Spatial Intelligence - Route Prediction and Planning Base on Geospatial Information

by

Ting Shen

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 7, 2022

Keywords:

Approved by

Wei-Shinn Ku, Professor of Computer Science and Software Engineering
Xiao Qin, Professor of Computer Science and Software Engineering
Kai Chang, Professor of Computer Science and Software Engineering
Alvin Lim, Professor of Computer Science and Software Engineering

Abstract

With the advances in localization techniques and popularity of mobile devices, Spatial Intelligence penetrated people's lives. What is Spatial Intelligence? It is 1) able to provide "smart" and intelligent services to assist people to make decisions, 2) based on efficient data processing and interactive data analytic, and 3) using big data on Location-based Social Networks. In this proposal, I focus two problems of Spatial Intelligence about route prediction and planning.

First, we study the problem of recommending time-sensitive venue sequence for mobile users using their check-in footprints on LBSNs. Most of the current studies in Point of Interest (POI) recommendation and prediction fail to address the following key challenges: (1) how to recommend an optimal time-sensitive visit sequence where each point's time is specified by users, (2) how to handle the scenario where the user-location matrix is very sparse (i.e., each user has a very limited number of check-ins, or to say, cold-start users), and (3) how to dig deep into the user's behavior to assist the recommendation system. Motivated by the challenges above, we propose a predictive framework that enables time-sensitive location sequence recommendation leveraging both the users' preference and social opinions, especially for cold-start users.

Next, we presents an exact solution and a heuristic solution to a UAV-assisted parcel delivery problem, in which UAVs can only be operated in Visual-Line-Of-Sight (VLOS) areas. In our proposed problem, we assume that trucks travel on road networks, and UAVs move in Euclidean spaces and can launch at any locations on roads. We first demonstrate the overview of our exact solution that iterates all permutations of destinations for an optimal delivery route. Given a specific delivery order, an intuitive approach needs to check all possible locations on roads in the VLOS areas and find a globally optimal location for every

destination if UAVs are used for delivery. To avoid high computational cost of searching the optimal location at run-time, we propose an advanced index-based alternative, which computes optimal delivery routes in a pre-processing stage. Due to the nature of NP-hard problems, we also propose a heuristic approach that utilizes delivery groups for the proposed problem of practical size. All proposed solutions are evaluated through extensive experiments.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Geospatial intelligence is a broad field that encompasses the intersection of geospatial data with social, political, environmental and numerous other factors. My research focus two main problems.

1.1 Location Recommendation Problem

With the rapid development of the positioning technology and personal devices such as smart-phones, a number of Location-based Social Networks (LBSNs) have emerged, e.g., Foursquare and GeoLife. With these applications, users are able to share their location data. For example, a customer may check in and leave a comment on a LBSN site for a restaurant she just visited. These records imply extensive knowledge about a user's interests and behavior, and mining such data has become an interesting research topic. Check-in records have two very important components, the location and the time. Because of the particularity of check-in record, it is possible to analyze user's action according to the user's mobility and activities in the physical world.

A location recommendation often simultaneously considers the following factors. (1) User record: a user is more likely to go to a category of locations she visits frequently. For example, food lover may be more interested in the restaurants. (2) Behavior of other users: the choice of other users are of high reference value, especially for the semantic similar users (i.e., they have similar interests in what types of restaurants to visit) or spatial similar users (they have similar range of activities). (3) User's current location: User's location limits the searching range, and therefore the places nearby have higher priority. (4) The current time: a user's visiting location is impacted greatly by the time of the day. Most of the existing

location-based recommender systems have the following limitations: (1) providing a list of separate locations which match a user's personal interest without the support of continuous location sequence recommendation[46] , (2) requiring a large number of check-in data from the target user and other users to make the predictive systems work (i.e., cold-start problem), (3) inability to consider temporal factors when making venue recommendation because user's preference on venues are actually time-dependent or time-sensitive. In this paper, we aim to develop a novel framework for recommending an optimal time-sensitive visit sequence that best matches the time slots specified by a target user.

However, developing such a recommendation system faces a few challenges. First, how can we make personalized location sequence recommendation by taking into account the impact of time on location sequence? For example, people tend to leave their workplaces to visit restaurants at lunchtime. Second, how can we make accurate location sequence recommendation even if the location-user matrix is sparse? Due to the sparsity of check-in data, it is insufficient to train an accurate predictive model by only using a user's own check-in data in order to make prediction or recommendation on various POIs. Furthermore, a user's preferences are not generally binary decisions but have a variety of granularity, such as "Food - Chinese food - Szechuan food". So an ideal location sequence recommender should be able to capture such hierarchy of venue categories. Last but not least, different people have different movement patterns and such spatial preferences are too complex to be represented using Euclidean distance only.

To the best of our knowledge, our work is the first attempt to address the time-aware location sequence recommendation in support of sparse user-location matrix. By taking full advantages of the similarities between mobile users, our proposed framework does not suffer from the cold-start problem and works well even if the target user has very few footprints (check-ins) on a LBSN site. The key contributions of this paper are as follows:

- We propose a new framework, TLSR+, that enables time-sensitive location sequence recommendation in support of sparse user-location matrix. TLSR+ extends our prior

work, TLSR [42], and makes location sequence recommendation by considering the following: individual preference, social opinions, geospatial factors, and users' temporal behavior patterns.

- In order to incorporate social options, we local experts (i.e., users with rich local knowledge) based on the HITS algorithm. We further screen out similar users considering both semantic and spatial factors.

- In semantic similarity modeling, we build a venue categorical semantic tree, i.e., Weighed Category Hierarchy (WCH), to capture each user's venue categorical preference based on users' check-in histories. Given any two users, their semantic similarity score can be calculated as the weighted sum of the Jensen-Shannon divergence of each level in their corresponding Normalized Weighed Category Hierarchies (NWCHs).

- In spatial similarity modeling, we employ Gaussian Mixture Model (GMM) to capture users' geographical movement patterns. The in-between user spatial similarity scores can then be estimated by computing the average log-likelihood that a user's check-in records fit with the GMMs learned from other users.

- By taking advantage of the check-ins from both semantically and spatially similar local experts, we train a Hidden Markov Model (HMM) for each user to predict her likelihood to visit each location category at any given time.

- To improve the accuracy of recommendations, we divide and model the check-in data depending on different time bucket. For the inquiry from different time bucket, we select the corresponding model for calculation. We choose weekday and weekend as a division because of significant differences in user behavior.

- We conduct extensive experiments to evaluate the performance of our approach against three baseline methods, MPV, LPR [6], and TLSR [42] using four real dataset (NYC Weekdays, NYC Weekends, Tokyo Weekdays, Tokyo Weekends). Results show that

TLSR+ significantly outperforms all baseline methods in temms of the hits rate and MRR.

The Chapter 3 present our time-aware location-based recommendation system. And experimental evaluation is followed by.

1.2 Ground/Aerial Parcel Delivery Problem

The Travelling Salesman Problem (TSP) finding an optimal route for a salesman, who plans to travel to each of a list of cities exactly once and return to the home city, has been extensively studied and applied in the parcel delivery service and other real applications [27] [20]. On the other hand, Unmanned Aerial Vehicles (UAVs), or drones, are developed for assisting traditional delivery vehicles (such as trucks) since the delivery could be completed in a shorter time period with lower maintenance cost by using UAVs in specific circumstances. Many UAV-assisted delivery projects have been initialized. For example, Prime Air is designed to deliver packages to customers in 30 minutes or less at Amazon [4]. DHL will start a project for delivering medications and other urgently needed goods to the North Sea island of Juist by DHL parcelcopters [23]. The project Wing targets delivering aid to isolated areas for disaster relief [22].

As UAVs are not limited by established infrastructure (e.g., roads), a new delivery model, a truck and a UAV, was proposed for parcel delivery [37]. In the model, every truck is equipped with a UAV and all packages can be delivered by either of the two. Fig. 1.1 and Fig. 1.2 display examples of TSP and the UAV-assisted parcel delivery problem. The delivery route starts at a distribution center h (indicated by a box), and eventually returns to the distribution center after reaching five destinations ($\{d_1, d_2, d_3, d_4, d_5\}$ indicated by circles). The delivery truck (indicated by the triangle) could stop at any locations on the road. In the example of the UAV-assisted parcel delivery problem in Fig. 1.2, the truck stops on road r and the UAV flies to d_4 for delivery.

