Decision Making for Parking

Aniruddha Maiti (email : tuf86648@temple.edu)

April 28, 2016

Abstract

.

1 Introduction

Finding the strategy to park based on the availability of open parking spot is important because it can save time, energy and manpower. There exist current efforts to develop algorithms which can provide effective strategy to a driver. In [1], [2], [3] and [4], strategies are proposed that allow drivers to reserve an available parking spot. In [5], [6] and [7], several agent based parking choice models were discussed. This work aims to develop a parking strategy in a scenario where the probability of the availability of an open parking spot of a given street is known but this knowledge is derived from a noisy dataset and thus not reliable. This project aims to incorporate the noisy information in the search strategy.

1.1 Problem Statement

Given any parking dataset containing historical parking information along the street side road segments and additional road network information (map), a problem can be formulated as follows :
Given current location of the car, the destination, and geometry of the road segments, identify the best parking spot.
If we do not consider the parking dataset, this problem reduces to a graph traversal problem. But in presence of a noisy dataset we need some methodology to extract information from this dataset and incorporate that information in the search strategy. This work aims to design a methodology which can incorporate additional information from the noisy dataset in the search algorithm.

2 Data Set

2.1 Data Set description

In this work, a total of five datasets with a total of 510,214 instances spanning over the year 2013 and 2014 are considered. The total number of samples and the time duration between which the data is collected is shown in Table-1.

The information of individual street segment consists of the name, identification number of the street, latitude and longitude of start and end location of streets, total number of available parking spots and total number of occupied parking spot. The total number of occupied parking spot and the total number of allowed parking spot can give us idea about the availability of the parking space in a particular street. Figure-1 and 1 are examples of such time series dataset.

Table 1: Dataset Description

Dataset No	**Start Date** (yyyy-mm-dd)	**End Date** (yyyy-mm-dd)	**# Samples**
1.	2013-07-28	2013-09-29	84624
2.	2013-09-29	2013-12-11	102780
3.	2013-12-11	2014-02-06	79743
4.	2014-02-06	2014-06-13	176503
5.	2014-06-13	2014-07-30	66564

† Time between two consecutive responses are different in each dataset

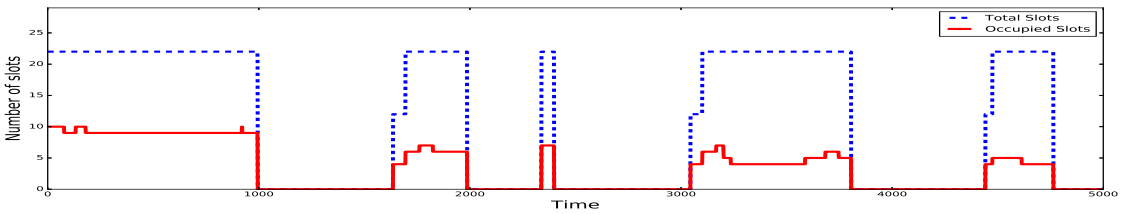


Figure 1: Number of allowed parking slots and the occupancy level of a selected street segment

Given this large amount of data, it might not be feasible to analyse the entire dataset before incorporating the information in the search strategy. Instead, some representative sections can be analysed and this small sampled dataset can be used to implement a search strategy. A total of 79 street segments are selected in this step for further analysis. In Figure-3, the all available street segments i and the street segments in the representative section are shown.

