fail (case 1) or 50% (or more) of the sensors fail (case 2). To see what sensor failure means, see Assumption (b) in Section 4.1.3. The choice of these sensor failure values is based on the requirement that we should be able to track the car for more than 90% (case 1) or 50% (case 2) or 25% (case 3) of the time the car is in the garage. The choice of these values is based on the following reasoning: while 100% tracking is ideal, due to sensor failure possibilities, it is not always possible to track the car at all times. Hence, in order to decide a strict yet realistic tracking scenario, we chose to decide the vehicular tracking to be correct if we were able to track the car for more than 75% (case 1) or 50% (case 2) of the time the car is in the garage. This has proved to be an effective metric in showing the efficacy of the algorithms T1 and T2.

d. Cost of the ultrasonic sensor is $3.9 (Futurlec Ultrasonic Sensors) [16].

e. The cars need to be at least 70ms apart in order to be able to be tracked accurately. [Appendix 2]

h. The minimum car size is that of a smart car (2500 mm) [Appendix 2 A2.2].

i. Detection Range of ultrasonic sensors is up to 18m and its frequency is 40 kHz (Futurlec Ultrasonic Sensors) [16].

j. Cars are assumed to be travelling at constant speed (independent of each other), inside the garage. (Lu et.al, 2009) [35] [Section 7.1 d]

B. Tracking Algorithms

a. There are sensors not only at the branching point on the floor entrance but also in the middle of every two opposite spots in each row (Algorithm T1)[Section 4.3.1 Part A].
b. There are sensors only at the branching point on the floor entrance and at the entrance of each row of spots. (Algorithm T2)[Section 4.3.1 Part B]
c. Branching points occur at the start of the parking rows (Figure 10) where the user can either choose to go to a spot in the wrong parking row or in the right parking row (Algorithm T1 and T2). They also occur at every sensor location within the parking row (in algorithm T1) where the user can choose to go left or right into an arbitrary parking spot which may or may not be his designated parking spot.
d. Cars can travel in only one direction in each row and cannot turn back. They will need to change levels/floors before coming back to the same floor. (Assumption (e) in Section 4.1.1)
e. Each algorithm T1 and T2 have been executed for 500 cars, which is the maximum capacity of our garage.
f. The arrival of cars is assumed to have a Poisson distribution with exponential inter-arrival times and arrival rate of 100 cars/hour (λ) (Flintsch et al., 2006) [56].
g. There is a 50% chance of the customer choosing the right spot (Section 4.3.1 Part A (b)).
h. The total time required for a sensor to be added to the actual path vector (Section 4.3.1 Part A) for the car is 186 ms (Appendix 2 Part A2.2).

i. If the user drives from the entrance to the exit without parking, we will track him but mark him as ‘incorrect tracking’ because he did not stop at the designated parking spot.

4.1.4 Reservation Optimization using Defragmentation

a. Parking reservations are to be made for within the next 24 hours from current time and the parking garage does not accept long-term reservations (greater than 24 hour duration reservations). This is standard procedure for most of the parking garages. One instance is mentioned here (JFK Airport Parking, 2009) [55].
b. The cancellation rate of the reservations is 15% (Section 4.1.5 Part A (b)).
c. The reservations are to be made in intervals of 30 minutes (Indian Institute of Technology, Bombay) [23].
d. Reservations can be made anytime between 12:00am (of the first day) and 12:00am (next day).
e. The reservations that are cancelled are having selected using random numbers having a uniform distribution.

4.1.5 Revenue Management

A. Booking Limit

a. For the pricing of the lower and higher price spots, the ratio chosen varied from 0.166 to 0.75 (lower price/higher price).
b. No show rates have a Gaussian distribution and have a mean at 15% of total reservations made. Since, to my knowledge, there is no historical data from parking garages, we have based this on the historical information from two diverse industries,

hotel (Bitran et al., 1989) [8] and airline industries (SITA, 2005) [44] and used the same.

c. For Poisson, the corporate car arrival rates varied from 5 cars/hour to 500 cars/hour to cover all real-world scenarios. The arrival rate of leisure customers is not fixed because our aim is to decide number of parking spots to be allocated to the corporate customers. Once this allocation is done, the rest of the parking spots are allotted to the leisure customers and as a result we do not need rate of leisure car arrivals.

d. For binomial distribution, no show rates of customers vary from 10% to 90%. We have chosen up to 90% as the worst case scenario but practically this almost never happens. While these values are given as input to the algorithm, when put into practice, only the actual no show-rate needs to be plugged in by the garage operator. Since the no show rate will occur between 10-90% (See Section 4.1.5 point c), we have chosen a broad range in order to account for possible no-show rate scenarios.

B. Overbooking-Algorithm 1(Probabilistic/Risk Model)

a. Overbooking is carried out for the entire parking garage capacity and not selectively based on booking class. See Appendix 2 A2.8 for details.

b. No show rate varies from 10% to 50% at intervals of 10%. [Section 4.1.5 Part A (b)]

c. The standard deviation of no show rate varies from 0.01 to 0.5. Since standard deviation indicated deviation from the mean and in point (b) we have a maximum of 50% of no show rate (i.e. 0.5 of total reservations), hence the maximum deviation (to give an overbooking value greater than actual capacity of parking garage) should be 0.5. We know that the mean standard deviation for the airline industry is 8% (i.e. 0.08) (Belobaba et al., 2009) [43], this value falls within our range of 0.01 to 0.5.

Since we do not have historical data to estimate standard deviation for the parking industry, we are using the values for the airline industry as they are similar in nature.

4.2 Experimental Setup

4.2.1 Software Setup

In this thesis, Java has been the main language used for coding the tracking algorithms as well as the reservation defragmentation algorithms. MATLAB has been used for the revenue management algorithms (booking limits and overbooking) due to the availability of mathematical distribution functions in MATLAB.

MySQL database on the WAMP server (Apache Tomcat) was used for storing reservations from which the reservation bitmap was generated for reservation defragmentation algorithms. JDBC (Java DataBase Connectivity) API was used to bring about the inter-link between Java and mySQL to run the algorithms.

4.2.2 Hardware Setup

A computer having 4GB RAM, T6500@ 2.10 GHz Core 2 Duo processor, 32 bit Windows 7 OS and 160 GB HDD was used to run simulations as part of the thesis.

4.3 Implementation Algorithms

4.3.1 Tracking

While there are a few systems that have been developed to track cars in the parking lots with various technologies (Section 2.2.2), to my knowledge, there is not much research done on tracking a car in real-time in a parking garage using ultrasonic sensor technology for positional guidance. The aim is to have real-time position of the vehicle in the parking garage using this algorithm.

A. Algorithm T1

a. <u>Illustration</u>

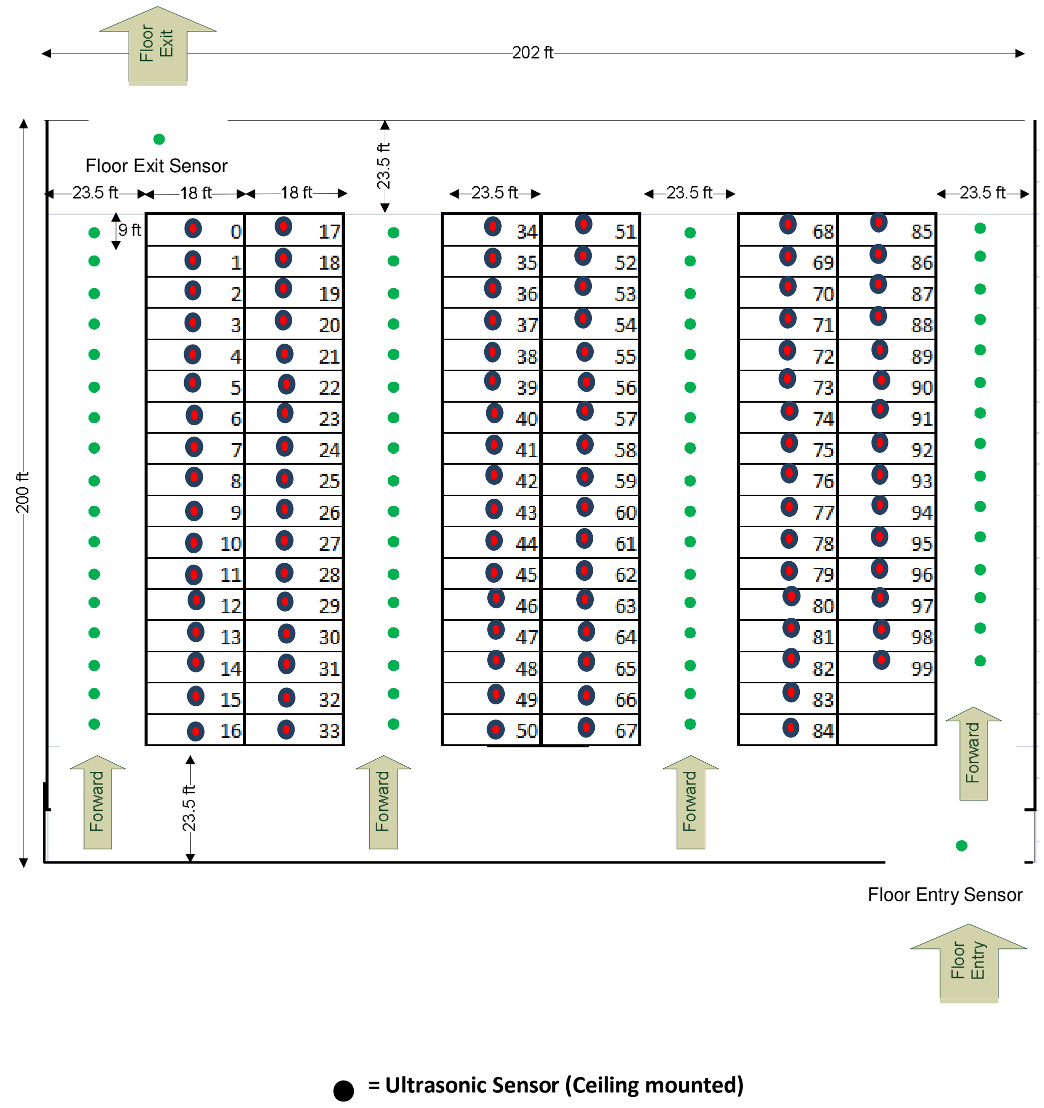


Figure 12: Floor plan for sensor layout for Algorithm T1 (For one floor of the garage in Fig. 10)

b. <u>Basis of Implementation</u>

This algorithm implemented is for a system that has an ultrasonic sensor in the middle of every two opposite spots in each row (See Figure 12). As there are more sensors used, there will be a higher cost issue but more accuracy since there is more information available (Appendix 2).

c. Algorithm Details

The assumptions for this algorithm are stated in Section 4.1.3. The algorithm uses path vectors to determine the position of the car. A path vector is nothing but a list ultrasonic sensors that the car has crossed or is about to cross (depending on the type of path vector). When the car arrives at the parking garage entrance (after having previously made a reservation), the customer is allotted a parking space by giving the nearest available spot. Based on this allotted parking space, the Parking Lot Server (PLS) generates a 'designated path vector', which consists of a list of sensors that the car should cross if he parks correctly at his designated parking spot. While there could be several long paths as seen in Figure 12, even if the car goes to the next floor or enters a wrong aisle, the car will be tracked but marked as wrong tracking. However, it is known that users have a tendency to park wrongly for various reasons such as seeing an empty spot before reaching their own spot. This has been taken into consideration by assuming that a customer will choose either his spot or another with a probability of 50%. This choice of probability does not affect any results. It is arbitrarily chosen since the choice of any value will give the same results as the probability value is not a parameter in the algorithm. As soon as the user enters the parking garage after getting his spot, the PLS instantiates another path vector for the user based on the actual path that the user will take. This is called the 'actual path vector' and consists of a list of all the sensors that the user's vehicle is crossing in real-time. It should be noted that after the user enters the parking garage, he can choose to go into any aisle as shown in Figure 12. The algorithm knows which aisle the customer entered into based on which floor sensor was set off (indicating which garage floor) and when the first row sensor of the aisle is set off (indicating which aisle of that garage floor). As the user crosses a sensor, after the processing time, it gets added to the 'actual path vector'. If the user keeps circling the parking garage and then chooses a parking spot, the algorithm will track his position and just mark the parking as wrong parking if he chooses the wrong spot to park finally. It should be mentioned that there are two types of sensors that are employed in the parking garage system:

a. Ultrasonic Tracking Sensors
b. Ultrasonic Occupancy Sensors

While (a) helps us track the real-time position of the car in the parking garage, (b) helps us determine whether the parking spot is occupied or not. One important observation is that the tracking sensors need to have many constraints due to the fact that they are tracking cars in motion whereas occupancy sensors are only used for stationary vehicles. These constraints include maximum speed of the moving car and minimum inter-car distance for accurate detection (Appendix 2). These constraints come into picture due to the response time of the ultrasonic sensors (as shown in Figure 11 part b) and have been explained in Appendix 2. For occupancy sensors, this response time is irrelevant since the duration of stay of cars at the spot is in hours where as the response time is in milliseconds. In other words, we can consider the response time of the ultrasonic occupancy sensors to be zero and that the characteristics of these ultrasonic sensors to be ideal (Figure 11 part a). Hence, the constraints of car speed and inter-car distance are not applicable to occupancy sensors. In our simulation, we have considered all sensors to be of the tracking sensors type which is a correct assumption because for occupancy monitoring, the ultrasonic sensor behaves like an occupancy sensor since response time is negligible as mentioned above.

The following features have been enabled using this tracking algorithm:

Feature 1: Our main aim is to determine that the car is driving to its designated spot and if it is not, this should be recognized and the driver error should be recorded. When the driver goes on a path that is different from the designated path, then a decision needs to be made quickly whether he/she is going on the right/wrong path. The 'actual path vector' is deemed to be wrong when the final parked spot is different from the designated parked spot.

Feature 2: It is our intention that the tracked path of the car is 100 % accurate whether the driver takes the right or the wrong path. This means that we should have every sensor tracked in the 'actual path vector'. This would ensure that we are not making errors in knowing the actual position of the vehicle. However, the inaccuracy/failure of the sensor causes inaccuracies in the algorithm. Consider Figure 12. Assume that the user enters a particular row where the designated spot is located. Further assume the user needs to cross 8 sensors to reach his final spot, and that the final 3 sensors fail. Since the last sensor recorded is nowhere near the spot where the user actually went and parked we will record this as a wrong parking. This is not correct though since the user actually parked in the right spot but we have marked it as an incorrect parking. There is no real provision in the algorithm to correct this issue, but there is a method by which we have counted the number of faulty recordings made. If the number of failed sensors along the path which the car travels as a percentage of actual path vector is less than 50% (case 1) [See Figure 13] or 75% (case 2) [See Figure 14] of the size of the designated path vector, then we mark this as a correct recording else it is a wrong recording. This implies that we cannot trust this actual path vector and we cannot know whether it is right or wrong. This is a flaw in the algorithm and some work can be done in the future to correct this/know the correct outcome (See Section 7.1). To know about the inter-dependence of tracking algorithms on reservation defragmentation algorithms, see Section 3.2.4.

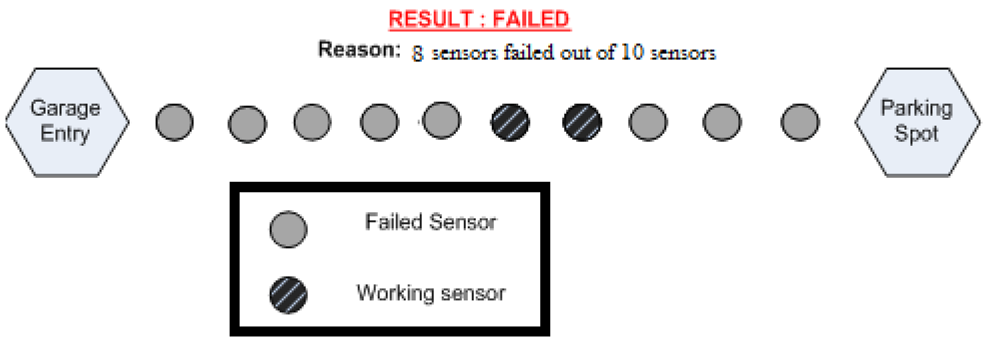


Figure 13: Example of at least 75% sensor failure

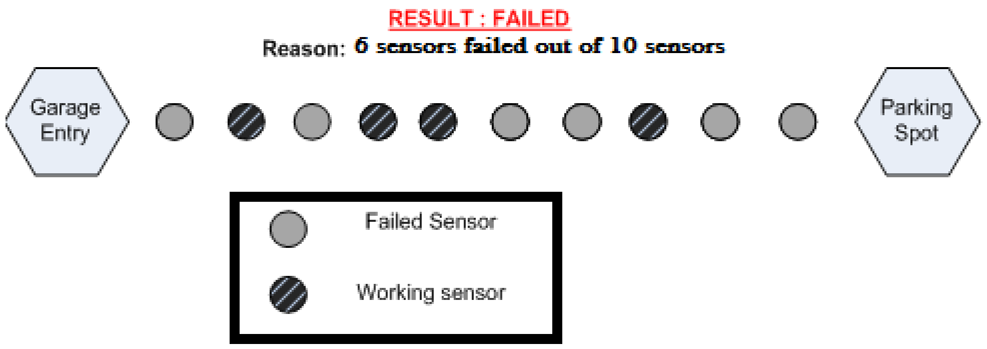


Figure 14: Example of at least 50% sensor failure

<u>Feature 3:</u> A very intuitive feature that forms a part of this algorithm has to do with dynamic allotment of spots. This takes care of allotting a new spot to a user A in case his spot has been taken by some user X. This feature is explained below.

1. Assume that users A and B made reservations such that user A arrives at the parking lot much before the user B. If user A is given some spot X and he instead parks instead at spot Y that has not been allotted to anyone, then it does not matter since only a record is made against user A that he has parked wrongly.
2. The bigger problem occurs if A and B arrive at the parking garage in such a way that both are given their spots X and Y and enter the parking garage together to head to their destinations. However, assume that user A instead of going to X goes to Y and arrives there first. This would mean that when user B reaches Y he will see it is occupied and will have to circle the parking lot to find his parking space which is exactly the problem being attempted to solve in this thesis.

In this case, the moment user A parks wrongly in Y, an alert is issued to user B on his/her smart-phone saying that there is a spot change warning and based on the position of the user B (since we know the real-time actual path vector of every user), an available spot nearest to B is allotted to user B. This implementation makes the use of hash maps in Java to determine whether both the users are in the parking garage and that A has parked in the spot of B. While alerting using smartphones has its risks, to my knowledge, there is no other method where the user can be alerted in real-time. In fact, feasibility studies for smartphone alerts are being tested by the San Francisco's Department of Parking and Traffic (RFID Journal) [40]. It should be noted that once the driver enters the parking garage, his vehicle will be tracked, even if he chooses to not park and exits the garage.

B. Algorithm T2

a. Illustration

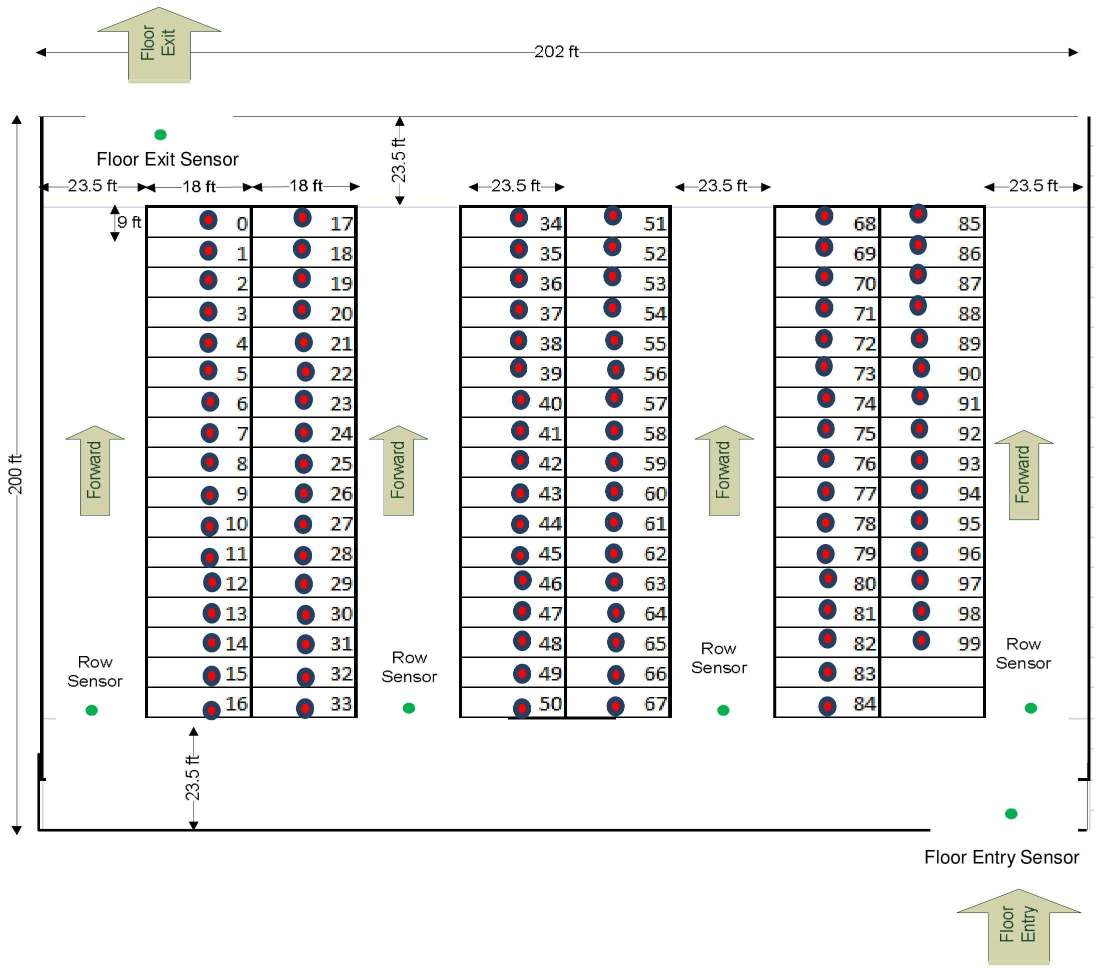


Figure 15: Floor plan of sensor layout for Algorithm T2 (for one floor of the garage in Fig. 10)

● = Ultrasonic Sensor (Ceiling mounted)

b. Basis of Implementation

This algorithm implemented is for a system that has an ultrasonic sensor at the beginning of each row of every level [Figure 15]. The purpose of these sensors is to just tell us as to which row the user got into and once the final spot occupancy sensor goes off, we will know where the user has parked. Between this main row sensor and the final spot sensor, there is no tracking of the car. As there are much fewer sensors used (as compared to Algorithm T1), there will be a lower cost involved but lesser accuracy since there is lesser information available (Appendix 2). Lesser the accuracy, lesser is our ability to track the car in the parking garage correctly. For a cost v/s accuracy comparison between Algorithm T1 and T2 see Appendix 2 A2.9.

c. Algorithm Details

This algorithm is conceptually the same as Algorithm T1 and implements the same features as Algorithm T1. However, it differs in the amount of tracking it actually does due to the reduced amount of sensors. It also makes the use of 'designated path vector' and 'actual path vector'.

d. Simulation Details for Algorithm T1 and T2

Multithreading is used to simulate the arrival of multiple cars. Sensor failure will occur with a probability of 0.02, 0.05, 0.1, 0.2 and 0.5. Simulations have been run for all these conditions. Poisson arrivals have been simulated with arrival rate (λ) 100 cars/hour (Flintsch et al., 2006) [56]. The simulation has been run for 500 vehicles. This means that the simulations can be run for 500 vehicles in the parking garage simultaneously and all their positions will be recorded accurately provided the speed limit is followed and minimum distance requirement is observed (Appendix 2 Part A2.3). See Section 3.2.4 for discussion of implications if speed limit is not obeyed and the tracking algorithm fails to track the vehicle correctly. We record the number of wrong parking as well as the number of wrong predictions that are made due to faulty sensor behavior.