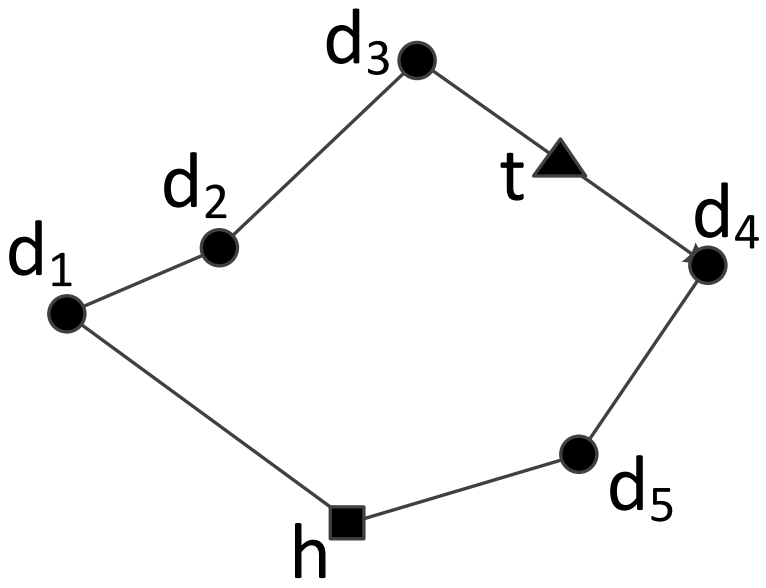


Figure 1.1: An example of TSP in a Euclidean space. The line segments indicate a delivery route.

The travelling salesman problem is a well-known NP-hard problem in the field of combinatorial optimization [38], and the new UAV-assisted parcel delivery problem is more challenging. First of all, the fastest delivery route among destinations is usually assumed to be computed in a pre-processing stage in TSP. However, if packages could be delivered by either the truck or the UAV, the truck can stop at any places on the road and the UAV can be used for delivery. So, the fastest route among locations where the truck stops cannot be pre-determined, and may greatly vary case by case. Second, only one search space (either a Euclidean space or a road network) is usually considered in TSP, but the UAV-assisted problem assumes that the truck travels on road networks while the UAV moves in Euclidean spaces. Thus, the fastest delivery route may consist of paths in both search spaces, which significantly increases the complexity of the problem. Third, many variants of UAV-assisted problems have been investigated [37] [24] [47]. Nevertheless, none of these studies takes UAV regulations into account. For example, the Federal Aviation Administration (FAA) does not allow UAVs to be operated beyond the Visual-Line-Of-Sight (VLOS) of operators in the United States [1]. Existing solutions are not applicable to any VLOS compliance problems

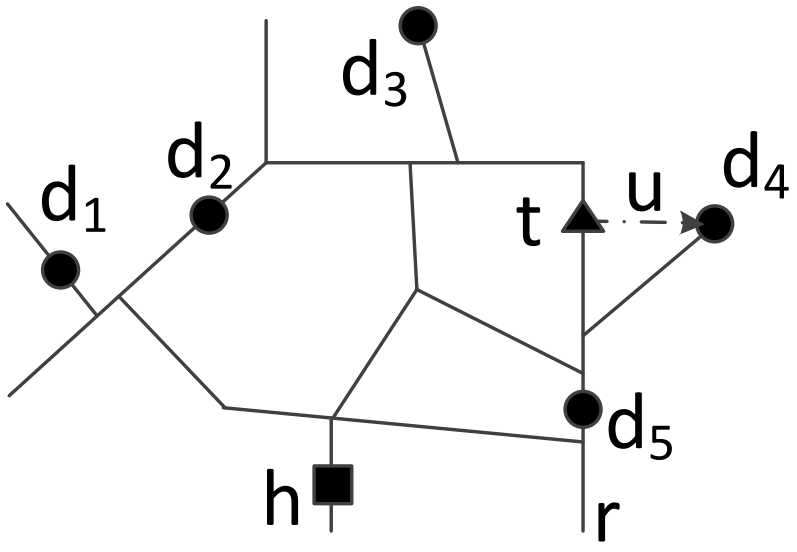


Figure 1.2: An example of UAV-assisted parcel delivery problem. The line segments indicate the road network.

because the optimal delivery route may vary greatly by the VLOS distance and speeds of the truck and the UAV.

Therefore, we propose a novel Ground/Aerial Parcel Delivery Problem (GAPDP) with consideration of VLOS compliance. We develop an exact solution and a heuristic solution for the problem in this paper. In our exact solution, we check all permutations of destinations for the fastest delivery route. To calculate the fastest delivery route for destinations in a given order, we present an index-based exact approach that relies on an index over the VLOS areas for reducing the cost of the route calculation. This approach finds the fastest delivery route for every destination from all its entrances to its exits at a pre-processing stage. The fastest routes can be retrieved from the index, and our method can "jump" from one destination to the other without computing routes in the VLOS areas. Additionally, as the proposed problem is NP-hard, we also propose a heuristic solution that utilizes delivery groups for the proposed problem of practical size. All proposed solutions are evaluated through extensive experiments.

The contributions of this study are summarized below:

- We propose and formally define a new Ground/Aerial Parcel Delivery Problem (GAPDP), in which packages could be delivered by either a truck or a UAV.
- We develop an index-based exact solution that pre-computes the fastest delivery routes in VLOS areas of destinations.
- We propose a heuristic solution that produces an approximation for problems of practical size.
- We evaluate our solutions through extensive experiments over a real-world road network.

The Chapter 4 shows our solution of the GAPDP problem.

Chapter 2

Literature Review

In this chapter, we review previous studies related to the recommendation system, travelling salesman problem, and UAV-assisted parcel delivery problems.

2.1 Stand-alone Location Recommendation

Most of the related researches focus on stand-alone location recommendation. Without taking personal preferences into account, Generic location recommendation systems, such as [55], encapsulate the public opinions on locations to provide people with the most popular venues in a city. In this work, they model multiple individuals' location histories with a tree-based hierarchical graph. Then a HITS-based inference model, which regards an individual's access on a location as a directed link from the user to that location, is proposed to predict the popularity of a location and the knowledge of a user. However, this kind of method is not customized for the user. They can't make an accurate recommendation since everyone's interest is varied.

Some location recommendation systems suggest locations by matching user's profile data against the location meta data. In [40], Ramaswamy et al. presents a social network-based recommender system that has been explicitly designed to work with low-end devices. They analyze information such as customer's address books to estimate the level of social affinity and combined them with the spatio-temporal context to identify the recommendations to be sent to an individual user. The social affinity computation and spatio-temporal contextual association are continuously tuned through user feedback. In [33], Kodama et al. select candidates taking into account the user's current location and preferences and using

semantic data. Furthermore, Skyline operator is also applied to make the final recommendations. However, the requirement of needing users' complete profile is a big shortcoming. Furthermore, this kind of systems may suffer from the recommendation quality issue.

Using users' location histories for making recommendations avoids the shortcoming of relying solely on profile data. And capturing the ratings from the other users improves the quality of recommendation. Motivated by the fact that user will share location preferences with similar users, [12, 28, 50] use Collaborative Filtering (CF) models to give personalized location recommendations that take into account other users' ratings. Ye et al. 2010 [50] uses the ratings of a user's friends to select candidate locations. In [16], they further extend the solution to consider the situation that user is traveling out of town. In [53], Ying et al. extend the recommendation system by considering the popularity of the candidate venues by analyzing the large scale user check-in behavior. [51] store users location histories using a matrix, where each row denotes a user and each column represents a location. [6] handles the cold-start mobile users by storing a user location history into the category space and modeling user preferences using WCH. In [52], Yin et al. further extend the problem by proposing a location-content-aware probabilistic generative model to quantify both the local preference and the item content information in the recommendation process.

There are also some studies in the literature that employ machine learning-based approaches in POI prediction or recommendation. [5] presents a system that clusters users' significant locations from GPS data and predicts location based on the Markov models. [34] uses a history of a driver's destinations to predict where a driver is going as a trip progresses. [3] designs a system based on the generation of a hidden Markov model from the past GPS log and the current location to predict a user's destination. [32] combines k-nearest neighbor and decision tree for predicting user's destination based on hidden Markov model. Unfortunately, these works neither solve the sequential recommendation problem, nor consider the temporal effect.