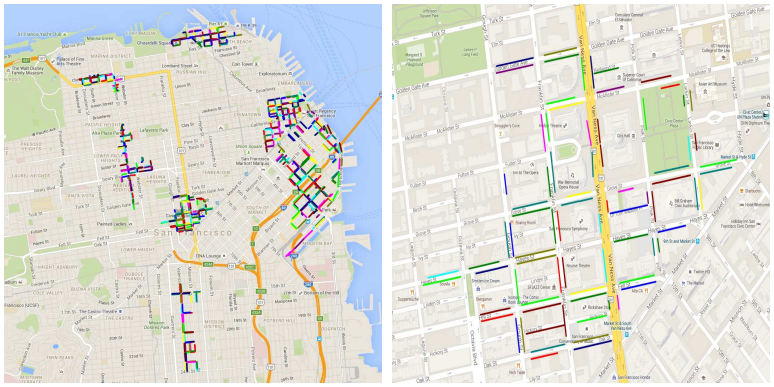


Figure 3: Overlay of the street segments for which information is available

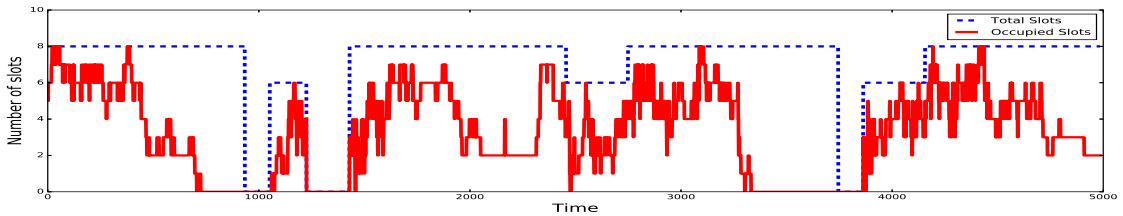


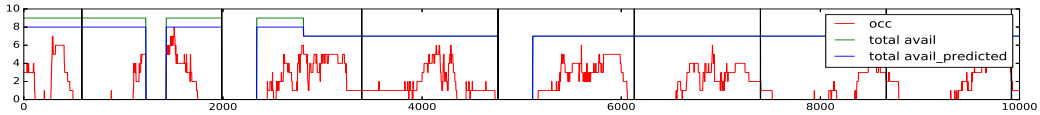
Figure 2: Number of allowed parking slots and the occupancy level of a another selected street segment

2.2 Data Cleaning

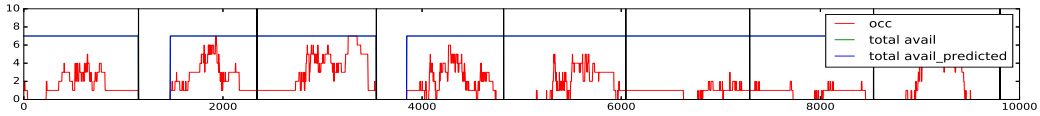
The information from the representative section of the dataset is extracted and processed. The parking data contains the occupancy information at different locations. The information is collected from sensors installed at parking locations. At a any given time, a fraction of the total number of sensors malfunctions. So, the available data set is noisy in some sense. A suitable algorithm needs to be developed in order to identify the sensors that are malfunctioning at a given time. Although majority of the sensors works properly in most of the street segment, some streets are identified where a few sensor malfunctions. A major challenge is to identify those malfunctioning sensors and exclude the faulty information from this study. Two simple assumptions are made in the data cleaning process.

- 1. Malfunctioning sensors not replaced in later stage, i.e. if three sensors started to malfunction after one weeks of installation, the number of malfunctioning sensors will be greater than or equal to three at the end time.
- 2. As the region concerned is very busy area, it is expected that the occupancy level will reach maximum value at least once in a week. This assumption is necessary because some street segments are found in the dataset for which the occupancy level never reaches to the maximum level in the entire time span of two months for which the data is collected. Given the place concerned here is down town San Francisco, one of the busiest location of the country, it is highly unlikely that all the parking spot of a given street segment will not be in the occupied state in the span of two months. In this case, the number of properly functioning sensors are taken to be equal to the highest number of occupied spot in the entire time span over which the data is collected. In some streets the number of allowed parking spot vary with time, in this case the same approach is taken for all the allowed parking levels.

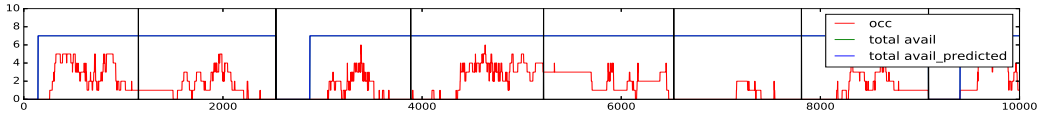
In Figure-4, 6 and 5, examples of the cleaned data is shown. Days are marked with a vertical black line. The red green and blue lines represent the number of currently occupied spot, the number of total available spot, and the predicted number of properly functioning sensors respectively.



(a)



(b)



(c)

Figure 4: Street ID 411002 predicted number of working sensors

It can be observed in Figure-4 that one sensor is predicted to be in non-performing state after a couple of days. This is so because the total occupancy level never reaches to the maximum number of allowed parking spot (9 in this case) for the rest of the time during which the data is collected.

