IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 1, 2021, accepted November 19, 2021, date of publication November 29, 2021,
date of current version December 7, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3131229*

# SCP-Tree: Finding Multiple Nearest Parking Spots With Minimal Group Travel Cost

**JINE TANG[1], YUPENG WANG[2,3], WEIJING LIU[4], XILING LUO[3,5], AND ZHANGBING ZHOU[6,7]**

[1]School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China
[2]School of Electronics and Information Engineering, Beihang University, Beijing 100191, China
[3]Comprehensive Transportation Big-Data Research Center, Hangzhou Innovation Research Institute of Beihang University, Hangzhou 310051, China
[4]Tianjin Key Laboratory of Aerospace Intelligent Equipment Technology, Tianjin Institute of Aerospace Mechanical and Electrical Equipment, Tianjin 300458, China
[5]Research Institute for Frontier Science, Beihang University, Beijing 100191, China
[6]School of Information Engineering, China University of Geosciences, Beijing 100083, China
[7]Computer Science Department, TELECOM SudParis, 91000 Evry, France

Corresponding author: Yupeng Wang (wangyup@buaa.edu.cn)

This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1601200, in part by the Open Project Program of Tianjin Key Laboratory of Aerospace Intelligent Equipment Technology, Tianjin Institute of Aerospace Mechanical and Electrical Equipment under Grant ZNZB-2021-01, and in part by the National Natural Science Foundation of China under Grant 61702232.

**ABSTRACT** Finding the nearest parking location in road networks is one of the most commonly faced challenges in everyday life of green transportation. A main challenge faced by the state-of-the-art existing parking allocation methods is to optimally offer the nearest parking location for a group of $m$ users at the cost of minimal overall traveling time to ensure the traffic and environmental sustainability. In this article, we model it as a Multiple Nearest Parking Location Allocation (MNPLA) problem, and devise a spatial index tree, called SCP-tree, to accelerate the nearest parking location allocation within the users' time constraints. During the search process in SCP-tree, we build a pruning strategy relevant to the Geographical Preference Estimation, travel time and parking capacity to determine which branch to visit so that the search accuracy can be improved. Considering the users' behaviors are often impacted by the geographical location and some personalized attribute information, we set the user priority based on them to help the parking officer determine the allocation sequence. We evaluate our allocation scheme using large real-world dataset with on-street parking sensor data, and extensive experimental results reveal (i) a minimum improvement of 15.9%, 1.4%, 96.9%, 160% in parking allocation time, average traveling time, I/O cost and service utility compared to the progressive methods, and (ii) a minimum improvement of 8.9%, 11.1%, 78.2%, 714% in parking allocation time, average traveling time, I/O cost and service utility compared to the baseline methods.

**INDEX TERMS** Green transportation, multiple nearest parking location allocation, minimal overall traveling time, SCP-tree, geographical preference estimation, user priority, parking sensor data.

## I. INTRODUCTION

In order to reduce energy uses and cut emissions that contribute to the climate and environment change, parking allocation is an important aspect required to be considered in green vehicular transportation, as search for parking by drivers is a significant contributor to the congestion in cities and thus also generates a lot of greenhouse gas emissions [1], [2]. Usually, drivers spend a great deal of time on searching for parking and on leaving sooner or later due to the anticipated parking and congestion problems. Moreover, the unplanned parking allocation can make the travellers, such as coming later, be guided to an unsatisfactory parking location. It is not easy to maximize the benefits of all the mobile users for finding the available parking locations to ensure the sustainability of urban environments [3], which is viewed as a change that improves the quality of life and conserves the time and natural resources.

In essence, vehicle parking can be viewed as a continuous query submitted by mobile users to obtain the available parking locations. Several existing works [4]–[7] compute for parking locations with different costs, and propose some

The associate editor coordinating the review of this manuscript and approving it for publication was M. Shamim Kaiser.

strategies to allocate, such as nash equilibrium, gravity based game-theoretic framework [4]–[6], bipartite matching based natural posted price mechanisms [7], exclusive closest pairs join operation [8] and so on. Each user achieves the preferred parking location by calculating and comparing the prices for available parking locations based on a given selection strategy. However, all these studies have not considered any time and personalized attribute constraint to determine the preference allocation order that (i) helps a centralized parking allocation system to manage the parking locations, and (ii) maximizes the benefits for all users.

In real mobile application scenarios, a mobile user usually wants to know which parking location to visit in order to minimize the time and overall travel distance, specifically the driving distance from the departing location to the parking spot and the walking distance from the parking to the destination. The parking locations are always selected as closer as possible to the destinations since walking may consume much time [9]. However, parking is competitive in nature because after making a choice to visit a particular available parking location, the success in obtaining that parking location will depend on if any other vehicles having a preference for that parking location also make the same choice. Considering this problem, we assume that the centralized parking allocation system has a preference order for each user with different personalized attributes and departing locations, which are deemed as two major factors in modeling users' preferences for selecting parking locations. Meanwhile, the distance from the parking locations to the destinations should be as shorter as possible, which also has an important impact on the traveling time.

Generally, in this article, the objective of the system designer is to set up the user priority in terms of personalized attributes and departing locations, and model the geographical preference to prompt the most effective allocation for available parking locations so that it can serve the maximum number of users within some driving time constraint. At the same time, it should reduce the walking distance from the parking locations to the destination locations as much as possible. Next, we present an example to demonstrate how to allocate the parking locations according to the rules relevant to distance, personalized attributes as well as destinations. A comparison about the allocation result with vehicle-slot (vs) pricing strategy is also given to motivate our work.

*Example 1:* Fig. 1 illustrates a motivating example of analysis with 8 departing locations and destination locations (see Fig. 1(a) and (c)) as well as 12 parking locations (see Fig. 1(b)). Each user $u_i$ has a profile tuple $\{l_{di}, age, occupation, driving\ age\}$, where $l_{di}$ is the departing location of $u_i$. The attributes afore-mentioned (e.g., age, occupation, driving age) are examples of some personalized attributes of $u_i$. Note that, this list is for illustration, and any other attribute can be similarly accommodated to our approach. The parking officer analyzes the four dominated factors according to the following classification mechanism:

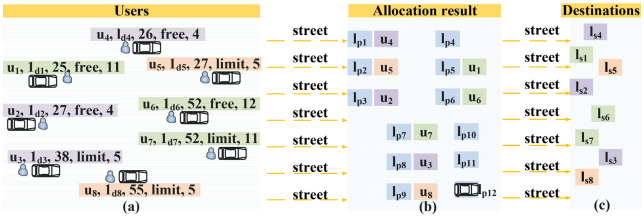



**FIGURE 1.** An example of parking allocation.

(1) In this example, the destinations $l_{s1}, l_{s2}, l_{s4}, l_{s5}, l_{s6}$ are nearer to parking locations $l_{pi}$ ($i \in [1, 6]$), while the destinations $l_{s3}, l_{s7}, l_{s8}$ are nearer to parking locations $l_{pi}$ ($i \in [7, 12]$). The officer allocates the $l_{pi}$ ($i \in [1, 6]$) to $u_1, u_2, u_4, u_5, u_6$ and $l_{pi}$ ($i \in [7, 12]$) to $u_3, u_7, u_8$, respectively according to the users' departing locations and personalized attributes.

(2) For the user whose age is over 50, and occupation is time limit, as well as driving age is less than 6 years, he/she can be allocated a *near* parking location, such as $u_7$, $u_8$.

(3) For the user whose age is below 30, and occupation is time limit, as well as driving age is less than 6 years, he/she can be allocated a *near* parking location, such as $u_5$.

(4) For the user whose age is over 50, and occupation is time free, as well as driving age is more than 10 years, he/she can be allocated a *farther* parking location, such as $u_6$.

(5) For the user whose age is below 30, and occupation is time free, as well as driving age is more than 10 years, he/she can be allocated a *farther* parking location, such as $u_1$.

(6) If more than one user are in the above classifications, they will be competitive in terms of the distance from the departing location to the parking location. The other users, such as $u_2$, $u_3$, and $u_4$, can be allocated mainly considering the departing locations.

Note that the above mentioned strategies are implicit personalization for the parking spot allocation. If there is an explicit preference or constraint for a parking spot, it can be easily accommodated in our approach. If we are to run the above example with vs pricing strategy, assume that it needs a total \$-cost of 60 cents on average driving 1 mile. At this case, the pricing authority can assign $u_1$ a greatly large quantity for $l_{p2}$ and 0 for $l_{p5}$. For $u_5$, the pricing authority will assign the prices as 0.5 miles for $l_{p5}$ and a greatly large quantity for $l_{p2}$. The cost that $u_5$ will have to pay for traveling 0.5 miles converts to 20 cents. Thus, by setting vs prices, the pricing authority stimulates $u_1$ and $u_5$ to choose $l_{p2}$ and $l_{p5}$ respectively according to the distance minimization assignment. By paying this 20 cents price, $u_5$ is not necessary to pay the same amount of both \$-cost and driving distance. $u_1$ pays more due to the disadvantages brought, but the pricing authority can compensate her from the money collected from $u_5$. The user cannot arbitrarily change her parking location and improve her cost. Thus, it is worthwhile for the drivers to travel a shorter distance in total by the pricing authority.

This paper is motivated by the fact that although there is an authority to price on-street parking locations considering

both the price and driving distance, the time and personalized attribute constraints still have a large influence on the traveling cost. Thus, we study the question: how to allocate the parking locations to users by considering the dominated constraint factors that have an influence on the overall traveling time? The answer is from the allocation result in Fig. 1(b), where the officer allocates $l_{p5}$, $l_{p2}$ to $u_1$, $u_5$ respectively by considering the users' occupation and driving age. The whole allocation is reasonable since these users belonging to (2) and (3) tend to drive slowly within the limited time, while these users belonging to (4) and (5) tend to drive faster without a time limit. Meanwhile, the traveling time from the parking slot to the users' destinations is within the walking time constraint.

In the following, we list the contributions of this article.

- We propose a new spatial index SCP-tree to represent the parking locations for efficient real-time parking allocation. The branch node regions that are within the walking time constraint to the users' destinations are first positioned. Then, the Lower Bounding Driving Time and parking capacity placed on each index node are used to prune the nodes that cannot satisfy the driving time constraint or have no available parking locations.
- We propose User Priority setting based on the personalized attribute and distance, combined with the estimation of geographical location proximity developed for each user, to maximize the benefit of all users. As for a set of users having the similar personalized attributes and closer departing locations, they can be combined into one group for further allocation, especially in peak hours when a large number of users request for parking.
- We evaluate our proposed method by the allocation time, traveling time, I/O cost and service utility metrics using a large public dataset provided by the local city council. The experimental results show our proposed method has a significantly efficient allocation performance.

The remainder of this article is organized as follows. In Section II, we present a review of related works. In Section III, we present the motivation, and relevant problem definition. In the following, we present the construction process of SCP-tree and parking location allocation scheme in Section IV. Section V contains the evaluation results. Section VI concludes this article.