2.2 Sequential Location Recommendations

Sequential location recommender systems recommend a series of locations to a user (such as a visiting route in a city), based on their preferences and the time constraints. For example, [44] provides personalized suggestion of a sequence of following visiting spots leveraging the geo-tagged photos. The recommendation is based on the data mining techniques to extract and differentiate the preferences of various users. In [54], Zhao et al. propose a visual feature enhanced tour recommender system by integrating visual features into the collaborative filtering model, to learn user interests by leveraging the historical travel records. These methods heavily rely on the geo-tagged photos and The effect of different times was not considered. Moreover, the complexity of computing is relatively high.

Other researches make sequential location recommendations by mining GPS trajectory. [15] present a graph model for socio-spatial networks that store information on frequently traveled routes and implement a query language that consists of graph traversal operations. In [9], the authors propose a route recommender system that comprises two components, familiar road network construction and route planning, which mined from the user's historical trajectories. In [21], Ge et al. develop a mobile recommender system which has the ability in recommending a sequence of pick-up points for taxi drivers or a sequence of potential parking positions, by learning energy-efficient transportation patterns from trajectories.

To the best of our knowledge, our work is the first attempt to address the time-aware location sequence recommendation in support of sparse user-location matrix. Taking full advantages of the similarities between mobile users, our proposed framework does not suffer from the cold-start problem and works well even if the target user has very few footprints (check-ins) on a LBSN site.

2.3 Travelling Salesman Problem

The Travelling Salesman Problem (TSP) is an NP-hard problem [38]. An intuitive method for an exact solution of the problem is to check all permutations of destinations, and return the shortest route among those permutations, the computational complexity of which is $O(n \cdot n!)$, where n denotes the number of destinations. Held and Karp reduced the computational cost to $O(n^2 \cdot 2^n)$ by utilizing dynamic programming [25]. Jones and Adamatzky approximated TSP in a Euclidean space by shrinking "blob" [31]. In their method, a "blob" of material emerging from low-level interactions of particles is placed over destinations. The shrinkage process automatically and morphologically adapts to the configuration of destinations. And finally, a TSP tour is captured by tracing the perimeter of the blob among the destinations. Moreover, there are many studies that have investigated heuristic solutions for TSP [20]. A hybrid heuristic approach was developed for the multi-commodity pickup-and-delivery travelling salesman problem [26]. In the pickup-and-delivery TSP, each customer could be either a producer or a consumer of a package, or both. In other words, a package is assumed to be picked up at a customer, and then delivered to another customer in the delivery process. GELS-GA is a hybrid metaheuristic algorithm for multiple Travelling Salesman Problem (mTSP) [29]. Rather than the traditional TSP, mTSP has two or more salesmen, and all of them must return to the places where they start. GELS-GA combines Gravitational Emulation Local Search (GELS) algorithm and Genetic Algorithm (GA) for achieving the global optimum in a high possibility. Xu *et al.* also proposed a two-phase heuristic algorithm for mTSP [48]. Their method specifically considers workload balance and minimizes the overall travelling cost. They improved the K-means algorithm by grouping visited cities based on their locations and capacity constraints, and then designed a route planning algorithm for a delivery route. However, aforementioned studies are different from our proposed GAPDP problem; the shortest delivery routes among destinations in GAPDP depend on speeds of trucks and UAVs, the delivery order, the delivery method for each destination, UAV regulations, and more.

2.4 Route Planning Query

The route planning query is a problem that finds an optimal route under a set of constraints. The Optimal Sequenced Route (OSR) was proposed to search for an optimal route starting from a given source location and travelling through a set of locations in different types in a given order. Sharifzadeh *et al.* developed an additively weighted Voronoi diagram based method to incrementally build the OSR in both vector and metric spaces [41]. Li *et al.* proposed a trip planning query without a particular order. Given a set of objects in different types, the optimal route connects the given starting location and destination, and passes through exactly one object in each object type on the route [35]. The a-autonomy shortest path and k-stops shortest path queries were studied in [45]. The a-autonomy shortest path consists of a sequence of objects from the source to the destination, where the distance of any two consecutive locations on the path is not greater than a. The k-stops shortest path query finds a route that consists of k intermediate stops on the path. Chen *et al.* proposed a partial sequenced route query, which satisfies a set of partial visiting orders in the travel plan [10]. However, the major difference from our study is that our route planning query searches the optimal route in both Euclidean space and road networks by utilizing trucks and UAVs. All aforementioned queries do not consider the assistance of UAVs in their route planning search.

2.5 Drone-assisted Parcel Delivery Problem

Agatz *et al.* studied the Travelling Salesman Problem with Drones (TSP-D), formulated the problem as an Mixed Integer linear Programming (MIP) model, and developed route first-cluster second heuristic approaches based on local search and dynamic programming [2]. Ha *et al.* proposed two heuristic methods, either of which utilizes route-first cluster-second or cluster-first route-second strategies to solve TSP-D [24]. They used a new mixed integer programming formulation in the cluster step for both heuristics. Wang *et al.*

focused primarily on the worst case of a vehicle routing problem with drones [47]. They assumed that a truck could be equipped with many drones, and their investigation showed that the results of the worst case depend on the number of drones on the truck and the relative speed of the drones to the truck. Murray and Chu proposed the flying sidekick TSP problem, in which customers can be served by either a driver-operated delivery truck or a UAV [37]. They solved the problem by utilizing an MIP formulation. Moreover, they also proposed a heuristic approach for parallel drone scheduling TSP problem. Ferrandez *et al.* explored the delivery time and energy consumption of a truck-drone delivery network [17]. Their proposed algorithm aims at minimizing the time of delivery by utilizing K-means clustering methods, and determining the optimal number of launch sites and drones per truck. Dorling *et al.* proposed two solutions for multi-trip vehicle routing problems [14]. Specifically, they first demonstrated an energy consumption model of drones, in which the energy consumption is approximately linear to the payload and battery weight. Then, they developed a cost function by using the energy consumption model and drone reuse. Finally, they proposed a method that minimizes costs by considering the limit of delivery time, and a method that minimizes the delivery time under a budget constraint. A Randomized Variable Neighborhood Descent (RVND) heuristic method was proposed to TSP-D problem [13]. In the method, practical restrictions, such as the flying time limit of drones and the limit of payload, are considered, and the RVND heuristic is used in local searches to find an optimal delivery route. Their experimental results show that the help of drones can save up to 20% of time in the last mile delivery.

However, all methods mentioned above are not applicable to our proposed problem due to the difference in assumptions of the problems. None of these studies considers the VLOS restriction. The optimal delivery route may vary greatly by setting different VLOS distances or speeds of the truck and the UAV. Many studies output an approximation [2] [24] [37]; while we propose an exact solution to the VLOS-compliance UAV-assisted parcel delivery problem. Moreover, Murray and Chu assumed that the truck could move on road networks

while the UAV is in flight; but the truck and the UAV cannot be operated simultaneously in our problem because the driver can only either drive the truck or operate the UAV at a particular time. In addition, in many countries, UAVs are not allowed to land at or takeoff from a moving vehicle. Agatz *et al.* assumed that the UAV can only land on and depart from the truck while it is parked at a customer location, while the truck is allowed to park at any locations on roads in our problem. More than one UAV can be used with a truck in the problems proposed by Wang *et al.* and Ferrandez *et al.*; while there is only one UAV available with a truck for delivery in our problem, as each UAV requires a dedicated operator to handle.