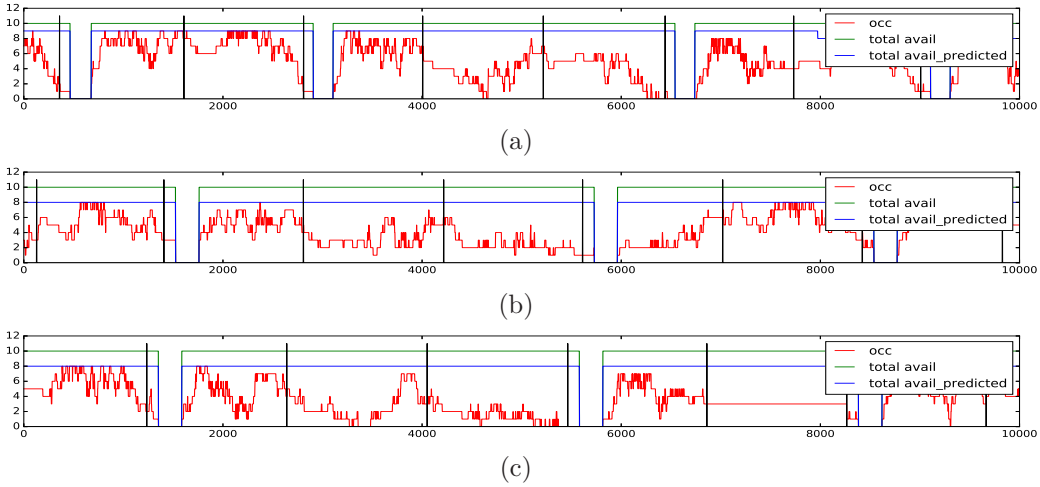


Figure 5: Street ID 411059 predicted number of working sensors

In Figure-5, it can be seen that one sensor was malfunctiong at the beginning, then after a few days one additional sensor started to malfunction.

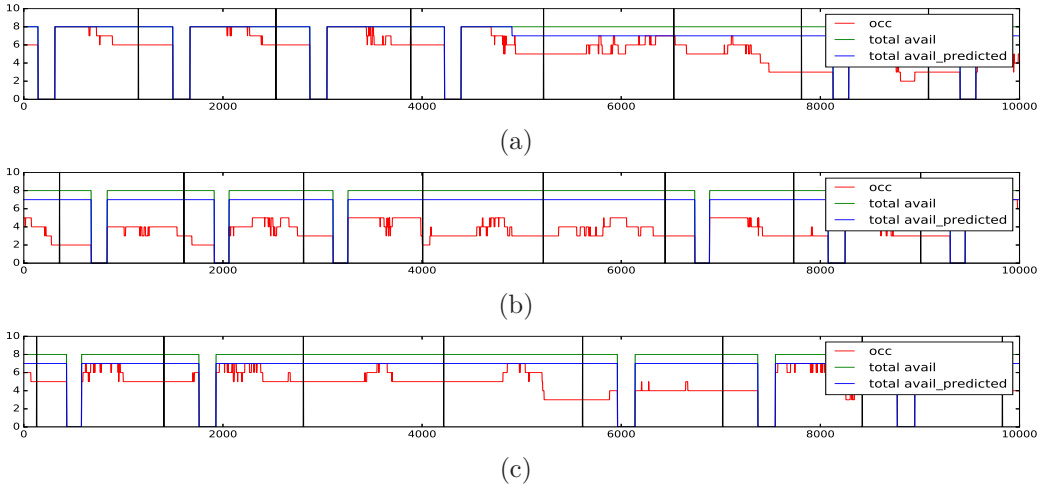


Figure 6: Street ID 612032 predicted number of working sensors

In Figure-6, the similar result can be observed. One sensor started malfunctioning after few days. Then for a long time the occupancy level does not reach the maximum (Figure-6c), but it again reached the maximum value of in Figure-6c. So it is not necessary to assume the any additional sensor died in that period. From Figure-4, 6 and 5, it can be concluded that the proposed method to identify the malfunctioning sensors is able to clean the data to some extent.

3 Proposed Search Algorithm

3.1 Search Algorithm

Given the geometry of the road segments, any suitable graph traversal algorithm can be applied to such problems. In this work A^* search algorithm is is used. A^* search uses heuristic in the search process. It is suitable in this specific problem because the uncertainty in the dataset can be incorporated in the heuristic of A^* search algorithm. In Algorithm-, the pseudo code of the algorithm is described.