4.3.2 Reservation Defragmentation

A. Introduction

There are three main metrics that are chosen to determine efficiency of implemented algorithms:

1. Decrease in number of occupied parking spots to accommodate more reservations
2. Increased defragmentation of reservations (measured by determining how many free time slots exist before the last reservation in the highest spot). Higher the defragmentation, lower is the number of time slots available.
3. Decrease in mean free time slot lengths in between reservations.

It should be noted that all the reservations are considered within a 24 hour period. The reason for doing this is that our parking garage does not deal with long-term parking but only with parking done on a daily basis. Hence, we only take into account those reservations that will be made 24 hours from the current time. Since we consider reservations in thirty minute durations, hence each parking spot has 48 possible half hour reservations. Hence, when the user makes reservations, he needs to do it in multiples of half-hour. We have considered the usage of defragmentation algorithms for

1. Advance reservations (Next day reservations)
2. Current reservations (Keeping track of current time of day and existing reservations)

We consider the effect of defragmentation on next day reservations, because they give us a clear indicator of the efficiency of the defragmentation algorithms using parameters such as increase in available parking spots (Section 5.2.4) and decrease in fragmented time slots (Section 5.2.2). The benefit of conducting defragmentation on next day reservations is that there is no constraint of time and the reservations can be moved around freely to provide high values of defragmentation (Section 5.2.2, 5.2.4).

In order to determine the effect of defragmentation for reservations made for a parking garage, we should also take into consideration current time of the day. In other words, consider the effect of the defragmentation algorithms on reservations that are currently in the parking garage or are scheduled to begin in a short time away from the current time. There are a few considerations that should be taken into account for such a scenario:

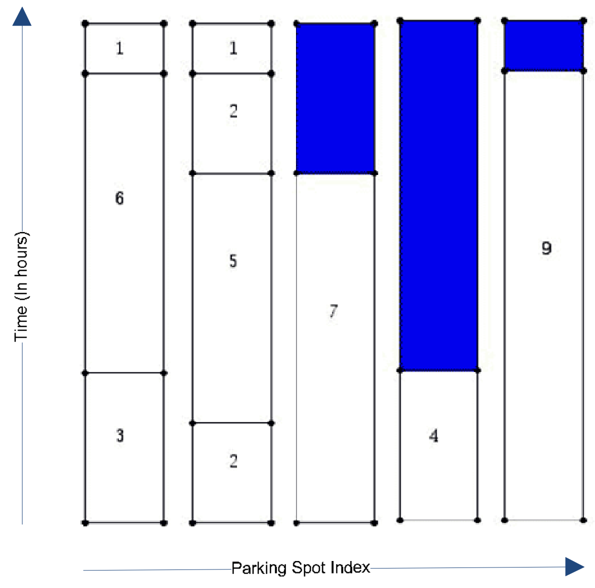
i. If we are keeping a tracking of current time of the day during the course of defragmentation, we should make sure that the reservations that are currently in the parking garage should not be moved during the defragmentation process since this does not make practical sense.

ii. In order to consider the effect of defragmentation on increasing occupancy of the garage, we need to keep the total time of consideration fixed. In other words, we need to fix the duration over which we are expecting to see an increase in occupancy, otherwise it will be difficult to ascertain the increased occupancy. Hence, we are observing parking garage occupancy over 24 hours consistent with our observations in the previous section (next –day reservations).

To determine the efficiency of defragmentation on current time reservations, the metrics we choose are a) increase in occupancy as well as b) decrease in *mean length of free time slots* (Section 5.2.6). The reason we do not choose increase in parking spots available and decrease in free time slots caused due to defragmentation as our metrics is because in current time reservations, everything is dynamic since everything is in real-time. Hence, it does not make sense to keep track of parameters which are applicable at only certain points of time (i.e. specific to number of reservations at that point of time). Hence, it was necessary to use metrics which would determine the efficiency of the algorithm over the span of the whole period for which defragmentation was carried out. Increase in occupancy was the appropriate metric for this purpose.

B. First Fit Algorithm

This is a greedy approximation algorithm (Robson, 1977) [39]. For each reservation, it attempts to place the reservation in the first parking spot that can accommodate the reservation (Figure 16). If no free parking spot is found, it rejects the reservation. This algorithm is used to allot parking spots to users when the reservations are first made. The black spaces in Figure 16 indicate the free time slots while the integers indicate the duration of the reservation (in hours).



☐ - Reservation made

■ - No reservation made

Figure 16: First Fit Algorithm

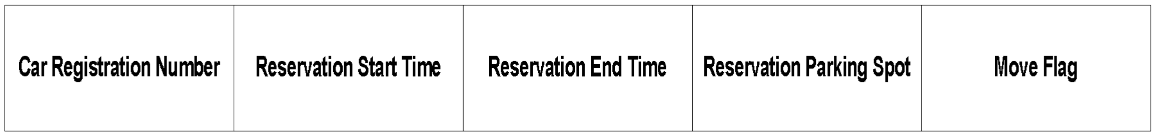
C. Algorithm R1

a. Basis of Implementation

This is an implementation of the algorithm mentioned in the book ‘Algorithm Design’ by Jon Kleinberg (Kleinberg et.al, 2005) [28]. It is an algorithm for interval scheduling and is used for job scheduling. While it is meant for interval scheduling, its application to parking reservations is straightforward, since we replace the parking reservations made by the customer as jobs and the parking spots as intervals of time.

b. Algorithm Details

The assumptions for this algorithm have been stated in Section 4.1.4. Assume that we have a reservation ‘j’ starting at time s_j and ending at $f_{j.}$ Two reservations are said to be *compatible* if they do not overlap in time in the same spot. Our goal is to find the maximum subset of mutually compatible reservations. This algorithm sorts all the reservations in increasing order of finish times. Once this is done, then we take each reservation sequentially and see if it is compatible with the previous one taken. Reservation 'j' is said to be compatible with a reservation 'A' if $s_j \geq f_A$. The algorithm involves sorting the ***'Reservation Defragmentation arraylist'*** (Figure 17) according to finish times and then inserting the reservations into the reservation bitmap.



Car Registration Number	Reservation Start Time	Reservation End Time	Reservation Parking Spot	Move Flag

Figure 17: Reservation Defragmentation arraylist

For the current time implementation of defragmentation, we maintain a timer that keeps a track of current time. Based on this time, we reference the value of the row of the reservation bitmap which corresponds to the current time. Since the algorithm runs defragmentation based on current time, we should not move the reservations that are currently in the parking garage. In order to know which these reservations are, we set the *'move flag'* to '1' to indicate that the reservation should not moved in the defragmentation process. Otherwise, *'move flag'* is set to '0' by default. If two reservations do not collide, then they are kept in the same spot (if either reservation is not currently in that spot and needs to be moved to that spot, then the movement of the reservation is done). If they collide, then they are kept in separate spots and this is continued for all reservations. However, since memory defragmentation will cause these reservations to move around and we will not be able to keep a track of which reservation was moved where using a bitmap only, we make use of a ***Summary vector*** (Figure 18) which has a summary of the registration number, start time and end time of reservation as well as the spot where it has been allocated. A prototype of the *summary vector* is shown in Figure 18.

Car Registration Number	**Reservation Start Time**	**Reservation Stop Time**	**Reserved Parking Spot**

Figure 18: Summary Vector

When reservation defragmentation occurs even the summary vector is re-arranged to reflect the new spot changes. This algorithm can be used for reservation defragmentation due to cancellation as well as inefficient allotment due to First Fit algorithm because all the reservations are sorted according to finish time and this will take care of the misplaced reservations due to First Fit (Section 4.3.2 Part B). The complexity of this algorithm is O (*n* log *n*) (Appendix 2 Part A2.5).

c. Flowchart

The flowchart for Algorithm R1 is given in Figure 19.

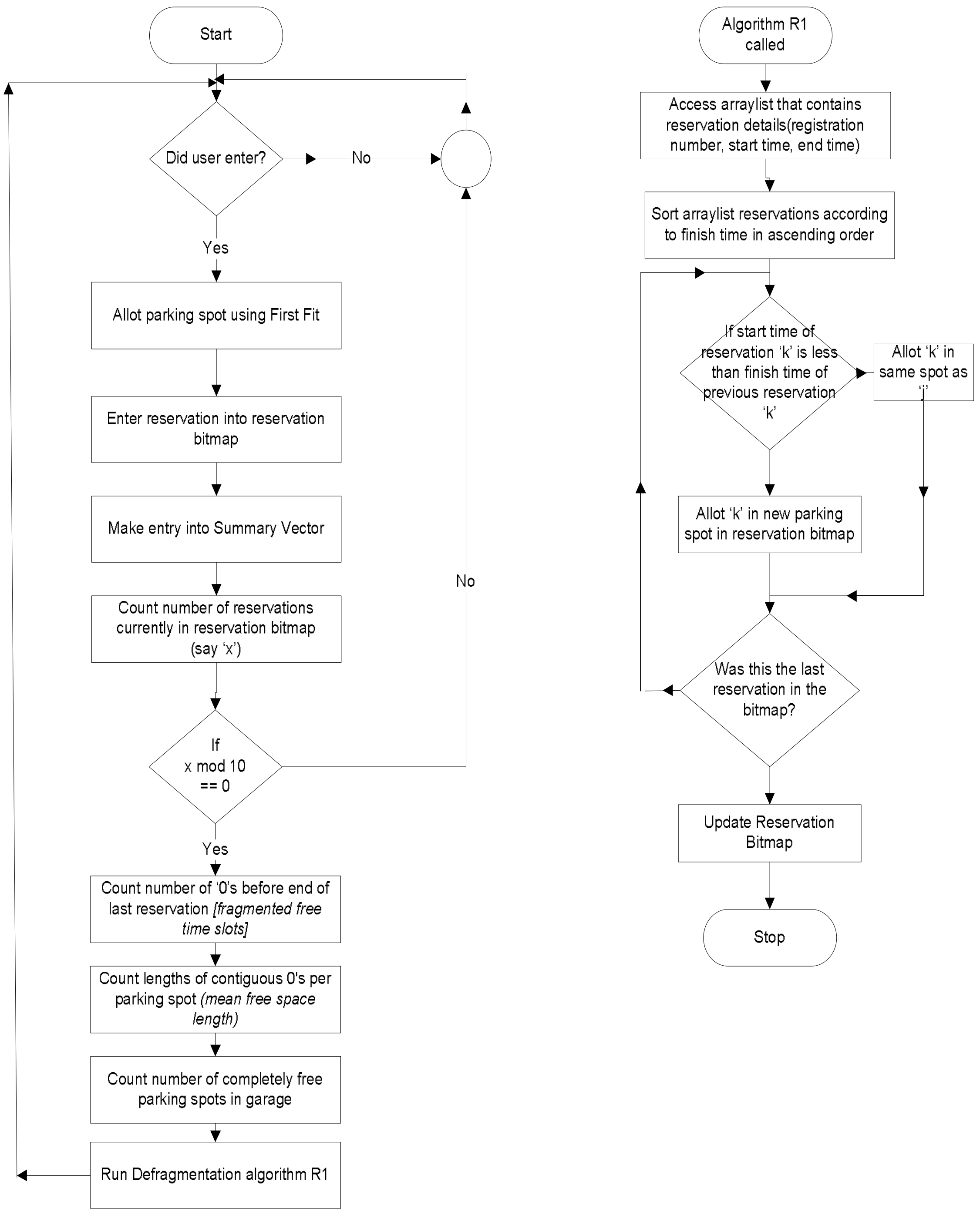

Start
Did user enter?
No
Yes
Allot parking spot using First Fit
Enter reservation into reservation bitmap
Make entry into Summary Vector
Count number of reservations currently in reservation bitmap (say 'x')
If x mod 10 == 0
No
Yes
Count number of '0's before end of last reservation [fragmented free time slots]
Count lengths of contiguous 0's per parking spot (mean free space length)
Count number of completely free parking spots in garage
Run Defragmentation algorithm R1
Algorithm R1 called
Access arraylist that contains reservation details(registration number, start time, end time)
Sort arraylist reservations according to finish time in ascending order
If start time of reservation 'k' is less than finish time of previous reservation 'k'
Allot 'k' in same spot as 'j'
Allot 'k' in new parking spot in reservation bitmap
Was this the last reservation in the bitmap?
Update Reservation Bitmap
Stop

Figure 19: Flowchart of Algorithm R1

D. Algorithm R2

a. Basis of Implementation

This algorithm is called ‘Recursive First Fit’. This algorithm is a modification of the first fit algorithm (Section 4.3.2 Part B) in which we keep performing first fit of the reservations at periodic intervals. This algorithm has been developed by us for the purpose of defragmentation. It is composed of two parts i) initial allotment using First Fit Algorithm (Robson, 1977) [39] ii) Defragmentation using First Fit Decreasing Algorithm (Lodi et.al, 1996) [53].

b. Algorithm Details

In this algorithm, whether cancellations occur or not, the existing reservations are re-arranged into the bitmap in a first fit manner. However, while the initial allotment of parking spots is done using First Fit algorithm, the defragmentation is done using ‘First Fit Decreasing’ algorithm (Lodi et al., 1996) [53]. In the first fit decreasing algorithm, we sort the incoming reservations in decreasing order of duration of reservation and then apply the first fit algorithm. Since first fit algorithm is run repeatedly at periodic intervals to generate reservation defragmentation, the algorithm is called ‘Recursive First Fit’. Figure 20 shows the operation of the algorithm. We can see that the integers represent the durations of the reservations while the x-axis represents the parking spot index. The reservations of largest duration are placed in first lower spots and smaller size durations are allocated. For the current time implementation of defragmentation, refer to section 4.3.2 Part C (b) since the same logic is applied even to algorithm R2.

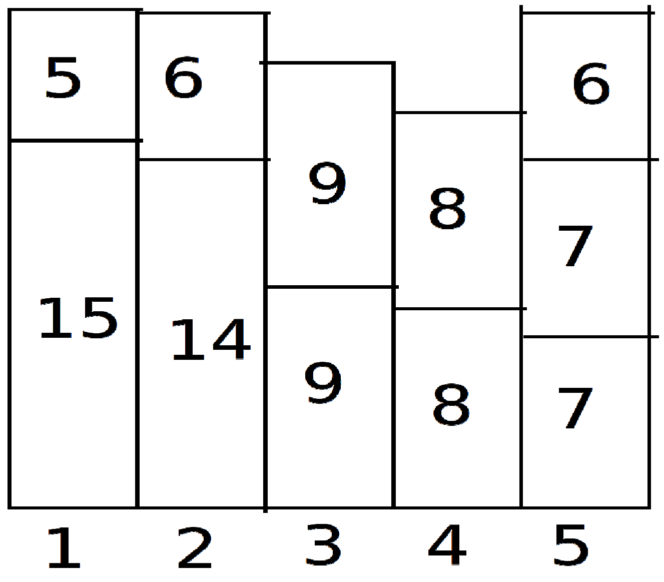


Figure 20: First Fit Decreasing Algorithm

c. <u>Flowchart:</u> Figure 21 shows the flowchart for Algorithm R2.

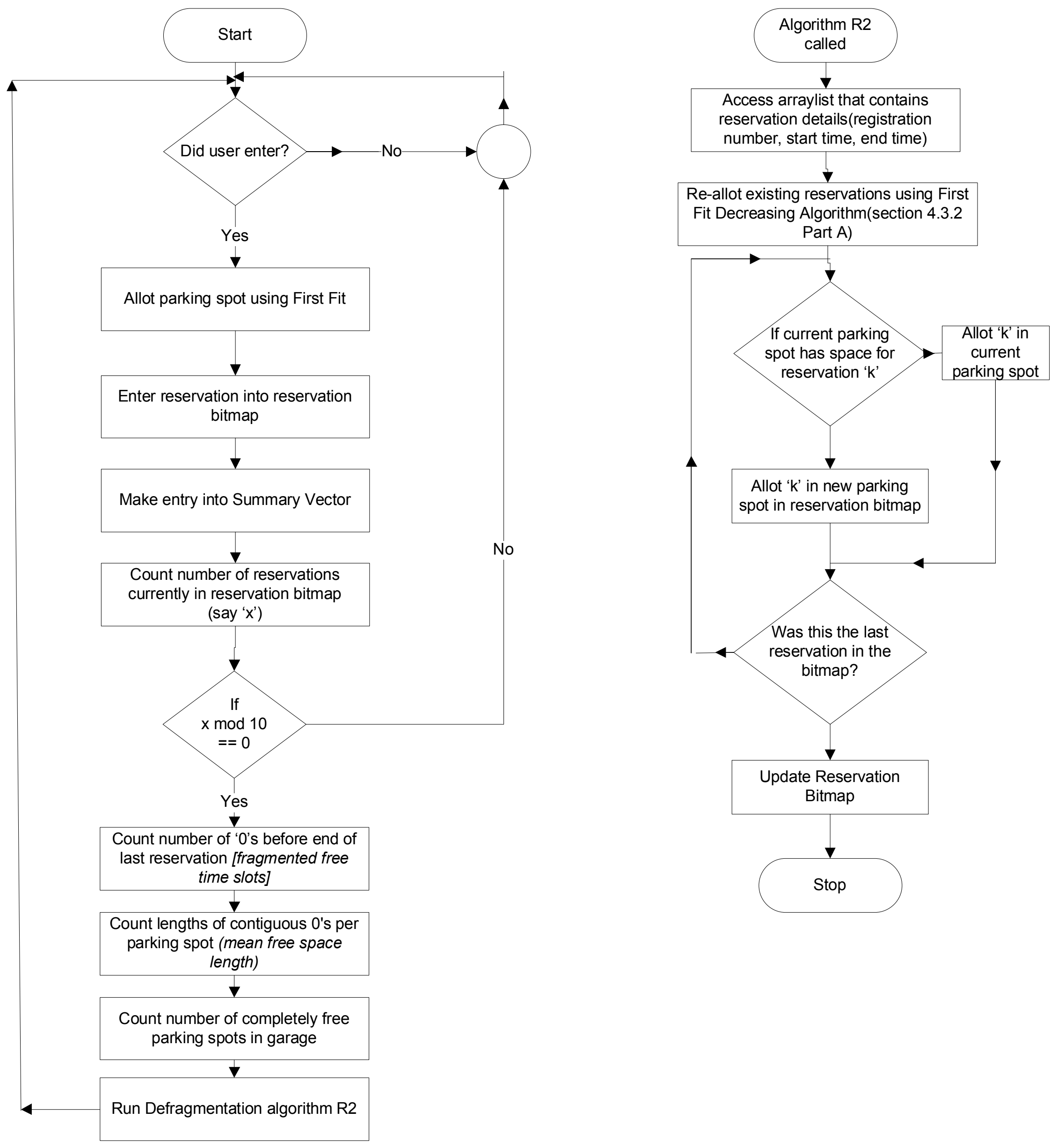

Start
Did user enter?
No
Yes
Allot parking spot using First Fit
Enter reservation into reservation bitmap
Make entry into Summary Vector
Count number of reservations currently in reservation bitmap (say 'x')
If x mod 10 == 0
No
Yes
Count number of '0's before end of last reservation *[fragmented free time slots]*
Count lengths of contiguous 0's per parking spot *(mean free space length)*
Count number of completely free parking spots in garage
Run Defragmentation algorithm R2
Algorithm R2 called
Access arraylist that contains reservation details(registration number, start time, end time)
Re-allot existing reservations using First Fit Decreasing Algorithm(section 4.3.2 Part A)
If current parking spot has space for reservation 'k'
Allot 'k' in current parking spot
Allot 'k' in new parking spot in reservation bitmap
Was this the last reservation in the bitmap?
Update Reservation Bitmap
Stop

Figure 21: Flowchart for Algorithm R2

E. Algorithm R3

a. Basis of Implementation

This algorithm was developed by us for the purpose of achieving higher levels of reservation defragmentation. The operation of this algorithm is to mimic the disk defragmentation technique in memory management where the free space in the memory is tracked so as to move other files into this free space (Jensen, 1994) [24].

b. Algorithm Details

The algorithm makes use of a data structure called 'free space vector' which is a Java vector that keeps a track of how much free space (i.e. number of time slots that are equal to '0') is present in each parking spot of the garage. The free space vector has a length of 500 entries (one entry for each parking spot) which are all initialized to '48' because if no reservations are made, all the time slots are free and there are 48 time slots per parking spot. When reservations are made, the time slots for the particular parking spot keep getting occupied and the '0''s become '1''s to indicate occupancy. The free space vector for each parking spot is generated by counting the total number of 0's in that parking spot, present in the reservation bitmap, since each "0" corresponds to a 30- minute interval that is available. Every time a reservation is made, the free space vector is checked to see if there is space in any spot to accommodate the new incoming reservation. Each time some cancellation occurs, the free space is created and the free space vector is updated for future reference. For the current time implementation of defragmentation, refer to section 4.3.2 Part C (b) since the same logic is applied even to algorithm R3.