Chapter 3

Time-aware Location Sequence Recommendation for Cold-start Mobile Users

3.1 Overview

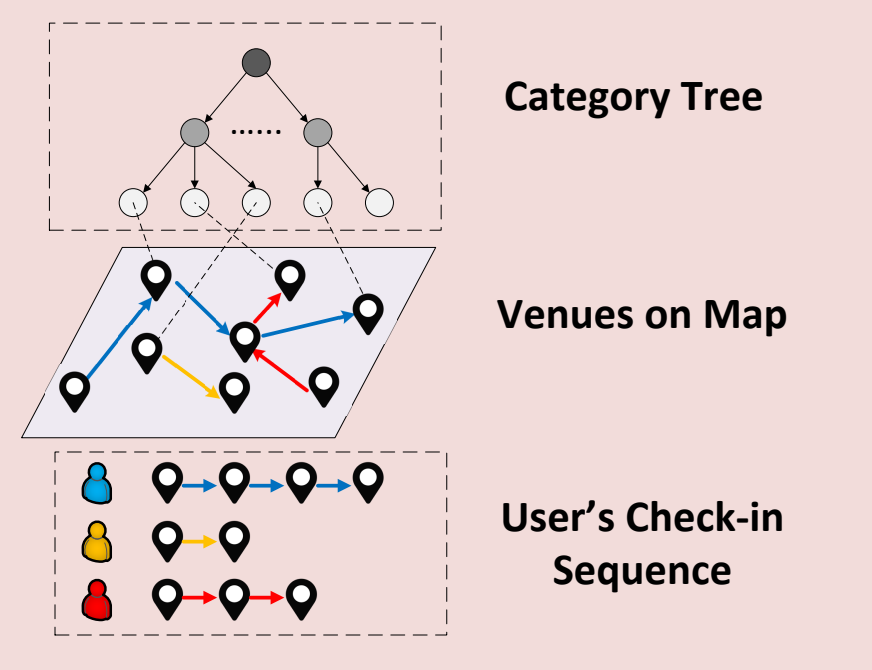


Figure 3.1: User Check-ins on Social Network.

In this section, we introduce the motivating example and the architecture of our time-aware location sequence recommender system.

3.1.1 Preliminary

In a location-based social network, users and venues are the basic elements. A user u maintains her profile information, such as user ID, name, gender and home location. A venue v holds the information of the venue ID, name, and real-world location. Venues can be

visited by users, and users save their check-in record in the system. As illustrated in Figure 3.1, the system maintains a sequence of check-in record for each user. Each check-in record shows the timestamp when the user check in and the venue where the user visit. A venue is also associated with a set of categories. In Foursquare the categories are represented by a category tree. For example, category "Food" includes category "Chinese Restaurant" and "American Restaurant", and "Chinese Restaurant" includes "Cantonese Restaurant" and "Szechuan Restaurant".

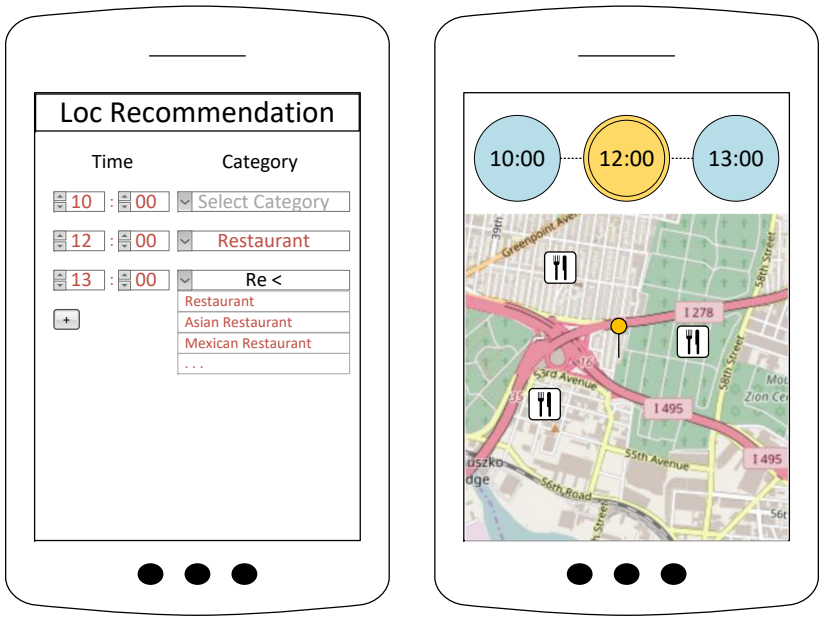


Figure 3.2: Example of An Application Scenario.

3.1.2 Application Scenario

As shown in Figure 3.2, our system allows a user to set the time schedule by entering a sequence of timestamps when she wants to visit multiple venues. The length of the sequence is decided by the user. For example, a user may set 10:00, 12:00, and 13:00 as the time for three visit activities. Then the user decides the search range by zoom in/out the map. The

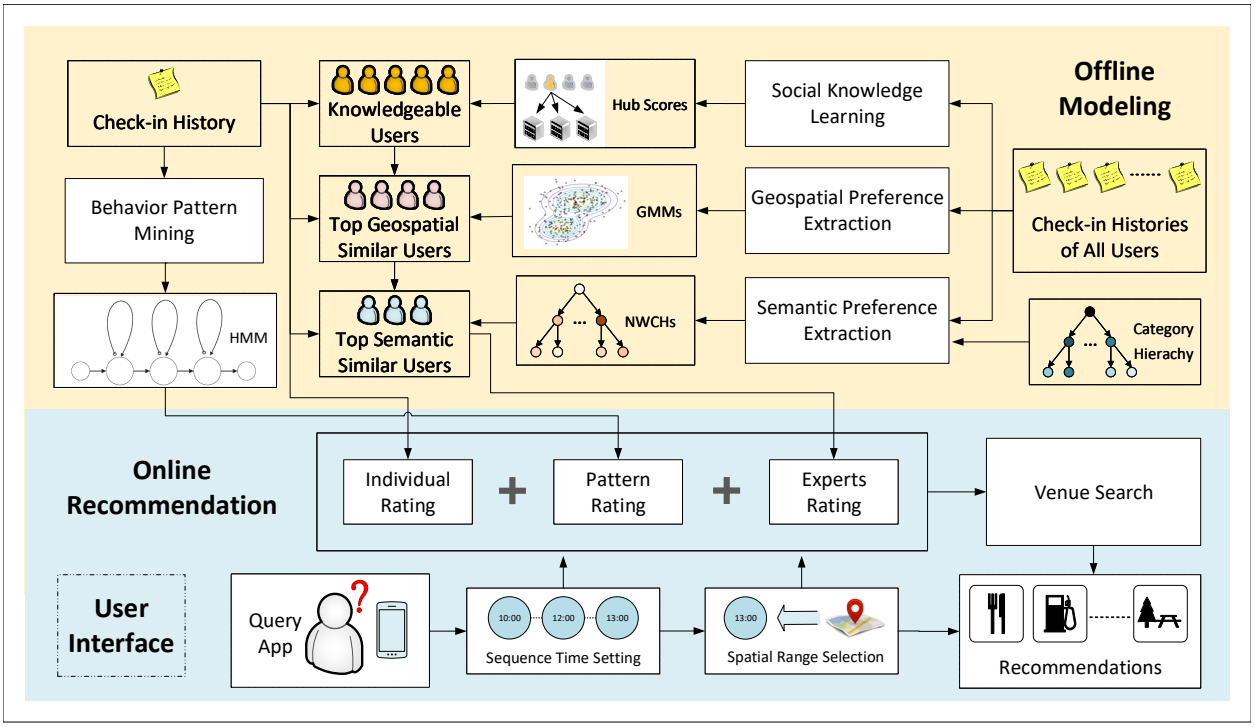


Figure 3.3: System Architecture.

user can either specify a search range for all the points, or specify a range for each point-in-time. The user can also choose a category for each point. The system will generate a list of venues with size N for each time slot that matches her input which give consideration to her preferences. For example, as shown in Figure 3.2, 3 restaurants are recommended for the second time slot.

3.1.3 System Architecture

Figure 3.3 shows the system architecture, which consists of offline modeling and online recommendation.

Offline Modeling.

The offline modeling is comprised of two major components, 1) Personalized Experts Discovery and 2) Behavior Pattern Mining. In personalized experts discovery, the personalized local experts for target users will be discovered. First, we search the users with rich

knowledge about different location categories. Hyperlink-Induced Topic Search algorithm is employed to evaluate a user's level of expertise. Second, we find out the most spatially similar users to the target user. In order to calculate spatial similarity between any two users, we create a Gaussian Mixture Model for each user according to her check-in records. The input for the GMM is a two-dimensional vector with the longitude and latitude of each check-in point. To calculate user u's similarity score to user v, we calculate the average probability of u's check-in points according to v's GMM. At last, we calculate similarity score in terms of their semantic preference and keep the most semantically similar users as the final experts. Given a predefined category hierarchy, we first model each user's semantic preferences using an NWCH. This extracts the features of users' semantic preferences. To compute the similarity score between two NWCH, the values at the same level are normalized and Jensen-Shannon divergence is employed. In behavior pattern mining, we adopt the Hidden Markov Model to mine the target user's behavior pattern. By treating the categories of venues as states, and time as observations, an HMM can be learned from a target user's check-in history.