## II. RELATED WORK

### A. PARKING ALLOCATION

The notable work, such as optimal system assignments, Nash equilibrium, the price of anarchy, as well as ridesharing relevant to the parking allocation has been studied in the field of transportation applications. U *et al.* [10] propose a capacity constrained assignment to identify the allocation with the optimal overall quality by employing novel edge-pruning strategies. Ayala *et al.* [4], [5] analyze the parking situations in competitive simulations through a game-theoretic framework and present the relevant algorithms to choose ideal parking locations. Ayala *et al.* [6] also propose a vehicle-slot pricing mechanism considering algorithmic game theory, where drivers may pay different prices for the same parking location. Meir *et al.* [7] study MAXDISTANCE and LINEARCOST valuation schemes for parking allocation through setting up connections with the online bipartite matching problem. In recent years, Tian *et al.* [11] propose Noah to perform large scale real-time ridesharing. This system accepts requests in real-time and assigns the request to a taxi which is either roaming or having a pre-engaged route. The assignment aims to guarantee the service constraints of the request in terms of waiting time and detour percentage. Cheng *et al.* [12] propose a utility-aware ridesharing on road networks with the goal of providing optimal rider schedules for vehicles to maximize the overall utility of riders. The focus of it is to assign each rider to a suitable vehicle, subject to the constraint of riders' deadlines of pickup or delivery of riders and the capacity of vehicles. Chen *et al.* [13] develop a price-and-time-aware ridesharing system by offering more options with different pick-up times and prices. The system assumes that the destination of a vehicle is not limited and that it can accommodate any number of rider groups during a trip as long as it satisfies a capacity constraint. Although these works on ridesharing and our work are both related with the allocation problem, they cannot be directly applied to solve our parking allocation problem. This is because (i) these works on ridesharing may require one vehicle to be allocated to multiple riders for completing its trip. The vehicles' number may be less than the riders' number. However, our work requires only one parking location allocated to one user to complete all the users' trips. The parking locations' number requires to be bigger than or equal to the users' number; (ii) these works on ridesharing aim to maximize the riders' utilities and minimize the vehicles' travel distances, while our work aims to minimize all the users' travel time by allocating the reasonable parking locations to users; (iii) these works on ridesharing suffer the constraints of vehicles' capacities, traveling time and riders' deadlines, while our work suffers the constraint of users' departing-destinations, geographical preference to parking location, personalized attributes that may give different traveling time, as well as parking slot' capacity; (iv) these works on ridesharing consider to satisfy the riders' deadlines in the same trip from one vehicle and the acceptable extra detour time from the shortest possible trip duration, while our work considers to satisfy all the users' traveling time in different trips from the departing locations to the parking locations. In general, all these studies have not considered any time and personalized attribute constraint to help the officer manage on-street parking allocation. In our work, we investigate the parking allocation problem in the context of a centralized model, where a centralized parking officer assigns parking locations by satisfying user priorities and maximizing the benefit of all users.

### B. PAIR QUERIES IN SPATIAL DATABASE

The pair queries in spatial database aim to discover some pairs of spatial objects with certain time or distance constraint. Corral *et al.* [14] address the problem of finding the K pairs of spatial objects formed from two data sets that have the K smallest distances between them, where each set is stored in a structure belonging to the R-tree family. However, due to ties of distances, the result of K-Closest Pair of these two point sets may not be unique for a specific pair. Unlike this situation, our parking allocation is certain for the Closest Pair relevant to each departing location. Zhou *et al.* [15] address the problem of querying the historical and spatial range close-pair moving objects, whose distance is closer than a user-specified distance constraint during the time interval TI and within the spatial range SR. Different from this query, parking allocation considers two kinds of static data objects, i.e., departing location and parking location, for the closet pair matching. U *et al.* [8] exploit space partitioning method to resolve the exclusive closest pairs problem, whose real application is the computation of car and parking slot assignments. In addition, joined data sets changing positions and content, and having a capacity constraint are also considered for parking allocation. U *et al.* [16] also propose an efficient algorithm for optimal capacity constrained assignment. Each customer is assigned to at most one provider. Every provider serves no more customers than its capacity. Meanwhile, the sum of Euclidean distances within the assigned provider-customer pairs is minimized. Different from the problem resolved above, our parking allocation focuses on point-to-point departing-parking location matching to maximize the service utility of all users. Chen *et al.* [17] retrieve the pairs with top-k maximal probabilities of being the closest pair given two uncertain datasets in which each spatial object is modeled by a set of sample points. Unlike this dataset, our parking allocation is schemed based on certain departing and parking locations.

### C. RELATED SPATIAL AND SPATIAL-TEXTUAL INDEX

Indexing as grid or tree structures is a popular and effective method to answer many spatial queries like range search or visit the k-nearest neighbors, often under some query constraints. Li *et al.* [18] propose an efficient index called IR-tree, which supports top-k document retrievals leveraging the unified representation of textual and spatial relevances. Choudhury *et al.* [19] extend IR-tree and construct a text-first index structure SIF for batch processing top-k spatial-textual queries with the help of block based inverted file. Zhong *et al.* [20] present a height-balanced and scalable index, namely G-tree, to efficiently support KNN queries and keyword-based KNN queries based on the assembly-based method. Kucuktunc *et al.* [21] introduce a diverse browsing method for diverse k-nearest neighbor searches based on the popular distance browsing feature of R-tree. Tao *et al.* [22] design an inverted index, named SI-index, which extends the conventional inverted index to deal with multidimensional data and perform keyword-augmented nearest neighbor search in time. Cao *et al.* [23] further propose to retrieve a group of spatio-textual objects such that the group's keywords cover the query's keywords, and such that the objects are nearest to the query location and have the smallest inter-object distances, with the help of an IR-tree. In order to provide users with more options, the authors also propose to find the top-k groups covering all the query keywords and rank according to their costs. As the advantages of spatial index in coping with KNN query, we construct a new SCP-tree based on R-tree [24], to represent the parking locations and answer the MNPLA problem (see Definition 1).

## III. PROBLEM FORMULATION

### A. MOTIVATION AND SYSTEM OVERVIEW

The Melbourne Transportation Council sets up in-ground sensor systems around Melbourne City Centre areas [25]. For each car parking area, the sensor can detect parking space availability and report it to the information centre periodically [26]. The system would send the message to the parking officer who is in charge of the parking allocation in some area [27], [28]. At a regular frequent time interval (e.g., 1 sec), the centralised system groups one or multiple new user queries that are received. The parking officer preprocesses some work in advance to analyze the users' personalized attributes and formulates the allocation priority for the users offline. When the users are ready to start their journeys and query for a parking spot, the parking officer will check the parking space availability and assign the parking locations according to the real-time departing locations, priority and destinations. During peak hours, the number of such users is likely to be more than 1, and the parking officer processing such users at a frequent interval as a group will improve the overall performance without any significant increase in the waiting time for response. Table 1 lists the notations used in this article.

### B. PROBLEM DEFINITION

*Definition 1 Multiple Nearest Parking Location Allocation (MNPLA):* Given a set of parking locations $l_p = (l_{p_1}, l_{p_2}, \cdots, l_{p_n})$, a set of users $U = (u_1, u_2, \cdots, u_m)$ with their departing locations $l_d = (l_{d_1}, l_{d_2}, \cdots, l_{d_m})$ and destination locations $l_s = (l_{s_1}, l_{s_2}, \cdots, l_{s_m})$ as well as a time constraint specified for users to reach from the departing

**TABLE 1. Notations and Their Descriptions.**

| Notations | Description |
|---|---|
| KNN | the k nearest neighbors |
| $l_d$ | the departing location |
| $l_s$ | the destination location |
| $l_p$ | the parking location |
| $k$ | the number of cluster centers |
| $M_0$ | the maximum number of child nodes in an index node |
| $m_0$ | the minimum number of child nodes in an index node |
| $f$ | the fanout of an SCP-tree index |
| MBR | the minimum boundary rectangle |
| $T_{lo}$ | the lower bounding driving time |
| $\theta_{td}$ | the driving time constraint |
| $\theta_{tw}$ | the walking time constraint |
| GPE | Geographical Preference Estimation |
| $u_{pr}$ | the priority of user $u$ |

locations to the parking locations (driving time) and from the parking locations to their final destinations (walking time), the MNPLA problem is to retrieve a subset $sl_p \subseteq l_p$ and assign the parking locations in $sl_p$ to mobile users based on (i) Geographical Preference Estimation (see Definition 6), which is used to determine the nearest parking location, (ii) user priority (see Definition 7), which is used to determine the optimal allocation order for minimizing the traveling time, and (iii) destinations, so that the allocation strategy can serve the maximum number of users within the time constraint and under the parking capacity constraint.

## IV. PARKING LOCATION ALLOCATION SCHEME

### A. OVERVIEW OF SCP-TREE

R-tree is a height-balanced index structure that has been widely adopted to dispose different kinds of queries, such as range query, KNN query etc., by pruning unpromising branch nodes. Inspired by this, we aim to devise a similar spatial index on parking locations. To this end, we recursively partition the parking space into sub-regions and construct a hierarchical index tree structure. However, R-tree allows more overlap and coverage among sibling nodes. Thus it calls for an effective method that can provide a single traversal path and less coverage for parking locations, to achieve high query efficiency and low storage cost.

To address this issue, we exploit clustering techniques such that parking locations in the same cluster are closely located, and parking locations in different clusters are far from each other as much as possible. Therefore, parking locations in the dense regions should be put into the same node region to avoid cascading overlap and large coverage, whereas parking locations in the sparse regions can have a simpler sub-tree structure and better operation performance. The SCP-tree organizes parking locations according to such clustering results. Each node of the tree corresponds to a cluster. Each node is associated with a value called node capacity which denotes the number of available parking locations in that node region. As vehicles arrive and leave, the parking space availability of the node region (i.e., the node capacity) increases or decreases accordingly. In SCP-tree, the parent node capacity is the summation of its child nodes' capacity. A single parking location capacity is "1" if it is available while it is "0" if it is unavailable. During query time, a node is visited only when its node capacity is greater than zero. Generally, SCP-tree has less overlap among sibling nodes than that of R-tree and its variants. In addition, due to the adjacent characteristic among parking locations in the same node, the coverage area of nodes is less than that of R-tree. In the following, we formally give the properties of SCP-tree:

- The root node is the entire MBR region covering all the parking locations.
- Each node stores a lower bounding driving time. It is computed by the minimum value from the driving time distances between all the departing locations of social-life-work-related users who stop in the locations indexed in its subtree and the parking locations of itself by the end of the query time (see Definition 2 as detailed explanation). This property is used to effectively do pruning the subtrees not within the driving time constraint.
- Each node has a parking capacity. It is computed by summarizing the parking availability of its child nodes. This property is used to prune the subtrees that currently do not have available parking locations.
- The interior nodes are the cluster regions where the number of parking locations is larger than a threshold, $M_0$. Similar to the R-tree, the maximum number of the child nodes of an interior node is set to this value, $M_0$.
- The leaf nodes are the clusters where the number of parking locations is between $m_0$ and $M_0$. We also set $m_0$ as the minimum number of parking locations in a leaf node similar to the R-tree.
- A parking location is contained in only one upper parent node, thus the traversal path to a parking location during the query time is unique. Additionally, the parking location capacity is "1" if it is available while it is "0" if it is unavailable.
- Unlike R-tree, SCP-tree is often an unbalanced tree. Thus, the leaf nodes are generally not at the same level.