c. Flowchart

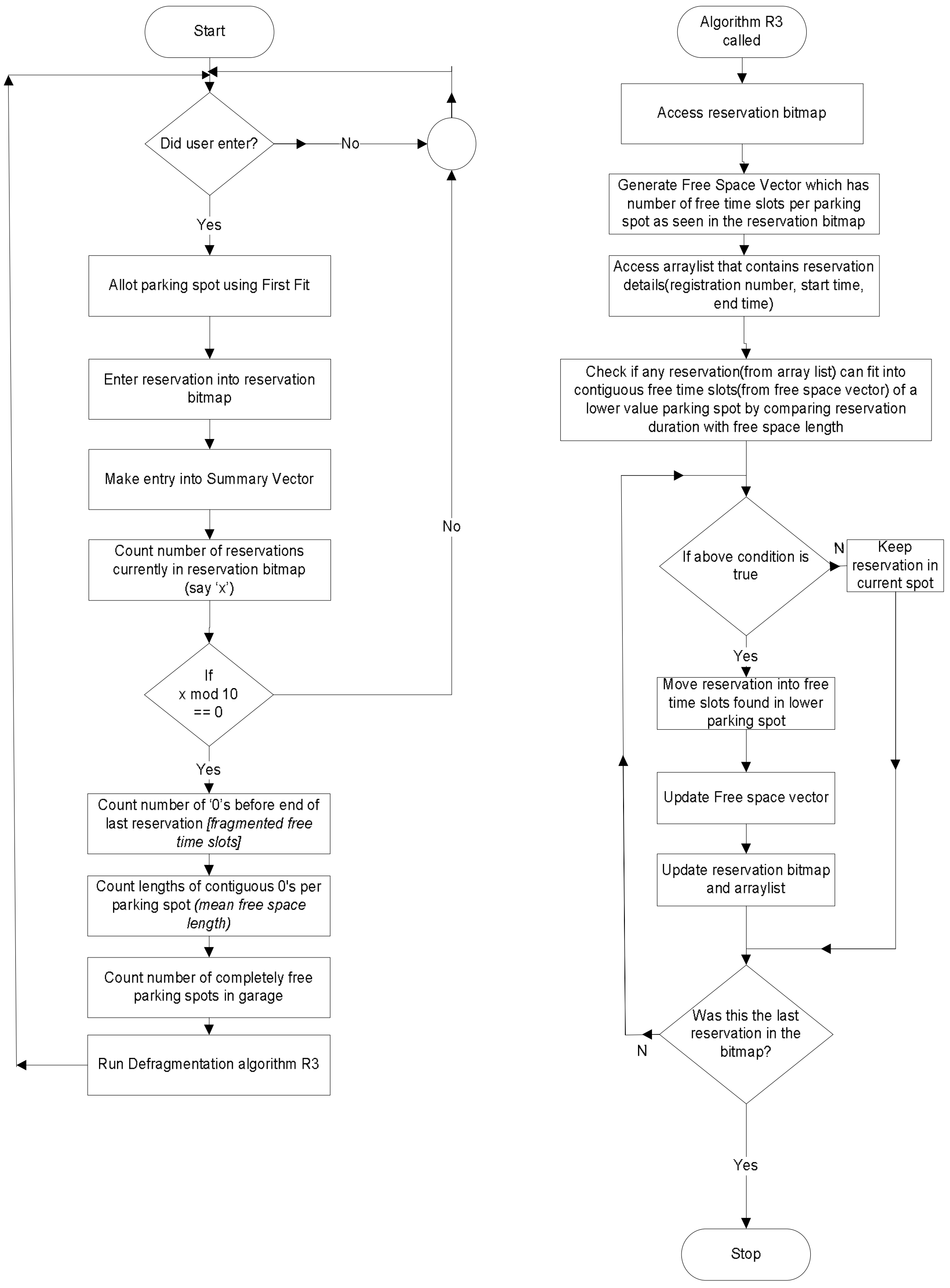

Start
Did user enter?
No
Yes
Allot parking spot using First Fit
Enter reservation into reservation bitmap
Make entry into Summary Vector
Count number of reservations currently in reservation bitmap (say 'x')
If x mod 10 == 0
No
Yes
Count number of '0's before end of last reservation [fragmented free time slots]
Count lengths of contiguous 0's per parking spot (mean free space length)
Count number of completely free parking spots in garage
Run Defragmentation algorithm R3
Algorithm R3 called
Access reservation bitmap
Generate Free Space Vector which has number of free time slots per parking spot as seen in the reservation bitmap
Access arraylist that contains reservation details(registration number, start time, end time)
Check if any reservation(from array list) can fit into contiguous free time slots(from free space vector) of a lower value parking spot by comparing reservation duration with free space length
If above condition is true
N
Keep reservation in current spot
Yes
Move reservation into free time slots found in lower parking spot
Update Free space vector
Update reservation bitmap and arraylist
Was this the last reservation in the bitmap?
N
Yes
Stop

Figure 22: Flowchart for algorithm R3

d. Simulation Details for Algorithm R1, R2, R3

Reservations are made using multithreading. The reservation defragmentation algorithm is run for 200, 500, 1000, and 1750 total reservations made for the day. The garage capacity is 500 spots. The reservations are made in such a way that arrivals will have Poisson distribution with exponential inter-arrival time and having an arrival rate of 20 cars/hour to 100 cars/hour. The reservations have random durations that are exponentially distributed (Figure 29). All the algorithms are run for 100 sets of up to 1750 reservations for 15% cancellation (Section 4.1.5 Part A) in order to prove the reliability of the algorithms under all conditions. In order to simulate cancellations, we have chosen cancellation rates of 15% since this is the usual cancellation rates that occur when reservations are made (Section 4.1.5 Part A). The simulations are run for block cancellations as well as random cancellations to ensure that all kinds of cancellations are accounted for.

4.3.3 Revenue Management

A. Booking Limit

ii. Basis of implementation

The algorithm used is based on the implementation of a paper whose application was for the hotel industry (Netessine et al., 2002) [37]. The technique is called “Expected Marginal Seat Revenue” (EMSR) analysis by Peter Belobaba at MIT (Belobaba, 1989) [6]. While the EMSR model has been developed for the airline industry, its application for the parking industry is straightforward. By correlating the two classes in Netessine’s paper to be corporate and leisure classes, and by fixing a fare distinct to each class (Rate Ratio (Appendix 1)), we ported over the concepts of booking limits to the parking industry. See Section 3.2.5 for more information about this.

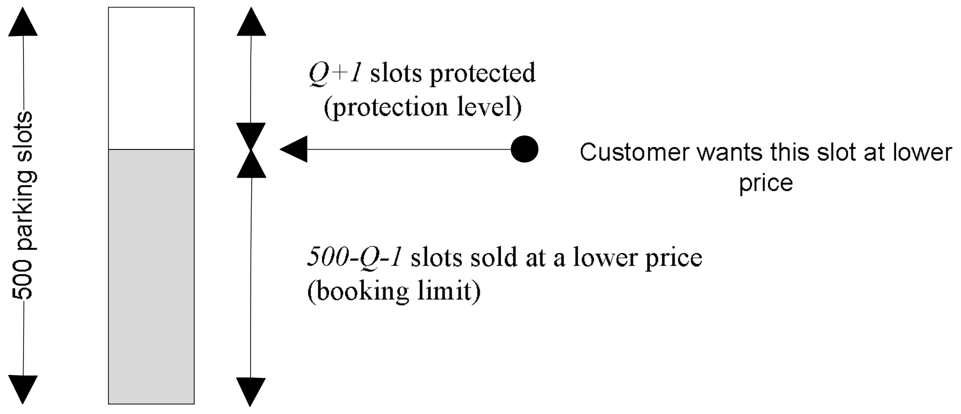

500 parking slots
Q+1 slots protected (protection level)
Customer wants this slot at lower price
500-Q-1 slots sold at a lower price (booking limit)

Figure 23: Booking limits in parking garage prototype

iii. <u>Algorithm Details</u>

The assumptions for this algorithm are stated in Section 4.1.5. Diagrammatic representation of booking limits has been provided in Figure 8 for better understanding. Based on the paper on booking limits (Netessine et al., 2002) [37], in our implementation, we have two fare classes (Corporate and Leisure) (introduced in Section 3.2.5) having prices R_h and R_l respectively (where $R_h > R_l$). As there are only two fare classes, the optimal booking limit for the leisure fare class is equal to C - Q* where 'C' is the total garage capacity and 'Q*' is the optimal protection level for the corporate class of cars. Let D be the random variable which represents the demand distribution at the higher fare. After the booking limits have been established, assume there is a high demand for the corporate class spots. Let Q be the protection level for the corporate class spots. If too many spots are protected, then we lose the revenue on Q -- D spots (assuming $Q \geq D$).

Let the penalty be R_l for the unsold parking spot. Let F(Q) is the cumulative probability i.e. Probability($D \leq Q$) where D is the random variable to describe the anticipated demand for the spot at the full price (corporate class cost). This distribution of D can either be estimated from the historical demand or forecasts based on the expected demand values. We have chosen the latter method, since there is no historical demand for the parking garage. With reference to the figure given, we can calculate the revenue change obtained by lowering the protection level from Q+1 to Q. If we do this then we will be leasing the (Q+1)th spot at a discount which guarantees revenue of R_h. Thus if we establish a protection level of Q+1 spots i.e. protect Q+1 spots for the corporate class, then the expected value of revenue is given by:

$$(1 - F(Q))\,(R_h) + F(Q)\,(\$0) = (1 - F(Q))\,(R_h)\ldots\ldots\ldots\ldots\ldots\ldots(1)$$

Hence, we should lower the protection level to *Q* as long as: $(1 - F(Q))\,(R_h) <= R_l$

After plugging in values for R_h and R_l into equation (1), we will get an equation that will give us the minimum value of the cumulative probability that is required. By corroborating it with the demand distribution, we can get the protection level (Q) that needs to be set for the higher priced spots. Booking limit is the difference between the capacity (C) and the protection level (Q) thus found. Consider Figure 24. If we choose a larger value of Q, then we would be protecting too many spots and thus leave too many spots unsold on average. The revenue from these unsold spots would be zero. If we keep Q at a smaller value, we are likely to sell too many spots at a lower price to leisure users and thereby turn away too many corporate clients. Since our aim is to maximize profits, it is not the best idea to do this. The ideal situation would be to protect Q spots and sell all of them to corporate customers in which case we get maximum revenue whereas practically we can allot as many of the ‘Q’ spots to the corporate class and give the rest to the leisure class in order to obtain maximum revenue.

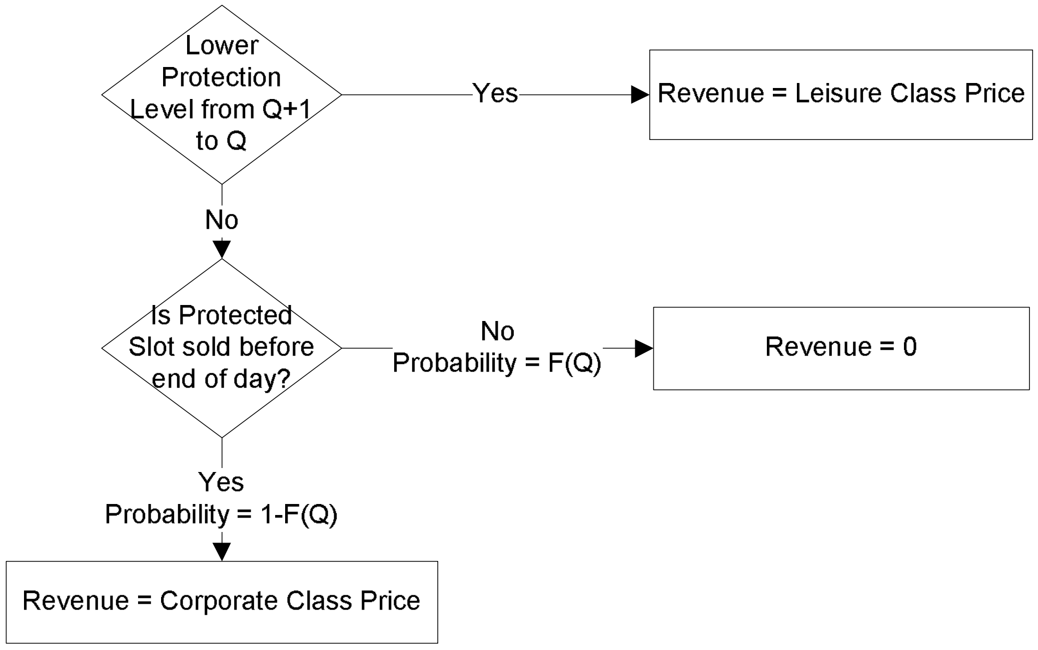

Lower Protection Level from Q+1 to Q
Yes
Revenue = Leisure Class Price
No
Is Protected Slot sold before end of day?
No
Probability = F(Q)
Revenue = 0
Yes
Probability = 1-F(Q)
Revenue = Corporate Class Price

Figure 24: Decision of selling protection level seat to leisure customer

iv. Simulation Details

The booking limit algorithm has been run for various distributions of customer arrivals (Leisure and corporate customers). It should be noted that the algorithm is independent of how many of the arrivals are corporate customers. The reason for this property is as follows: Since this is a static method and not a dynamically altering algorithm, a certain protection level is decided prior to any arrivals at all and that number is kept fixed. The protection level only depends on how many total arrivals occur. This is based directly from the algorithm proposed by Netessine (Netessine et al., 2002) [37] because the conditions in the parking and hotel industry for this case are identical.

1. Poisson distribution: This was chosen since most of the arrivals are modeled using Poisson distribution with a certain arrival rate of corporate customers. As we did not

have access to historical data, the simulations were run for varying values of corporate car arrival rate ranging from 5 cars/hour (light traffic) to 500 cars/hour (heavy traffic). The algorithm was run for various values of corporate customer arrival rate such as 5, 10, 25, 50, 100, 150, 250, 400, 500 cars /hour. It has been run against various values of ‘cost ratio’ i.e. ratio of cost of lower class (leisure) spots to cost of higher class (corporate) spots. These values are ranging from 0.166 to 0.75. Since we do not have access to absolute values of parking space pricing as this is not a widely implemented methodology, we have simulated the application of booking limits for a wide range of possible ratio values. In the airline industry the economy class to first class cost ratio is up to as low as 0.25 (Kayak.com) [58] (Malcolm, 2011) [59].

2. Binomial Distribution: The Binomial distribution may better fit the vehicle counts where the arrivals are rather uniformly spread over time (it may happen in heavy traffic) (Tarko, 2010) [38]. It has been run for various values of probability of corporate customer arrival ranging from 0.1 to 0.9 (since probability > 0 and probability ≤ 1) and for various values of ratio of cost of lower class (leisure) to cost of higher class (corporate) ranging from 0.166 to 0.75 based on trial and error. For the ratio of leisure to corporate class fares, since we do not have any historical data, we tried to model the prices similar to the airline economy class to first class rates (Malcolm, 2011) [59]. Comparison with the latter is not direct since the prices depend on flight route, type of flight etc. However, in order to capture the essence of the difference between the cost of the two classes, different flight routes were chosen

(Kayak.com) [58] and the range of 0.166 to 0.75 was decided as broadly encompassing most of the possible fare ratios that could occur with the two classes.

B. Overbooking – Algorithm OB1 (Probabilistic/Risk Model)

a. Basis of implementation

In order to evaluate overbooking, we had to ensure that the distributions that we used were applicable to the parking industry so that the concepts used for overbooking (Belobaba, 1987)[5] could be directly applicable to our line of work.

b. Algorithm details

Let the No-Show rate be NSR. The algorithm incorporates the uncertainty of the no-show rate for future flight (in our case parking reservation). Our aim is to find that value of overbooking (AU) such that we have a minimum number of denied parkings i.e. we reduce the customers turned away due to the parking garage being full. Let us assume that there are AU confirmed bookings. Let us further assume that there we have a Gaussian distribution of No-Show rates (Belobaba, 1987). We need to ensure that the number of customers who show up are less than or equal to the full capacity of the garage (CAP). Thus, we have to find an optimal show-up rate (SUR*), so that:

$$\text{AU} \times \text{SUR}^* = \text{CAP},$$

$$\text{Probability } [\text{AU} \times \text{SUR}^* > \text{CAP}] = 5\%$$

From the Gaussian distribution, we know SUR* will satisfy:

$$Z = 1.645 = (\text{SUR}^* - \text{SUR}) / (\text{STD})$$

Where Z= 1.645 from the standard normal curve for 95% confidence, SUR is the mean show-up rate and is the same as (1—NSR), SUR* is the optimal show-up rate, STD is the standard deviation of the show-up rate.

The optimal value of overbooking (AU) i.e., the value of overbooking which yields a minimum number of denied parkings, when we know the capacity of the garage (CAP), show-up rate of customers (SUR), standard deviation of No-Show rate knowing that denied parking should be 0 with 95% confidence(STD), is given by:

$$\mathbf{AU = CAP / (SUR + 1.645*STD) = CAP / (1-NSR + 1.645*STD)}$$

c. Simulation Details

The distribution of no-show rate for the customers is assumed to be Gaussian in nature. Hence, we need values of μ (mean) and σ (standard deviation). For purposes of simulation, the algorithm has been run for values of μ and σ ranging from 0.1 to 0.5 and 0.01 to 0.5 respectively. See Section 4.1.5 Part A.

Chapter 5: Results

5.1 Tracking

5.1.1 10% Tolerance Permitted (Algorithm T1/T2)

a. Details

If we consider that we can afford to not track the car for half of all the sensors that it actually crosses, then the inaccuracy of the algorithm is given by Figure 25. This implies that if the path traversed by a car has less than or equal to 10% of the sensors that have failed, then we declare the tracking to be accurate. If the car has more than 10% of the sensors that have failed, then the tracking is said to be *inaccurate*. A tracking is also said to be *inaccurate* when the car is declared to have parked in the wrong place when it actually parked in the right spot. The error occurs due to the failure of the tracking sensor due to which the algorithm is unable to differentiate between a right and wrong parking. This is a stringent case to test efficiency of the algorithm as compared to 50% tolerance and 75% tolerance cases.

b. Observations

In Figure 25, we can see that as the sensor failure rate increases, the inaccuracy of the algorithms also increases exponentially. It can also be seen that the inaccuracy of the Algorithm T2 is higher than inaccuracy of Algorithm T1. This is because algorithm T1 uses higher number of sensors. Since it uses more sensors, hence algorithm T1 loses a lesser percentage of sensors for the same sensor failure rate when compared to algorithm

T2 which uses fewer sensors. Hence, algorithm T1 is more accurate as compared to algorithm T2.

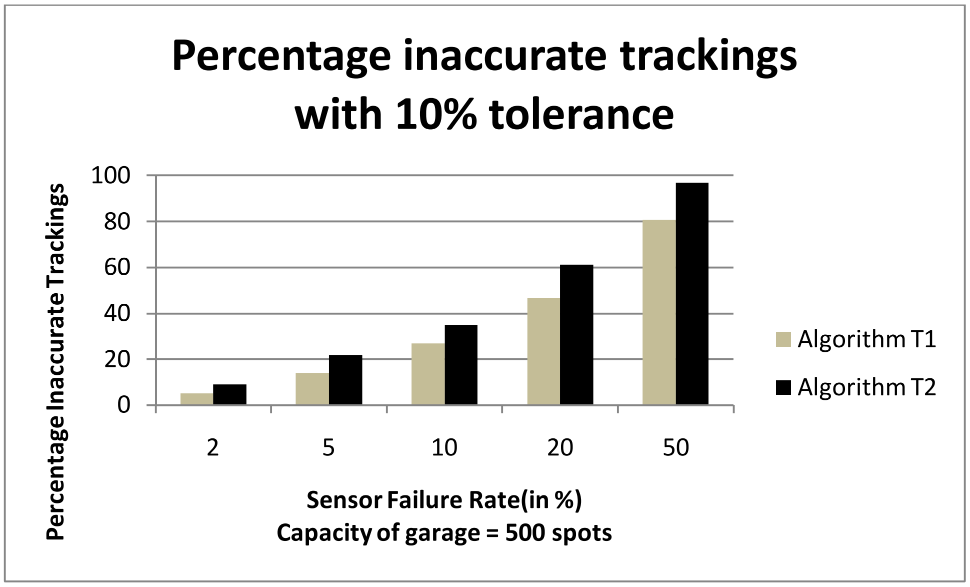


Figure 25: Inaccuracy percentage with 10% tolerance

c. Inference

From Figure 25, we can see that as the failure rate of sensors increases, the performance of the algorithms also degrades exponentially, since the sensor does not transmit any information about the car. We can see that the inaccuracy characteristics are exponential increasing in nature. However, despite the number of sensors being used for algorithm T1 are much more than the sensors being used for algorithm T2, the inaccuracy of the algorithms is not markedly different. The reason for this is that the inaccuracy is calculated as a percentage and not absolute value of the number of sensors that fail.

5.1.2 50% Tolerance Permitted (Algorithm T1/T2)

a. Details

If we consider that we can afford to not track the car for half of all the sensors that it actually crosses, then the inaccuracy of the algorithm is given by Figure 26. This implies that if the path traversed by a car has less than or equal to 50% of the sensors that have failed, then we declare the tracking to be accurate. If the car has more than 50% of the sensors that have failed, then the tracking is said to be *inaccurate*. A tracking is also said to be *inaccurate* when the car is declared to have parked in the wrong place when it actually parked in the right spot. The error occurs due to the failure of the tracking sensor due to which the algorithm is unable to differentiate between a right and wrong parking.

b. Observations

In Figure 26, we can see that as the sensor failure rate increases, the inaccuracy of the algorithms also increases exponentially. It can also be seen that the inaccuracy of the Algorithm T2 is higher than inaccuracy of Algorithm T1. This is because algorithm T1 uses higher number of sensors. Since it uses more sensors, hence algorithm T1 loses a lesser percentage of sensors for the same sensor failure rate when compared to algorithm T2 which uses fewer sensors. Hence, algorithm T1 is more accurate as compared to algorithm T2. Table 2 summarizes these key observations. To know what an inaccurate tracking is for 50% tolerance, see Section 5.1.2 Part a.

