# University of Illinois at Chicago

Department of Computer Science

Spring 2016

## Project Report On

## Spatio Temporal Resource Search

## Submitted by

## Team 7

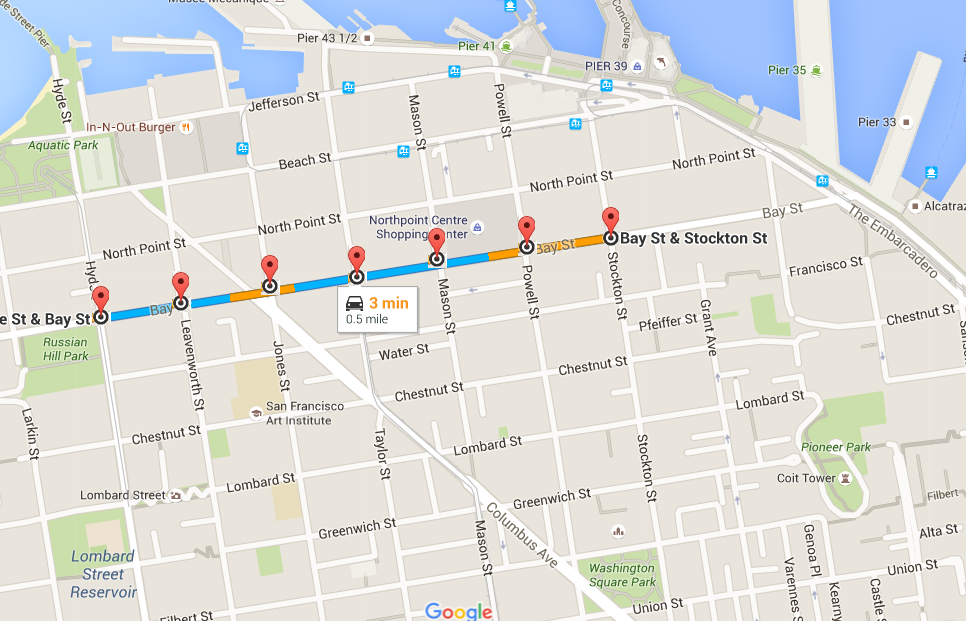
|  |  |
| --- | --- |
| **Swapnil Akolkar** | **665395774** |
| **Vishalaxi Tandel** | **659468038** |
| **Raghavendra Bableshwar** | **664260979** |

## Under the guidance of

## Prof. Ouri Wolfson

Department of Computer Science

University of Illinois at Chicago  
Chicago, IL 60607-7053



# Abstract

The project aims at analyzing various algorithms on given Spatio-Temporal dataset. The dataset contains information about parking and de-parking activities for a range of a month. The given data holds about 1.5 million records. We need to devise and implement algorithms and examine their behavior. We would then simulate an incoming car and using the given dataset, run the algorithm to determine the travel time suggested by each algorithm before getting the parking slot.

We have categorized our algorithm into three different categories based on the level of information the algorithm has.

Following table gives a snapshot of the three algorithms proposed:

|  |  |
| --- | --- |
| **Algorithm** | **Level of Information** |
| Baseline | No Information |
| Probabilistic | Historical Information |
| Deterministic | Real time Information |

# Table of Contents

[Introduction](#h.2s8eyo1)

[Goal of the project](#h.17dp8vu)

[Design Overview](#h.3rdcrjn)

[Problem and Solution approach](#h.26in1rg)

[Implementation details](#h.lnxbz9)

[Experiment and Simulations](#h.35nkun2)

[Result and Analysis](#h.1ksv4uv)

[Future Work](#h.44sinio)

[Conclusion](#h.2jxsxqh)

[References](#h.z337ya)

# Introduction

Like every other resources, parking slots are limited and number of people contending to take on a parking slot are more than the available number of parking slots. Moreover, the information about availability of parking slot is not readily available. Due to lack of information, the drivers keep looking for a parking slot which not only cause the traffic congestion in the neighborhood but also wastes fuel.