Fig. 3 gives an example of SCP-tree constructed from the parking locations shown in Fig. 2 by setting $M_0$ as 4. Each non-leaf node corresponds to a sub-region of $R_i$ ($i \in [0, 8]$). For example, the node $R_4$ corresponds to the upper-right sub-region shown in Fig. 2, which includes parking locations $l_{p5}, l_{p6}, l_{p7}, l_{p8}$.

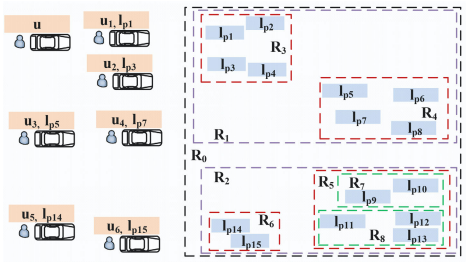



**FIGURE 2.** An example of hierarchical parking cluster.

### B. SCP-TREE CONSTRUCTION

In this article, we adopt the classic $K$-means [29] algorithm for clustering parking locations. Roughly, this method selects $k$ parking locations as initial cluster centers. Then the distances between the parking locations and the cluster centers are computed, and each parking location is assigned to a cluster when the distance between this parking location and the center of the corresponding cluster is minimum. Then the center of the cluster is updated accordingly. The clustering procedure terminates when no update occurs to the cluster centers. Usually, the number of parking locations is quite larger than $M_0$ (i.e., $n_{l_p} \gg M_0$) before clustering. As a result, the parking locations are divided into clusters.

Algorithm 1 presents the pseudo-code of our SCP-tree construction procedure. We first select $k$ ($m_0 \leq k \leq M_0$, $m_0 = M_0/2$) parking locations as the initial cluster centers,

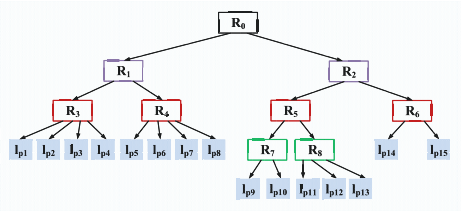



**FIGURE 3.** The SCP-tree constructed corresponding to Fig. 2.

**Algorithm 1** SCPTreeBuilt

**Require:** : $S_{l_p}$: the set of parking locations
**Ensure:** : $nd_{rt}$: the root node of constructed SCP-tree
1: $k_1 \leftarrow$ the number of parking locations in $S_{l_p}$
2: **if** $k_1 > M_0$ **then**
3: $S_{cls} \leftarrow$ select $k$ initial cluster centers ($m_0 \le k \le M_0$, $m_0 = M_0/2$), and cluster parking locations using K-means method, with each cluster containing $k_0$ ($k_0 \ge m_0$) parking locations
4: **foreach** $\vartheta \in S_{cls}$ **do**
5: $k_1 \leftarrow$ the number of parking locations in $\vartheta$
6: **if** $m_0 \le k_1 \le M_0$ **then**
7: set $\vartheta$ as a leaf node of $nd_{rt}$
8: $\vartheta.T_{lo} \leftarrow$ the minimum time value computed by Definition 2
9: **else**
10: $nd_{cld} \leftarrow$ SCPTreeBuilt(parking locations in $\vartheta$)
11: set $nd_{cld}$ as a child node of $nd_{rt}$
12: $nd_{cld}.T_{lo} \leftarrow$ the minimum time value computed by Definition 2
13: **end if**
14: **end for**
15: **end if**

and apply the $K$-means method to group parking locations into $k$ clusters (line 3), with each cluster containing $k_0$ ($k_0 \ge m_0$) parking locations. If the number of parking locations in a sub-cluster is within $m_0$ and $M_0$, this sub-cluster is set as a leaf node of SCP-tree (lines 6-7). When a sub-cluster covers parking locations in the dense regions, the number of parking locations in that cluster is inevitably larger than the maximum value (i.e., $M_0$). We call such cluster as an overflow cluster, which needs to be re-clustered to form smaller clusters (lines 10). Additionally, for each child, a node stores a lower bounding driving time $T_{lo}$, constructed by selecting the minimum value at each time point among all the parking locations indexed therein (line 8, 12) (see Definition 2). For each node, it also has a parking availability capacity, which is defined in Definition 4.

Recall that the relevant users that stop in the parking locations indexed in the subtree covering region $R_{st}$ by the end of the query time $t_0$ are independent for different query users with specific personalized attributes. To obtain a tight lower bounding driving time, we need the users having the similar social-life-work-related attributes with the query user $u$ as large as possible since (i) the social-life-work-related users usually have a closer social connection, similar check-in activities as well as similar travel routine recommendation [30]; (ii) the users who have a closer social connection may have better trust in terms of their travel recommendation; (iii) the users who show more similar check-in behaviors should have more similar tastes with the active user, thus suggestions from those users are more worthy.

*Definition 2 (Lower Bounding Driving Time(LBDT)):* Let $L_d = \{l_{d_1}, l_{d_2}, \cdots, l_{d_{m'}}\}$ be the departing locations of a set of users $U = \{u_1, u_2, \cdots, u_{m'}\}$ who have a larger social-life-work-related attributes similarity with the query user $u$ (see Definition 3) and stop in the parking locations indexed in subtree $R_{st}$ by the end of the query time $t_0$. Let $L_p = \{l_{p_1}, l_{p_2}, \cdots, l_{p_{n'}}\}$ be the parking locations in the cluster region of $R_{st}$. Then the lower bounding driving time $T_{lo}$ for $R_{st}$ is defined as

$$T_{lo} = \min_{i \in [1,m'], j \in [1,n']} TimeDistDrive(l_{d_i}, l_{p_j}) \tag{1}$$

where $TimeDistDrive(\ ,\ )$ is the time required from $l_{d_i}$ to $l_{p_j}$ by driving.

*Definition 3 (User Similarity (USI)):* Given two users $u_i$ and $u_j$, Ye *et al.* [30] present the technique to derive the similarity between them based on both their social connections and similarity of their check-in activities. We follow the same similarity measure in this article, which can be calculated as below.

$$USI_{i,j} = \alpha \cdot \frac{F_i \bigcap F_j}{F_i \bigcup F_j} + (1-\alpha) \cdot \frac{L_i \bigcap L_j}{L_i \bigcup L_j} \tag{2}$$

where $\alpha$ corresponds to the tuning parameter within the range of $[0, 1]$; F denotes the friend set, which refers to the socially connected friends of a user according to the location-based social networks; and L represents the interests set, which refers to the set of point-of-interests a user has life-work-related check-in activities.

*Definition 4 (Node Capacity (NC)):* Let $C_{nd}$ be the parking availability capacity of node $nd$, and $C_{cl} = \{C_{cl_1}, C_{cl_2}, \cdots, C_{cl_z}\}$ be the parking availability capacity of the $z$ child nodes of $nd$. Then $C_{nd}$ is defined as

$$C_{nd} = \begin{cases} 1 & \textit{if nd is an available location;} \\ 0 & \textit{if nd is an unavailable location;} \\ \sum_{i=1}^{z} C_{cl_i} & \textit{otherwise} \end{cases}$$

### 1) SPACE COMPLEXITY OF SCP-TREE

The space requirement of SCP-tree includes: (i) parking location information $S_{pl}$, (ii) tree hierarchy $S_{tr}$, and (iii) time and capacity constraint $S_{tc}$. The overall storage for SCP-tree, denoted as $S_{SCP-tree}$, can be estimated as $S_{SCP-tree} = S_{pl} + S_{tr} + S_{tc}$. Specifically, $S_{pl}$ involves $|L|$ parking locations, in total having $|W|$ parking location information. As SCP-tree is not a balanced tree, for $L_{st} = |L| - |L'| + |u|$ nodes in the first balance layer of SCP-tree, the height is $log_f L_{st}$, where $|L'|$ is the number of parking locations not at the first balance layer, and $|u|$ is the number of parent nodes generated at the first balance layer. Assume the root is at level 0, the number of SCP-tree nodes in the layers higher than and equal to the first balance layer is given as $\sum_{h=0}^{\log_f L_{st}-1} f^h$. For each parent node $j \in u$, it can be considered as the root node of

another sub-index tree and a similar height can be computed for $j$ through finding the first balance layer. This iteration computation process terminates until all the remaining leaf nodes are in the same level. Given $p$ as the reducing scale of the number of nodes in the first balance layer in each height computation process of the following sub-index trees, the number of nodes in the first balance layer in the following sub-index trees is approximately as $\frac{L_{st}}{p^i}(i = 0, 1, \cdots)$. Thus, $S_{tr}$ can be given as $\sum_{i\geq 0}\sum_{h=0}^{\log_f \frac{L_{st}}{p^i}-1} f^h$. Moreover, since each node is associated with one lower bounding driving time and a parking available capacity, the storage for all "lower bounding driving time" and "parking availability capacity" is $(1+1)\sum_{i\geq 0}\sum_{h=0}^{\log_f \frac{L_{st}}{p^i}-1} f^h = 2\cdot\sum_{i\geq 0}\sum_{h=0}^{\log_f \frac{L_{st}}{p^i}-1} f^h$. Therefore, the overall size of SCP-tree can be given as below.

$$\begin{aligned} S_{SCP-tree} &= O(|W|) + O(\sum_{i\geq 0}\sum_{h=0}^{\log_f \frac{L_{st}}{p^i}-1} f^h) \\ &= O(|W|) + O(\sum_{i\geq 0}\frac{f^{\log_f \frac{L_{st}}{p^i}}-1}{f-1}) \\ &\approx O(|W|) + \sum_{i\geq 0} O(\frac{L_{st}}{p^i}) \\ &\approx O(|W|) + O(L_{st}) \end{aligned} \tag{3}$$

#### 2) TIME COMPLEXITY OF SCP-TREE

Given $p'$ as the reducing scale of the number of parking locations in each clustering process, the number of parking locations in the following sub-cluster regions is approximately as $\frac{n}{p'^i}$ $(i = 0, 1, \cdots)$ from the high level to the low level, where $n$ is the total number of parking locations. The time complexity of clustering parking locations is $O(\frac{n}{p'^i} \times k \times t)$, where $k$ is the number of clusters and $t$ is the iterative times. Usually, $k \ll n$ and $t \ll n$. In our clustering algorithm, $k$ is a constant and $t$ is typically quite small. Thus, the time complexity of Algorithm 1 is $O(n)$.