% of sensor failure	Algo. T1 (% of incoming cars tracked inaccurately)	Algo. T2 (% of incoming cars tracked inaccurately)
2%	0.8%	3.8%
10%	3.2%	11.8%
50%	59.8%	69%

Table 2: Key observations for tracking algorithms for 50% tolerance

c. Inference

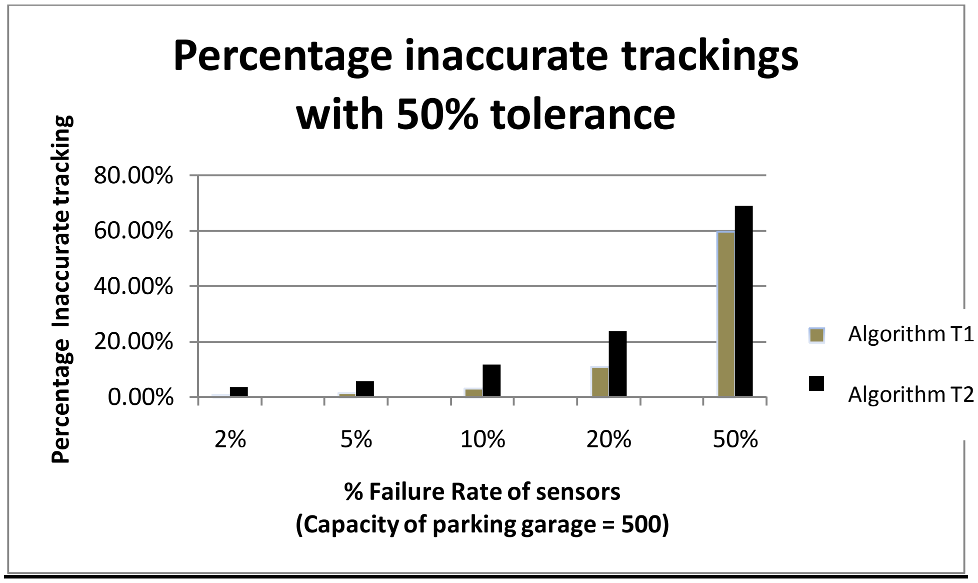


Figure 26: Inaccuracy percentage with 50% tolerance

From Figure 26, we can see that as the failure rate of sensors increases, the performance of the algorithms also degrades exponentially, since the sensor does not transmit any information about the car. We can see that the inaccuracy characteristics are exponential increasing in nature. However, despite the number of sensors being used for algorithm T1 are much more than the sensors being used for algorithm T2, the inaccuracy of the algorithms is not markedly different. The reason for this is that the inaccuracy is calculated as a percentage and not absolute value of the number of sensors that fail.

5.1.3 75% Tolerance Permitted (Algorithm T1/T2)

a. Details

If we consider that we can afford to not track the car for 75% of the total sensors that it actually crosses (i.e. 75% tolerance), then the inaccuracy of the algorithm is given by Fig. 27. This implies that if the path traversed by a car has less than or equal to 75% of the sensors that have failed then we declare the tracking to be accurate. If the car has more than 75% of the sensors that have failed, then the tracking is said to be *inaccurate*. A tracking is also said to be *inaccurate* when the car is declared to have parked in the wrong place when it actually parked in the right spot. The error occurs due to the failure of the tracking sensor due to which the algorithm is unable to differentiate between a right and wrong parking.

b. Observations

In Table 3, we can see that as the sensor failure rate increases, the inaccuracy of the algorithms also increases exponentially. To know what an inaccurate tracking for 75% tolerance means, see Section 5.1.3 Part a. It can also be seen that the inaccuracy of the Algorithm T2 is higher than inaccuracy of Algorithm T1. This is because algorithm T1 uses higher number of sensors. Since it uses more sensors, hence algorithm T1 loses a lesser percentage of sensors for the same sensor failure rate when compared to algorithm

T2 which uses fewer sensors. Hence, algorithm T1 is more accurate as compared to algorithm T2.

Percentage of sensor failure	**Algo. T1** (% of incoming cars tracked inaccurately)	**Algo. T2** (% of incoming cars tracked inaccurately)
2%	0.4%	1.2%
10%	3%	9.8%
50%	36%	51.2%

Table 3: Key observations for tracking algorithms for 75% tolerance

c. Inference

From Figure 27, we can see that as the failure rate of sensors increases, the performance of the algorithms degrades, since the sensor does not transmit any information about the car. We can see that the inaccuracy characteristics are almost exponential increasing in nature. However, despite the number of sensors being used for algorithm T1 are much more than the sensors being used for algorithm T2, the inaccuracy of the algorithms is not markedly different. When we consider inaccuracy of the algorithm, we are stating how many times the algorithm fails to tell us that the car parked in a spot where it actually did. The reason for this to happen is that a) a sensor fails just before the car parks and hence the algorithm cannot determine where the car actually parked or that b) a series of sensors failed that caused the total number of failed sensors to go above the tolerance level (75%) causing an inaccurate reading. If (a) occurs, then both algorithm T1 and T2 are equally likely to give an inaccurate reading since it is the matter of only a single sensor failing which can happen in either algorithm but for (b) to occur a lot more sensors need to fail for algorithm T1 to give high inaccurate values. We can see from Table 7 that for low values of sensor failure rate the accuracy of algorithm 1 is 3 to 4 times that of algorithm 2. However, as the sensor failure rate increases, this gap of accuracy difference narrows down between the two algorithms. The reason this occurs is that while condition (a) and (b) occur a lot more frequently, the increased occurrence of condition (a) causes comparatively high values of inaccuracy in algorithm T1 (36% for 50% sensor failure). Since algorithm T2 employs such few sensors, it affected in a more severe way by (b) as a result of which it has inaccuracy of 51.2% (for 50% sensor failure). As we are considering percentage inaccuracy, we see the smaller gap between the two algorithms. To justify why algorithm T1 is better than T2, if we were to compare the amount of information provided by the two algorithms for a worst case scenario, say of reaching spot 0 on floor 1 using algorithm T1(figure 12) and algorithm T2 (Figure 15), we can see that T1 will give 19 points of information/sensors (17(row sensors)+1(floor sensor)+1(occupancy sensor) = 19 sensors) before the car finally parks, where as T2 will give 3 points/sensor outputs. Hence, if we consider actual information provided by the two algorithms T1 gives over 6 times more information than algorithm T2.

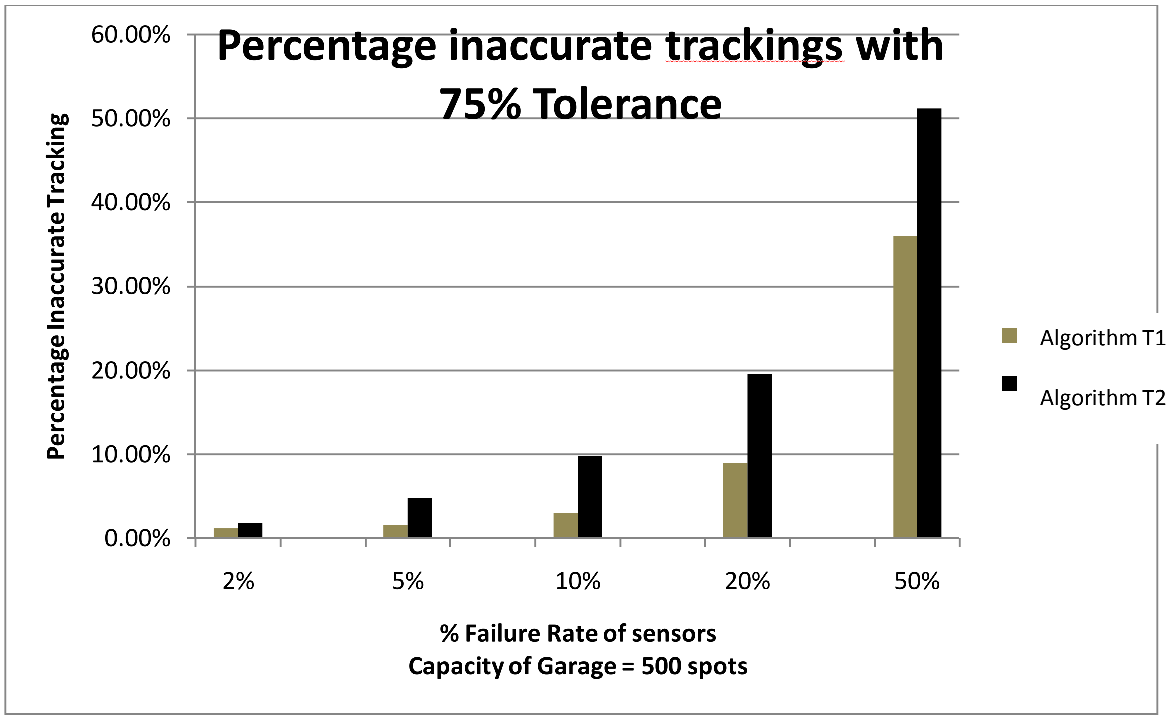


Figure 27: Inaccuracy percentage with 75% tolerance

5.1.4 Average Information Points provided by Algorithm T1 and T2

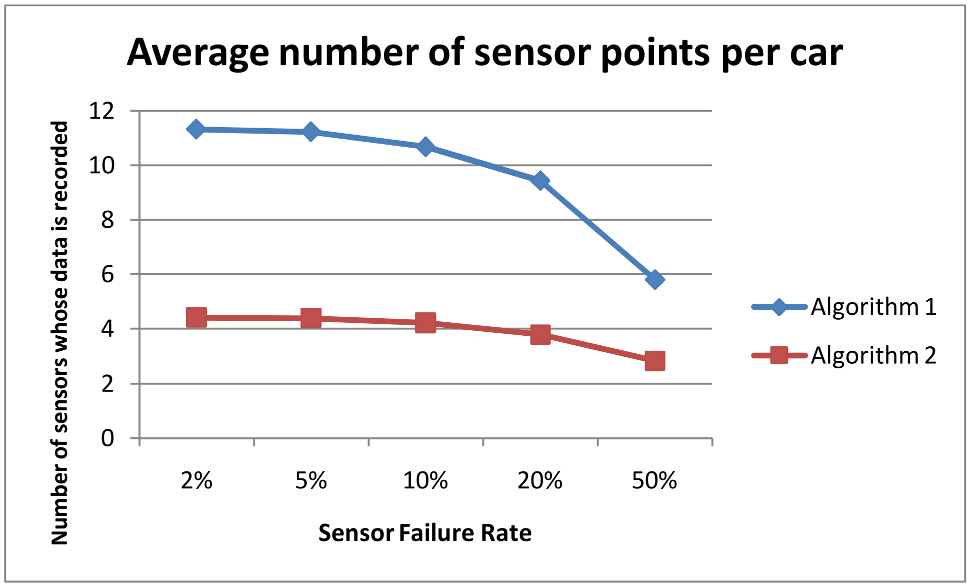


Figure 28: Average number of sensor points provided by T1 and T2

While the tolerance metric tells us which tracking algorithm provides more accurate readings as to position of the car, Figure 28 tells us how much information each tracking algorithm provides us. To obtain this information, the number of sensor readings from 500 cars (capacity of the garage) was totaled and the average number of sensor points per car was plotted for both algorithms T1 and T2. The readings obtained are as shown in Figure 28. From the figure 28, we can clearly see that Algorithm T1 provides more information than Algorithm T2. From Table 4, we can see that on an average, Algorithm T1 provides about 2.5 times more information about each car than Algorithm T2.

Percentage of Sensor Failure	**Avg. number of sensor readings for Algorithm T1**	**Avg. number of sensor readings for Algorithm T2**	**Ratio of number of readings given by T1 to readings given by T2**
2%	11.314	4.416	2.56
5%	11.22	4.39	2.55
10%	10.67	4.222	2.52
20%	9.432	3.796	2.48
50%	5.816	2.826	2.05

Table 4: Average number of sensor readings per car

5.2 Reservation Defragmentation

5.2.1 Input Datasets Considered

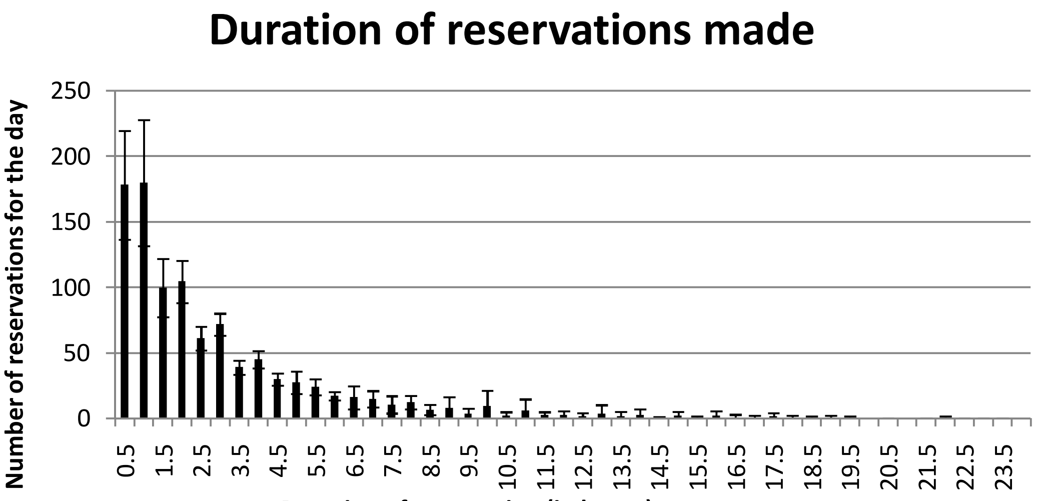


Figure 29: Histogram of average duration of reservations chosen as sample data. Error bars are standard deviation

From the Figure 29, we can see that the average duration of reservations varies from 0.5 hours to 22 hours which is representative of most of the possible reservations made in a 24 hour period. Since we have taken into account such a large range of reservation durations, we believe it is indicative of real-world parking reservations since we have covered possible reservation durations for a 24-hour period.This input dataset is used by algorithms R1 (Section 4.3.2 Part C), R2 (Section 4.3.2 Part D) and R3 (Section 4.3.2 Part E). These algorithms use reservations of different durations and re-arrange them in order to carry out reservation defragmentation. We have chosen to perform defragmentation simulation for 15% cancellation rate of reservations (Section 4.1.5 Part A (c) and Appendix A2.6). It should be noted that the gains observed by use of algorithms is dependent on the distribution of reservations that are made. We have chosen an exponential distribution of reservations but with a different distribution we could see other values of gains observed.

5.2.2 Decrease in Fragmented Free Time Slots with Block Cancellation (Next Day Reservations)

a. Details

We have chosen to perform analysis of the algorithms with block cancellation to take into consideration bulk cancellations by tour groups/corporate. In this set of results, we will observe the effect our algorithm has on reducing the total number of free time slots in between reservations in all the parking spots after block cancellation has been done. We term these free time slots in between reservations as ***fragmented free time slots*** since they are fragmented and not contiguous in nature. The aim is to pack the reservations as close to each other as possible in order to reduce the free time slots in between reservations. This is done by re-arranging the reservations in the parking spots (Section 4.3.2). The algorithm has been run for 15% cancellations. The algorithms take incoming reservations and the cancellation percentage as inputs. The graph obtained in Figure 30 has been plotted for average values of percentage decrease in fragmented free time slots obtained after the algorithms were run over 100 datasets. The error bars (Figure 30) indicate the standard deviation of these values over 100 dataset runs.

The reason for choosing this metric is to make sure there is an decrease in free time slots in between reservations after defragmentation which will be an indication the reservations are now packed more tightly (i.e. better defragmentation) so that more reservations (of any duration) can be accommodated in the parking garage.

b. Observations

See Table 5 for some of the key observations obtained from simulations performed.

		Percentage decrease in fragmented time slots		
Number of Reservations made	**% Cancellation**	**Method R1**	**Method R2**	**Method R3**
200	15%	15.72%	26.74%	24.62%
1000	15%	22.92%	46.31%	36.48%
1750	15%	22.81%	46.81%	38.21%

Table 5: Percentage decrease in fragmented time slots using block cancellation

The percentage values tell us the reduction in free time slots in between reservations that is obtained when the original configuration of reservations (after first fit algorithm followed by cancellation) is defragmented to obtain the new arrangement of reservations (after applying algorithms R1, R2 and R3). Table 5 discusses the percentage reduction in free time slots provided by algorithms R1, R2 and R3 when 200, 1000 and 1750 reservations are made and 15% of these reservations are cancelled. (Section 4.1.5 Part A). For example from Table 5, for 1000 reservations, algorithm R2 reduces free time slots between reservations by 46.31% if 15% of reservations were cancelled. (Section 4.1.5 Part A).

c. Inference

We can see that although that 15% of reservations have been cancelled (Figure 30) the three algorithms result in higher percentage values of free space creation. This is because when we consider the percentage decrease in free time slots, we are considering the combined effect of i) cancellation (i.e. deletion from the reservation bitmap) of reservations of certain durations which get freed when cancellation of 15% of reservations takes place and ii) the effect of defragmentation of the remaining reservations. It can be seen from the Figure 30, algorithm R2 results in most defragmentation followed by algorithm R3 and then algorithm R1. Standard deviation values seem comparatively lower for algorithm R2 implying more reliability than the other two algorithms. The reduction in standard deviation values for algorithm 2 is statistically significant since the reduction is substantial where the deviation of algorithm 3 is about 1.5 times that of algorithm 2 for 1000 reservations (Figure 30). As we increase the number of reservations, for algorithm R1 and R2, we see a steady decrease in fragmented free time slots (more reservation defragmentation). From Table 5, we can see that for algorithm R1 for 1000 reservations at 15% cancellation, the decrease in fragmented time slots is 22.92%. The reason we observer greater decrease when we increase the number of reservations is that there is more possibility of moving a certain reservations in between the free time slots that are in between existing reservations when there are greater number of reservations. From Table 5 we can see that there is a statistically significant decrease obtained in free time slots when the reservations are increased from 200 (26.74% for R2) to 1000 (46.31% for R2) for all three algorithms. However, the decrease observed in 1000 reservations is not statistically significant from 1750 reservations. It should be noted that while the decrease observed for 1750 reservations is high (46.81% for R2), the reason it is not much more different from 1000 is because there even with 15% cancellation, there are a lot of reservations which are not contiguous. Hence, there is a certain extent to which so many reservations can be compacted to which is seen by the near similar values of percentage decrease in 1000 and 1750 reservations.

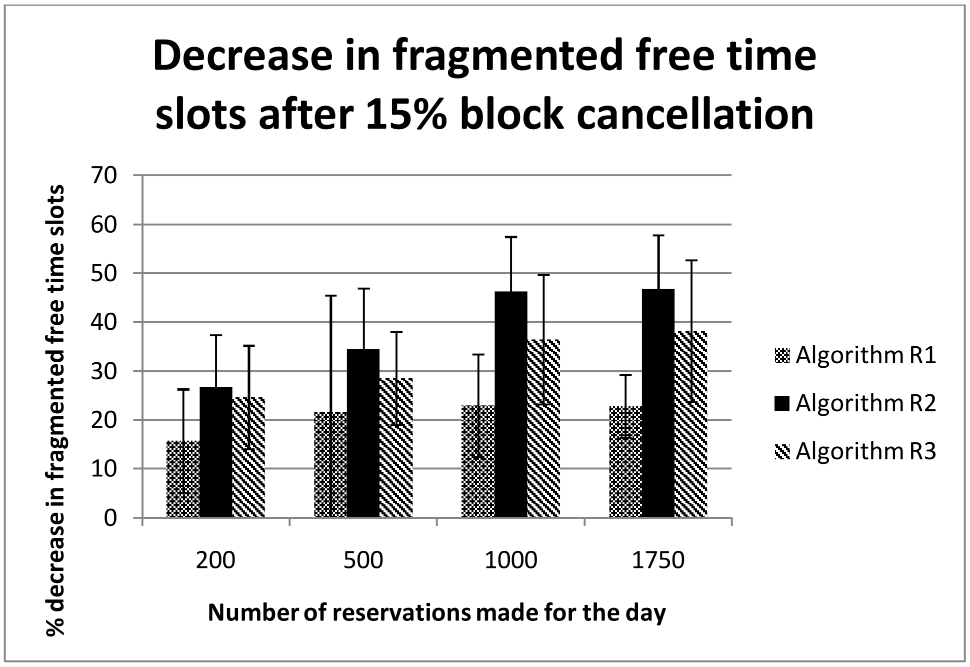


Figure 30: Percentage decrease in fragmented free time slots available Error bars are standard deviation. For parameters refer to Table 9

5.2.3 Decrease in Fragmented Free Time Slots with Random Cancellations (Next Day Reservations)

a. Details

We have chosen to perform analysis of the algorithms with random cancellation to take into account arbitrary cancellations by customers who made reservations. In this set of results, we will observe the effect our algorithm has on reducing the total number of free time slots in between reservations in all the parking spots after random cancellation has been done. We term these free time slots in between reservations as *fragmented free time slots* since they are fragmented and not contiguous in nature. The aim is to pack the reservations as close to each other as possible in order to reduce the free time slots in between reservations. This is done by re-arranging the reservations in the parking spots (Section 4.3.2). The algorithm has been run for 15% cancellations. The algorithms take incoming reservations and the cancellation percentage as inputs. The graph obtained in Figure 31 has been plotted for average values of percentage decrease in fragmented free time slots obtained after the algorithms were run over 100 datasets. The error bars (Figure 31) indicate the standard deviation of these values over 100 dataset runs.

The reason for choosing this metric is to make sure there is an decrease in free time slots in between reservations after defragmentation which will be an indication the reservations are now packed more tightly (i.e. better defragmentation) so that more reservations (of any duration) can be accommodated in the parking garage.

b. Observations

See Table 6 for some of the key observations obtained from simulations performed.

Number of Reservations made	% Cancellation	Method R1	Method R2	Method R3
200	15%	17.64%	31.82%	20.47%
1000	15%	22.19%	42.74%	30.3%
1750	15%	23.12%	46.23%	34.65%

Table 6: Percentage decrease in fragmented time slots using random cancellation

The percentage values tell us the reduction in free time slots in between reservations that is obtained when the original configuration of reservations (after first fit algorithm followed by random cancellation) is defragmented to obtain the new arrangement of reservations (after applying algorithms R1, R2 and R3). Table 6 discusses the percentage reduction in free time slots provided by algorithms R1, R2 and R3 when 200 or 1000 reservations are made and 15% of these reservations are cancelled. (Section 4.1.5 Part A). For example, for 1000 reservations, algorithm R2 reduces free time slots between reservations by 42.74% if 15% of reservations were cancelled. (Section 4.1.5 Part A).

d. Inference

We can see that although that 15% of reservations have been cancelled (Figure 31), the three algorithms result in much higher percentage values of free space creation. This is because when we consider the percentage decrease in free time slots, we are considering the combined effect of i) cancellation (i.e. deletion from the reservation bitmap) of lengths of the durations that are getting freed when cancellation of 15% of reservations takes place and ii) the effect of defragmentation of the remaining reservations. It can be seen from the Figure 31, algorithm R2 results in most defragmentation followed by algorithm R3 and then algorithm R1. Standard deviation values seem comparatively lower for algorithm R2 implying more reliability than the other two algorithms. The reduction in standard deviation values for algorithm R2 is statistically significant since the reduction is substantial where the deviation of algorithm R3 is about six times that of algorithm R2 for 1000 reservations (Figure 31). As we increase the number of reservations, for algorithm R1 and R2, we see a steady decrease in fragmented free time slots (more reservation defragmentation). From Table 6, we can see that for algorithm R1 for 1000 reservations at 15% cancellation, the decrease in fragmented time slots is 22.19%. The reason we observer greater decrease when we increase the number of reservations is that there is more possibility of moving a certain reservations in between the free time slots that are in between existing reservations when there are greater number of reservations. From Table 6 we can see that there is a statistically significant decrease obtained in free time slots when the reservations are increased from 200 (31.82% for R2) to 1000 (42.74% for R2) for all three algorithms. However, the decrease observed in 1000 reservations is not statistically significant from 1750 reservations. It should be noted that while the decrease observed for 1750 reservations is high (46.23% for R2), the reason it is not much more different from 1000 is because there even with 15% cancellation, there are a lot of reservations which are not contiguous. Hence, there is a certain extent to which so many reservations can be compacted to which is seen by the near similar values of percentage decrease in 1000 and 1750 reservations.