Online Recommendation.

The online recommendation part has three components. 1)Individual Preference, 2) Pattern Prediction and 3) Social Opinion. The first component computes every location's individual rating. We extract the venues in range from a user's check-in history, and compute a score for each venue by counting the frequency of the visit. The second component predicts the category that the user will visit. Giving the observation sequence (time slots), a forward algorithm is employed to decode the category sequence. With the probabilities of each category we got, a pattern score is calculated for each venue in range. The third component is the experts score computation, which is based on the personalized local experts we found in offline approach. Finally, the three score will be summed up, and the top N venues with highest score will be selected as the locations for recommendation.

3.2 Personalized Experts Discovery

In this section, we generates a list of experienced "experts" tailored to each user's preference. The process is composed of three steps: 1) knowledgeable users discovery, which selects the candidate local experts base on their experience, 2) geographic similar users filter, which filter the experts base on the user's geographic preference, and 3) semantic similar users filter, which selects more professional experts according to the semantic preference of the user from the candidates.

3.2.1 Knowledgeable Users Discovery

Some people have rich visiting experience of some specific categories, such as"Asian Restauran" or "Park", we call them "Knowledgeable Users". These users, as local consultants, can find high quality venues which can attract other users. To find out the knowledgeable users, this component computes the users' expertise of each category base on users' check-in history.

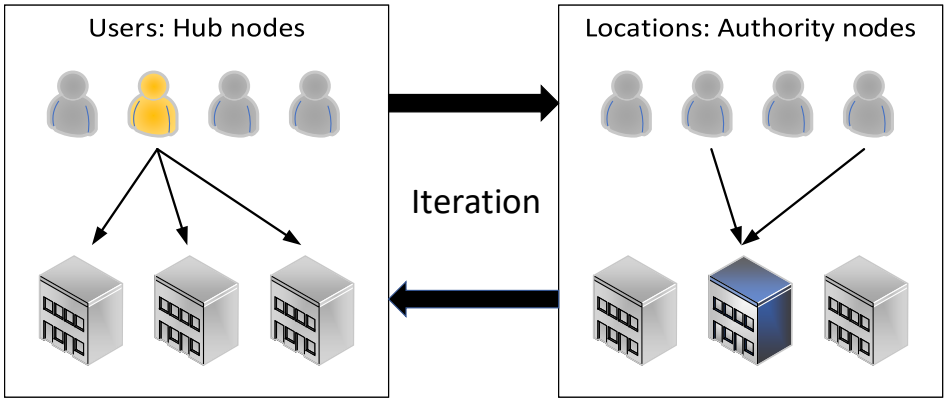


Figure 3.4: HITS Algorithm for Social Knowledge Learning

We divide the users' check-in history into groups by the categories of venues, then apply Hyperlink-Induced Topic Search (HITS) algorithm [6] for each group. The algorithm calculates the hub score for each user, which represents its knowledge, and the authority score for each venue, which denotes its popularity. As shown in 3.4, a directed link between

a user and a venue means the user has been to the place. Thus a user-location matrix for each category can be generated. A user's hub score can be represented by the sum of the authority scores of the venues that visited by the user. Likewise, a venue's authority score can be represented by the sum of the hub scores of the users who have visited this venue. The equations are shown below, where $v.c$ denotes the category of venue v, $v.\mathcal{U}$ denotes all the users that visited venue v.

$$v.authority = \sum_{u \in v.\mathcal{U}} u.hub(v.c) \tag{3.1}$$

$$u.hub(c) = \sum_{u.v \in c} v.authority \tag{3.2}$$

With the initial authority and hub scores to be set as the number of a user's visits, the authority and hub scores could be calculated by iterative method using user-category matrix M.

$$\mathcal{A}_n = M^T \cdot M \cdot \mathcal{A}_{n-1} \tag{3.3}$$

$$\mathcal{H}_n = M \cdot M^T \cdot \mathcal{H}_{n-1} \tag{3.4}$$

We select Knowledgeable Users by algorithm 1. One thing to note is that the size of the result could be less than N_k, because the users selected for different categories are likely to overlap. This algorithm could be implemented at any level of the category tree, which depends on how fine of the categorization of the data set.

3.2.2 Geographic Similar Users Filter

To find out experts that are more relevant, this module identify the experts that have preferences on similar geographical areas with the user.

Algorithm 1: Knowledgeable Users Selection

Input: (1) Total number of users selected N_k, (2) A user's visit records $u.V$, and (3) Selected level of category tree l

Output: A set of selected Knowledgeable Users U

$U \leftarrow \emptyset$;
$k \leftarrow \text{Size}(u.V)$;
for c in categories of level l **do**
    `// Find out records that locations belong to c`
    $V_c \leftarrow \text{Filter}(u.V, c)$;
    `// Select top-k users based on hub scores`
    $k_c \leftarrow \text{Size}(V_c)$;
    $n \leftarrow |N_k * k_c/k|$;
    $U_c \leftarrow \text{Top}(n, c)$;
    $U \leftarrow U \cup U_c$;
end
if $u \in U$ **then**
    $U \leftarrow U - u$
end
return U

According to common sense, users have preferences on different geographical areas. It is because there may be several frequently visited neighborhoods for a particular user (e.g., around the user's home or around the user's work place). As shown in Figure 3.5, the distribution of the check-ins from a sample user indicates some geographical clusters where she frequently checked in. Gaussian Mixture Model (GMM) is appropriate for analysing this kind of data. Therefore, we use GMM to cluster check-in locations.

A Gaussian mixture model is a probabilistic model that assumes all the data points are generated from a mixture of a finite number of Gaussian distributions with unknown parameters[36]. Some prior works [11, 19, 56] utilize the GMM to model individual users' check-in behaviors in LBSNs. A Gaussian mixture model can be represented by $\lambda = \{\phi, \mu, \Sigma\}$, and the equations to evaluate a vector $\overrightarrow{z}$ are shown as follows:

$$p(\overrightarrow{z}) = \sum_{i=1}^{K} \phi_i \mathcal{N}(\overrightarrow{z} \mid \overrightarrow{\mu_i}, \Sigma_i) \tag{3.5}$$

$$\mathcal{N}(\overrightarrow{z} \mid \overrightarrow{\mu_i}, \Sigma_i) = \frac{1}{\sqrt{(2\pi)^K |\Sigma_i|}} \exp\left(-\frac{1}{2}(\overrightarrow{z} - \overrightarrow{\mu_i})^T \Sigma_i (\overrightarrow{z} - \overrightarrow{\mu_i})\right) \tag{3.6}$$

$$\sum_{i=1}^{K} \phi_i = 1 \tag{3.7}$$

By training a GMM for each user based on her coordinates of check-ins, we could find out top geographic similar users for any user. Assume that the GMM for user u is λ_u, and $Z = \{\overrightarrow{z_1}, ..., \overrightarrow{z_T}\}$ are the feature vectors (a vector contains the coordinates of a check-ins, $z_t = (lat_t, lon_t)$) of user u'. We can compute the average log-likelihood using

$$\log p(Z \mid \lambda_u) = \frac{1}{T} \sum_{t=1}^{T} \log p(\overrightarrow{z_t} \mid \lambda_u), \tag{3.8}$$

which is the spatial similarity score from user u' to user u. The larger the value is, the more similar the two users are in terms of their spatial preferences. From the users previously selected, we choose candidates with top N_g scores. Normally, the scope of people's activities on weekdays is different from that on weekends. So by separating the data, we calculate two list of the candidates – weekday experts and weekend experts.

Expectation maximization (EM) is the most commonly used algorithm to estimate the GMM's parameters when we know the component number K. Expectation maximization for mixture models consists of three steps.