### C. PARKING LOCATION ALLOCATION

In our parking location allocation scheme, the most common objective is to minimize the total distance of all the vehicles' allocation, that is also to maximize the total service utility across all the users in descending of user priority. Here, a kernel density estimation (KDE) [31], [32], which can be used for arbitrary distribution estimation and without the assumption that the form of the distance distribution is known, is employed to personalize the geographical preference for each user. The intuition is that a user's parking behavior is significantly affected by the geographical positions of parking locations, which have been considered as a major impact factor in minimizing his/her traveling time.

*Definition 5 (Geographical Preference Model (GPM)):* Given a parking region composed of some parking locations, let $U = \{u_1, u_2, \cdots, u_m\}$ be the set of $m$ users that have visited this parking region, and $D$ be the set of distances between each pair of user locations. For a new user $u_i$, we define $d_{ij} = ||u_i, u_j||$ as the Euclidean distance between $u_i$ and $u_j$ $(\in U)$. The kernel density of $d_{ij}$ is defined as:

$$F(d_{ij}) = \frac{1}{|D|b}\sum_{d'\in D} k((d_{ij} - d')/b) \tag{4}$$

$$b = (\frac{4\widehat{\sigma}^5}{3n})^{\frac{1}{5}} \approx 1.06\widehat{\sigma}n^{-\frac{1}{5}} \tag{5}$$

*Definition 6 (Geographical Preference Estimation (GPE)):* The geographical preference of $u_i$ on parking region Rg, denoted by GeoRating($u_i$, Rg), can be calculated by taking the average of all its probability densities, i.e.,

$$GeoRating(u_i, Rg) = \frac{1}{m}\sum_{j=1}^{m} F(d_{ij}) \tag{6}$$

*Definition 7 (User Priority (PR)):* Given a set of attributes $A = \{a_1, a_2, \cdots, a_n\}$ that a user $u_i$ has, let $W_{u_i,a_j}$ be a weighting factor of parking officer's semantic importance on $a_j$ of $u_i$. Some possible semantic weights are extremely important, very important, important, and somewhat important. The overall personalized trade-off priority $u_{pr_i}$ of $u_i$ endowed by the parking officer can be described using the individual personalized attribute $a$, the preference weight $w$ and the aggregation operator $\otimes$ given as

$$u_{pr_i} = (W_1 \cdot a_1) \otimes (W_2 \cdot a_2) \otimes \cdots \otimes (W_n \cdot a_n) \tag{7}$$

In our allocation scheme, we assume that a user provides a profile before he or she starts moving. A profile should cover all categorical attributes, such as the personalized attribute, location, distance etc. And such preference order information summarized from the profiles of different users is stored by the officer for parking allocation. In the following, we give an example for explanation.

*Example 2* Suppose a user Alice has the following preferences:

- The distant departing location regions are preferable because it may require to spend more time before parking the car. These preferences can be expressed as an order between the semantic distances: remote $\succ$ medium $\succ$ near. The weight order of these preferences, for example, can be expressed as $W_{remote} = 0.8 > W_{medium} = 0.5 > W_{near} = 0.2$.
- She prefers to drive slowly because of her senior age. This means that there exist preference relationships between the semantic attributes: old $\succ$ middle-aged $\succ$ young. The weight order of these preferences, for example, can be expressed as $W_{old} = 0.8 > W_{middle-aged} = 0.5 > W_{young} = 0.2$.

It is noteworthy that in *peak hours*, a lot of users will be requesting for parking. To help parking allocation in peak hour, some analysis and decision are required as soon as possible by sharing and reducing computations. To achieve this goal, we can divide the users into groups using their

approximate priorities. Therefore, a decending queue can be constructed based on both the users' and groups' priority. In the following, we extend the Geographical Preference Estimation of a single user to the Geographical Preference Estimation for *a group of users*.

*Definition 8 Group Geographical Preference Model (GGPM):* Similar to Definition 5, for a group of users $ug_i$, we define $Ad_{ij} = \frac{\sum_{u_{i'} \in ug_i} ||u_{i'}, u_j||}{N_{ug_i}}$ as the average Euclidean distance between each user in $ug_i$ and $u_j (\in U)$, where $N_{ug_i}$ is the number of users in $ug_i$. The kernel density of $Ad_{ij}$ is defined as:

$$F(Ad_{ij}) = \frac{1}{|D|b} \sum_{d' \in D} k((Ad_{ij} - d')/b) \tag{8}$$

*Definition 9 Group Geographical Preference Estimation (GGPE):* The geographical preference of $ug_i$ on parking region Rg, denoted by GeoRating($ug_i$, Rg), can be calculated by taking the average of all its probability densities, i.e.,

$$GeoRating(ug_i, Rg) = \frac{1}{m} \sum_{j=1}^{m} F(Ad_{ij}) \tag{9}$$

**Algorithm 2** ParkingLocationAllocation

**Require:** $S_{l_p}$: the set of parking locations
$S_{ur}$: a set of users
$\theta_{td}$: the driving time constraint
$\theta_{tw}$: the walking time constraint
**Ensure:** $S_{npl}$: the set of nearest parking locations to each user
1: **foreach** $u_i \in S_{ur}$ **do**
2: $u_{pr_i} \leftarrow$ the priority computed from Eq. 7
3: **end for**
4: $u_{pq} \leftarrow$ the user priority group queue constructed from $u_{pr_i}$ in descending order
5: SCP-tree $\leftarrow$ SCPTreeBuilt($S_{l_p}$)
6: $nd_{rt} \leftarrow$ the root node of SCP-tree
7: $S_{cl} \leftarrow$ the child nodes of $nd_{rt}$
8: $u_{pq}(\theta_{tw}) \leftarrow$ ParkingLocationAllocationDestination($S_{l_p}, S_{cl}, S_{ur}, \theta_{tw}$)
9: **foreach** $u_{pq}(\varphi \in S_{cl}, \theta_{tw}) \in u_{pq}(\theta_{tw})$ **do**
10: **foreach** $\varphi \in u_{pq}(\varphi \in S_{cl}, \theta_{tw})$ **do**
11: **if** $\vartheta = u_i$ **then**
12: $l_{nf_i} \leftarrow$ SingleUserParkingAllocation($u_i, \theta_{td}, \varphi$)
13: $l_{np_i} \leftarrow$ the nearest parking location in $l_{nf_i}$ to the departing location of $u_i$
14: $S_{npl} \leftarrow S_{npl} \bigcup l_{np_i}$
15: **else**
16: $l_{nf_{i'}} \leftarrow$ SingleGroupUserParkingAllocation($ug_i, \theta_{td}, \varphi$)
17: $l_{np_{i'}} \leftarrow$ the nearest parking location in $l_{nf_{i'}}$ to the departing location of each user in $u_{gi}$
18: $S_{npl} \leftarrow S_{npl} \bigcup l_{np_{i'}}$
19: **end if**
20: **end for**
21: **end for**

Algorithm 2 shows the details of parking location allocation for a set of users. We first compute the priority for each user (lines 1-3), which is used to serve the maximum number of users in terms of distance. The priority queue is maintained in descending order of the users' or user groups' priority value. For the users or user groups in this priority queue, we re-divide them into several sub-priority queues by computing the traveling time between their destinations and the branch node parking regions (line 8). For the users or user groups in each sub-priority queue, we search the nearest leaf node according to Algorithm 4 (line 12) and 5 (line 16), which introduce the parking location allocation mechanism for a single user and a group of users, respectively. Since more than one parking location may be allocated to a user, we select the nearest one as the final allocation result. As the time complexity of Algorithm 3 is $O(n_{ur})$, and the time complexity of Algorithm 4 and Algorithm 5 is $O(n)$ after-mentioned, the time complexity of parking location allocation can be derived as $O(n_{ur} + n_{ur} \times n)$, where $n_{ur}$ is the total number of users and user groups, and $n$ is the number of parking locations. Usually, $n_{ur}$ is a constant and $n_{ur} \ll n$. Therefore, the time complexity of Algorithm 2 can be derived as $O(n)$.

**Algorithm 3** ParkingLocationAllocationDestination

**Require:** $S_{l_p}$: the set of parking locations
$S_{cl}$: the child nodes of the root node of SCP-tree
$S_{ur}$: a set of users
$\theta_{tw}$: the walking time constraint
**Ensure:** $u_{pq}(\theta_t) \leftarrow$ user priority group queue constructed with $\theta_{tw}$ constraint
1: **foreach** $u_i \in S_{ur}$ **do**
2: $u_{pr_i} \leftarrow$ the priority computed from Eq. 7
3: **end for**
4: $u_{pq} \leftarrow$ the user priority group queue constructed from $u_{pr_i}$ in descending order
5: **foreach** $\vartheta \in u_{pq}$ **do**
6: **foreach** $\varphi \in S_{cl}$ **do**
7: **if** TimeDistWalk($\vartheta.destination, \varphi$) $< \theta_{tw}$ **then**
8: $u_{pq}(\varphi, \theta_{tw}) \leftarrow u_{pq}(\varphi) \bigcup \vartheta$
9: **end if**
10: **end for**
11: **end for**
12: **foreach** $\varphi \in S_{cl}$ **do**
13: $u_{pq}(\theta_{tw}) \leftarrow u_{pq}(\theta_{tw}) \bigcup u_{pq}(\varphi, \theta_{tw})$
14: **end for**

The steps for re-dividing the users or user groups in descending order in the priority queue are given in detail in Algorithm 3, which aims to guarantee the walking time satisfying the query requirement. Through computing the traveling time between the users or user groups' destinations and the branch node parking region (line 7) according to Eq. 10, each user or user group can be divided into a sub-priority queue correlated with a child branch region of the root node (line 8). The time complexity of Algorithm 3 is $O(|u_{pq}| \cdot |S_{cl}|)$, where $|u_{pq}|$ represents the number of members in the priority queue and $|S_{cl}|$ represents the child nodes' number of the root node. Since $|u_{pq}| \leq n_{ur}$, $|S_{cl}| \leq M_0$ ($M_0$ is a constant), the time complexity of Algorithm 3 in the worst case is $O(n_{ur})$, where $n_{ur}$ is the number of users.