Algorithm A^* search algorithm

Input : Graph : G, StartNode : s, EndNode : g
1. Set PriorityQueue Q as Empty
2. $Q.enqueue(s)$
3. $Parent(s) = Null$
4. $Visited(s) = True$
5. $gScore(s) = 0$
6. $fScore(s) = 0$

while Q is not Empty **do**
  $u = Q.dequeue()$
  $Visited(u) = True$
  if $u == g$ **then**
    Break
  end if
  for Each neighbour v of u **do**
    if $u == g$ **then**
      Break
    end if
    $newgScore := gScore[current] + dist(u, v)$
    if $Visited(u)! = True$ or $newgScore < gScore[v]$ **then**
      $gScore[v] = new_gScore$
      $Parent(v) = u$
      $fScore[v] := gScore[v] + heuristic(v, g)$
      $Q.enqueue(v, fScore[v])$
    end if
  end for
end while
return List $Parent$

Here, the metric $gScore$ represents the distance that is already covered to reach to the current node and the metric $fScore$ represents the tentative distance remains to be covered. Given the graph of the road network, the uncertainty from the dataset can be incorporated into the $fScore$. Two things should be considered here, parking availability and walking distance from Parking spot to destination. These information is available in the dataset. The measure of walking distance can be measured using latitude and longitude information accurately. But the probability that a parking space would be available is uncertain. NARS (Non-Axiomatic Reasoning System) like concept can be used in such a scenario [8] [9]. We have some probability measure of a parking spot being available which can be calculated from the dataset but the confidence on this probability measure depends on how much data we have and the quality of the data. Now confidence can be measured directly using NARS ($w/(w+k)$ where w is the number of experiments and k is some constant), but in this particular problem the time series of all the roads are of same length. So number of samples are same for each case. To incorporate the confidence measure, the variance of the data is used instead. If the occupancy level fluctuates more then there is less certainty of an open parking slot. The heuristic used in this work is presented in eq.(1).

$$heuristic(v, g) = k_1 * (1/prob) + k_2 * (1/confidence) + k_3 * walk(curr, target) \quad (1)$$

Both low probability and low confidence measure increases the tentative distance and A^* algorithm will be more unlikely to produce that path as a solution. k_1, k_2 and k_3 are three constants that can control the trade off between parking availability and Walking distance.

3.2 Implementation

Google Map APIs have in built search algorithms that can produce best path on google map given start and end points. But the proposed method uses parking availability information that could not be incorporated into Google Map API. So Google Map is not used in this project. The data processing and implementation of the proposed method is done using Python programming language. The python implementation of the proposed algorithm is shown below.

```
def A_STAR(graph, start, target) :

    param1=10
    param2=15
    param3=10000
    AStarPriorityQueue=PriorityQueue()
    AStarPriorityQueue.insert(start, 0)
    parent_list={}
    accumulativeCost={}
    parent_list[start]=None
    accumulativeCost[start]=0

    while not AStarPriorityQueue.isEmpty() :
        curr=AStarPriorityQueue.remove()

        if curr == target :
            break
        for indx, child in enumerate(graph.neighbors(curr)[0]) :
            updated_cost=accumulativeCost[curr] + graph.neighbors(curr)[1][indx]

            if child not in accumulativeCost or updated_cost < accumulativeCost[child] :
                accumulativeCost[child]=updated_cost
                parent_list[child]=curr

                prob, confidence=calculateProbConfidence(curr)
                priority=updated_cost + param1 * (1/prob) + param2 * (1/confidence) + param3 * walk(curr, target)
                AStarPriorityQueue.insert(child, priority)
                parent_list[child]=curr
    return parent_list, accumulativeCost
```