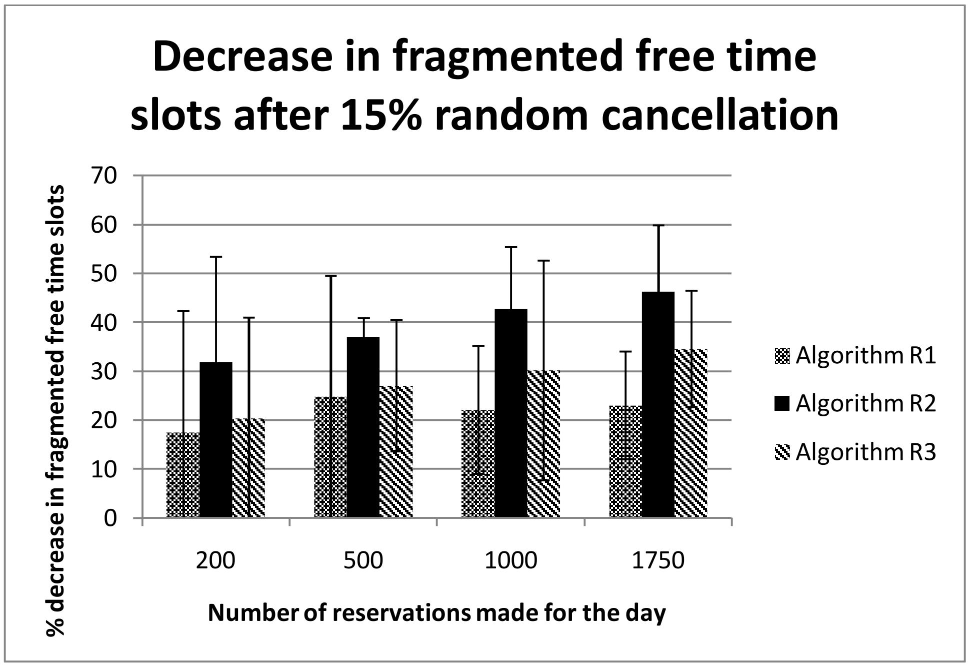


Figure 31: Decrease in fragmented free time slots. Error bars are standard deviation. For parameters refer to Table 9.

5.2.4 Decrease in Occupied Parking Spots with Block Cancellations (Next Day Reservations)

a. Details

We have chosen to perform analysis of the algorithms with block cancellation to take into account bulk cancellations corporate customers. In this set of results, we will observe the effect our algorithm has on increasing the number of completely free parking spots after block cancellation has been done. The aim is to pack the reservations as close to each other as possible in order to make the existing reservations occupy as few parking spots as possible. This is done by re-arranging the reservations in the parking spots (Section 4.3.2). The number of completely free parking spots (i.e. parking spot having no reservation) after the parking spot of the last reservation are counted. This number is compared before and after defragmentation and the percentage decrease of occupied parking spots after defragmentation is reported as percentage decrease in Figure 32. The algorithm has been run for 15% cancellations. The algorithms take incoming reservations and the cancellation percentage as inputs. The graph obtained in Figure 32 has been plotted for average values of percentage decrease in occupied parking spots obtained after the algorithms were run over 100 datasets. The error bars (Figure 32) indicate the standard deviation of these values over 100 dataset runs. The reason for choosing this metric is to make sure there is an decrease in number of occupied parking spots after defragmentation, so that more reservations (of any duration) can be accommodated in the parking garage easily by allotting an empty parking spot to that reservation.

b. Observations

See Table 7 for some of the key observations obtained from simulations performed.

		Percentage decrease in occupied parking spots		
Number of Reservations made	**% Cancellation**	**Method R1**	**Method R2**	**Method R3**
200	15%	8.14%	10.65%	12.99%
1000	15%	7.18%	14.62%	15.13%
1750	15%	4.23%	13.64%	14.91%

Table 7: Percentage decrease in occupied parking spots using block cancellation

The percentage values in Table 7 tell us the decrease in occupied parking spots obtained when the original configuration of reservations (after first fit algorithm followed by random cancellation) is defragmented to obtain the new arrangement of reservations (after applying algorithms R1, R2 and R3). Table 7 discusses the percentage decrease in occupied parking spots provided by algorithms R1, R2 and R3 when 200, 1000 and 1750 reservations are made and 15% of these reservations are cancelled. (Section 4.1.5 Part A). For example, for 1000 reservations, algorithm R2 increases free parking spots by 14.62% if 15% of reservations were cancelled (Section 4.1.5 Part A).

c. Inference

It can be seen from the Figure 32, algorithm R3 results in most defragmentation followed by algorithm R2 and then algorithm R1 although the decrease is not statistically different between algorithm R2 and R3. Standard deviation values are comparatively lower for algorithm R2 implying more reliability than the other two algorithms. The reduction in standard deviation values for algorithm R2 (Figure 32) is statistically significant since the reduction is substantial where the deviation of algorithm R3 is about two and a half times that of algorithm R2 for 1000 reservations (Figure 32). As we increase the number of reservations, we see a gradual decrease in free parking spots. The reason for this is that as the number of reservations increase, there is an increase in the number of parking spots occupied by the vehicles as well. Defragmentation can ensure tight packing of reservations but the maximum capacity of the garage is fixed at 500 vehicles. As the number of reservations increases to 1750, about 95% of total possible reservations (Section 5.2.7), even defragmentation will not be able to increase the number of free parking spots by a lot, which is why the percentage decrease in occupied parking spots decreases with increase in number of reservations made. From Table 7, we can see that for algorithm R2 for 1000 reservations at 15% cancellation, the decrease in occupied parking spots is 14.62.

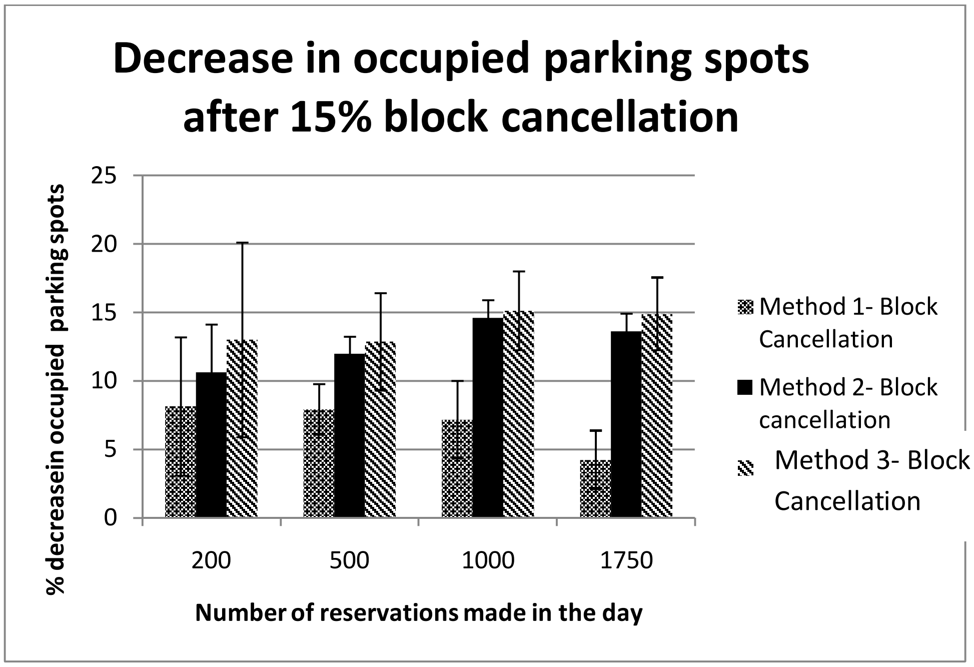


Figure 32: Percentage increase in parking spots available. Error bars are standard deviation. For parameters refer to Table 9.

5.2.5 Decrease in Occupied Parking Spots with Random Cancellations (Next Day Reservations)

a. Details

We have chosen to perform analysis of the algorithms with random cancellation to take into account arbitrary cancellations by customers who made reservations. In this set of results, we will observe the effect our algorithm has on increasing the number of completely free parking spots after random cancellation has been done. The aim is to pack the reservations as close to each other as possible in order to make the existing reservations occupy as few parking spots as possible. This is done by re-arranging the reservations in the parking spots (Section 4.3.2). The number of completely free parking spots (i.e. parking spot having no reservation) after the parking spot of the last reservation are counted. This number is compared before and after defragmentation and the percentage decrease of occupied parking spots after defragmentation is reported as percentage decrease in Figure 33. The algorithm has been run for 15% cancellations. The algorithms take incoming reservations and the cancellation percentage as inputs. The graphs obtained in Figure 33 has been plotted for average values of percentage decrease in occupied parking spots obtained after the algorithms were run over 100 datasets. The error bars (Figure 33) indicate the standard deviation of these values over 100 dataset runs.

The reason for choosing this metric is to make sure there is an decrease in number of occupied parking spots after defragmentation, so that more reservations (of any duration) can be accommodated in the parking garage easily by allotting an empty parking spot to that reservation.

b. Observations

See Table 8 for some of the key observations obtained from simulations performed. The percentage values in Table 8 tell us the decrease in occupied spots obtained when the original configuration of reservations (after first fit algorithm followed by random cancellation) is defragmented to obtain the new arrangement of reservations (after applying algorithms R1, R2 and R3). Table 8 discusses the percentage decrease in occupied parking spots provided by algorithms R1, R2 and R3 when 200 or 1000 reservations are made and 15% of these reservations are cancelled. (Section 4.1.5 Part A). For example, for 1000 reservations, algorithm R2 decreases occupied parking spots by 13.44% if 15% of reservations were cancelled (Section 4.1.5 Part A).

		Percentage decrease in occupied parking spots		
Number of Reservations made	**% Cancellation**	**Method R1**	**Method R2**	**Method R3**
200	15%	7.48%	13.47%	12.02%
1000	15%	6.14%	13.44%	14.71%
1750	15%	4.41%	12.29%	14.58%

Table 8: Percentage decrease in occupied parking spots using random cancellation

c. Inference

It can be seen from the Figure 33, algorithm R3 results in higher defragmentation followed by algorithm R2 and then algorithm R1 although the decrease is not statistically different between algorithms R2 and R3. For example, for 1000 reservations, algorithm R3 decrease occupied parking spots by 14.71% while algorithm R2 decreases it by 13.44% for 15% cancellation. Standard deviation values are comparatively lower for algorithm R2 implying more reliability than the other two algorithms. The reduction in standard deviation values for algorithm R2 is statistically significant since the reduction is substantial where the deviation of algorithm R3 is about three times that of algorithm R2 for 1000 reservations (Figure 33). As we increase the number of reservations, we see a gradual decrease in free parking spots (Table 8). The reason for this is that as the number of reservations increase, there is an increase in the number of parking spots occupied by the vehicles as well. Defragmentation can ensure tight packing of reservations but the maximum capacity of the garage is fixed at 500 vehicles. As the number of reservations increases to 1750, about 95% of total possible reservations

(Section 5.2.7), even defragmentation will not be able to increase the number of free parking spots by a lot, which is why the percentage decrease in occupied parking spots decreases with increase in number of reservations made. From Table 8, we can see that for algorithm R2 for 1000 reservations at 15% cancellation, the decrease in occupied parking spots is 13.44%.

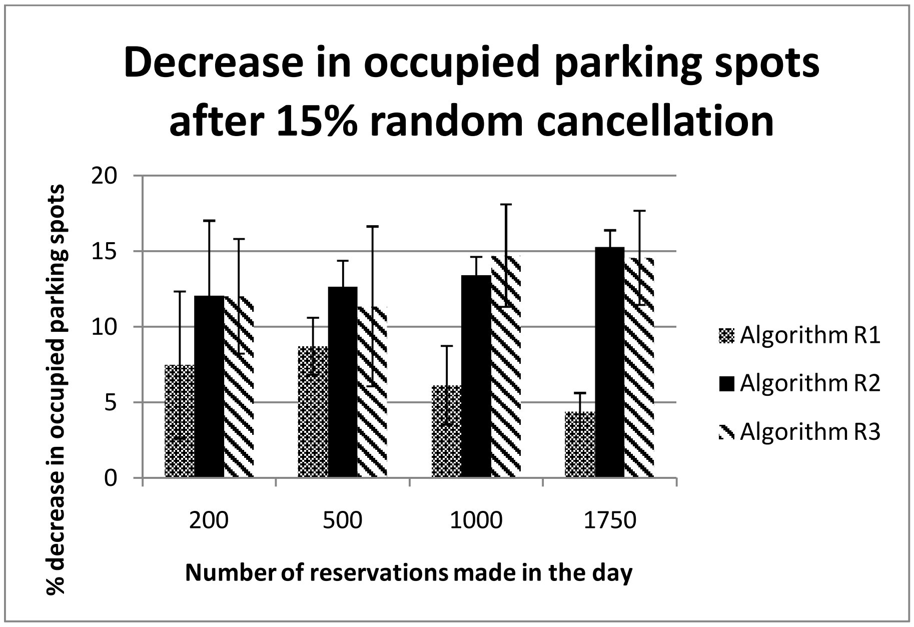


Figure 33: Percentage decrease in occupied parking spots available given. For parameters refer to Table 9.

5.2.6 Decrease in Mean length of Contiguous Free Time Slots in between Reservations with 15% Cancellations (Next day Reservations)

a. Observation and Inferences

Figures 34 and 35 give an indication of the average number of contiguous free time slots (*Mean length of contiguous free time slots*) that exist in between reservations per parking spot, before and after the reservation defragmentation algorithm(R1, R2 and R3) has been applied. ***Mean length of contiguous free time slots*** is the average length of contiguous free time slots that occur in a particular parking spot. The *Mean length of contiguous free* *time slots* per parking spot is calculated by counting the number of times we have any (more than zero) free time slots ('0' in the reservation bitmap), say 'n', and adding up the lengths of the contiguous '0''s that occur, say 'l'.

'*Mean length of contiguous free time slots*' for parking spot 'x' = $\frac{\sum_{i=1}^{i=n} l_i}{n}$......(I)

Where l_i = length of the 'i'th contiguous '0' chain in spot x ,

n = number of sets of contiguous '0''s found in parking spot 'x'

As can be seen from the figure 34 and 35, the *mean length of contiguous free time slots* is comparatively lower in post-defragmentation (Figure 35) as opposed to the pre-defragmentation (Figure 34) as indicated by lower y-axis values. Figure 34 shows many fluctuations with high mean length values indicating a lack of defragmentation of reservations prior to application of any defragmentation algorithm. Figure 35 has lower and more consistent values of mean length showing that there is a greater amount of packing that occurs due to the defragmentation algorithms because there are fewer spaces post-defragmentation. We can also see that the *mean length of contiguous free time slots* is lower at lower spot numbers. This shows that the defragmentation algorithms have a tendency to pack the reservations to the initial spots. From the Figure 35, we can see that Algorithm R2 has the lowest values of mean length indicating highest defragmentation provided as opposed to algorithm R1 and R3. Figure 35 also tells us from what parking spot numbers we get completely free spots (i.e. no reservations at all), from the straight line obtained at *Mean length of contiguous free time slots* = 48. The earlier this happens the more efficient the algorithm. In order to see the percentage of parking spots that are freed by the algorithm R1, R2 and R3, see sections 5.4.4 where the cancellation is 15%. The initial portion of the line graph (Figure 35) should be low and steady to show that the *mean length of contiguous free time slots* is constantly low.

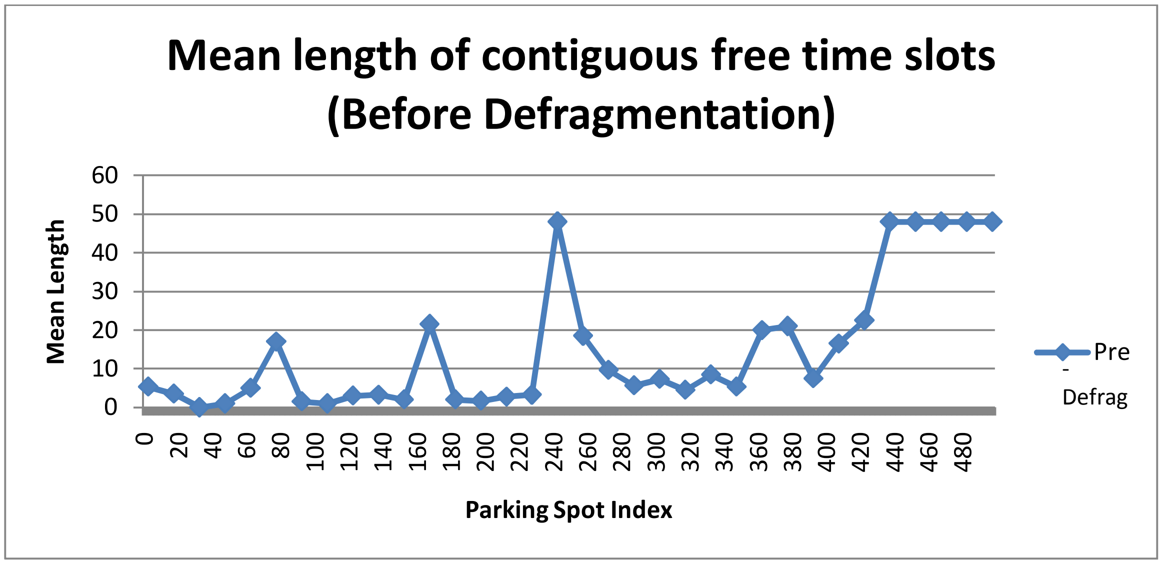


Figure 34: Mean length of contiguous free time slots. For parameters refer to Table 9.

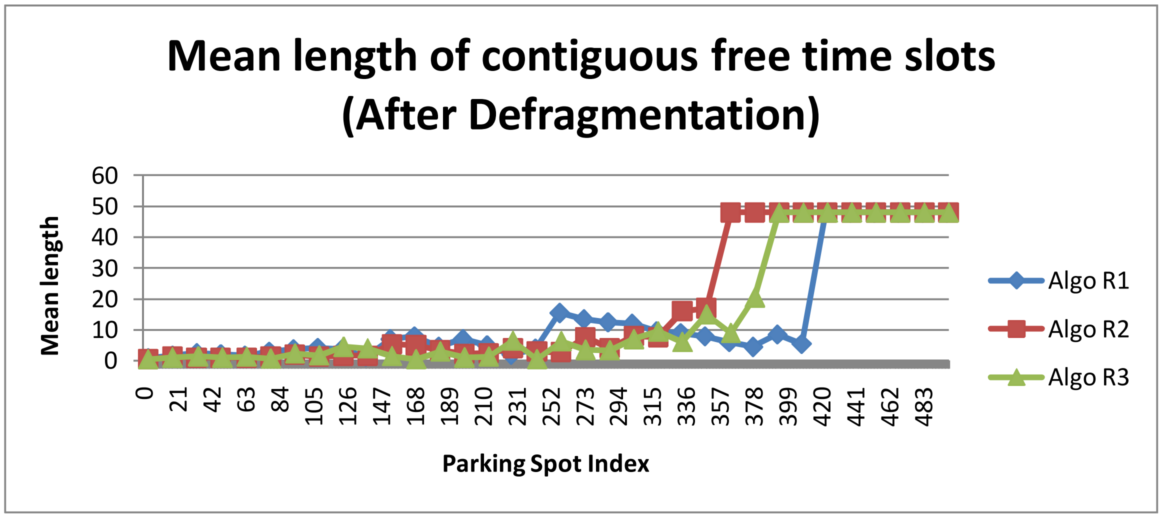


Figure 35: Mean length of contiguous free time slots for Method R1, R2. R3 For parameters refer to Table 9.

5.2.7 Increase in Maximum Occupancy of Parking Garage Considering 15% Random Cancellations (Current Day Reservations)

The reason we chose increase in maximum occupancy of the parking garage as a metric has been explained in detail in Section 4.3.2. When we ran the first fit algorithm for allotment of reservations when they were made, we determined the average maximum occupancy of the parking garage to be 1820 vehicles. This was determined after simulation over 100 dataset runs. Hence, when we are considering real-time, our aim is to ensure that by the end of the day, we are able to accommodate more than 1820 vehicles to prove the efficiency of the algorithms. After running the algorithms over 100 dataset runs, we determined the increase in occupancy caused by algorithms R1, R2 and R3 to be as shown in Figure 36.

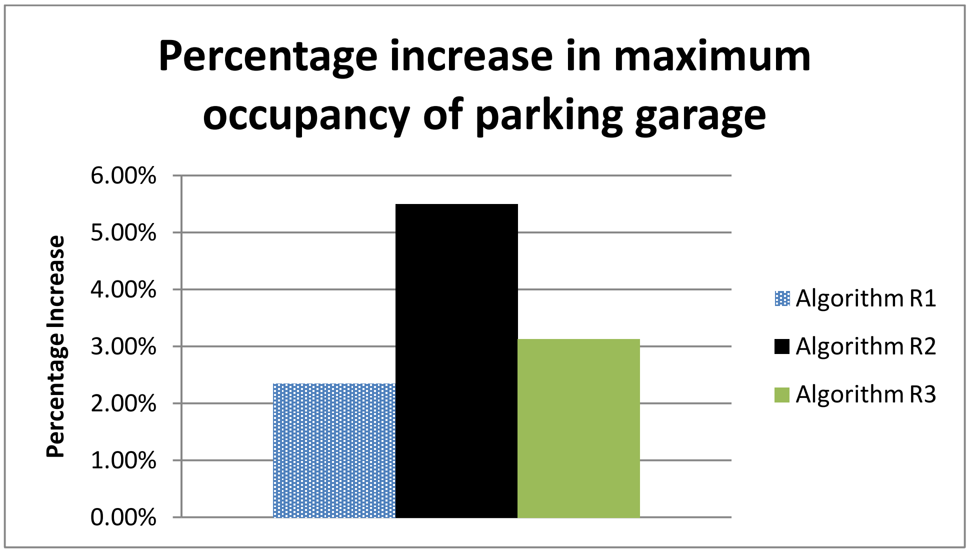


Figure 36: Percentage Increase in Maximum Occupancy of parking garage using Defragmentation Algorithms. For parameters refer to Table 9.