*Definition 10 (Walking Time (WT)):* Let $R_s$ be a destination region including $S = \{l_{s1}, \cdots, l_{sm}\}$ locations, $R_p$ be a parking region including $P = \{l_{p1}, \cdots, l_{pn}\}$ locations. The walking time from $R_s$ to $R_p$ with a traveling speed $v$ is defined as:

$$TimeDistWalk(R_s, R_p) = \frac{\max\limits_{i \in [1,m], j \in [1,n]} ||l_{si}, l_{pj}||}{v} \tag{10}$$

The steps for processing the parking location allocation for a single user are given in detail in Algorithm 4: (i) We first

**Algorithm 4** SingleUserParkingAllocation

**Require:** $u_i$: a user $u_i$
$\theta_{td}$: the driving time constraint
$nd_{st}$: a subtree node in SCP-tree
**Ensure:** $l_{nf_i}$: the nearest leaf node to $u_i$
1: $S_{cld} \leftarrow$ the child nodes of $nd_{st}$
2: $GeoRating \leftarrow 0$
3: **foreach** $c_i \in S_{cld}$ **do**
4: $T_{loi} \leftarrow$ the lower bounding driving time of $c_i$
5: $C_{ci} \leftarrow$ the parking available capacity of $c_i$
6: **if** $T_{loi} < \theta_{td}$ and $C_{ci} \neq 0$ **then**
7: $GeoRating(u_i, c_i) \leftarrow$ compute the geographical preference estimation of $u_i$ to $c_i$ according to Eq. 6 along SCP-tree
8: **if** $GeoRating(u_i, c_i) > GeoRating$ **then**
9: $c_0 \leftarrow c_i$
10: **end if**
11: $GeoRating \leftarrow GeoRating(u_i, c_i)$
12: **end if**
13: **end for**
14: $l_{nf_i} \leftarrow$ SingleUserParkingAllocation$(u_i, \theta_{td}, c_0)$

**Algorithm 5** SingleGroupUserParkingAllocation

**Require:** $ug_i$: a group of users having an approximate priority
$\theta_{td}$: the driving time constraint
$nd_{st}$: a subtree node in SCP-tree
**Ensure:** $l_{nf_{i'}}$: the nearest leaf node to each user in $ug_i$
1: $S_{cld} \leftarrow$ the child nodes of $nd_{st}$
2: $GeoRating \leftarrow 0$
3: $flag \leftarrow 0$
4: **foreach** $c_i \in S_{cld}$ **do**
5: **if** $N_{c_i} > N_{ug_i}$ **then**
6: flag = flag + 1
7: **end if**
8: **end for**
9: **if** $flag = N_{S_{cld}}$ **then**
10: **foreach** $c_i \in S_{cld}$ **do**
11: $T_{loi} \leftarrow$ the lower bounding driving time of $c_i$
12: $C_{ci} \leftarrow$ the parking available capacity of $c_i$
13: **if** $T_{loi} < \theta_{td}$ and $C_{ci} \neq 0$ **then**
14: $GeoRating(u_{gi}, c_i) \leftarrow$ compute the geographical preference estimation of $u_{gi}$ to $c_i$ according to Eq. 9 along SCP-tree
15: **if** $GeoRating(u_{gi}, c_i) > GeoRating$ **then**
16: $c_0 \leftarrow c_i$
17: **end if**
18: $GeoRating \leftarrow GeoRating(u_{gi}, c_i)$
19: **end if**
20: **end for**
21: $l_{nf_{i'}} \leftarrow$ SingleGroupUserParkingAllocation$(ug_i, \theta_{td}, c_0)$
22: **else**
23: **foreach** $u_{i'} \in u_{gi}$ **do**
24: $l_{nf_{i'}} \leftarrow$ SingleUserParkingAllocation$(u_{i'}, \theta_{td}, nd_{st})$
25: **end for**
26: **end if**

determine the lower bounding driving time of each node (line 4), which is used to serve the users within the driving time constraint. (ii) The Geographical Preference Estimation to the nodes that satisfy the driving time constraint and parking capacity constraint are computed for the user $u_i$ along the SCP-tree (line 5-11) until the nearest leaf node to $u_i$ is returned (line 13). The time complexity of Algorithm 4 is $O(n)$, where $n$ is the number of parking locations.

The pseudocode for allocating the parking locations for a group of users is shown in Algorithm 5. We first determine the lower bounding driving time of each node (line 11), which is used to serve the users within the driving time constraint. The user group computes the Geographical Preference Estimation to the nodes that satisfy the driving time constraint along the SCP-tree (line 12-18) until the nearest leaf node to $ug_i$ is returned (line 20). When one of the child nodes of a certain non-leaf node cannot have more parking locations than that required, the parking location allocation of a single user is performed for each user in $ug_i$ (line 23) until the nearest leaf node to each user in $ug_i$ is returned. As the time complexity of Algorithm 4 is $O(n)$ afore-mentioned, the time complexity of Algorithm 5 can be derived as $O(n)$, where $n$ is the number of parking locations.

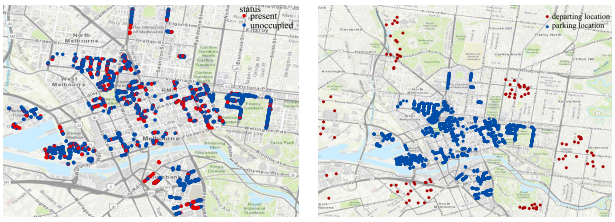
*Example 3:* Here we use an example to illustrate the traversal process over SCP-tree shown in Fig. 2. When users' priority is determined by Definition 7, we carry on traversing SCP-tree to achieve the parking location of current user $u$ in Fig. 2. Supposed that the walking time from $u'$ destination to parking region $R_1$ is within the time constraint $\theta_{tw}$, we carry on searching the parking location along $R_1$ branch. We assume that $T_{lo}$ at each node $R_i$ ($i \in [0, 8]$) is lower than the driving time threshold $\theta_{td}$ specified by the query and all the parking locations are available currently. First, we access $R_3$ and $R_4$, and compute $GeoRating(u, R_3) > GeoRating(u, R_4)$ based on $u_1$, $u_2$, $u_3$ and $u_4$, who have already visited parking locations $l_{p_1}$, $l_{p_3}$, $l_{p_5}$ and $l_{p_7}$. Finally, we acquire the nearest parking location $l_{p_1}$. When the parking available capacity of $R_4$ is "0", $R_4$ is pruned and we focus on $R_3$ for the nearest parking location allocation.

## V. IMPLEMENTATION AND EVALUATION

### A. EXPERIMENTAL SETUP

Our real parking locations dataset is from Melbourne City Centre, which consists of 2908 parking locations [33], [34]. In this experiment, we use the parking space availability data from the on street parking bay sensors events records recently published online, that include different bays, streets, status as well as spatial information such as the latitude and longitude of the parking locations (see Table 2 as an example). Fig. 4(a) reveals the parking location availability in Melbourne City Centre within a certain time interval. Further, we extract 100 users' departing locations that disperse in the administrative regions of Melbourne City during 2016/1/4-2016/1/10 8:00-17:00. The parking locations and extracted departing locations are visualized in Fig. 4(b). In addition, we also extract 130 users' destination locations from the neighborhood of parking locations. The user percent for each attribute combination shown in Table 3 is extracted from a real-world dataset such as China Daily [35].

Furthermore, we extract the data from the four time divisions (Morning (AM), Afternoon (PM), Monday to Friday (MF), Saturday to Sunday (SS)), to compare the parking location performance. The parking locations in the four time divisions are counted from parking violation sensor datasets in January, 2016 (see Table 4 as an example). And the values



(a) Parking availability within a certain time interval (b) The departing locations and parking locations

**FIGURE 4.** Parking availability, departing locations and parking locations.

**TABLE 2.** Parking Locations Without Time Division in Experiments.

| BayId | StreetId | Status | Location |
|---|---|---|---|
| 6379 | 13628E | Unoccupied | (-37.819425816280599,144.94449225725817) |
| 2526 | C2986 | Present | (-37.813101635840611,144.95945362700032) |
| 1761 | 2384N | Unoccupied | (-37.814371236105046,144.96246424218043) |
| 1371 | 1526E | Unoccupied | (-37.817549044707249,144.95358534336572) |
| 4694 | 8640E | Present | (-37.825634397962318,144.96259843892904) |

**TABLE 3.** User Attribute Percent Settings in Experiments.

| Percent | Age | occupation | Driving Age |
|---|---|---|---|
| 10% | $> 50$ | limit | $< 6$ |
| 20% | $> 50$ | free | $> 10$ |
| 10% | $< 30$ | limit | $< 6$ |
| 20% | $< 30$ | free | $> 10$ |
| 40% | null | limit/free | null |

tested for each parameter in our experiment are detailed in Table 5, where the default parameter values are given in bold.

For the sake of understanding the effectiveness of SCP-tree, time constraints, GPE and priority, the four methods with progressive constraints are tested in our experiment (see Table 6), where ''yes'' indicates that the method has the corresponding constraint, otherwise, it is labeled as ''no''. Meanwhile, we also give another three typical baseline methods to illustrate the performance of proposed method:

- *BPMT*: An allocation scheme that provides a pricing mechanism to guarantee the optimal social welfare by using ad auction theory [7].
- *NHEQ*: An allocation scheme that assigns vehicle-slot pair prices, which guarantees that no driver will pay a higher total cost than the one she can obtain by behaving selfishly [6].
- *ECP*: An allocation scheme that recursively searches for the closest pair objects from two data sets [8].

In the following, we aim to compare these methods in terms of allocation time, I/O cost, traveling time and service utility. Meanwhile, we also test and analyze the parking allocation performance about group based allocation scheme, which includes the following aspects: group and non-group based allocation time (GAI, AI), group and non-group based traveling time (GTT, TT), group and non-group based accessed pages (GAP, AP), as well as group and non-group based service utility (GSU, SU).

To evaluate the service utility for allocation scheme, we adopt the following three kinds of utilities similar to that in Literature [12]:

- Parking Location-related Utility ($\xi_p(u_i, p_j) \in [0, 1]$): This utility reflects the preference of user $u_i$ towards the parking location $p_j$. It can be estimated with the categorically stated parking experience relevant to the geographical preference and personalized attributes.
- User-related Utility ($\xi_u(u_i, p_j)$): This utility will change when other users with similar tastes in parking location $p_j$ change. Specifically, we define user-related utility in the following form:

$$\xi_u(u_i, p_j) = \sum_{u_{i'} \in T_j^k} \frac{USI(u_i, u_{i'})}{T_j^k - u_i} \tag{11}$$

where $T_j^k$ indicates the users parking in $p_j$ during the trajectories $Tr_i^j$ connecting $u_i$ and $p_j$, and $USI(u_i, u_{i'})$ is the social similarity of $u_i$ and $u_{i'}$. Intuitively, for a user $u_i$, sharing with other high-similarity users for a larger portion of $Tr_i^j$, will lead to a higher user-related utility.
- Trajectory-related Utility ($\xi_t(u_i, p_j) \in [0, 1]$): This utility is to measure the satisfaction level of the user $u_i$ towards the routes $Tr_i^j$. In general, the trajectory-related utility decreases when the extra travel cost is incurred (e.g., caused by detours, weather, traffic congestion etc) for $u_i$ parking in $p_j$. We define the trajectory-related utility as

$$\xi_t(u_i, p_j) = \frac{Cost(l_{di}, l_{pj})}{\sum_{tr_k \in Tr_i^j} Cost(tr_k)} \tag{12}$$