Different cities tries to implement solutions to address the parking issues. The city of San Francisco implemented a sensor-based approach to solve this problem. The city installed sensors in the ground to track the availability of the parking slot at that specific time. However, the installation and the maintenance of these sensors were economically not feasible.

Our project is trying to address the problem of finding a parking slot without incurring the cost of the installing and maintaining the sensors. The project aims at implementing various algorithms working at different levels of information to address the issue better.

## Following are the three approaches that we have devised based on level of information:

## 1. Baseline Approach:

This approach simulates the parking slot search which is done by a driver who does not have any information of the available parking slot. The algorithm will have knowledge of the road network to search for a parking slot, and the availability of a slot in a particular block at a particular time. So the input for this algorithm is just the *block\_id* and the *timestamp* at which the car is searching for the parking slot.

## 2. Probabilistic Approach:

This approach has an historical information of parking and de-parking activities for each block and for different time frames. Using this information, we will generate a predictive model. This model will take a block id and timestamp as input and will return the expected number of slots for various block. The algorithm then, computes a ratio of expected number of slots and the distance between the nodes to be travelled. Based on the ratio, a node is selected. Series of nodes, make a route. Once a parking slot is found in any of the blocks the algorithm stops and returns the list of nodes.

## 3. Deterministic Approach:

For this approach, we consider the real time information of parking and de-parking activities. The given dataset provides information about number of parking slots available for each block and for different time frames. The algorithm takes current block id of user and timestamp as an input. The algorithm then, computes a force value and the distance between the nodes to be travelled. Based upon the least distance and highest force value, a node is selected. Series of nodes, make a route. Once a parking slot is found in any of the blocks the algorithm stops and returns the list of nodes.

# Goal of the project

The goal of the project is to compare and analyze various approaches of finding a parking slot for a given block id and timestamp. We compare each of the algorithms with each other to interpret how availability of various levels of information control the performance. The higher level of information comes at a high cost as sensors are needed and they are expensive.

The project deals with identifying various algorithms/approaches based on the level of information (about the availability of slots), this information is fed to the algorithm to get the route for an available parking slot.

Upon completion of the project and the experiments involved, we can comprehensively compare the different approaches with each other using total time the car travels before finding a parking spot. This will help us gauge the pros and cons of each approach.

# Design Overview

## 

**Fig: Architecture of the system proposed**

## Assumptions:

1. The input data will be the road network given in csv files
2. The speed of the vehicle which is searching for the parking slot shall be 10 miles/hour (15 ft./sec approximately)
3. Normal availability of google map APIs to provide route between origin and destination coordinates

# Problem and Solution Approach

As discussed earlier, there can be multiple solutions to the problem in varying conditions. The aim/goal of the project is evaluating these solutions under different conditions.

The solution has been divided into three tasks/approaches, each of which deals with the same data but under different conditions which try to simulate a real life situation. The approaches and the level of information available in these approaches are as follows:

|  |  |  |  |
| --- | --- | --- | --- |
| Approach | Level of Information | Maintenance cost | Accuracy |
| Baseline | No information | Low | Low |
| Probabilistic | Historical data | Moderate | Medium |
| Deterministic | Real time data | High | High |

We are going to record travel time of car to find a parking slot to measure efficiency of each algorithmic approach. The metric will give a comprehensive way to compare tasks and simulate different environment and compare each other.

# Implementation details

## Baseline Algorithm:

Input: Block id, timestamp

Output: Set of nodes travelled during parking slot search

selectPath (block b, timestamp t):

while the algorithm has not found a parking slot:

retVal = findSlot (block b, timestamp t)

if retVal == -1 i.e. no parking slot was found in the block b:

select the next block which is incident on the end node of block b

else

add retVal to the route and return

findSlot (block b, timestamp t):