From Figure 36, we can see that algorithm R1 causes an increase of 2.34% and algorithm R2 causes an increase of 5.5% in maximum occupancy of the parking garage. Algorithm

R3 causes and increase of 3.1% in maximum capacity of garage but due to the much larger execution times of Algorithm R3 as opposed to R1 and R2, (Section 6.2 Part iii), Algorithm R3 is practically infeasible for such an application since everything is needed dynamically due to operation in real-time. It should be noted that defragmentation was carried out after every 50 reservations. From Figure 36, we can see that the increase observed is not as significant as the values observed in next-day reservation defragmentation (Section 5.2.2 – Section 5.2.5). There are two reasons for this:

a. In current time reservations, the reservations which are currently in the parking garage cannot be moved due to practical reasons. Hence, this reduces amount of defragmentation possible since there will be reservations will not be moved due to this immovable reservations and thereby will increase the inter-reservation free time slots.
b. Since we have taken current time into effect and the total time of consideration is fixed (24 hours) (Section 4.3.2), hence the amount of space available in the reservation bitmap for defragmentation also keeps reducing. The reason for this is that each row in the reservation bitmap corresponds to 30 minutes. As time progresses, we go lower in the reservation bitmap one row at a time. The rows above the current time cannot be used for moving reservations, since those rows indicate time that has already passed. Hence, the space available for defragmentation keeps reducing.

Due to the above two reasons, we do not observe very high values in increase in maximum occupancy but since we do observe a positive increase in maximum occupancy, this is significant.

Simulation case	Garage Capacity	Number of reservations	Percentage Cancellation	Cancellation type	Percentage Occupancy
Figure 30	500	As per X-axis	15%	Block	-
Figure 31	500	As per X-axis	15%	Random	-
Figure 32	500	As per X-axis	15%	Block	-
Figure 33	500	As per X-axis	15%	Random	-
Figure 34	500	1750	-	-	95%
Figure 35	500	1750	15%	Random	-
Figure 36	500	1820	-	-	-

Table 9: Parameters Table

5.3 Booking Limits

5.3.1 Poisson Distribution of Arrival

a. Details

The booking limit problem was simulated with Poisson distribution of arrival which is the favored mathematical distribution to model arrival of objects. The parameters that are to be given as input to the algorithm include the arrival rate of the customers (corporate customers), *cost ratio* (ratio of leisure class rate to corporate class rate) and capacity of the garage.

b. Observations

The legend in the graph is the cost ratio (Appendix 1). Our metric is to determine how many spots we should set aside for corporate class from the 500 spots that are available in the parking lot known as *protection level*. Table 10 is derived from Figure 37. In Table

10, we can see that an increase in arrival rate causes an increase in required protection level.

Corporate Arrival Rate(Cars/hour)	**Cost Ratio** **(See Appendix 1)**	**Protection Level (As % of garage capacity)**
50	0.5	10%
250	0.5	50%
500	0.5	100%

Table 10: Key Observations for booking limits with Poisson distribution of arrival

c. Inference

Consider Figure 37. With increasing corporate car arrival rate, there is increase in protection level and with increase in cost ratio we observe the same phenomenon. The increase in protection level follows almost an exponential curve. It implies that the higher the arrival rate, the greater should be the allocation of spots to corporate customers to increase revenue. If the protection level exceeded 100%, we have ignored this condition since it does not make any practical sense.

One very important observation is the fact that the increase in protection level is proportionately higher when there is an increase in corporate arrival rate as compared to increase in rate ratio (R_l/R_h). Also, we see that there are more spots allocated to the corporate class when there is a higher rate ratio (for a particular arrival rate). The reason for this can be explained as follows: If there is a smaller difference between the corporate class and leisure class, then an incoming customer will find it sensible to take the corporate class spot since he will get all the benefits for almost the same price as leisure class. Hence, the garage will see an increase in revenue due to higher class spot occupancy. On the other hand, if the difference between leisure class and corporate class is more, an incoming customer will think about whether or not the corporate class spot should be taken or not. In this case, it is more prudent to allocate comparatively lesser number of spots to the corporate class to make sure that there is no loss of revenue in over-estimating number of corporate spots that will be occupied.

For obtaining profits gained from imposing protection levels, see Appendix 2 Part A2.4.

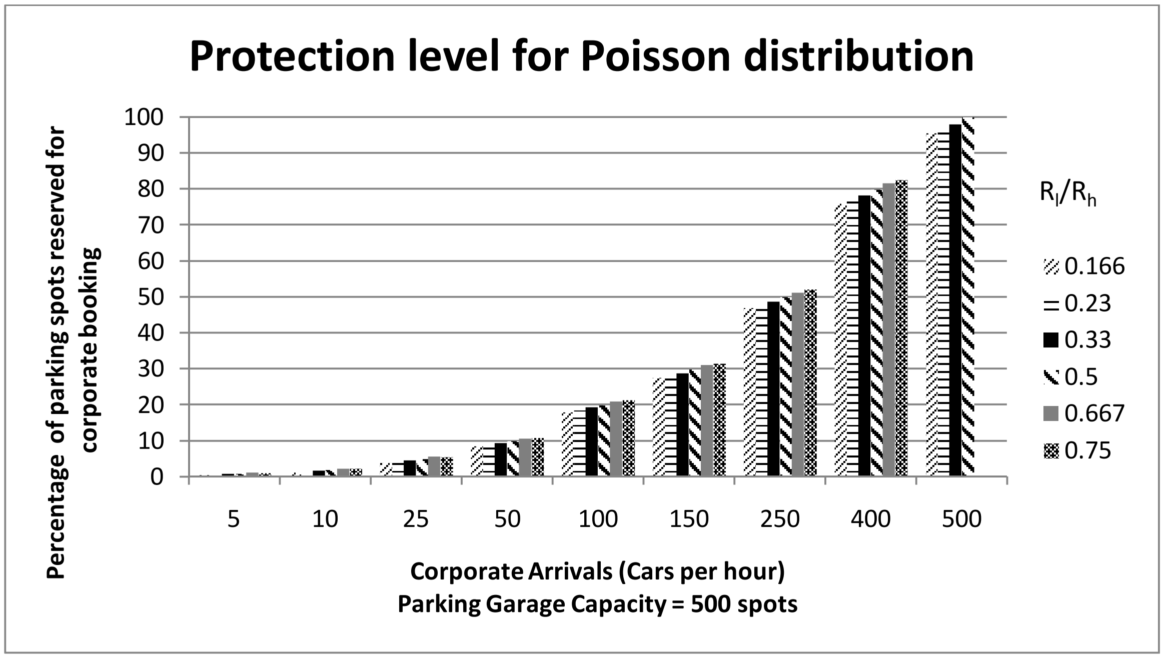


Figure 37: Protection Level given Poisson distribution of arrival

5.3.2 Booking Limits for Binomial Distribution

a. Observations

This model suggests relatively higher values of protection level as opposed to other probability distributions. The parameters that are to be given as input to the algorithm include the arrival rate of the customers (all customers), *cost ratio* (ratio of leisure class rate to corporate class rate) and the probability of arrival of a customer (show-rate).

Probability of corporate customer arriving	**Cost Ratio** **(Appendix 1)**	**Protection Level (As % of garage capacity)**
0.1	0.5	10%
0.5	0.5	50%
0.9	0.5	90%

Table 11: Key Observations for booking limits with Binomial Distribution of arrival

b. Inference

The reason for choosing Binomial distribution for analyzing incoming traffic at the parking garage has been mentioned in Section 4.3.3 Part A. It can be seen from Figure 38 that as the probability 'p' of a customer being entering increases, the value of protection level also increases. With increasing value of rate ratio, there is an increase in protection level. With increasing value of probability that a customer shows up, there is an increase in protection level as well for a particular binomial probability value (Table 11). This seems correct since we want to sell as many spots as possible to the corporate class to increase revenue. It can be seen that the increase in protection levels is almost linear with increase in probability of corporate customer or even increase in rate ratio [Appendix 1].

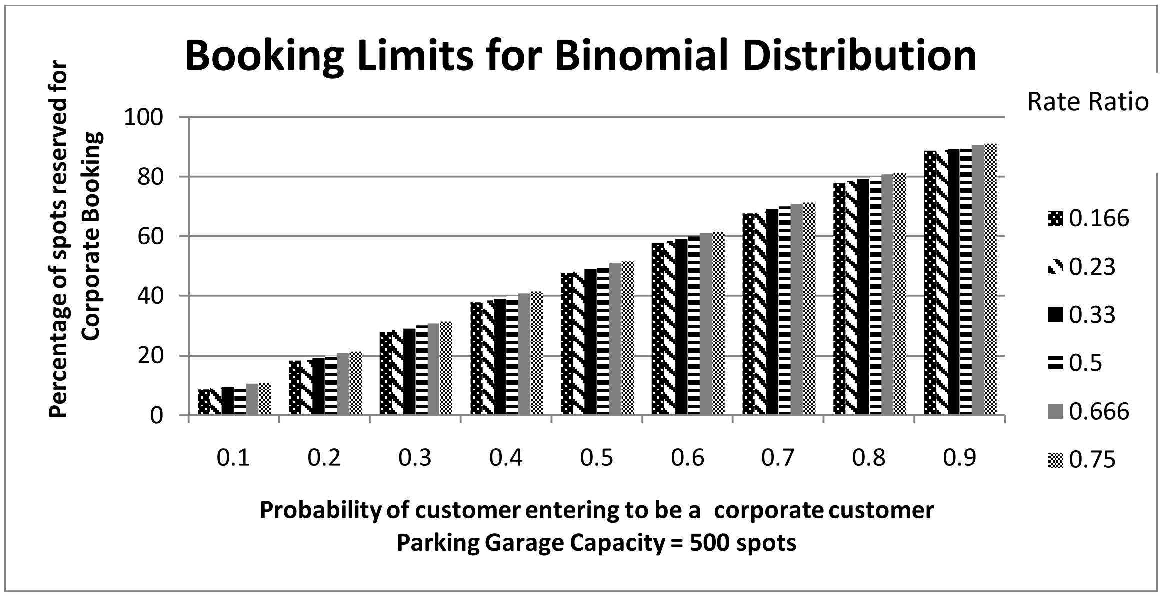


Figure 38: Protection Level given Binomial distribution of arrival

One very important observation is the fact that the increase in protection level is proportionately higher when there is an increase in corporate arrival probability as compared to increase in rate ratio (R_l/R_h). Also, we see that there are more spots allocated to the corporate class when there is a higher rate ratio (for a particular arrival probability). The reason for this can be explained as follows: If there is a smaller difference between the corporate class and leisure class, then an incoming customer will find it sensible to take the corporate class spot since he will get all the benefits for almost the same price as leisure class. Hence, the garage will see an increase in revenue due to higher class spot occupancy. On the other hand, if the difference between leisure class and corporate class is more, an incoming customer will think about whether or not the corporate class spot should be taken or not. In this case, it is more prudent to allocate comparatively lesser number of spots to the corporate class to make sure that there is no loss of revenue in over-estimating number of corporate spots that will be occupied.

5.4 Overbooking

5.4.1 Probabilistic/Risk Model (Algorithm OB1)

a. Observations

The algorithm has been described in Section 4.3.3 Part B. The algorithm takes as input mean and standard deviation of probability that customers do not show up at the parking garage. The algorithm also takes the capacity of the parking garage as input. Table 12 shows some of the values from Figure 39 in tabular form. To know about the selection of values of no show rate and standard deviation (**σ**), see Section 4.1.5 Part B.

No-show rate probability	Sigma (σ)	Overbooking as % of garage capacity
0.2	0.08	107.34%
0.5	0.01	193.63%
0.5	0.25	109.74%

Table 12: Key Observations for overbooking algorithm

b. Inference

Consider Figure 39. We can see that as we increase the value of mean of no-show rate (μ) we get increased values of overbooking i.e. as we increase μ; it means we are increasing the mean value of the no-show rate. This means that there are more people who do not show up. Hence, we need to overbook a greater number of spots. However, as we increase values of standard deviation of no-show rate (σ), we get lower values of overbooking since there is more deviation from the mean value of no-shows. The reason why values lesser than 100% of parking capacity have been shown in the Figure 39 is to demonstrate that the algorithm runs only for certain values of mean and standard deviation of no-show rate (values that are greater than 100%). The values for which overbooking capacity is greater than 100% of garage capacity should be considered by the parking management if overbooking is to be implemented.

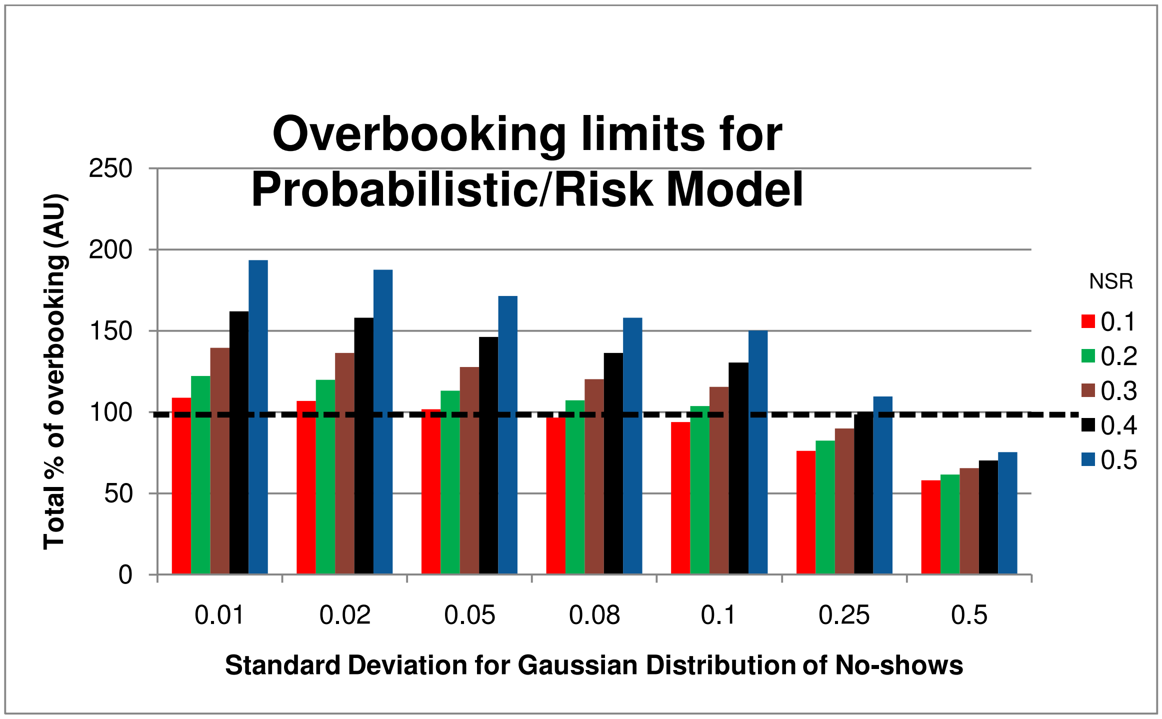


Figure 39: Overbooking given Gaussian No-show distribution [Probabilistic Model] with linear trendline

Chapter 6: Analysis of Results

At the beginning of the thesis, we sought to answer the following questions:

6.1 Tracking Position of Cars

i. What is the cost and accuracy of maintaining and developing a system that uses algorithm T1 and T2?

Cost of developing a sensor system with Algorithm T1 (more sensors) is $1526 and the cost of enforcing algorithm T2 (lesser sensors) in a parking lot is $142 [Appendix 2 A2.1]. The accuracy of algorithm T1 and T2 is dependent on a variety of conditions but it is found that algorithm T1 performs better than algorithm T2 in terms of correct tracking (Section 5.1.1, 5.1.2 and 5.1.3).

ii. Which method is the optimum one in terms of trade-off between cost and accuracy?

While algorithm T1 is more accurate as it uses more sensors, it entails a higher cost for the same reason (Section 5.1.1, 5.1.2 and 5.1.3). Algorithm T2, on the other hand, uses fewer sensors and hence costs lesser (Section 5.1.1, 5.1.2 and 5.1.3). The usage of algorithm T1 involves a cost nearly eleven times the cost that algorithm T2 would entail (Appendix 2 A2.1). For a sensor failure rate of 2%, algorithm T1 provides accuracy about five times that of algorithm T2 (Section 5.1.3). Depending on the budget availability and the accuracy required, a choice can be made between the two algorithms.

iii. What is the scalability of this algorithm?

The two algorithms were developed for tracking 500 users simultaneously. There was no instance of data of a particular user getting lost. Hence, these algorithms can be scaled to higher number of arrivals although practically such a situation will almost never arise because 500 users within the parking lot moving at the same time is a very high number.

iv What metrics were developed in order to discuss the efficiency of the algorithms?

Two sets of metrics were developed:

a. Tolerance Limits: In order to ensure that we obtained minimum number of inaccurate trackings, we set tolerance limits of 10%, 50% and 75% in order to have an estimate of failed sensor readings. 10% tolerance limits are highly stringent and do not permit more than 10% of sensor failures along the path of the car to fail for a reading to be recorded as accurate. In this manner we established a metric for tracking duration of path for which the car is inside the garage.
b. Average number of information points: By obtaining the average number of sensor readings per car, we were able to compare the information provided by the two algorithms T1 and T2.

6.2 Reservation Defragmentation

i. What is the input dataset considered?

We have run simulations for 50 reservations to 1000 reservations for a parking garage that has a capacity to hold a maximum 500 cars. The algorithms were simulated over 10 datasets that included reservations of durations 0 to 22 hours (Figure 29) that are exponentially distributed.

ii. Determining optimum solution based on complexity.

From Figure 38, we can see that for 1000 reservations, algorithm R2 takes about 1.2 times the time required by algorithm R1 for defragmentation. From the figure 38 and Figure 39, we can see that algorithm R3 takes about 25 times the time required for algorithm R2. Based on the complexity analysis, either algorithm R1 or R2 are better than algorithm R3 (Appendix 2).

iv. What is the scalability of the algorithms R1, R2 and R3?

The algorithms have been developed for a 500 spot parking garage with a capacity to take up to a maximum of 1820 reservations (Section 5.2.7). However, even prior to cancellation, the parking garage spots do not get entirely full due to the efficiency of the First Fit algorithm. Post reservation defragmentation, there are even more available spots. Thus, the algorithms can be scaled to perform reservation defragmentation for much more reservations since no errors are recorded during reservation defragmentation for even 1000 reservations. However, as the number of reservations increases, time of execution of algorithm 3 increases exponentially (Figure 41) making it inefficient for higher number of reservations. This is because of the search mechanism of free space that is used by Algorithm R3. The complexity analysis for algorithm R1, R2 and R3 has been done in Appendix 2.

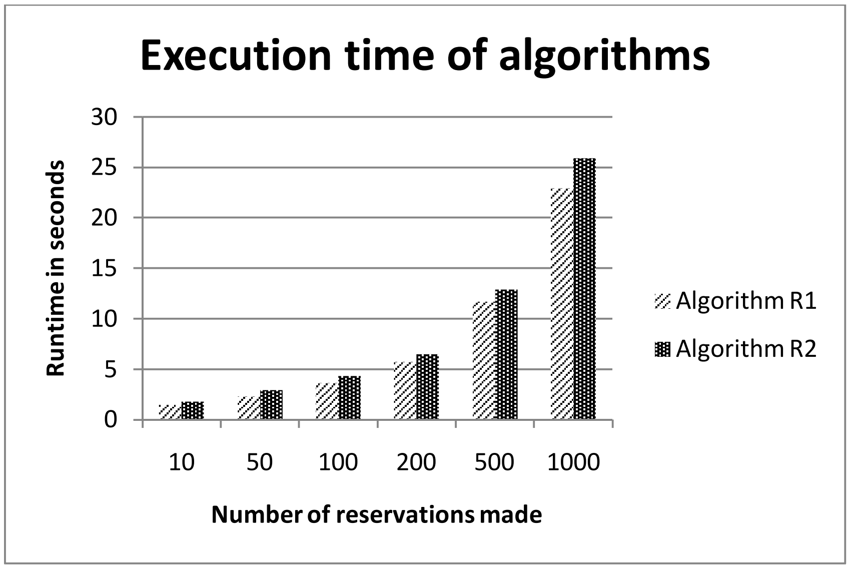


Figure 40: Execution time of reservation defragmentation algorithms R1, R2

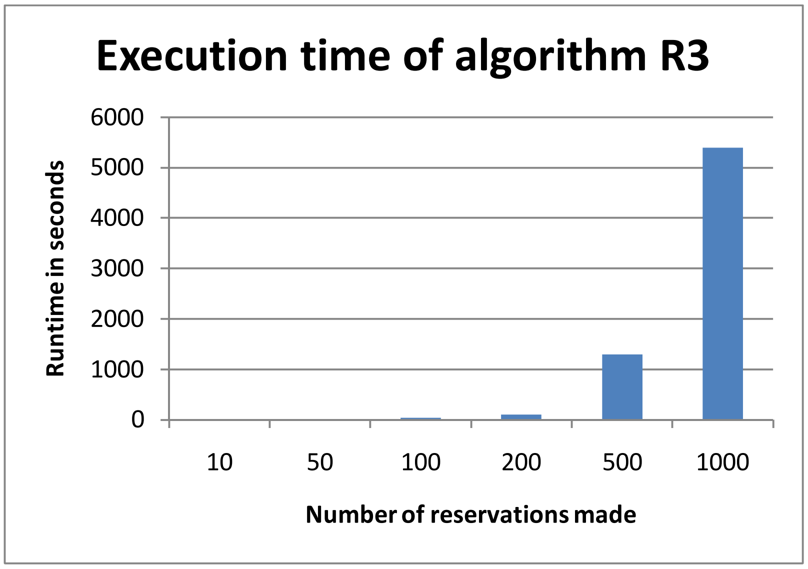


Figure 41: Execution time of reservation defragmentation algorithms R3

The hardware specifications of the device used for executing these algorithms have been mentioned in Section 4.2.2. The execution times mentioned in the Figures 40 and 41 are the result of running algorithm simulations on machine in Section 4.2.2.

v. *What is the efficiency of these algorithms?*

Although, algorithm R1 on its own performs quite well, it results in the least reservation defragmentation as opposed to the performance of algorithm 2 and algorithm 3 (Section 5.4.2, 5.4.3, 5.4.4, 5.4.5). Detailed observations are mentioned in chapter 5 in sections 5.4.2, 5.4.3, 5.4.4, 5.4.5.

iv *Which algorithm provided the best results?*

Algorithm R2 provided the best results for the metrics that were developed for the reservation defragmentation as far as random cancellations was concerned (Section 5.2.3, 5.2.5, 5.2.6, 5.2.7). However, for block cancellations, Algorithm R3 provided a better reduction in occupied spots (Fig 22).