3.2.1 Calculation of Parking availability

The first two terms in eq. (1) is calculated from the dataset. Occupancy probability is the ratio of average occupancy and average number of maximum allowed parking space in a given street segment. The confidence is the variance of the data. Instead of averaging the data, specific time windows can be used to measure the probability and confidence because

often the parking availability depends on the time of the day. The Python implementation is shown bellow.

```
# This is the used heuristic function for A start
def calculateProbConfidence(street1, street2) :
        #### Open Data File
    f1 = open('Selected_Street_LatLong.pckl', 'rb')
    Selected_Steer_Lat_Long = pickle.load(f1)
    f1.close()

    sample_db=lite.connect('ParkData_Selected_Clean.db')
    streetNum1=street1
    strName1=Selected_Steer_Lat_Long[streetNum1][0]
    strName1= "'B" + strName1 + "'"

    streetData1=[]

    with sample_db:

            #query1="select * from ParkDataTable where StreetId='B411011' and Date between '09-24' and
'09-29'"
            query1="select_*_from_ParkDataTable_where_StreetId=_" + strName1 + "and_Date='09-07'__order_by_Time_asc"
            cur= sample_db.cursor()
            for row in cur.execute(query1):
                streetData1.append(row)

            occ1=np.array([a[3] for a in streetData1])
            tavail11=np.array([a[4] for a in streetData1])

            prob=1-np.mean(occ1)/np.mean(tavail11)
            confidence=np.var(occ1)
    return prob, confidence
```

3.2.2 Calculation of Walking Distance

The probability and the confidence measure takes care of the Parking availability part of the heuristic. The second part is the walking distance from parking location to the destination. This is essential because parking distance should not be far from the actual destination. To incorporate this information, the latitude and longitude information of the the streets are used. The implementation is shown below.

```
def walk(street1, street2) :
        #### Open Data File
    f1 = open('Selected_Street_LatLong.pckl', 'rb')
    Selected_Steer_Lat_Long = pickle.load(f1)
    f1.close()

    streetNum1=street1
    lat1=(Selected_Steer_Lat_Long[streetNum1][2] -Selected_Steer_Lat_Long[streetNum1][4])/2
    long1=(Selected_Steer_Lat_Long[streetNum1][3] -Selected_Steer_Lat_Long[streetNum1][5])/2

    streetNum2=street2
    lat2=(Selected_Steer_Lat_Long[streetNum2][2] -Selected_Steer_Lat_Long[streetNum2][4])/2
    long2=(Selected_Steer_Lat_Long[streetNum2][3] -Selected_Steer_Lat_Long[streetNum2][5])/2

    dist = np.sqrt((lat1-lat2)^2 + (long1-long2)^2)

    return dist
```

4 Conclusion

In this work a small scale analysis is performed in the SFPark parking dataset. A section of the dataset is cleaned and analysed. A small section of the road network graph is built manually. The Data cleaning section of this project was time consuming. Although a small section of the data was used, it took time to manually build the street graph. Given a pre-built graph the system is able to produce reasonable solution if parameters k_1, k_2 and k_3 are tuned properly. But given the small road network graph, the tuning of the parameters is a problematic task. More detailed investigation is required before carrying out the the proposed method on the entire dataset successfully.

References

[1] A. Alhammad, F. Siewe, and A. H. Al-Bayatti, "An infostation-based context-aware on-street parking system," in *Computer Systems and Industrial Informatics (ICCSII), 2012 International Conference on*, pp. 1–6, IEEE, 2012.

[2] H. Wang and W. He, "A reservation-based smart parking system," in *Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on*, pp. 690–695, IEEE, 2011.

[3] E. Kokolaki, M. Karaliopoulos, and I. Stavrakakis, "Leveraging information in parking assistance systems," *Vehicular Technology, IEEE Transactions on*, vol. 62, no. 9, pp. 4309–4317, 2013.

[4] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin, "Parking slot assignment games," in *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 299–308, ACM, 2011.

[5] G. Tasseron, K. Martens, and R. van der Heijden, "The potential impact of vehicle-to-vehicle and sensor-to-vehicle communication in urban parking," *Intelligent Transportation Systems Magazine, IEEE*, vol. 7, no. 2, pp. 22–33, 2015.

[6] I. Benenson, K. Martens, and S. Birfir, "Parkagent: An agent-based model of parking in the city," *Computers, Environment and Urban Systems*, vol. 32, no. 6, pp. 431–439, 2008.

[7] R. Waraich and K. Axhausen, "Agent-based parking choice model," *Transportation Research Record: Journal of the Transportation Research Board*, no. 2319, pp. 39–46, 2012.

[8] P. Wang, "Confidence as higher-order uncertainty.," in *ISIPTA*, pp. 352–361, 2001.

[9] W. Pei, "From nars to a thinking machine," in *Advances in Artificial General Intelligence: Concepts, Architectures and Algorithms: Proceedings of the AGI Workshop*, vol. 157, pp. 75–93, 2006.