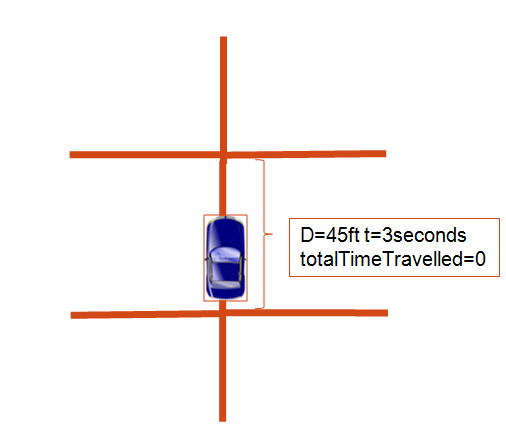
calculate the time ‘t1’ required to traverse through the block

if there exists a parking slot within the range of t and t1

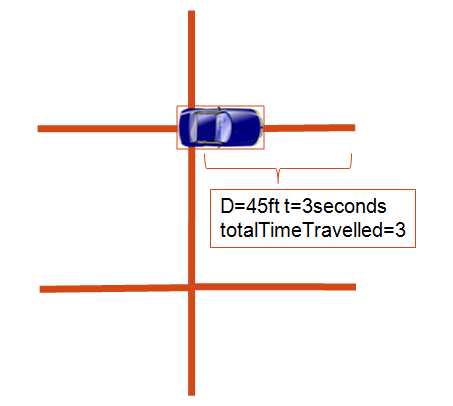
return endNode of block b

else:

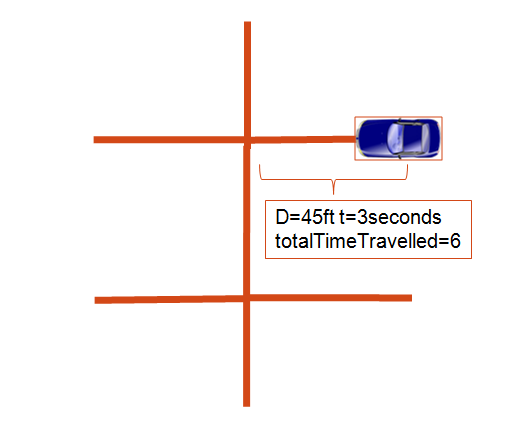
return -1 indicating no parking slots were available



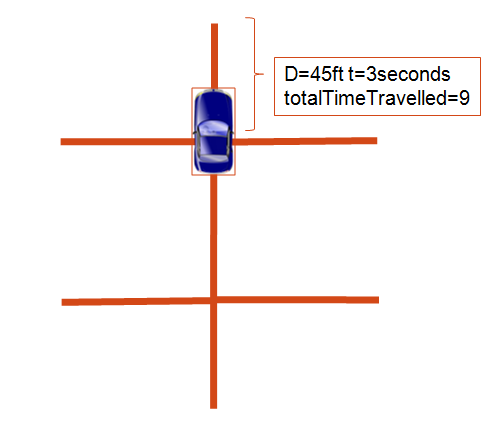
**Fig: Car starts looking for a parking slot and travels in a block**



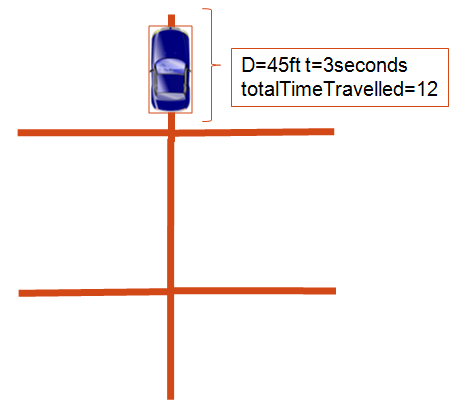
**Fig: Car takes a right turn to look for a parking slot and accordingly travel time is updated**