### B. EXPERIMENTAL RESULTS

#### 1) COMPARISON WITH PROGRESSIVE METHODS

##### a: EFFECT OF USER NUMBER

We begin by studying the impact of user number from 10 to 130. In this set of experiment, the parking location number is set as 2908. From Fig. 5(a), we can see the method *NNNN* without any order and pruning optimization performs worst in terms of parking location allocation time. With the help of SCP-tree, the method *NNNS* is at least 10 times faster than *NNNN*. This means that the SCP-tree gives a better pruning effect on the parking location search with geographical approximation. When pruning in the SCP-tree, we adopt the Geographical Preference Estimation, lower boundary driving time and node capacity as the three criteria to accurately carry on judgment. This eliminates the distant parking locations that are not within the time constraints. The experiment results of method *NGTS* verify this effectiveness. Additionally, we design user priority based on the distance and personalized attributes. Leveraging the order of user priority, the allocation process incurs less comparison and adjustment for maximizing the benefits of all users. Overall, the approach *PGTS* has a better improvement than the other three progressive methods in terms of parking location allocation time. We also notice that the parking location allocation time increases with the number of users in the order of $NNNN \succ NNNS \succ NGTS \succ PGTS$. This is to be expected

**TABLE 4.** Parking Locations for the Four Time Divisions in Experiments.

| ArrivalTime | DepartionTime | StreetId | Sign | Area | Location |
|---|---|---|---|---|---|
| 01/07/2016 01:09:04 PM | 01/07/2016 01:13:50 PM | 13085S | 2P TKT A M-F 7:30-18:30 | Docklands | (-37.82111917,144.9429514) |
| 01/04/2016 09:21:57 AM | 01/04/2016 09:33:35 AM | 12443S | 1P MTR M-SAT 7:30-18:30 | Jolimont | (-37.81196723,144.9759003) |
| 01/05/2016 11:52:25 AM | 01/05/2016 12:03:28 PM | 1457W | 1P MTR M-F 10:00-16:00 | Family | (-37.81673883,144.9557671) |
| 01/04/2016 12:38:46 PM | 01/04/2016 12:43:37 PM | 12E | 2P TKT A M-SAT 7:30-20:30 | Twin Towers | (-37.81469214,144.9747051) |
| 01/07/2016 07:30:00 AM | 01/07/2016 08:18:46 AM | C6376 | 4P MTR M-F 7:30-18:30 | Queensberry | (-37.80405567,144.950951) |

**TABLE 5.** Parameter and Value Settings in Experiments.

| Parameter Description | Value |
|---|---|
| number of users | 10, 25, 40, 55, 70, 85, **100**, 115, 130 |
| number of parking locations | 500, 1000, 1500, 2000, 2500, **2908** |
| number of parking locations (AM) | 500, 1000, 1500, 2000, **2327** |
| number of parking locations (PM) | 100, 200, 300, 400, 500, **581** |
| number of parking locations (MF) | 500, 1000, 1500, 2000, **2010** |
| number of parking locations (SS) | 100, 300, 500, 700, **898** |
| the traveling speed of user | 7, 10, 13, **16**, 19, 22 (m/s) |

**TABLE 6.** Progressive Methods Settings in Experiments.

| Method | Priority | GPE | Time Constraints | SCP-tree |
|---|---|---|---|---|
| NNNN | no | no | no | no |
| NNNS | no | no | no | yes |
| NGTS | no | yes | yes | yes |
| PGTS | yes | yes | yes | yes |

since the parking officer has to do more work to consider all the users' requirements.

Here, we study the distribution of average traveling time in terms of the number of users. Fig. 5(b) compares the average traveling time for the four approaches. Note that, the four approaches have a very similar traveling time distribution with the increase of user number. This demonstrates the superiority of our approach, as the other candidate approaches provide an approximately optimal allocation result in about 1100s. Some factors, such as weather, age, habit, road condition etc, have an impact on the traveling time. Compared with the parking location allocation time, the difference value of average traveling time is very small, at most 100$s$.

Next, we evaluate index I/O cost of the four approaches in Fig. 5(c). It is clear that all approaches follow a linear trend with the number of users, consistent to the behavior determined in our cost model. Meanwhile, we can also see a significant difference in terms of magnitudes among all those approaches due to the efficient pruning-based spatial index search algorithm facilitated by the Geographical Preference Estimation and user priority.

We also evaluate the service utility of our proposed approach in terms of the number of users in Fig. 5(d). As the user number increases, the generated service utility also grows since more users' benefits are maximized for allocating the nearest available parking locations. Since the service utility is evaluated in terms of user preference towards the parking location and route, as well as user similarity, *PGTS* performs best among the four methods.

In summary, we observe that the parking location allocation time, I/O cost as well as service utility present an increasing trend with the user number for each approach. The method *NNNN* has the largest parking allocation time, I/O cost, and the minimum service utility. The next worst method is *NNNS*, and the best method is *PGTS*. The allocation time, I/O cost, service utility and increment speed reflect the influence of SCP-tree, GPE, time constraint, user priority and user similarity on the allocation scheme. This implies that the method including more dominate factors provides better allocation performance. Although the four methods have a big difference in the parking location allocation time, I/O cost and user utility to get the optimal allocation strategy, the average traveling time of the final allocation result for each method has a smaller fluctuation time range, always in [1000$s$, 1200$s$]. The experiment results indicate the four factors, namely SCP-tree, GPE, time constraint and user priority, along with the user similarity greatly improve the allocation performance.

### b: EFFECT OF PARKING LOCATION NUMBER

To further explore the performance of our method, Fig. 5(e), Fig. 5(f), Fig. 5(g), Fig. 5(h) present the parking location allocation time, I/O cost, average travel time and service utility changing over the number of parking locations. We set the user number as 100 in this experiment. We observe that the parking location allocation time and I/O cost increase with the number of parking locations, as the parking allocation scheme needs to query and give a judgement over the gradually increasing parking range. The four methods *NNNN*, *NNNS*, *NGTS*, *PGTS* generate less and less allocation time and I/O cost for achieving the optimal allocation plan, as they gradually consider the SCP-tree, GPE, time constraint and user priority for effective pruning and filtering. The allocation time and I/O cost always increase with the number of parking locations regardless of the strategies considered in the allocation scheme.

We also test the average traveling time of the four approaches in terms of the number of parking locations. The traveling time distribution in Fig. 5(f) shows that the average traveling time for the allocation result of each method always fluctuates around 1100s in all cases. This is an interesting insight, which suggests that the other three progressive approaches tend to have the approximately average traveling time with *PGTS* at the expense of the parking allocation time, I/O cost and service utility. As for the service utility, *PGTS* performs best among the four methods, which can be explained as that in the former set of experiments.

### c: EFFECT OF TRAVELING SPEED

Fig. 5(i), Fig. 5(j), Fig. 5(k), Fig. 5(l) further study the parking location allocation time, I/O cost, average travel