Initialization Step. This step is to assign parameters to model base on the dataset $X = \{x_1, ..., x_n\}$. To initialize the means $\hat{\mu}_1, ..., \hat{\mu}_k$, we randomly assign samples from the dataset X. For example, choose x_1, x_{10}, and x_{24} as $\hat{\mu}_1$, $\hat{\mu}_2$, and $\hat{\mu}_3$ when $K = 3$. All the component variance will be set to the sample variance with formula $\hat{\sigma}_1^2, ...\hat{\sigma}_K^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2$. At last, all component distribution will prior estimate to the uniform distribution, i.e $\hat{\phi}_1, ..., \hat{\phi}_K = \frac{1}{K}$.

Expectation Step

This step first calculate the expectation of the component assignments C_k for all the data sample x_i. Then $\hat{\gamma}_{ik}$, the probability that x_i is generated by component C_k, will be computed. For $\forall i, k$:

$$\hat{\gamma}_{ik} = \frac{\hat{\phi}_k \mathcal{N}(x_i \mid \hat{\mu}_k, \hat{\sigma}_k)}{\sum_{j=1}^{K} \mathcal{N}(x_i \mid \hat{\mu}_j, \hat{\sigma}_j)} \tag{3.9}$$

Maximization Step

This step updates the parameters $\hat{\phi}$, $\hat{\mu}$, and $\hat{\sigma}$ using the expectations calculated in the expectation step:

$$\hat{\phi}_k = \sum_{i=1}^{N} \frac{\hat{\gamma}_{ik}}{N} \tag{3.10}$$

$$\hat{\mu}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} x_i}{\sum_{i=1}^{N} \hat{\gamma}_{ik}} \tag{3.11}$$

$$\hat{\sigma}_k = \frac{\sum_{i=1}^{N} \hat{\gamma}_{ik} (x_i - \hat{\mu}_k)^2}{\sum_{i=1}^{N} \hat{\gamma}_{ik}} \tag{3.12}$$

The entire iterative process repeats until the algorithm converges, giving a maximum likelihood estimate.

3.2.3 Semantic Similar Users Filter

This module filter users who have the same preferences from the candidates. This is motivated by the fact that the individuals with same interests tend to visit the same place. However, a person always has multiple interests. Furthermore, each class of interest could have different granularities. For example, a user may have interest in "Food" and "Entertainment", and in the category "Food" she likes "Chinese Restaurant".

To solve this problem, we extracts a user's semantic preference based on the categories of her visited places referring to Jie Bao's method [6]. Figure 3.6 shows our process of

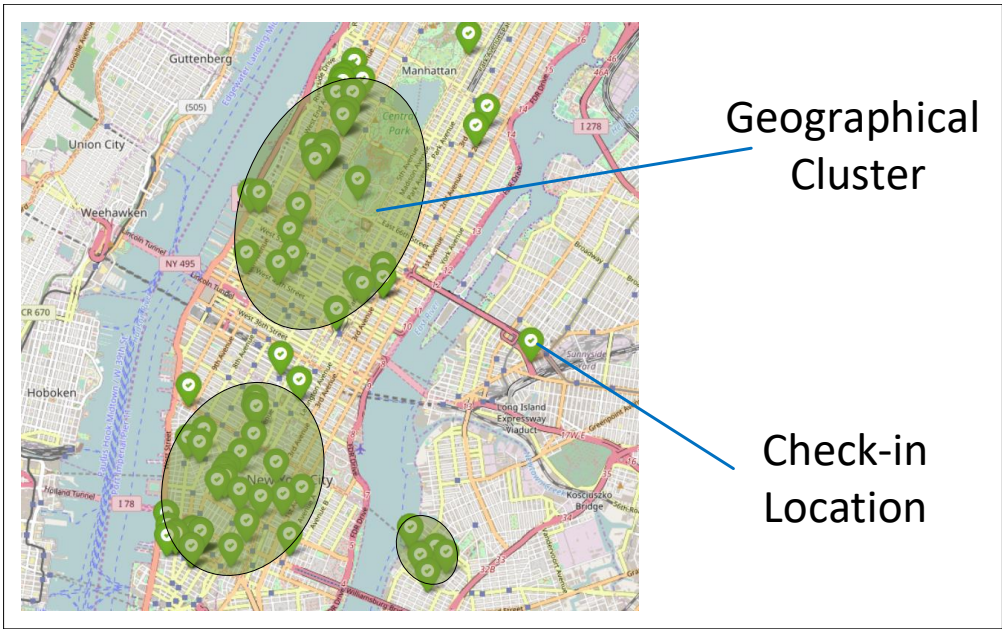


Figure 3.5: Venue Check-in Preference on Geographical Areas.

calculating a user's Normalized Weighted Category Hierarchy (NWCH). First, we generate a Term Frequency (TF) tree for the user. In a TF tree, each node is associated with the frequency of a user's check-ins at the corresponding category, and a deeper level denotes the categories of a finer granularity. Next, an Inverse Document Frequency (IDF) tree will be calculated. As shown in the Figure 3.6, the value of each node represents the IDF value of the category. Third, the two values for the same node will be multiplied to create the user's Weighted Category Hierarchy (WCH). The Equation 3.13 shows how to compute the TF-IDF value of category c in the hierarchy of user u (i.e., $u.w_c$).

$$u.w_c = \frac{|u.\mathcal{V}(c)|}{|u.\mathcal{V}|} \times \lg \frac{|\mathcal{U}|}{|\mathcal{U}(c)|} \tag{3.13}$$

$$u.\mathcal{V}(c) := \{u.v_i \in c\} \tag{3.14}$$

$$\mathcal{U}(c) := \{u_j : \exists u_j.v_k \in c\} \tag{3.15}$$

The first term of the equation is the TF value, where $|u.\mathcal{V}(c)|$ is user u's number of visits to venues in category c, and $|u.\mathcal{V}|$ is the user u's total number of visits. The second term of the equation is IDF value, where $|\mathcal{U}|$ is the number of all the users and $|\mathcal{U}(c)|$ is the number of users who have visited category c.

Such WCH is able to capture an individual's preferences in different categories of different granularity. For the requirement of the following calculation, we convert each WCH into Normalized WCH (NWCH) by normalizing all the TF-IDF scores for each category at the same level so that they sum to 1.

To compute the similarity score of any two users based on their NWCHs, we employ the Jensen-Shannon divergence [18], which is originally defined as follows:

$$D_{JS}(p \parallel q) = \frac{1}{2}D_{KL}(p \parallel m) + \frac{1}{2}D_{KL}(q \parallel m) \tag{3.16}$$

with

$$m(x) = \frac{1}{2}(p(x) + q(x)) \tag{3.17}$$

$$D_{KL} = -\sum_{x} p(x) \log \frac{q(x)}{p(x)} \tag{3.18}$$

where p and q are two probability distributions. The "semantic distance" of two users can be calculated as the weighed sum of the J-S divergence of each level in the two corresponding NWCH, shown in Algorithm 2. For each user, we keep a list of the candidates E (which are selected from the above process) with the top N_s smallest values to her.

Furthermore, dividing a user's check-in data according to time, we can get multiple semantic trees, which is called semantic forests. Therefore, different expert lists will be generated at different time, which can help get more accurate results. In our system, we keep two group of experts for a user. One is formed by the data through Monday to Friday. The other one is constructed by the rest of data.