**Fig: Car reaches other end of the right turn and finds that there are no parking slot**



**Fig: Car returns from the right turn and follows straight path to look for a parking slot**



**Fig: Car travels till the other end of the straight path, accordingly the amount of time travelled gets updated**

## 

## 

## 

## 

## 

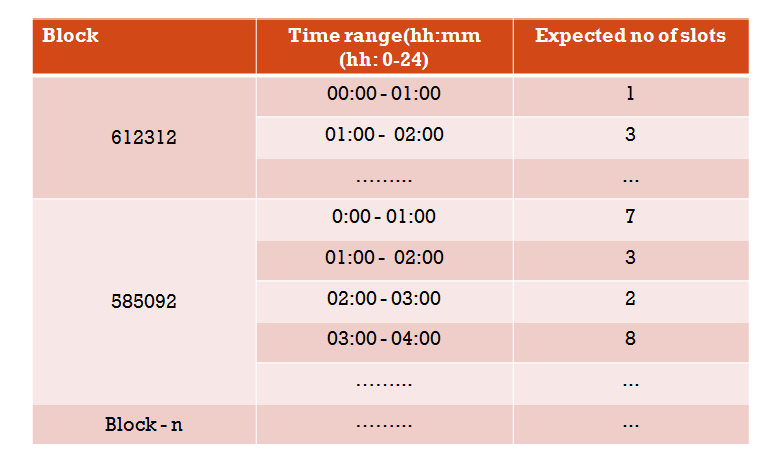
## 

## 

## Probabilistic

**Uses historical information and generates a predictive model in the following manner:**

1. For each block, for each time range(hourly)expected no of slots available is calculated using projection data(dbProjection\_4\_6\_125\_5\_12.csv)
2. Expected number of slots is computed by finding the weighted average of the number of slots available for a particular time range



**Fig: An Example of predictive model based on historic data**

**Algorithm:**

1. Get the starting currentNodeID from inputBlockID and add it to nodeList
2. Repeat the following until parking slot is found:

a. Find all the adjacent nodes of currentNodeID (node1,node2,node3)

b. For each adjacent node, find corresponding blockIDs

c. For each block calculate the ratio of Expected slots to length of block (Distance)

d. Pick the block with highest ratio(N/D)

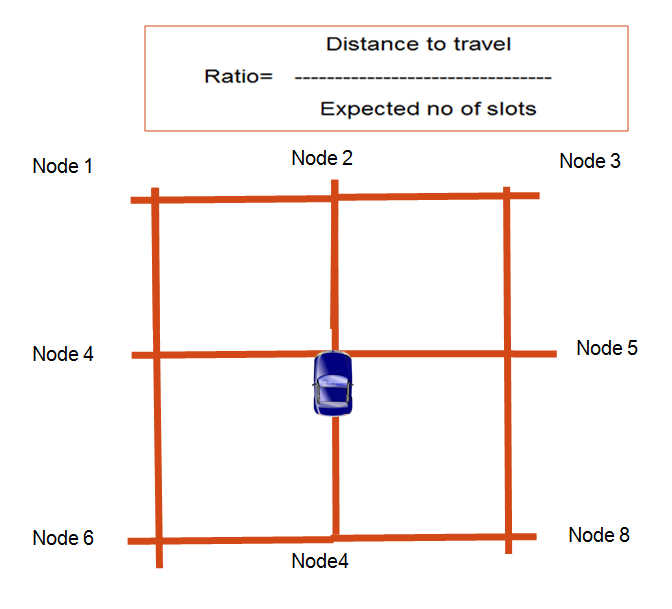
e. Calculate the time required to traverse that block in seconds (Distance/Speed(=10 miles/hr))

f. Find the sum of total time taken for all the blocks travelled

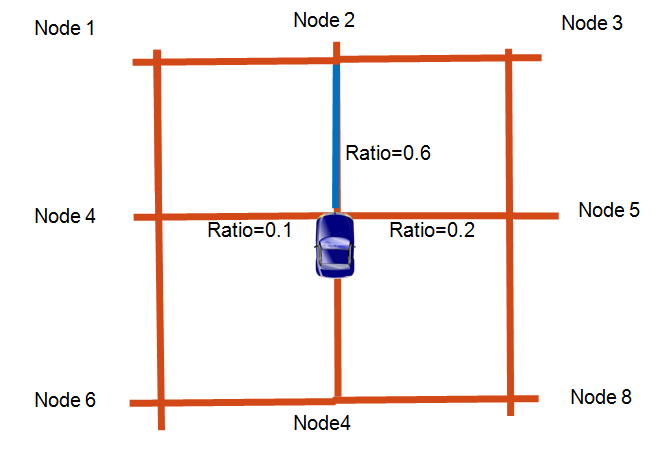
g. Return the node list when parking is found and break from the loop