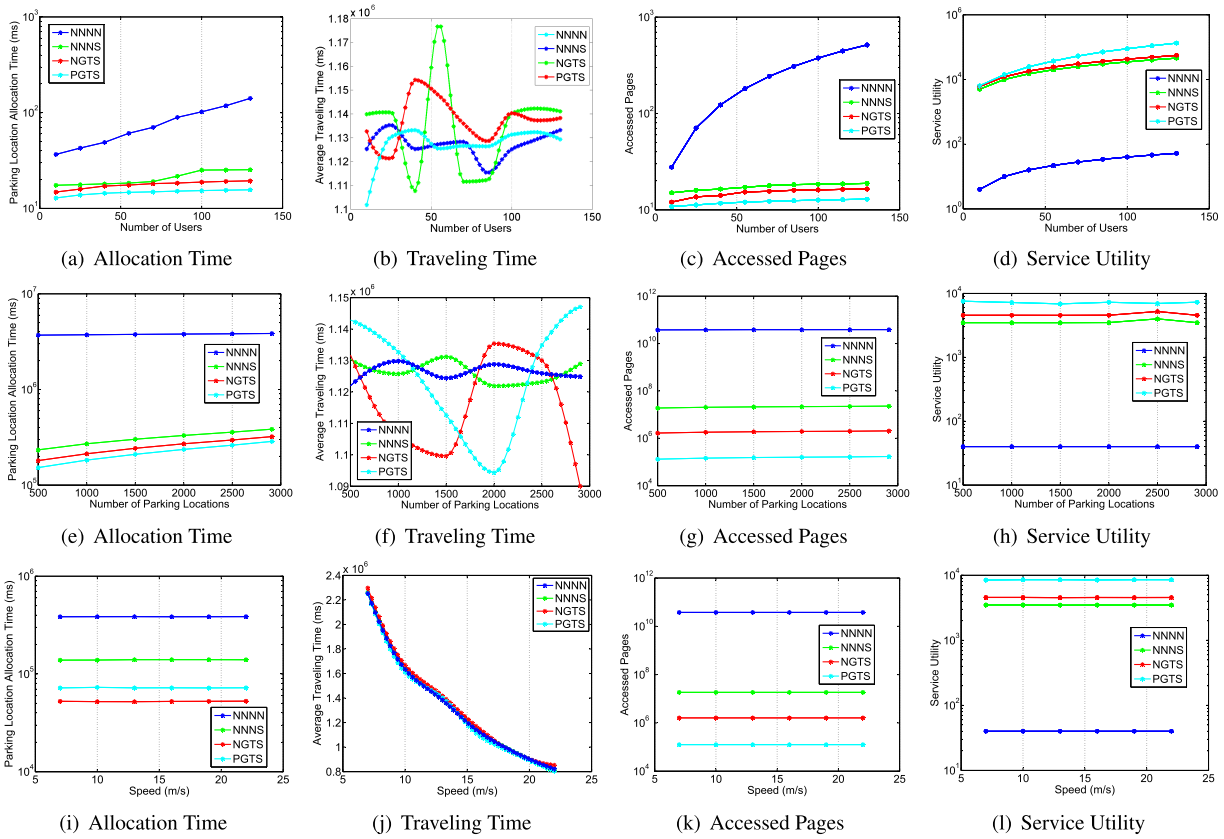


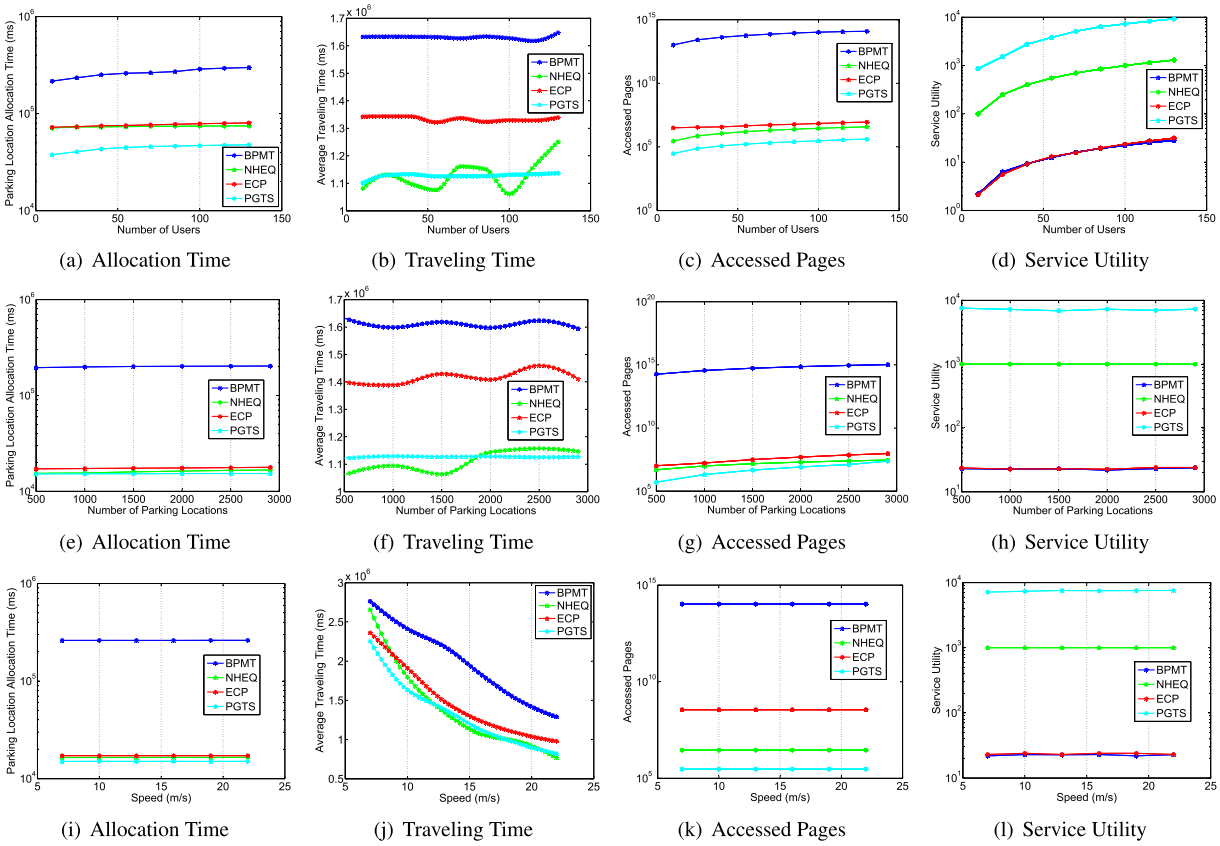

**FIGURE 5.** Progressive methods comparison without time division.

**TABLE 7.** Performance Comparison of Progressive Methods in Four Time Divisions.

| Time Division | Performance | NNNN | NNNS | NGTS | PGTS |
|---|---|---|---|---|---|
| AM | parking allocation time (ms) | 156669.22 | 86978.02 | 77325.80 | 65017.03 |
| | average traveling time (ms) | 1128794.67 | 1132540.93 | 1145432.29 | 1128762.19 |
| | accessed pages | 9244343328.7 | 80738817.92 | 7986299.21 | 241548.68 |
| | service utility | 41.02 | 3499.68 | 4541.71 | 7297.35 |
| PM | parking allocation time (ms) | 150643.02 | 13585.90 | 79482.66 | 65208.14 |
| | average traveling time (ms) | 1129941.54 | 1163579.96 | 1129266.69 | 1124388.645 |
| | accessed pages | 2310856671.3 | 20158682.08 | 1994000.79 | 60309.32 |
| | service utility | 40.09 | 3489.72 | 4551.81 | 7287.14 |
| MF | parking allocation time (ms) | 166447.14 | 108401.47 | 81719.39 | 66676.59 |
| | average traveling time (ms) | 1118794.68 | 1122540.93 | 1125432.29 | 1138762.19 |
| | accessed pages | 7914151306.7 | 69740018.91 | 6898350.41 | 208643.25 |
| | service utility | 40.19 | 3499.82 | 4557.88 | 7289.26 |
| SS | parking allocation time (ms) | 152769.82 | 90363.45 | 79058.30 | 65056.76 |
| | average traveling time (ms) | 1125070.46 | 1138836.73 | 1123002.90 | 1175070.46 |
| | accessed pages | 3541048693.3 | 31157481.09 | 3081949.59 | 93214.75 |
| | service utility | 41.39 | 3499.44 | 4559.47 | 7299.35 |

time and service utility for varying the traveling speed of users. In this set of experiments, we set the user number as 100 and the number of parking locations as 2908. The experiment results match our intuition that the allocation time, I/O cost and service utility are mainly influenced by two dominated factors, namely the user number and parking location number, along with some other impact factors, such as the user preference, user similarity, weather, road condition etc. We see that varying the traveling speed of users has no obvious impact on the allocation time, I/O cost and service utility. We further observe that the allocation time and I/O cost decrease by a large margin in the order of *NNNN*, *NNNS*, *NGTS*, *PGTS*, while the service utility increases in the order of *NNNN*, *NNNS*, *NGTS*, *PGTS*. The reasons behind this can be explained as that in the experiments for varying the number of users and parking locations.

Fig. 5(j) illustrates the traveling time performance with the traveling speed according to the allocation result of the four approaches. As shown in the figure, increasing the traveling speed has absolutely reduced the traveling time for all the four allocation approaches since the allocation result of each approach has the approximately same parking location for the




(a) Allocation Time (b) Traveling Time (c) Accessed Pages (d) Service Utility

(e) Allocation Time (f) Traveling Time (g) Accessed Pages (h) Service Utility

(i) Allocation Time (j) Traveling Time (k) Accessed Pages (l) Service Utility

**FIGURE 6.** Baseline methods comparison without time division.

**TABLE 8.** Performance Comparison of Baseline Methods in Four Time Divisions.

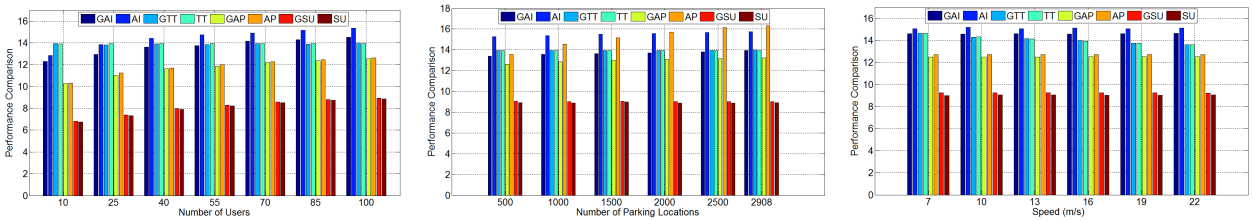
| Time Division | Performance | BPMT | NHEQ | ECP | PGTS |
|---|---|---|---|---|---|
| AM | parking allocation time (ms) | 113306.15 | 71625.64 | 74608.99 | 65285.44 |
| | average traveling time (ms) | 1628276.44 | 1261690.01 | 1329876.44 | 1121325.97 |
| | accessed pages | 1818723200 | 2390682 | 5692310 | 519013 |
| | service utility | 23.81 | 1007 | 22.38 | 7197.14 |
| PM | parking allocation time (ms) | 105858.88 | 64178.36 | 67161.72 | 57838.16 |
| | average traveling time (ms) | 1639276.83 | 1262589.61 | 1339858.65 | 1131890.15 |
| | accessed pages | 1048723200 | 1009682 | 2327001 | 251858 |
| | service utility | 22.36 | 1000 | 23.96 | 7256.14 |
| MF | parking allocation time (ms) | 113036.05 | 71017.71 | 74225.09 | 65285.44 |
| | average traveling time (ms) | 1657256.49 | 1222589.61 | 1395869.65 | 1141890.41 |
| | accessed pages | 1818723200 | 2031593 | 5261354 | 500206 |
| | service utility | 23.69 | 1020 | 23.42 | 7287.15 |
| SS | parking allocation time (ms) | 108077.36 | 66396.86 | 69380.21 | 60056.66 |
| | average traveling time (ms) | 1626468.67 | 1266589.73 | 1321104.53 | 1113682.41 |
| | accessed pages | 1048723200 | 1535783 | 2718256 | 283147 |
| | service utility | 23.73 | 1011 | 23.07 | 7296.24 |

same user. The average traveling time of the four methods can reach to the maximum value of 2300$s$ and minimum value of 810$s$ when the user has the slowest traveling speed $7m/s$ and the highest traveling speed $22m/s$, respectively. The difference between traveling time values only reaches to the maximum value of 180$s$.

To further illustrate the parking allocation performance, we compare the performance of four progressive methods in different time divisions (see Table 7). In this set of experiments, we adopt the default parameters values given in Table 5. The results present the similar variant trend for parking allocation performance as that without time division.

### 2) COMPARISON WITH BASELINE METHODS

Fig. 6 illustrates the experimental results on the comparison with the baseline methods without time division. The results are quite similar to that of the previous experiments. *PGTS* demonstrates a significant advantage over the other methods in terms of the overall parking location allocation time, average traveling time, I/O cost and service utility, while the




(a) Group vs Non-group (b) Group vs Non-group (c) Group vs Non-group

**FIGURE 7.** Performance comparison with group-based allocation.

four methods have the almost similar performance variant trend under different parameter settings. In all cases, *PGTS* efficiently allocates the parking locations and requires much less I/O cost as well as generates a higher service utility than *BPMT*, *NHEQ* and *ECP* for varying the number of users, the number of parking locations and traveling speed. This can be attributed to the ideas that the layer partitions for parking locations are at an arbitrary order in *BPMT* while *NHEQ* executes the auction algorithm for pricing parking locations in rounds or iterations beginning with arbitrary allocations and prices. Meanwhile, *ECP* only considers the pair nearest distance for parking location allocation. A significant adjustment is still required for maximizing all the users' benefits since the users' personalized attributes also have an important impact on the traveling time. Moreover, *PGTS* reduces the average traveling time compared to *BPMT*, *NHEQ* and *ECP* since they do not consider any time and personalized attribute constraint as the preference allocation order in the allocation process. The intuition behind is that the factors, such as the time and personalized attributes, can impact and postpone the traveling to some extent.

To further illustrate the parking allocation performance, we compare the performance of four baseline methods in different time divisions (see Table 8). In this set of experiments, we also adopt the default parameter values given in Table 5. The results present the similar variant trend for parking allocation performance as that without time division.

### 3) COMPARISON WITH GROUP-BASED ALLOCATION METHOD

In the following, we present group based allocation performance in peak hours. Fig. 7 shows the experiment results for varying the number of users, the number of parking locations and traveling speed. In all cases, group based allocation method requires less allocation time and I/O cost than that of the non-group allocation scheme, while having a higher service utility and approximate traveling time trend. The reason behind is that the users with approximate priority are grouped together, which can facilitate the officer to make some analysis and decision as soon as possible by sharing and reducing some of the computations when querying along the SCP-tree. Meanwhile, the user group provides a higher user-related utility since "group" indicates that the users in this group have a higher social similarity and "users gathering together at peak hours" indicates they have a higher probability to park in the regions along the same or adjacent trajectories.