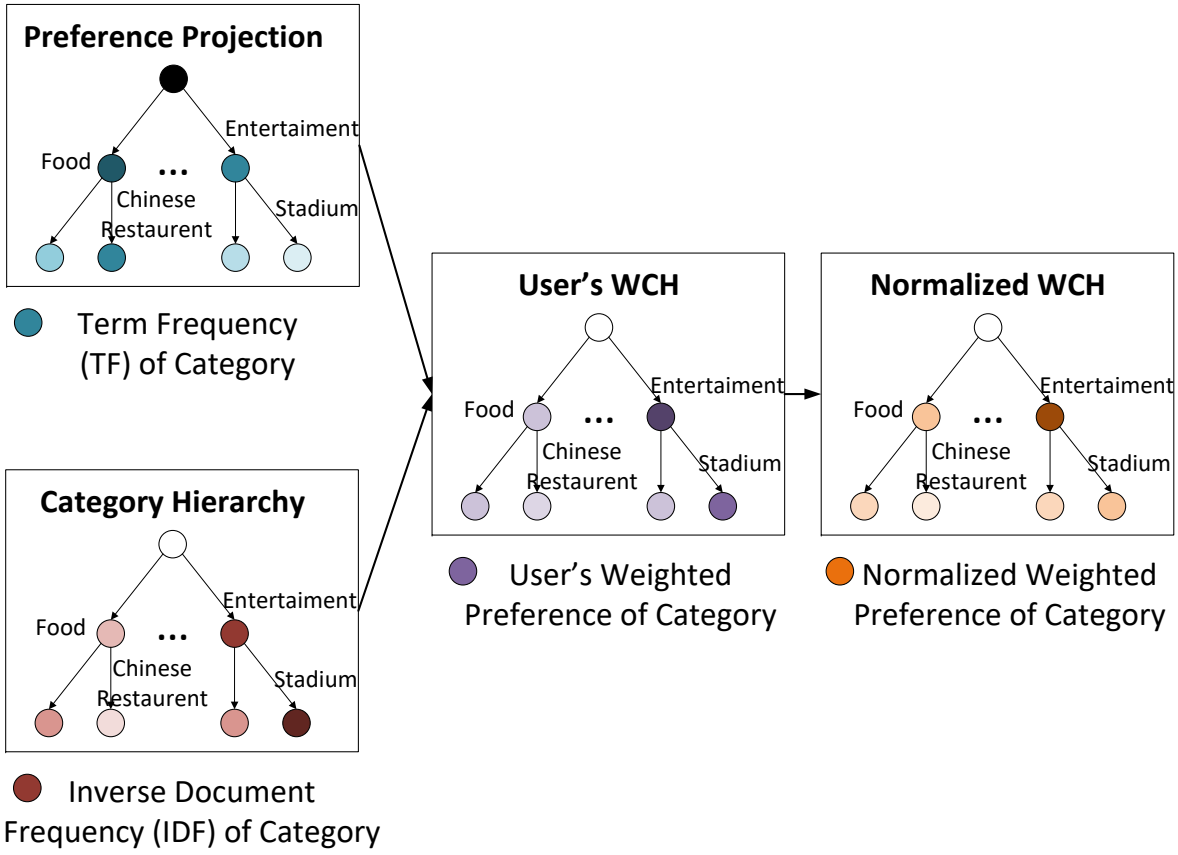


Figure 3.6: User WCH Construction.

Algorithm 2: Divergence of Two WCHs

Input: Two user's WCH tree (1) $u_1.wch$, (2) $u_2.wch$, and (3) Weights for levels w_l
Output: Divergence metric of two users' WCHs d

$d \leftarrow 0$;
for level l from root to bottom in semantic tree **do**
  $jsd \leftarrow 0$;
  foreach c in categories of level l **do**
    $p \leftarrow u_1.wch_c$;
    $q \leftarrow u_2.wch_c$;
    if $p = 0$ and $q = 0$ **then**
      continue;
    else if $p = 0$ or $q = 0$ **then**
      $jsd \leftarrow jsd + 1$;
    else
      $m \leftarrow 0.5 * (p + q)$;
      $jsd \leftarrow jsd + 0.5 * (p * log(p/m) + q * log(q/m))$;
    end
  end
  $d \leftarrow d + jsd * w_l$
end
return d

3.3 Behavior Pattern Mining

Table 3.1: Notations used in Hidden Markov Model.

Elements	Meaning
$S = \{s_1, s_2, \ldots, s_K\}$	Hidden State space, which cannot be observed directly
$O = \{o_1, o_2, \ldots, o_N\}$	Observation space, can be observed directly
$\Pi = (\pi_1, \pi_2, \ldots, \pi_K)$	Initial Probabilities, such that π_i stores the probability that $x_1 = s_i$
$A = (a_{11}, \ldots, a_{ij}, \ldots, a_{KK})$	Transition Matrix of size $K \times K$, such that a_{ij} stores the transition probability of transiting from state s_i to state s_j
$B = (b_{11}, \ldots, b_{ij}, \ldots, b_{KN})$	Emissions Matrix of size $K \times N$, such that b_{ij} stores the probability of observing o_j from state s_i.

In this section, we model the user's action patterns using Hidden Markov Model.

A Hidden Markov Model (HMM) [7, 39] is a state modeling technique that deals with a situation which you observe a sequence of emissions but do not know the sequence of states the model went through to generate those emissions. In a hidden Markov model, the state is not directly visible, but the output is visible, as Figure 3.7 shows. The sequence

of observations generated by an HMM gives some information about the sequence of states. Using hidden Markov models can discover the sequence of hidden states from the observed data. The notations used in HMM is shown on Table 3.1.

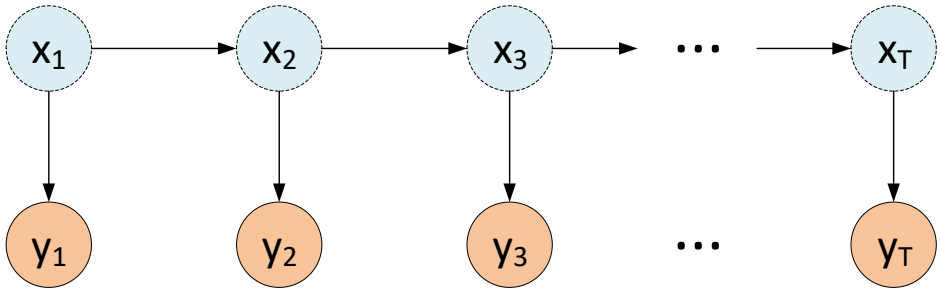


Figure 3.7: Hidden Markov Model.

The use of HMM to model users' preference on temporal venue visit sequence is motivated by the observation that an user's next visit largely depends on her current visit. Here we treat the categories of venues as states and the check-in timestamp as observations. Because of continuity of time, we have to do temporal clustering. The easiest way is to divide a day into a certain number of slots, and treat the check-in timestamp in a slot as the same observation. Therefore, for every day, we get a category path $X = (x_1, x_2, \ldots, x_T)$, which is a sequence of states $x_n \in S = \{s_1, s_2, \ldots, s_K\}$, and a time slot path $Y = (y_1, y_2, \ldots, y_T)$, which is a sequence of states $y_n \in O = \{o_1, o_2, \ldots, o_N\}$.

Figure 3.8 shows a simple representation in our HMM model. In this example, $S = \{Food, Collage, Home, \ldots\}$, and every hour correspond to a time slot. As it shows, a typical user has higher probability to go home after she leaves her work office, and she has lower probability to go to food store if she just visited a restaurant. The transition probability between different categories would form a transition matrix. Furthermore, the visit frequency for each venue is different. A fast food restaurant would have a higher probability to be visited at lunch time than during working hours. Therefore, each category has its own emission matrix. We treat the check-ins in one day as one sequence. We train a personalized

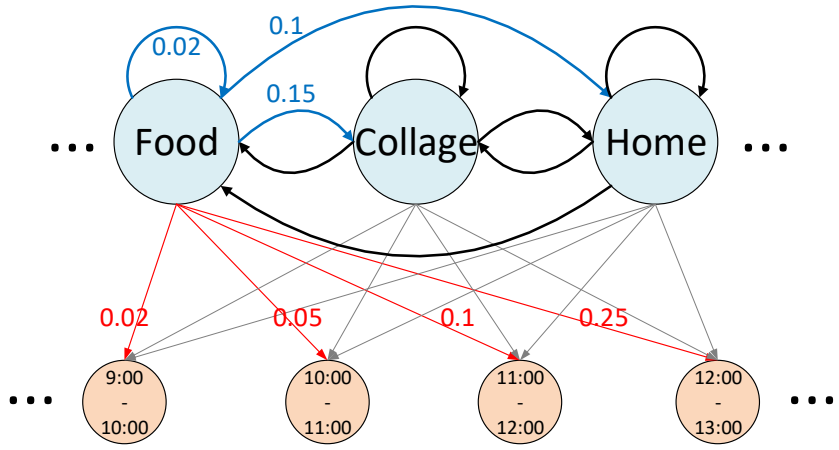


Figure 3.8: HMM Modeling for Check-in Category Prediction.

Hidden Markov Model (HMM) for time-sensitive location sequence recommendation by using all the sequences from the target user, shown in Algorithm 3.

3.4 Online Recommendation

In this section, we discuss the online recommendation of our system, which consists of 1) category inference, which calculate the probability of different categories for each time slot specified by users using Viterbi algorithm, and 2) venue search, which infers a predication score of the candidate locations the user would visit using category prediction and personalized experts.