3. Pass the node list to output module

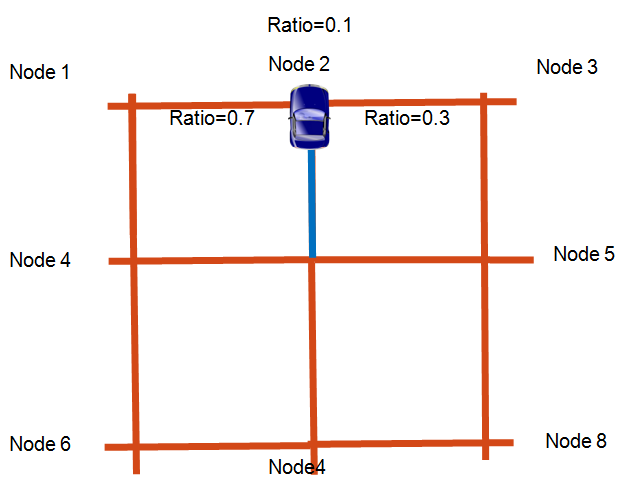
4. Output module displays the route on map



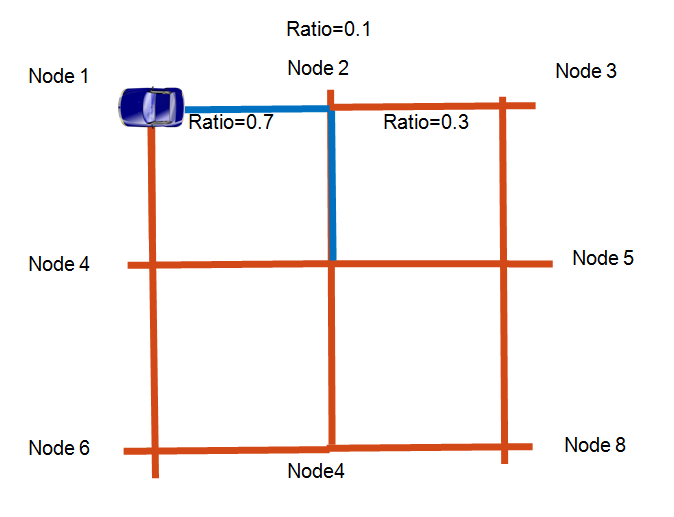
**Fig: Consider a road network as shown, let’s say the car starts from the location shown**



**Fig: The car takes a straight path based on the highest value of ratio**



**Fig: The car reaches the other end of the selected block and now, again calculates ratio of the corresponding adjacent blocks**



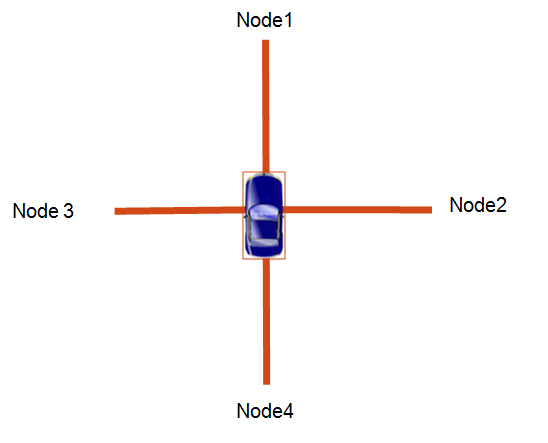
**Fig: The car selects the path with highest ratio and then travels till the other end of the block. If a parking slot is found in the block then the algorithm stops. The path in blue color indicates the route.**