## VI. CONCLUSION

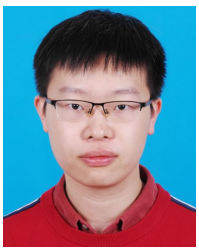
In this article, we propose a novel parking location allocation scheme for mobile users to support the sustainability in green transportation systems. The focus of this work is to construct an SCP-tree by incorporating a lower boundary driving time and a node parking capacity into the design mechanism. Furthermore, we present a Geographical Preference Estimation based search strategy along the branch nodes that are within the walking time requirement in descending order of user priority. We evaluate the performance with four progressive methods and also compare our approach against three baselines. The experimental results show that our proposed method has a theoretical and practical superiority in the parking allocation performance. However, the method has to be executed by parking officer continuously for efficient allocation. As future work, we would like to extend this paper to provide means to automatically and intelligently realize the parking allocation combined with the technology of Internet of Things and machine learning.

## REFERENCES

[1] B. Yang, N. Fantini, and C. S. Jensen, "iPark: Identifying parking spaces from trajectories," in *Proc. 16th Int. Conf. Extending Database Technol.*, 2013, pp. 705–708.
[2] E. H.-C. Lu and C.-H. Liao, "Prediction-based parking allocation framework in urban environments," *Int. J. Geographical Inf. Sci.*, vol. 34, no. 9, pp. 1873–1901, Sep. 2020.
[3] J. Arellano-Verdejo, F. Alonso-Pecina, E. Alba, and A. G. Arenas, "Optimal allocation of public parking spots in a smart city: Problem characterisation and first algorithms," *J. Exp. Theor. Artif. Intell.*, vol. 31, no. 4, pp. 575–597, Jul. 2019.
[4] D. Ayala, O. Wolfson, B. Xu, B. Dasgupta, and J. Lin, "Parking slot assignment games," in *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst. (GIS)*, 2011, pp. 299–308.
[5] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Parking in competitive settings: A gravitational approach," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 27–32.
[6] D. Ayala, O. Wolfson, B. Xu, B. DasGupta, and J. Lin, "Pricing of parking for congestion reduction," in *Proc. 20th Int. Conf. Adv. Geographic Inf. Syst. (SIGSPATIAL)*, 2012, pp. 43–51.
[7] R. Meir, Y. Chen, and M. Feldman, "Efficient parking allocation as online bipartite matching with posted prices," in *Proc. Int. Conf. Auto. Agents Multi-Agent Syst.*, 2013, pp. 303–310.
[8] N. Mamoulis and M. L. Yiu, "Computation and monitoring of exclusive closest pairs," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 12, pp. 1641–1654, Dec. 2008.
[9] Y. Kc and C.-S. Kang, "A connected car-based parking location service system," in *Proc. IEEE Int. Conf. Internet Things Intell. Syst. (IoTaIS)*, Nov. 2019, pp. 167–171.

[10] H. U. Leong, M. L. Yiu, K. Mouratidis, and N. Mamoulis, "Capacity constrained assignment in spatial databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2008, pp. 1–12.
[11] C. Tian, H. Yan, L. Zhi, F. Bastani, and R. Jin, "Noah: A dynamic ridesharing system," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 2013, pp. 985–988.
[12] P. Cheng, H. Xin, and L. Chen, "Utility-aware ridesharing on road networks," in *Proc. ACM Int. Conf. Manage. Data*, May 2017, pp. 1197–1210.
[13] L. Chen, Y. Gao, Z. Liu, X. Xiao, C. S. Jensen, and Y. Zhu, "PTRider: A price-and-time-aware ridesharing system," in *Proc. VLDB Endowment*, 2018, pp. 1938–1941.
[14] A. Corral, Y. Manolopoulos, Y. Theodoridis, and M. Vassilakopoulos, "Closest pair queries in spatial databases," in *Proc. ACM SIGMOD Int. Conf. Manage. Data (SIGMOD)*, 2000, pp. 189–200.
[15] P. Zhou, D. Zhang, B. Salzberg, G. Cooperman, and G. Kollios, "Close pair queries in moving object databases," in *Proc. Int. Workshop Geographic Inf. Syst. (GIS)*, 2005, pp. 2–11.
[16] H. U. Leong, K. Mouratidis, M. L. Yiu, and N. Mamoulis, "Optimal matching between spatial datasets under capacity constraints," *ACM Trans. Database Syst.*, vol. 35, no. 2, pp. 1–43, 2010.
[17] M. Chen, Z. Jia, Y. Gu, and G. Yu, "Top-$K$ probabilistic closest pairs query in uncertain spatial databases," in *Proc. Asia–Pacific Web Conf. Web Technol. Appl.*, 2011, pp. 53–64.
[18] Z. Li, K. C. K. Lee, B. Zheng, W.-C. Lee, D.-C. Lee, and X. Wang, "IR-tree: An efficient index for geographic document search," *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 4, pp. 585–599, Apr. 2011.
[19] F. M. Choudhury, J. S. Culpepper, and Z. Bao, "Batch processing of top-$K$ spatial-textual queries," *ACM Trans. Spatial Algorithms Syst.*, vol. 3, no. 4, p. 13, 2018.
[20] R. Zhong, G. Li, K. L. Tan, L. Zhou, and Z. Gong, "G-tree: An efficient and scalable index for spatial search on road networks," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 8, pp. 2175–2189, Aug. 2015.
[21] O. Kucuktunc and H. Ferhatosmanoglu, "λ-diverse nearest neighbors browsing for multidimensional data," *IEEE Trans. Knowl. Data Eng.*, vol. 25, no. 3, pp. 481–493, Mar. 2013.
[22] Y. Tao and C. Sheng, "Fast nearest neighbor search with keywords," *IEEE Trans. Knowl. Data Eng.*, vol. 26, no. 4, pp. 878–888, Apr. 2014.
[23] X. Cao, G. Cong, T. Guo, C. Jensen, and B. C. Ooi, "Efficient processing of spatial group keyword queries," *ACM Trans. Database Syst.*, vol. 40, no. 2, p. 13, 2015.
[24] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD Int. Conf. Manage. Data*, 1984, pp. 47–57.
[25] W. Shao, Y. Zhang, B. Guo, K. Qin, J. Chan, and F. D. Salim, "Parking availability prediction with long short term memory model," in *Proc. 13th Int. Conf. Green, Pervasive, Cloud Comput.*, 2018, pp. 124–137.
[26] W. Shao, S. Zhao, Z. Zhang, S. Wang, M. S. Rahaman, A. Song, and F. D. Salim, "FADACS: A few-shot adversarial domain adaptation architecture for context-aware parking availability sensing," in *Proc. IEEE Int. Conf. Pervasive Comput. Commun. (PerCom)*, Mar. 2021, pp. 1–10.
[27] K. K. Qin, W. Shao, Y. Ren, J. Chan, and F. D. Salim, "Solving multiple travelling officers problem with population-based optimization algorithms," *Neural Comput. Appl.*, vol. 32, no. 16, pp. 12033–12059, Aug. 2020.
[28] W. Shao, S. Tan, S. Zhao, K. K. Qin, X. Hei, J. Chan, and F. D. Salim, "Incorporating LSTM auto-encoders in optimizations to solve parking officer patrolling problem," *ACM Trans. Spatial Algorithms Syst.*, vol. 6, no. 3, p. 20, 2020.
[29] A. R. Khan, S. A. Madani, K. Hayat, and S. U. Khan, "Clustering-based power-controlled routing for mobile wireless sensor networks," *Int. J. Commun. Syst.*, vol. 25, no. 4, pp. 529–542, 2012.
[30] M. Ye, P. Yin, W.-C. Lee, and D.-L. Lee, "Exploiting geographical influence for collaborative point-of-interest recommendation," in *Proc. 34th Int. ACM SIGIR Conf. Res. Develop. Inf. (SIGIR)*, 2011, pp. 325–334.
[31] Y. Lyu, C.-Y. Chow, R. Wang, and V. C. S. Lee, "Using multi-criteria decision making for personalized point-of-interest recommendations," in *Proc. 22nd ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, Nov. 2014, pp. 461–464.
[32] J.-D. Zhang and C.-Y. Chow, "iGSLR: Personalized geo-social location recommendation: A kernel density estimation approach," in *Proc. 21st ACM SIGSPATIAL Int. Conf. Adv. Geographic Inf. Syst.*, 2013, pp. 1–10.
[33] W. Shao, F. D. Salim, A. Song, and A. Bouguettaya, "Clustering big spatiotemporal-interval data," *IEEE Trans. Big Data*, vol. 2, no. 3, pp. 190–203, Sep. 2016.
[34] W. Shao, F. Salim, T. Gu, T. Dinh, and J. Chan, "Traveling officer problem: Managing car parking violations efficiently using sensor data," *IEEE Internet Things J.*, vol. 5, no. 2, pp. 802–810, Apr. 2018.
[35] [Online]. Available: http://www.chinadaily.com.cn/

**JINE TANG** received the Ph.D. degree from the China University of Geosciences, Beijing, China, in 2014. She is currently an Associate Professor with the School of Artificial Intelligence, Hebei University of Technology, Tianjin, China. Her research interests include spatial-temporal database and IoT search.



**YUPENG WANG** received the M.S. degree in transportation information engineering from Beihang University, Beijing, China, in 2021, where he is currently pursuing the Ph.D. degree with the School of Electronics and Information Engineering. His research interests include traffic forecasting and spatio-temporal data mining.

**WEIJING LIU** received the M.Sc. degree from Yanshan University, in 2016. She is currently an Engineer with the Tianjin Key Laboratory of Aerospace Intelligent Equipment Technology, Tianjin Institute of Aerospace Mechanical and Electrical Equipment. Her research interests include hardware circuit design and software development.

**XILING LUO** received the B.E. and Ph.D. degrees from the School of Electronics and Information Engineering, Beihang University, Beijing, China, in 1996 and 2003, respectively. He is currently a Professor with the Research Institute for Frontier Science, Beihang University. He is also the Principle Investigator of the Comprehensive Transportation Big-Data Research Center, Hangzhou Innovation Research Institute of Beihang University. His research interests include comprehensive transportation information systems and air traffic managements.

**ZHANGBING ZHOU** received the Ph.D. degree in computer science from the Digital Enterprise Research Institute (DERI), Galway, Ireland, in 2010. He worked as a Software Engineer with Huawei Technology Company Ltd., Beijing, China, for one year and served as a member of the Technical Staff at Bell Laboratories, Lucent Technologies, Beijing, for five years. He is currently a Professor with the China University of Geosciences, Beijing, and as an Adjunct Professor at TELECOM SudParis, Evry, France. He has authored over 100 refereed papers. His research interests include process-aware information systems and sensor network middleware. He has served as an Associate or a Guest Editor of over ten journals.