6.3 Revenue Management

i. *How are these techniques affected by the number of users that enter the parking garage i.e. scalability of proposed algorithm?*

While simulations have been run for parking lot of capacity 500 vehicles, the algorithms developed for booking limits and overbooking are both scalable for parking lots of any sizes because the algorithms are based on mathematical formulae which have no bounds (Section 4.3.3 Part A and Section 4.3.3 Part B). These algorithms are applicable to real-world scenarios because they are simulated with a number of probability distribution models to study the effect of multiple customer arrival scenarios.

ii. *What metrics were developed for revenue management and were they directly ported from other industries?*

For revenue management, booking limits and overbooking were the techniques used. However, their application was not as direct as other industries (Airline and Hotel) in which they are applied. For booking limits in order to obtain the cumulative probability for corporate arrival spot demand, we had to use distributions suitable to the parking industry such as Poisson and Binomial distribution (Section 4.3.3 Part A). In case of overbooking, we had to overbook each 30 minute time slot instead of the parking spot as a whole (Section 3.2.6 (b)). The reason for doing this is that in an airline, the seat is occupied for the entire duration of the flight and hence can be overbooked. However, for a parking garage, the parking spot is not necessarily booked for the whole day since many people use the same spot during the course of the day. Hence, the static reference for overbooking in this case is the time slot of reservation which is what we overbook.

6.4 Comparisons with Other Algorithms

a. Performance comparisons

Since there is no other reservation defragmentation algorithm in the parking industry, our benchmark of comparison is the memory defragmentation algorithms. While we cannot make a direct comparison due to the difference in parameters and size of the dataset being worked on, a rough comparison can be made. There are also major conceptual differences between the two systems. For example, in memory allocators, types of objects to be stored in the heap storage are predetermined. The reservations in the parking garage, on the other hand, are not predetermined and reservations can be of random durations. Memory efficiency of allocators can be improved by up to 18.85% by using the budgeting method (Lee et al., 2010) [31]. Our algorithms R1, R2 and R3 provide time slot efficiency increase of up to 23%, 46% and 37% respectively (in case of 15%

cancellations). The reason for the large difference is as follows. It should be noted that our method considers cancellations followed by defragmentation of the remaining reservations whereas the budgeting method works on fixed data which do not take cancellations into account. It should be noted that the performance comparison of these two systems is not fully appropriate and has only been introduced in order to show that there are similarly modeled systems in practice.

6.5 Key Observations

1. In case of booking limits algorithm (Figure 37), as the values of arrival rate goes on increasing, for a particular rate ratio (higher corporate rate to lower leisure rate), the protection level (spots set aside for the corporate class) goes on increasing. The protection level is a number that is determined by the garage management prior to the entry of the first user. Hence, the parking garage will regulate whether or not they should sell a spot to a leisure customer. Once all the leisure parking spots are taken, the remaining parking spots need to be sold at corporate class rate whether or not the customer is actually a corporate customer or not. This is necessary in order to obtain maximum profit for the garage using protection level policy.

Chapter 7: Future Work

7.1 Tracking Real-Time Position

a. It is our intention that the tracked path of the car is 100 % accurate whether the driver takes the right or the wrong path. However, the failure of the sensor causes inaccuracies in the algorithm. If sensor failure occurs such that the last sensor recorded is nowhere near the spot where the user actually went and parked, we will record this as a wrong parking. This is not correct though since the user actually parked in the right spot but we have marked it as an incorrect parking. There is no provision in the algorithm to correct this issue. This implies that we cannot trust this actual path vector and we cannot know whether it is right or wrong. This is a flaw in the algorithm and some work can be done in the future to correct this. In order to determine the final parking spot, sensors which have a much lower failure rate could be used, so that reliability is maximum. Alternately, each spot could have multiple sensors, so that even if one sensor fails, there is information from the other sensor about the occupancy status of that parking spot. While these methods will increase the cost of the system, in order to deal with this situation, such methods can be applied.

b. Currently, we track real-time position of only arriving cars and not of departing cars. It is more important to detect the motion of the car to the parking spot rather than away from it because this enables us to track wrong parking. There is no real value addition with tracking a car leaving the parking lot. The probability that a car will park in some spot and then leave only to park in another spot is practically very low and has been considered to be zero. However, if the need arises, departing cars can be tracked in the future. Currently, the algorithms only detect arrival sensing and track the car position from the entrance to the destination parking space.

c. Ultrasonic sensors have been used in this simulation. However, other sensors such as photo-sensitive or infrared sensors can be also be used.

d. Currently, the tracking algorithms have been developed for a parking garage having a particular layout [Section 4.1.1]. Being able to generalize the algorithm over a layout of any kind can be worked on in the future. Consider a parking garage having a different layout as opposed to the one in Figure 42.

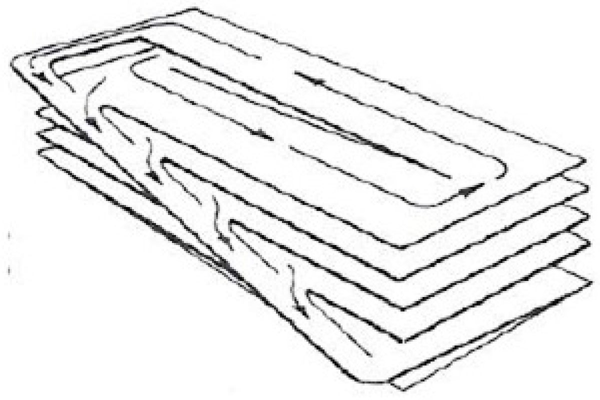


Figure 42: Prototype of alternate parking garage design

Source: Khairunnur B.M.S., et al [26]

In this we can see that we cannot have the four rows of sensors as shown in Figure 12 and Figure 15. Since our algorithm has data structures that are modeled according to the four row model, we cannot apply our algorithm directly to the parking garage shown in Figure 40. Also, the entry and exit points of floors in the two parking garages (Figure 12 and Figure 42) are very different and since our algorithm uses floor sensors for tracking car position as well, change in such configurations will not give correct results if our algorithm is not modified for the new parking garage.

e. After a car enters a parking lot, it is assumed to have a constant speed of travel. This is necessary in order to be able to predict and then corroborate which sensors the car is crossing. However, this assumption may not always be true because of the layout of the parking lot where the car might have to slow down in some places (such as when the car nears a bend in the path) and speed up in others (when the car has straight open road in front). However, in order to determine exactly which car crossed at time ‘x’, we would have to calculate delay threshold levels for the sensors (i.e. how much offset from pre-determined time is permitted to accommodate for variation in speed) to permit change of speed by car to still give us accurate tracking. Since this is very dependent on the layout, we have left this as future work and given the general methodology to approach the tracking problem. In worst case, our algorithm will not be able to track the car if the layout of the parking garage does not match the layout chosen by us. To obtain good results, the algorithm would have to be modified to suit the new garage layout.

f. The tracking system assumes that the user will move in only one direction in the parking lot floor. While this is a valid assumption, there is a possibility that this rule might be violated by some users despite the presence of signboards that suggest the single direction in which the cars should move. While the present system will not be able to deal with this situation based on the algorithms developed and will record the tracking as a wrong tracking, algorithms will have to be developed, in the future, to deal with this problem.

g. Currently, the number of sensors on each floor of the parking garage are fixed, independent of the failure rate of the sensor. However, in order to make sure that the car is tracked even with high failure rates, we could increase the number of sensors on the parking floor (i.e. increase the density of the sensors), in order to capture more positions of the car on the garage floor. The increase in density based on failure rate of sensor is a study that can be carried out as future work.

7.2 Reservation Defragmentation

a. While allotting reservations into parking spots at the time a reservation is created, we have used First Fit Algorithm (Section 4.3.2 Part B). While it is a fast and easy method to implement, it might not always be the most efficient in terms of number of parking spots allotted. The first fit algorithm checks the lowest spot that has space for the incoming reservation. If there is no space in a parking spot, it will place it in a new parking spot. This greedy approach might result in usage of more parking spots than is actually needed which is a disadvantage. Studies can be undertaken to determine if there are better algorithms for this purpose with same or better efficiency.

b. In the algorithm R3 (Section 4.3.2 Part E), in order to enable reservation defragmentation, whenever reservation cancellations occur, an existing reservation will not be moved into a spot higher than its current position but only into a spot lower than its current value. This ensures that the process of defragmenting does not instead result in the reservations being spread among a larger number of spots. However, the drawback of this approach is that it is possible that better reservation defragmentation would have been possible if certain reservations were moved to spots higher than their current spot. The metric which would determine when reservations should be moved to higher spots was not worked on in this thesis but could be worked on in the future. Since, we got good results with the current implementation, this problem was not looked into in detail but its existence has been acknowledged. To my knowledge, current disk defragmentation algorithms also function in a manner similar to our implemented algorithms and do not defragment in both directions. The approach chosen by disk defragmentation algorithms is to carry out multiple passes over the data so as to achieve maximum defragmentation which is the approach chosen by us for algorithm R3 (Section 4.3.2 Part E).

7.3 Revenue Management

7.3.1 Booking Limits

This thesis focused on a two-fare class structure that catered only to leisure and corporate classes. However, n-fare class structure can also be looked into and experiments can be run to determine if this method of setting protection levels is more profitable than the 2-fare class method. If such an implementation of having n-booking classes results in higher profits, then implementation of such a system can be studied.

7.3.2 Overbooking

We have run overbooking algorithms over the entire capacity of the parking garage considering only one-fare class. We have not linked the booking limits and the overbooking algorithms. Hence, the overbooking algorithms do not affect the two-fare structure in any manner in our implementation. However, one of the advantages of having overbooking can be ‘guaranteed reservations’ for the pricier class (corporate class) of reservations. In other words, we can overbook the cheaper (leisure) class parking spots but keep the corporate class spots free from being overbooked. In this case, the corporate class reservation will always be served whereas there will be a probability of denied parking for the leisure class user if the parking garage gets full. This can also be used as an incentive for the corporate class to generate more corporate customers. Hence, overbooking on certain fare classes can be worked on some time in the future and its impact on revenue management studied.

7.4 Walk-In customers

The parking system in this thesis has been studied with respect to reservations being made. While walk-ins have not been incorporated, work can be done in the future, to study the behavior of walk-ins on the system as a whole. The following behavior is expected on the three aspects studied as part of the thesis:

1. Tracking cars: In the current implementation, since our aim was to develop a tracking system for cars within the parking garage, we did not make use of cars that had reservations previously. In fact, reservation duration does not play a part in the tracking system developed. Hence, the incorporation of walk-ins in this scenario is very straightforward since, like all other cars, even walk-in customers will be allotted a parking spot number to which they should park at and the car will be tracked upon entry into the garage.
2. Reservation Defragmentation: In case of reservation defragmentation implementation, the defragmentation has been done purely on the basis of reservations made beforehand. Walk-in customers could be implemented in the following manner. Currently, all reservations are stored in a ***Reservation bitmap***

(See Section 3.2.3) and references to the reservations are stored in the ***Summary Vector*** (See Section 4.3.2 Part C). If there is a walk-in customer, and there is availability for the walk-in customer's requirement, then the reservation can be made on the spot and the entry can be recorded in the reservation bitmap as well as the summary vector. Otherwise, the walk-in customer will have to be turned away due to lack of space. The reason this kind of implementation would work is that the defragmentation algorithms only take in the reservation bitmap entries and the summary vector entries. As long as reservation validation (i.e. to ensure that the walk-in reservation is not overlapping with any other previous reservation) is done, the algorithm will be able to defragment even walk-in reservations. Finer details about whether or not the walk-in customer will have to register with the parking garage at the spot of entry or will be given a guest ID for the duration of the parking, can be worked on in the future.

3. <u>Revenue Management</u>: In the current implementation, revenue management techniques of booking limits and overbooking are both static methods. Hence as a result, a value for the next day is calculated on the previous day in order to maximize revenue. Incorporation of walk-ins in the future is straightforward since any new walk-in customer will be directed to the parking spot (if available at the time) either at the corporate/leisure fare (depending on the customer's willingness to pay) in case of booking limits. In case of overbooking, the situation will be slightly more complicated. In overbooking, since parking spots would have been reserved by a customer, if a certain walk-in customer comes in, the decision of whether or not to provide him a parking spot in the garage would depend on the following conditions:

a. If parking spots are vacant for the duration of the walk-in customers stay in the parking garage and there are no reservations against the to-be-allotted parking spot, then the walk-in customer should be permitted to park in the garage.

b. If parking spots are vacant for the duration of the walk-in customers stay in the parking garage and there are reservations against the to-be-allotted parking spot and that the customer 'X' whose reservation exists has not arrived at the parking garage, then a decision should be made whether to allot the slot to the walk-in customer (assuming 'X' will not show up at all) or to not allot the slot to the walk-in customer (based on the level of overbooking made on the parking garage spots). This study can be done in the future.

Chapter 8: Bibliography

1. Andrews R. (2011), **‘**Parking Lots & Garages in the US’**,** *IBISWorld Industry Report 81293*, 3.

2. Anon. (2005), ‘The Basics of Revenue management’, *IDeaS,* 10, http://www.adhp.org/pdf/1-theBasicsofRM.pdf

3. Arenberg Y. (1991), ‘Reservations and Overbooking’, *Eastern Economic Journal,* 100-108.

4. Basu P., Little T.D.C. (2004), ‘Wireless Ad Hoc Discovery of Parking Meters’, *Proc. MobiSys 2004 Workshop on Applications of Mobile Embedded Systems (WAMES 2004), Boston, MA, USA,* 1-10.

5. Belobaba, P.P. (1987), ‘Survey Paper--Airline Yield Management: An Overview of Seat Inventory Control’**,** *Transportation Science 1987* 21, 63-73

6.Belobaba P., P. (1989), ‘Application of a Probabilistic Decision Model to Airline Seat Inventory Control’, *Opns. Res.* 37, 183- 197.

7.Beckman, M.J., Bobkoski, F. (1958), ‘Airline Demand: An Analysis of Some Frequency Distributions’, *Naval Research Logistics Quarterly,* **5**, 43-51

8.Bitran G., Leong T-Y (1989),’ Hotel Sales and Reservations Planning’, *MIT Sloan School Working Paper #3108-89-MSA*, 1-9

9.Brucker P., Kravchenko S. (2005), ‘Polynomial Algorithm for Parallel Machine Mean Flow Time Scheduling Problem with Release Dates’, *Computational Science and Its Applications – ICCSA 2005, Lecture Notes in Computer Science Volume 3483*, 151-189

10. Carlo Gavazzi, Automation Components,

http://www.gavazzionline.com/pdf/ParkingGuidanceSystemBrochure.pdf

11. Charette R. (2007), ‘Smart Parking Systems Make It Easier to Find a Parking Space’, *IEEE Spectrum October 2007*, 1-3
12. Defoe D., Cholleti S., and Cytron R.K. (2005), ‘Upper bound for defragmenting buddy heaps’, In *Proceedings of the 2005 ACM SIGPLAN/SIGBED conference on Languages, compilers, and tools for embedded systems* (LCTES '05). ACM, New York, NY, USA, 222-229.
13. Delot, T., Cenerario, N., Ilarri, S., Lecomte, S. (2009), ‘A cooperative reservation protocol for parking spaces in vehicular ad hoc networks’, *In Mobility Conference*, 1-8.
14. Duin C.W., Van Sluis E. (2006), ‘On the Complexity of Adjacent Resource Scheduling’, *J. of Scheduling* 9, 1 (February 2006), 49-62.
15. Eijnden F., (2009), ‘Revenue Management at Park ’N Fly’, *MET Volume 17 Issue 1*, 18-24
16. Futurlec Ultrasonic Sensors, http://www.futurlec.com/Ultrasonic_Sensors.shtml
17. Ganchev I., O’Droma M., Meere D. (2008), ‘Intelligent Car Parking Locator Service’, *International Journal "Information Technologies and Knowledge" Vol.2*, 166-173
18. Geraghty M.K., Johnson E., (1997), ‘Revenue management saves National Car Rental’, Interfaces 27, *Institute for Operations Research and management Sciences*, 107-127
19. Goble W.M. (2002), ‘Getting Failure Data rate’, *Exida LLC*, 1-7

20. Gonçalves J.F, de Magalhães Mendes J.J., Resende M.G.C. (2005), ‘A hybrid genetic algorithm for the job shop scheduling problem’, *European Journal of Operational Research, 2005,* 77~95

21. Hadjinicola G., Panayi C. (1997), ‘The Overbooking Problem in Hotels with Multiple Tour Operators’, *International Journal of Operations & Product Management, Vol. 17*, No. 9, 874-885;

22. Inaba K., Shibui M., Naganawa T., Ogiwara M., and Yoshikai N. (2001), ‘Intelligent Parking Reservation Service on the Internet’, In *Proceedings of the 2001 Symposium on Applications and the Internet-Workshops (SAINT 2001 Workshops)* (SAINT-W '01), IEEE Computer Society, Washington, DC, USA, 159-

23. Mathew T. (2009), “Lecture notes in Transportation Systems Engineering”, Indian Institute of Technology(Bombay), http://www.civil.iitb.ac.in/tvm/1100_LnTse/124_lntse/plain/plain.html

24. Jensen C. (1994), Fragmentation: The Condition, the Cause, the Cure, *Executive Software International* (ISBN 0-9640049-0-9).

25. Jun Y. (2010), ‘A System Framework of Active Parking Guidance and Information System’, *Information Engineering (ICIE), 2010 WASE International Conference on* ,vol.2,no., 150-154

26. Khairunnur B.M.S., Nuur L.B.K., Multi Storey Car Parking, 1-34 http://www.scribd.com/doc/44120965/2194924-8-Multi-Storey-Car-Parking

27. Klappenecker A., Lee H., Welch J.L (2010), ‘Finding available parking spaces made Easy’. *In Proceedings of DIALM-PODC* , 49-52

28. Kleinberg J., Tardos E. (2005), ‘Algorithm Design’, *Pearson-Addison Wesley 2005*,

3-8

29. Klophaus, R. and Pölt, S. (2007), ‘Airline Overbooking with Dynamic Spoilage Costs’, *Journal of Revenue and Pricing Management*, 1-18
30. LaGanga L., Lawrence S.R.,(2009), ‘Appointment Scheduling With Overbooking To Mitigate Productivity Loss From No-Shows’, 1-29
31. Lee J., Yi J. (2010), ‘Improving Memory Efficiency of Dynamic Memory Allocators for Real-Time Embedded Systems’, *ETRI Journal, Volume 33, Number 2, April 2011, 230-239*
32. Lee, S., Yoon, D., Ghosh, A. (2008) , "Intelligent parking lot application using wireless sensor networks," *Collaborative Technologies and Systems, 2008. CTS 2008. International Symposium on* , vol., no., 48-57
33. Littlewood, K. (1972), ‘Forecasting and Control of Passengers’, in *Proceedings 12th AGIFORS Symposium*, American Airlines, New York
34. Local Motion http://alexandriava.gov/localmotion/info/default.aspx?id=14576
35. Lu R., Lin X., Zhu H., Shen X. (2009) , ‘SPARK: A New VANET-Based Smart Parking Scheme for Large Parking Lots’, *INFOCOM 2009, IEEE* , vol., no., 1413-1421.
36. Marsic I., Software Engineering Course Project Parking Lot/Garage, Rutgers University, http://www.ece.rutgers.edu/~marsic/books/SE/projects/ParkingLot/ParkingLot.pdf
37. Netessine S. and Shumsky R. (2002), Introduction to the Theory and Practice of Yield Management, *INFORMS Transactions on Education, 3*, 34-44.
38. Tarko A. (2010), “Highway Transportation Characteristics”: CE 463, School of Civil

Engineering, Purdue University, web.ics.purdue.edu/~tarko/CE463/Laboratory/Lab1/Lab1assign.doc

39. Robson J.M. (1977), 'Worst case fragmentation of first fit and best bit storage allocation strategies', ACM Computer Journal, 20(3), 242-244.

40. Swedberg C. (2007), "SF Uses Wireless Sensors to Help Manage Parking" *RFID Journal*, pp 1-2, http://www.rfidjournal.com/article/view/3625

41. Rothstein M. (1985), "O.R. and the Airline Overbooking Problem*,"* *Opns. Res. 33*, 237–248.

42. Shoup D. (2007), "Cruising for parking," *Access, vol. 30*, 16–22.

43. Belobaba P., Odoni A., Barnhart C. (2009), *The Global Airline Industry (1st Ed.),* John Wiley and Sons, pp. 94-95, ISBN: 978-0-470-74077-4.

44. SITA (2005), 'Revenue Integrity- The Solution to Airline Revenue Leakage', 4, www.sita.aero/file/569/Revenue_Integrity_white_paper.pdf

45. Smith B.C., Leimkuhler J. F. and R. M. Darrow,(1992), ' Yield Management at American Airlines'. *Interfaces*, 22:8.31

46. Tang VWS., Zheng Y., Cao J. (2006), 'An Intelligent Car Park Management System based on Wireless Sensor Network', *2006 1st International Symposium on Pervasive Computing and Applications*, 65-70.

47. Temecula Municipal Code (California) (2010), Quality Code Publishing, Seattle, WA,http://qcode.us/codes/temecula/view.php?topic=17-17_24-17_24_050&frames=on

48. Tumlin J., Millard-Ball A. (2004), "The Mythology of Parking", Line Magazine Dec 2004, http://www.walkablestreets.com/parking.htm

49. Urban Parking Concepts, http://www.urbanparkingconcepts.com/docs/systems.html,

2008

50. Van der Mei R., Pak K., Soomer M., Koole G. (2009), ‘Revenue Management in the Parking Industry:New Business Opportunities’, *ERCIM NEWS 78*, 25-26.

51. White P. (2007), “No Vacancy: Park Slopes Parking Problem And How to Fix It,” http://www.transalt.org/newsroom/releases/126

52. Yoo S., Chong P.K., Kim T., Kang J., Kim D., Shin C., Sung K., Jang B.(2008), ‘PGS: Parking Guidance System based on wireless sensor network’, *Wireless Pervasive Computing, 2008. ISWPC 2008. 3rd International Symposium on* , 218-222.