## Deterministic

1. Take block id and timestamp as an user provided input
2. Extract the first node of the block and it to the output list of nodes
3. Get the list of all the adjacent nodes of the first node
4. Find blocks for each of the adjacent nodes. In case of block with operational = 2, we get list of blocks for a node pair.
5. Find distances between each of the adjacent nodes from node 1.
6. Calculate the amount of time, say new\_time that will be to travel each of the adjacent nodes
7. Get the number of available slots between the input time and the new\_time in the each of the adjacent blocks.
8. Calculate force value for each and every adjacent nodes

|  |
| --- |
| Force value (current\_block, adjacent\_block) =no\_of\_available\_slots (adjacent\_block)/ (distance (current\_block, adjacent\_block))2 |

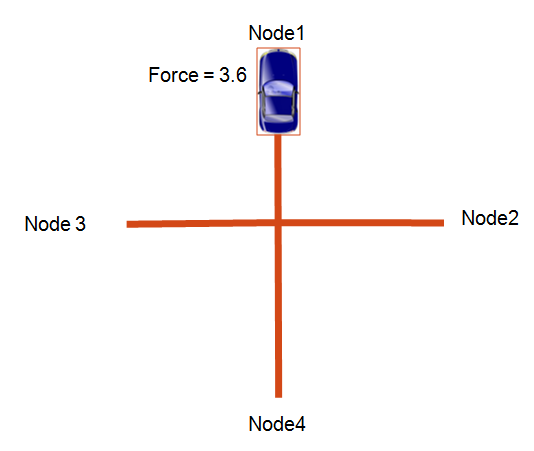
1. Select a block with the highest force value.
2. Add the node with maximum force value in the output list of nodes
3. If the corresponding block for this above added node has available slots then stop the algorithm
4. Return the list of output node



**Fig: Consider a road network as shown, let’s say the car starts from the location shown**

# 

**Fig: The algorithm calculates force value for each of the adjacent block based on current block using the formula**



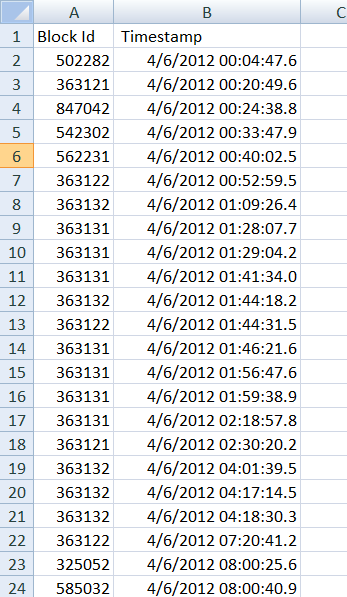
**Fig: The algorithm selects a straight path as it has the highest force value, the car travels till the other end of the block and if a parking slot is available the algorithm stops and returns the list of nodes travelled.**

You can find the full implementation of the algorithms in form of a web application in the below link:

<https://bitbucket.org/c0deForLife/spatiotemporalresourcesearch_web>

# Experiment and Simulations

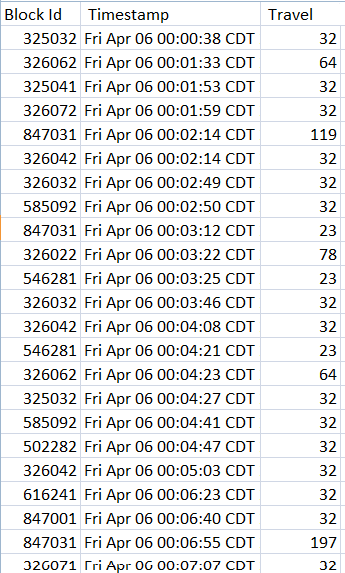
We have analyzed the results for 60 different blocks for 100 different randomly selected time stamps. We have provided the simulation data in Simulations.csv. The below image is a representation of the simulation data that we have used for all the three algorithms. Each algorithm has been run for various congestion levels – 0%, 20%, 40%, 60% and 80%. Each congestion level has 6000 simulation output. Thus, resulting in overall 90,000 simulations for three algorithms.



