This document is used to introduce our experimental data and program codes.

1. Experimental Data

Four datasets are used in this research as follows:

1.1 merge.csv

merge.csv is the main experimental data for parking occupancy prediction that contains 1,456,458 records, 2,026 parking spaces and 109 parking blocks from Apr. 1, 2011 to Oct. 8, 2012. This data contains the following attributes:

- 01. BID: Parking block ID
- 02-03. X, Y: Geographical coordinates of the parking block
- 04. nLots: The number of parking space in the parking block
- 05-06. dow, tod: Day of week and time of day
- 07-08. tpr(C), hp(mm).: temperature and humidity
- 09-18. K1_OCC: Occupancy from the K1 most recent time, where K1 is from 1 to 10.
- 19-28. K2_NN: Occupancy from the K2 nearest blocks, where K2 is from 1 to 10.
- 29-38. K3_RMSE: Occupancy from the K3 most similar blocks by RMSE, where K3 is from 1 to 10.
- 39-48. K4_RMSETOD: Occupancy from the K4 most similar blocks by RMSE and TOD, where K4 is from 1 to 10.
- 49-53. T_K5_OCC: Ground-truth occupancy for the next K5 hour, where K5 is from 1 to 5.
- 54-58. T_K6_OL3: Ground-truth occupancy level using OL3 for the next K6 hour, where K6 is from 1 to 5.
- 59-63. T_K7_OL4: Ground-truth occupancy level using OL4 for the next K7 hour, where K7 is from 1 to 5.
- 64-68. T_K8_OL5: Ground-truth occupancy level using OL5 for the next K8 hour, where K8 is from 1 to 5. (Not used in this research)

1.2 common_109_withXY_title.csv

common_109_withXY_title.csv is used to evaluate the performance of parking allocation strategies. This data records the information of each parking block. It contains 5 attributes:

- 01. BID: Parking block ID
- 02-03. POINT_X, POINT_Y: Geographical coordinates of the parking block
- 04. NumberOfParkingLots: The number of parking space in the parking block
- 05. total_time: The total occupied time of the parking block (Not used in this

research)

1.3 predict.csv

predict.csv is also used to evaluate the performance of parking allocation strategies. This data records the predicted and ground-truth number of parking spaces in each time slot. It contains 6 attributes:

- 01. BID: Parking block ID
- 02. is_week: Weekday or weekend
- 03-04. DOW, TOD: Day of week and time of day
- 05. Pred: The predicted number of parking spaces based on C5.0 classification model
- 06. Truth: The ground-truth number of parking spaces

1.4 test.csv

test.csv is used to simulate the start location and destination of a user based on the default setting for the parking allocation strategies. It contains 6 attributes:

- 01-02. User ID (identification only, not used in this research)
- 03-04. The x and y coordinate of start location
- 05-06. The x and y coordinate of destination

2. Experimental Program Code

The experiment consists of two parts. The first part aims to evaluate the accuracies of parking occupancy prediction models including Naïve Bayes, C5.0 decision tree, random forest and regression model run by the packages ‘e1071’, ‘C50’, ‘randomForest’ and ‘stats’ in R language 3.4.0, respectively. The model with the best performance (C5.0) will be utilized in the second part. The second part evaluates the performance of the parking allocation strategies under various system conditions. The performance evaluations of parking allocation strategies were implemented in Python 3.6 using an AMD Ryzen 7 3.7 GHz computer with 16 GB of memory running in Windows 10.

2.1 Parking Occupancy Prediction

From Section 5.2.1 to 5.2.4, the prediction accuracy and RMSE are evaluated by R language. We write four R programs based on merge.csv to evaluate the prediction performance. If the data cannot be loaded into R program, please add the full file path of merge.csv. Each program contains 4 methods (Method 1: Naive Bayes, Method 2: C5.0, Method 3: Random Forest, and Method 4: Regression), and will output 2 values that mean the classification accuracy and RMSE of predicted parking occupancy rate. Please manually change some parameters in each program:

- Section521.R: Parameter Setting (OL <- 3 or OL<- 4). After setting OL value, the program will generate accuracy and RMSE for 4 methods under various K1 to K4 from 1 to 10.
  - Input: OL variable (3 or 4) and merge.csv.
  - Output: Accuracy and RMSE for 4 methods under various K1 to K4 from 1 to 10.
- Section522.R: Feature Combination. This program doesn't need to set any parameter. The program will automatically generate accuracy and RMSE for 4 methods under various feature combinations.
  - Input: merge.csv.
  - Output: Accuracy and RMSE for 4 methods under various feature combinations.
- Section523.R: Robustness of Prediction Model (mode <-1 or mode <-2). The parameter mode is used to switch training data ratio and k-fold cross validation. After setting mode value, the program will generate accuracy and RMSE for 4 methods under various training data ratios or k-folder cross validation.
  - Input: mode variable (1 or 2) and merge.csv.
  - Output: Accuracy and RMSE for 4 methods under various training data ratios or k-folder cross validation.
- Section524.R: Parking Occupancy Prediction at k Hours Later. This program doesn't need to set any parameter. The program will automatically generate accuracy and RMSE for 4 methods under various k hours later.
  - Input: merge.csv.
  - Output: Accuracy and RMSE for 4 methods under various k hours later.

2.2. Parking Allocation Strategy

From Section 5.3.1 to 5.3.3, the average driving distance, average walking distance and the number of users who failed to find a parking space are evaluated by Python language. These codes can be opened by Jupyiter notebook. To simulate users looking for parking spaces, we first write creat_new_data.ipynb to generate the start location and destination of users.

- creat_new_data.ipynb: In the third block of this program, the function generate_user can be found. The last parameter of generate_user function is used to control the distribution density of users' destinations. In this research, the parameter can be set as 1689, 3378 and 5066. test.csv is generated by creat_new_data.ipynb based on the default setting 3378.
  - Input: Distribution density of user's destinations (1689, 3378 or 5066), common_109_withXY_title.csv and predict.csv.

- Output: test.csv.

Then, we write three programs Normal-IOT.ipynb, matching.ipynb and p_matching.ipynb for Normal, Matching and Prediction-based Matching strategies, respectively. There are three necessary settings:

1. For all experimental evaluations, input test.csv must be inputted to simulate the user parking request. Section 5.3.1 and 5.3.2 can directly use the default file test.csv, but Section 5.3.3 must generate various destination distribution files using creat_new_data.ipynb.
2. For all experimental evaluations, the query time of user parking request must be set. Since merge.csv starts from 00AM Wednesday, the time point 0 means 00AM Wednesday. Hence, if the allocation performance evaluation time is 8AM, the 8th time point should be extracted. In each program, please search the comment #change the query time, then the query time points can be changed in the next line. For example, if the performance evaluation time is 17PM in weekday and weekend, the time point vector should be set as [17, 41, 65, 89, 113, 137, 161].
3. For the Section 5.3.1, there are three collision solutions need to choose including Walking Distance First (WDF), Driving Distance First (DDF) and Transfer Distance First (TDF). Hence, please search the comment #choose the strategy, then one of the following collision solutions can be chosen for performance evaluation.

- user.des_nearestblock = nearest_block(key, dist(user.Start_X, user.Start_Y, values.point_X, values.point_Y)) #Driving Distance First
- user.des_nearestblock = nearest_block(key, dist(user.des_X, user.des_Y, values.point_X, values.point_Y)) #Walking Distance First
- user.des_nearestblock = nearest_block(key, dist(user.des_X, user.des_Y, values.point_X, values.point_Y) + dist(user.Start_X, user.Start_Y, values.point_X, values.point_Y)) #Transfer Distance First

Finally, we summarize all settings in each section.

- Section 5.3.1:
  - The user query is from test.csv.
  - 17PM in everyday, the time point vector is [17, 41, 65, 89, 113, 137, 161].
  - Collision solution can be changed according to the experimental requirement.
- Section 5.3.2:
  - The user query is from test.csv.
  - 8AM in weekday, the time point vector is [8, 32, 56, 128, 152].
  - 11AM in weekday, the time point vector is [11, 35, 59, 131, 155].
  - 14PM in weekday, the time point vector is [14, 38, 62, 134, 158].

  - 17PM in weekday, the time point vector is [17, 41, 65, 137, 161].
  - 20PM in weekday, the time point vector is [20, 44, 68, 140, 164].
  - 8AM in weekend, the time point vector is [80, 104].
  - 11AM in weekend, the time point vector is [83, 107].
  - 14PM in weekend, the time point vector is [86, 110].
  - 17PM in weekend, the time point vector is [89, 113].
  - 20PM in weekend, the time point vector is [92, 116].
  - Collision solution is set as Walking Distance First.
- Section 5.3.3:
  - The user query needs to be generated by creat_new_data.ipynb.
  - 17PM in everyday, the time point vector is [17, 41, 65, 89, 113, 137, 161].
  - Collision solution is set as Walking Distance First.