53. Lodi A., Marro G., Martello S., Toth P. (1996), ‘Algorithms for Two-Dimensional Bin Packing and Assignment Problems’, *Universitµa Degli Studi Di Bologna,* 46-60

54. Cormen, Thomas H.; Leiserson, Charles E., Rivest, Ronald L., Stein, Clifford (2001) [1990]. "2.3: Designing algorithms". *Introduction to Algorithms* (2nd ed.). MIT Press and McGraw-Hill. pp. 27–37. ISBN 0-262-03293-7.

55. JFK Airport Parking, 2009. http://www.jfk-airport.net/JFK-parking.html

56. Flintsch A., Rakha H.(2006), ‘Virginia Tech Parking Management Study’, *Virginia Tech Parking System* http://www.facilities.vt.edu/documents/tcs/parking/Parking_Management_Study_200607.pdf

57. GMV Innovating Solutions, http://www.gmv.com/transportation/parking_management.htm

58. Kayak.com, http://www.kayak.com/#flights/EWR-BOM/2011-08-02/2011-08-09, http://www.kayak.com/#flights/EWR-BOM/2011-08-02/2011-08-09/first, (As of July 21, 2011).

59. P. Malcolm, 2011, http://www.shenzhen-jp.org/208471-Difference-Between-First-Class-and-Economy-Flight-Tickets.html

60. Belobaba P., Odoni A., Barnhart C. (2009), *The Global Airline Industry (1st Ed.),* John Wiley and Sons, pp. 96-97, ISBN: 978-0-470-74077-4.

61. Rex S. Toh, Frederick Dekay, Hotel room-inventory management: an overbooking model, The Cornell Hotel and Restaurant Administration Quarterly, Volume 43, Issue 4, August 2002, Pages 79-90, ISSN 0010-8804, DOI: 10.1016/S0010-8804(02)80044-1.

Appendix 1: Glossary

A1.1 General

a. Parking Lot Server (PLS): The PLS is the central computing system for the parking garage that maintains the accounts of all the users registered with the parking lot as well as tracks usage of the parking lot by the customers.

A1.2 Revenue Management

a. Revenue Management: Revenue Management is the application of disciplined analytics that forecasts user actions and tries to enhance product availability and price to maximize revenue growth.

b. Booking Limit: Booking limit is the maximum number of parking spots that may be sold at the lower price

c. Protection Level: Protection Level as the maximum number of parking spots that may be sold at the higher price. It is equal to the difference between garage capacity and booking limit.

d. Overbooking: Overbooking is the sale of a service in excess of actual capacity. In our case, it refers to excess booking of the parking garage that is greater than the actual capacity of the garage.

e. Two-fare class: when there are two types of services categorized on the basis of facilities available such that usage of each has its own distinct fare, then we have established a two-fare class.

f. Leisure Class: Since we have two-fare booking in our parking garage for the booking limits problem to be studied, the lower class (cheaper fare) that is sold to the daily users is called Leisure Class.

g. Corporate Class: Since we have two-fare booking in our parking garage for the booking limits problem to be studied, the upper class (cheaper fare) that is sold to the corporate users is called Corporate Class.

h. Compensation Ratio: It is the ratio of the denied parking cost to the actual parking cost

i. Cost Ratio (R_l /R_h): It is the ratio of the fare of the leisure class to the fare of the corporate class.

j. No Show Rate (NSR): It is the percentage of customers who make reservations but do not show up.

k. Denied parking (DP): It is the practice of denying parking to customers who have made reservations due to the capacity of the garage getting full. This is due to the practice of overbooking.

l. Spoilage/spoilage cost: It is the loss that is caused due to the inability to see a spot in the parking garage

m. Overbooked Ratio (OR = AU/CAP): it is the ratio of the total overbooking (AU) done to the actual capacity of the parking garage (CAP). It is generally >1.

n. Show-up rate: it is the percentage of customers who made reservations who actually show up

o. Algorithm OB1 : Overbooking Algorithm 1 (Probabilistic/Risk Model)

A1.3 Tracking

a. Actual path vector: It is a real-time list of sensors that are crossed by a vehicle in a parking garage when the car travels to the parking spot that it actually parks in. It is maintained by the PLS.

b. Designated path vector: It is a list of sensors that should be crossed by a vehicle in a parking garage if it parks at the spot that is designated to it when it enters the parking garage. It is generated by the PLS.

c. Sensor Failure Rate: It is the probability that an ultrasonic sensor in the garage will fail. It helps determine if a car passing under a sensor will be detected or not.

c. Algorithm T1 : High Cost algorithm with higher number of sensors

d. Algorithm T2 : Low cost tracking algorithm with lesser number of sensors

A1.4 Reservation Defragmentation

a. Reservation Defragmentation: Pockets of smaller reservations that are scattered in spots all over the parking garage are caused due to several reasons (such as inefficient allocation and cancellation). Hence, we need to coagulate all reservations as much as possible to free more space. The problem of rearranging the reservations in a bitmap for efficient use is called Reservation Defragmentation.

b. Reservation System: It is the technology infrastructure that helps a customer reserve a parking spot in a parking garage in advance of his/her arrival at the garage.

c. Reservation Bitmap: It is a 2 dimensional array made up of 1's and 0's. Each 1 represents an occupied status for that spot for 30 minutes while each 0 represents vacant status for that spot for 30 minutes. In this way, a bitmap is constructed for each day and it has *N* rows for time and *M* columns for parking spot identifiers.

d. Algorithm R1: Reservation defragmentation algorithm 1 (Sorting finish times)

e. Algorithm R2: Reservation defragmentation algorithm 2 (Recursive first fit)

f. Algorithm R3: Reservation defragmentation algorithm 3 (Free Space Vector method)

g . Mean length of contiguous free time slots : Mean length of contiguous free time slots is the average length of contiguous free time slots that occur in a particular parking spot.

Appendix 2: Mathematical Calculations

A2.1 Ultrasonic Cost Calculations

a. Algorithm T1 (High Accuracy) [Based on Figure 12]

Number of tracking ultrasonic sensors required per floor (n): 68

Total Floors (f)	: 5
Spacing between row sensors	: 9 feet (based on parking spot size. See Figure 12)
Total ultrasonic sensors required (N)	: n*f = 340
Cost per ultrasonic sensor (c1)	: $3.9
Sensor purchase cost (c2)	: $1326
Maintenance cost (c3)	:$200
Total Cost (C)	: c2 + c3 = **$1526**

b. Algorithm T2 (Low Cost) [Based on Figure 15]

Number of tracking ultrasonic sensors required per floor (n):6

Total Floors (f)	: 5
Total ultrasonic sensors required (N)	: n*f = 30
Cost per ultrasonic sensor (c1)	:$3.9
Sensor purchase cost (c2)	:$117
Maintenance cost (c3)	:$25
Total Cost (C)	: c2 + c3 = **$142**

Source for sensor cost and sensor information:

Futurlec Ultrasonic Sensors (http://www.futurlec.com/Ultrasonic_Sensors.shtml) [16]

A2.2 Maximum Speed of Car

It is necessary to know what the maximum permissible speed of the car can be in order to determine two consecutive cars to be separate rather than one long car. The reason this is coming into the picture is because the ultrasonic sensor has a certain response time to rise to ‘1’ i.e. to detect the presence of a car and a certain response time to fall to ‘0’ i.e. to determine the car has left the sensor. If the cars are traveling too fast, and another car comes during the response time period of the previous car then we will not be able to distinguish between the two. Hence, the time required for the sensor to reach a stable ‘1’ value will determine the maximum speed of car. In fact, to determine the maximum speed of the car, we need to use the length of the smallest car because if the sensor can successfully detect a fast moving small car such as a smart car then it can easily detect a fast moving longer vehicle. We assume the height of the ceiling to be 15 ft. let the speed of ultrasound in air be 330 m/sec. The ultrasonic sensor keeps sending out ultrasonic beams out periodically. Assuming the car height to be 6ft, the ultrasonic beam has to travel from the ceiling to the top of the car i.e. 15ft – 6ft = 9 ft. In meters, the distance between ceiling and top of the car is 9ft/3.3 = 2.727 meters. The time it takes for the beam to return in case of a car

$$\mathbf{t = \frac{2*d}{s}} \qquad \textbf{(A1)}$$

where t= time required for reflected beam to received after transmitting

d = distance from the ceiling to the top of the car(for reflection)

s = speed of ultrasound in air

Plugging in the values known into equation A1,

Where $d = 2.727$ m and $s = 330$m/s;

$$t = (2 * 2.727) / 330$$

$$t = 16 \text{ ms}$$

where, 't' is the time required for the beam to reflect off the car after the car has come under the ultrasonic sensor. Since there are multiple sensors feeding information to the PLS(Parking Lot Server), there will be a certain processing time before the sensor triggered is actually registered in the actual path vector which is a path vector of cars in real-time. Let this processing time be 100ms. Hence, the total time required for the sensor to be added to the path vector of the vehicle indicating real-time position is 100ms + 16ms = 116ms. If we find the maximum speed that the smallest car (Smart car) can travel at, then we have established the boundary condition because any bigger car can easily be detected if it travels at this speed limit. We know response time is 70ms. However, the ultrasonic sensor has threshold values for '1' and '0' and this is within 2% of the ideal values. In other words, if the voltage value reaches greater than 98% of ideal '1', we still consider it a '1' and if voltage value reaches 2% of ideal '0', we still consider it a '0'. Based on this if the response time to be taken into consideration is 96% of actual response time [100-2-2 = 96] and a smart car is used to find the critical maximum speed, we can say,

If a smart car has speed 's' m/s, then in 70ms it has travelled (70ms) * s. Distance remaining now is 2.5m – ((70ms)*speed). 2.5 meters is the length of the smart car. As processing (adding sensor to the spot) takes 116ms, we know,

$$[(116\text{ms}) * s] *[(70 \text{ ms})*s] < = 2.5 \qquad \textbf{(A2)}$$

Plugging in the values to get the value of maximum permissible speed, and we get

$$[(0.116*\text{speed}) + 0.07 * \text{speed}] < = 2.5$$

$$\textbf{Speed} \leq \textbf{30.511mph}$$

A2.3 Minimum Distance between Cars

In order to obtain minimum distance between cars that needs to be maintained, we need to know when the ultrasonic sensor reaches a stable 0 i.e. it deregisters the previous car from its local memory and can register the next arriving car. This is the same as the response time of the sensor, since response time is also the time required for the state to reduce to 0 from 1 (‘occupied’ to ‘not occupied’). Hence, there should be atleast a 70 ms time difference between consecutive cars. In terms of distance this translates into atleast 0.5 ft for a car travelling at 5 mph and atleast 3 ft for a car travelling at 30.51 mph. (using d = s*t; where d = distance between cars, s = speed of 2nd car, t = time interval between two cars at a sensor)

A2.4 Profits from Imposing Protection Levels

Setting protection levels translates directly into profit for the parking manager. Consider an arrival rate of 100 cars/hour (Flintsch et al., 2006) [56] and cost ratio (Appendix 1) of 0.33. Let the cost of corporate class be \$5/hour. From the definition of cost ratio, the cost of leisure class is \$1.65/hour. We now need to compare revenues from a) when no protection levels are used b) when protection levels are used. We will study these two cases for maximum possible revenue i.e. when all spots are occupied for the whole day (24 hours).

a. When no protection levels are used: This condition implies that all spots are sold at the price of the leisure class rate. The revenue in such a case would be

Each spot is occupied for 24 hours. Hourly rate is \$1.65. Hence, each spot generates

$$24 \times 1.65 = \$39.6$$

Since there are 500 spots and each spot generates \$39.6 a day, the entire parking garage has a daily revenue of

$$500 \times \$39.6 = \$19800 \qquad \text{(I)}$$

b. When protection levels are used: This condition implies that some spots are sold at corporate class rate and some are sold at leisure class rate. From Figure 20, for the condition of cost ratio 0.33 and arrival rate 100 cars/hour, we can see protection level is 19.2%. 19.2% of the garage's 500 spots equals

$$19.2\% \text{ of } 500 = 96 \text{ parking spots}$$

This implies 96 garage spots should be sold at corporate class rate and 404 garage spots should be sold at leisure class rate.

Calculating the revenue for corporate class spots:

Each spot is occupied for 24 hours. Hourly rate is \$5. Hence, each spot generates

$$24 \times 5 = \$120$$

Since there are 96 spots and each spot generates \$120 a day, the corporate class spots have a daily revenue of

$$96 \times \$120 = \$11520 \qquad \text{(i)}$$

Calculating the revenue for leisure class spots:

Each spot is occupied for 24 hours. Hourly rate is \$1.65. Hence, each spot generates

$$24 \times 1.65 = \$39.6$$

Since there are 404 spots and each spot generates \$39.6 a day, the leisure class spots have a daily revenue of

$$404 \times \$39.6 = \$15998.4 \qquad \text{(ii)}$$

Adding equations (i) and (ii) to get total daily revenue of parking garage when protection levels are used, we get

$$15998.4+11520 = \$27518.4 \qquad \text{(II)}$$

Comparing (I) and (II), the percentage increase in revenue due to protection levels equals:

$$((27518.4 - 19800)/19800) \times 100 = 38.98\%$$

Hence, the percentage increase in revenue (for arrival rate 100 cars/hour and cost ratio 0.33), is 38.98% which is significant.

A2.5 Complexity Analysis

a. <u>Algorithm R1 [Sorting finish times]</u>

The algorithm involves sorting the ***'Reservation Defragmentation arraylist'*** according to finish times and then inserting the reservations into the reservation bitmap.

Car Registration Number	Reservation Start Time	Reservation End Time	Reservation Parking Spot	Move Flag

Figure 43: Reservation Defragmentation arraylist

Since the algorithm runs defragmentation based on current time, we should not move the reservations that are currently in the parking garage. In order to know which these reservations are, we set the *'move flag'* to '1' to indicate that the reservation should not moved in the defragmentation process. Otherwise, *'move flag'* is set to '0' by default.

Since merge sort has been used to sort the arraylist, complexity of merge sort is O (n log n). Inserting the each reservation into the bitmap is O (1) and for 'n' reservations this is equal to O (n). Hence, total time complexity is

$$O\ (n \log n) + O\ (n) = \mathbf{O\ (\boldsymbol{n} \log \boldsymbol{n})}\ \text{[approximately]}$$

b. <u>Algorithm R2 [Recursive First fit]</u>

The defragmentation uses First Fit Decreasing algorithm which directly packs the reservations into finite parking spots after having sorted the reservations in decreasing order of reservation duration Section 4.3.2 Part D. The current item is packed into the first spot which can accommodate it, or a new one, if no such bin exists. Sorting is done using merge sort which has a time complexity of O (n log n). The first fit algorithm has time complexity O (k^2) (Lodi et al., 1996) [53] where 'k' is the number of parking spots. However, since 'n' reservations need to be fitted, the first fit algorithm (each of O (k^2)) is done for 'n' reservations. The total time complexity is

$$\mathbf{O\ (\boldsymbol{n} \log \boldsymbol{n}) + O\ (k^2 \times \boldsymbol{n}) = O\ (\boldsymbol{n} \times (k^2 + \log \boldsymbol{n}))}$$

c. <u>Algorithm R3 [Free Space Vector]</u>

In this algorithm, we try to fit a reservation into the free time slots between reservations that are placed in parking spots lower than the current reservation being considered. In the worst case, if we have a total of 'n' reservations, then we would have to consider the free time slots in between the prior ‘n-1’ reservations. The number of sets of free time slots in between ‘n’ reservations is ‘n-1’. Hence, the number of sets of free time slots in between ‘n-1’ reservations is ‘n-2’. The total time complexity in this case is

$$\mathrm{O}\ (n \times (n\text{-}2)) = \mathbf{O}\ (\boldsymbol{n^2})\ \text{[approximately]}$$

A2.6 Choice of Cancellation Rate for Reservation Defragmentation Algorithms

We have chosen to simulate reservations with a 15% cancellation rate (Section 4.1.5 Part A (c)). However, to show that the choice of cancellation rate does not affect our defragmentation algorithms, we also tested the dataset for 30% cancellation rate. We ran the algorithms for 30% cancellation rate over 100 datasets to find percentage decrease in occupied parking spots (Section 5.2.4).

For percentage increase in free parking spots:

Number of Reservations made	% Cancellation	Method R1	Method R2	Method R3
200	15%	8.14%	10.65%	12.99%
200	30%	19.17%	23.01%	24.2%
1000	15%	7.18%	14.62%	15.13%
1000	30%	31.65%	35.17%	37.07%

Table 13: Percentage decrease in occupied parking spots using block cancellation

We can see for both 200 and 1000 reservations as the cancellation rate increases from 15% to 30%, there is a proportional gain (almost two times), in the percentage increase in free parking spots due to defragmentation. For example, for Method R3, for 200 reservations, we can see that there is 12.99% increase in parking spots available for 15% cancellation, whereas there is 24.2% increase in parking spots available when the cancellation rate is 30%.

Since the choice of cancellation rate does not affect the actual defragmentation done by the algorithms, we have simulated cancellation rate of 15% and results for this have been shown in Section 5.2.

A2.7 Possible Implementation of Driver and Parking Garage System Inter-Communication

When the user enters the garage, at the garage entrance, he has to submit his registration number (with the parking garage) into a keypad installed for this purpose. When the Parking Lot Server (PLS) receives this information, it cross-references it against the existing registered customers database and if this customer is not registered, denies entry to the user. On the other hand, if the registration number is valid, then the PLS will have information about the user such as the mobile phone number, history of usage of parking garage etc. Along with user validation, the PLS also provides the user with the parking spot index that the user should park his/her car at on an LED display board at the garage entrance. Thus, when the car enters the parking garage, the PLS knows the saved mobile phone number of the user. In the event of any change in the allotted parking spot, the PLS can notify the user of this change through two ways:

1. <u>SMS (Short Messaging Service):</u> SMS can be sent to the user's cellular phone regarding the change in the parking spot. There is a company GMV (Innovating Solutions) that has a system which allows the parking service user to access information in real time to find out whether the parking zone where he or she desires to park is occupied or not (GMV, Innovating Solutions) [57]. This is done via SMS

mobile messaging or an Internet connection. While the system is designed for street parking, it could be modified for garage parking as well.

2. Smart-phone application: A smartphone application (Android/iOS/Blackberry) could be developed for the client side (user) for the purpose of obtaining updates of the parking spot from the PLS when the user enters the parking garage. Since the PLS will know the phone number of the user and the user will have the smartphone application required, he could obtain the required information in real-time from the PLS. While alerting using smartphones has its risks, to my knowledge, there is no other method where the user can be alerted in real-time. In fact, feasibility studies for smartphone alerts are being tested by the San Francisco's Department of Parking and Traffic (Swedberg, 2007) [40].

The availability of Global Position System (GPS) has been widely used in land vehicle navigation applications. However, the positioning systems based on GPS may not be suitable for real-time parking navigation. The reason lies in the fact that the precision of many common used GPSs may not reach positioning each parking space, and more importantly, the status of a parking space is dynamic. A parking space that is vacant at the current time could be occupied in the next time. For these reasons, GPS is not a suitable method of guiding vehicles within the parking garage.

It should be noted that it is possible that the user does not check his parking spot change alert received (either through SMS or smartphone application message). In such a case, the user will not know that his original parking spot has been occupied by someone else. The user, say ‘A’, will have to drive to the spot to see it for himself. In such a situation, he could either check for any new updates received from the PLS or drive to the nearest parking spot and park there. The new parking spot will now be marked as occupied by this user ‘A’. Three conditions are now possible.

i.) In case the parking spot was not allotted to anyone else, then there is no issue and the algorithm will only mark this user ‘A’ as a wrong tracking (due to parking at a wrong spot).

ii.) If the parking spot had been allotted to another user , say ‘X’ (before the arrival of user ‘X’), then when X does arrive he will be allotted a new parking spot before entering the garage. This is not a problematic case either. ‘A’ will still be marked as wrong tracking.

iii.) The problem will arise in case ‘X’ already entered the parking garage and his parking space was then occupied by user ‘A’. In this case, the parking spot change alert (by SMS or smartphone) will now be sent to user ‘X’. ‘A’ will be marked as wrong tracking. If ‘X’ pays heed to the message, he will not be marked as wrong tracking. If ‘X’ however parks at any other arbitrary parking spot (not given by the PLS), then ‘X’ will also be marked as wrong tracking (for parking at the wrong spot).

A2.8 Reason for Overbooking over entire garage capacity

While overbooking is a concept used in the airline industry, to my knowledge, there is no application of overbooking in the parking industry. The aim of overbooking in this thesis is to introduce the practice of overbooking into the parking industry. During the course of the thesis, booking classes and overbooking are implemented as two separate revenue management practices with no overlap between them. An overlap of the two practices has been discussed as part of future work in Section 7.3.2.

It should be noted that in the overbooking algorithm, one of the input parameters is 'Capacity that needs to be overbooked'. In our existing implementation, we have overbooked all 500 parking spots (See Section 4.1.1). In order to discuss the modification in the algorithm required if we consider

a. Completely protecting corporate class from overbooking: For this case, we should count only the number of parking spots of the leisure class. In this case, however, booking limits will also need to be implemented.

b. Not completely protecting corporate class from overbooking but carrying out overbooking on a small section of corporate class spots as well as the rest of the garage: For this case, we should count the number of parking spots of the corporate class that we want to overbook as well as all the parking spots of the leisure class as well. In this case as well, booking limits will need to be implemented.

A2.9 Cost and accuracy comparisons between Algorithm T1 and Algorithm T2

In order to choose between the layout for Algorithm T1 and Algorithm T2, we have given a cost comparison and accuracy comparison between the two in this section. Based on the requirements of the parking garage, the operator can choose either of the two layouts.

a. **Cost Comparison (Refer to Appendix 2 A2.1)**

Algorithm T1: $1526

Algorithm T2: $142

Order of Magnitude Algorithm T1 is more expensive than Algorithm T2:

$1526 / $142 = 10.75

b. **Accuracy Comparison (See Section 5.1.2)**

Worst case accuracy comparison: (For 2% sensor failure and 50% tolerance permitted)

Algorithm T1: 0.8%

Algorithm T2: 3.8%

Order of Magnitude Algorithm T1 is more accurate than Algorithm T2:

3.8% / 0.8% = 4.75

We can see that Algorithm T1 is about 11 times more expensive to implement than Algorithm T2 but is about 5 times more accurate than Algorithm T2 and provides about 6 times more information than Algorithm T2 (See Section 5.1.3 (d)).