**Simulation Procedure:**

1. Set the congestion value in Congestion.properties file
2. Go to SpatioTemporal.java, we have a main method to run the simulations in an automated way
3. Initialize an appropriate value for the variable “localDataFilePath” with location of “data files” folder.
4. Configure the project in eclipse to redirect standard output to a csv file.
5. There are code snippets commented for each algorithm. Un-comment a snippet you wish to run and then run the code as a Java application.
6. The output can be viewed in the output csv file.

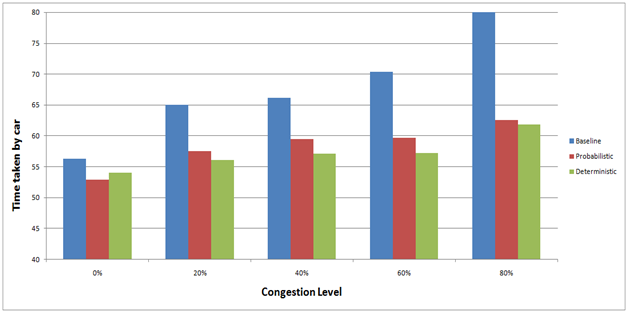
Each algorithm generates the travel time that the car will take to locate a parking spot. We take average of these travel time at each congestion level for each algorithm to plot the graph.



**Fig.: Sample result for the simulations**

# Result and Analysis

Based upon the simulation results we have plotted a graph of time taken by the car to obtain the parking slot versus the level of congestion by different approaches.



## Analysis for baseline approach:

Baseline approach has the highest amount of time needed to find a parking slot even with no congestion, this is because out of all the algorithms, baseline has minimal information available.

As we see, for the baseline approach when we increase the congestion level, the amount of travel time goes on increasing. The reason behind that is with the increase in congestion level, the number of available parking slots reduces because of which the car keep looking for the parking slot leading to increase in the travel time.

## Analysis for Probabilistic approach:

Probabilistic approach has lesser amount of time needed to find a parking slot as compared to baseline when there is no congestion. The reason is that since this approach has a predictive model which acts as an indicator of whether a parking slot would be available or not. Thus, it skips a few extra blocks that baseline might travel in search of parking slot and reducing the travel time.

However, we see that as the congestion level increases the probabilistic starts taking more time, since it has keep re-routing the car at every node when it doesn’t find a free a slot leading to increase in overall travel time.

## Analysis for Deterministic approach:

Deterministic approach takes least amount of time needed to find a parking slot as compared to baseline and probabilistic when there is no congestion. The reason is that since this approach has an access to real time information about availability of parking slot. Thus, it considers only those blocks that surely has a parking slot and in turn avoids unnecessary travel to other blocks. Moreover, if all the blocks have same number of available blocks it picks nearest block thus, reducing the travel time.

However, we see that as the congestion level increases the deterministic also starts taking more time.

# Future Work

1. If we can access the real time traffic information we can analyze the performance of our algorithm in a better way
2. For our project, we have an assumption that every vehicle moves 10 miles/hour, which may not true in the real world scenario.
3. The predictive model was generated with the given data which was for a range of one month. If more historical data is made available, the predictive model can be more accurate and give better results.

# Conclusion

In this project, our goal was to evaluate the parking under different circumstances. We have devised various algorithms that can be used to study the parking problem. For the deterministic approach, we implemented a greedy approach which tries to find the available parking slot based on highest force value.

For the historical interpretation, we generated a predictive model, which gave expected available slots for a given timeframe.

Finally, we have a blind uninformed search which uses data not more than the geographic location of the blocks and the number of operational slots of these blocks.

From the data analysis, we can conclude that having deterministic information is better for faster parking but at the expense of installation of expensive sensors whereas, the probabilistic approach could be good model for finding the availability. However, the accuracy is proportional to the data used to generate the predictive model. The uninformed can be used as the last resort to direct the user to the blocks for manual inspection of the availability.

# References

1. https://maps.googleapis.com/maps/api/directions/
2. http://www.w3schools.com/googleapi/google\_maps\_overlays.asp
3. San Francisco Park website: http://sfpark.org/