3.4.1 HMM inference on categorical sequence

Reviewing the application scenarios of our system, a user could choose a sequence of time slots. Therefore we get a sequence of observations $Y = (y_1, y_2, \ldots, y_T)$. While there is no visiting record before the query at the same day, we can use the precalculated Initial Probability Distribution Π. Otherwise, suppose the category of site visited last time is s_i, Π should be reset as $(A_{i1}, A_{i2}, \ldots, A_{iK})$.

Algorithm 3: HMM training

Input: (1) Category sequences $Cseqs$, (2) Check-in time slot sequences $Tseqs$, (3) Number of sequences n

Output: (1) Transition matrix A, (2) Emissions matrix B, and (3) Initial Probability Distribution Π

create transition matrix A of all 0;
create emissions matrix B of all 0;
// iterate check-in sequences of each day
for $i \leftarrow 0$ **to** length$(Cseqs) - 1$ **do**
    $X \leftarrow Cseqs_i$;
    $Y \leftarrow Tseqs_i$;
    $B[X_0][Y_0] \leftarrow B[X_0][Y_0] + 1$;
    // iterate check-in records in one day
    for $j \leftarrow 1$ **to** length$(X) - 1$ **do**
        $B[X_j][Y_j] \leftarrow B[X_j][Y_j] + 1$;
        $A[X_{j-1}][X_j] \leftarrow A[X_{j-1}][X_j] + 1$;
    end
end
// Normalize matrix A and B, let sum of the elements in all the rows = 1
Normalize(A);
Normalize(B);
return A, B

Suppose the sequence of state is $X = (x_1, x_2, \ldots, x_T)$, we define the forwards probabilities as

$$\alpha_t(j) := p(y_1, y_2 \ldots y_t; x_t = j) \tag{3.19}$$

So

$$\alpha_1(j) := \pi_j b_j(y_1) \tag{3.20}$$

Here we replace the notation b_{jy_1} by $b_j(y_1)$ just to make the equation clearer. Further, we can compute α_t from α_{t-1} using forward method:

$$\alpha_t(j) = \sum_{i=1}^{K} \alpha_{t-1}(i) a_{ij} b_j(y_t) \tag{3.21}$$

Algorithm 4 shows the full process. With an HMM for each user, applying this algorithm can find out the probabilities for each state (category) for every time slot specified by the user.

Algorithm 4: Forward Algorithm

Input: (1) Transition matrix A, (2) Emissions matrix B, (3) Initial probabilities Π, (4) Observations Y of length T, and (5) size of state space K

Output: Probability matrix $forward$

```
create a probability matrix forward[K, T] of all 0;
for s ← 0 to K − 1 do
    forward[s, 0] ← Π[s] ∗ B[s][Y_0];
end
for t ← 1 to T − 1 do
    for s ← 0 to K − 1 do
        for i ← 0 to K − 1 do
            p = forward[i, t − 1] ∗ A[i][s] ∗ B[s][Y_t];
            forward[s, t] ← forward[s, t] + p;
        end
    end
end
return forward
```

3.4.2 Venue search

In this step, we compute the recommendation ratings of the venues for a user at a specified time slot, by which a list of recommended locations can be generated. Assume at time slot t, we have calculated the probabilities for different categories using HMM. The probability for the user of visiting a venue v can be computed by

$$p_p(v) = \alpha_t(v.c) \tag{3.22}$$

where $v.c$ means the category of the venue. We can easily figure out that the same type of locations would share a same value. Therefore, more criteria are needed to better evaluate a location's recommendation level.

An intuitive approach is considering the user's frequency of visiting to the venue. Because it's easier for users to visit places they've been to. With the history check-in record of the user, we can come out with a individual visiting probability which is calculated by

$$p_i(v) = \frac{|\{u.v_i = v\}|}{|\{u.v_i \in R_q\}|} \tag{3.23}$$

where $\{u.v_i = v\}$ means all the user's check-ins to the location v, and $\{u.v_i \in R_q\}$ means all the user's check-ins in the bounding box of query range. The calculation above is based on the analysis of user's own check-in records. However, a user may not have enough amount of check-in records in local, especially for the cold-start users. Furthermore, a user not always visit familiar venues, and it is also important to refer to the public's choices. Reviewing Section 3.2, we have already get the user's own local experts. We evaluate a venue by the experts' voting, i.e., a place gets one point while an expert has visited. The equation is shown below, where W_e is the weight.

$$p_e(v) = \frac{|\{u_j \in E : \exists u_j.v_k = v\}|}{|E|} \tag{3.24}$$

With the individual weight w, the final predication would be calculated by

$$p(v) = p_p(v) \cdot (p_i(v) \cdot w + p_e(v) \cdot (1 - w)) \tag{3.25}$$

We don't need to come out with the scores for every venue in the map. We just need to retrieve venues in the bounding box determined by the user at time slot t, and the system will return the top-N venues with the highest scores to the user as the location recommendations. In the same way, we can make multiple venue recommendations according to each time slot and range set by the user.

3.5 Experimental Evaluation

In this section, extensive experiments were conducted to evaluate the effectiveness and efficiency of our proposed recommender system. We first describe the settings of experiments, including the dataset, baseline approaches, and the method of evaluation. After that, we report the results followed by some discussions.

3.5.1 Experiment Settings

Datasets. We study Tokyo, the largest city in Japan and New York City, the largest city in the United States. The dataset of Tokyo includes 573703 check-in records collected from Foursquare from April 12, 2012 to February 16, 2013[49]. The dataset of New York City includes 227428 check-in records collected from Foursquare from April 12, 2012 to February 16, 2013[49]. From the dataset, we collects the following information: 1) user profile information, consisting of a user's ID, name, and home city; 2) venue profile information, including the venue's ID, name, address, GPS coordinates, and its categories; and 3) user's check-in histories, containing the check-in timestamp and the location ID.

Approaches for comparison. We compare our proposed recommender algorithm (TLSR+) with the following approaches.

Table 3.2: First-Level Category Distribution in Our Dataset.

Category	**Frequency**	**Ratio**
College & University	5950	0.0262
Food	51933	0.2284
Residence	19817	0.0871
Travel & Transport	32634	0.1435
Outdoors & Recreation	30442	0.1339
Arts & Entertainment	10708	0.0471
Shop & Service	31080	0.1367
Nightlife Spot	16851	0.0741
Professional & Other Places	28012	0.1232

(1) Most-Popular-Venues-based (MPV). This approach employ document frequency for venue recommendation. In this system, each venue keeps and updates a "popularity degree", which is the number of users who have visited the venue. Given a user-specified geospatial range, the approach chooses the top-N popular venues as the output. As a baseline, the approach neither considers users' personal preference, nor utilizes time factor.

(2) Location-based and Preference-aware Recommendation (LPR). This approach is proposed by Bao et al. [6]. It not only consider user personal preferences, but also take account of social opinion. The author models each individual's personal preferences with a weighted category hierarchy, and selects candidate local experts that match the user's preferences. The candidate venues will also be picked and rate by the selected local experts. At last, the top-k ranked locations will be chose for recommendation. This approach originally focuses on time irrelevant recommendation, therefore a little modification was made to suit our situation.

(3) Time-aware Location Sequence Recommendation (TLSR). This approach comes from our past work [42]. It first finds out semantic similar users by comparing users' hierarchical semantic trees. Then selects geospatial similar users using GMM. During online part, the records from both groups of users are combined with the query user's check-in records, and HMM will be applied to search the final recommended locations.

Table 3.4 shows the applied factors of the approaches.

Table 3.3: Second-Level Category Distribution in Our Dataset.

Category	Frequency	Ratio
Home (private)	15334	0.0674
Bar	13236	0.0582
Office	12554	0.0552
Athletics & Sports	9874	0.0434
Metro Station	9348	0.0411
Coffee Shop	7510	0.0330
Train Station	6408	0.0282
Food & Drink Shop	6197	0.0272
States & Municipalities	5436	0.0239
Asian Restaurant	5136	0.0226
Park	4689	0.0206
Bus Station	4474	0.0197
Deli / Bodega	4214	0.0185
Residential Building	4185	0.0184
American Restaurant	3701	0.0163
Building	3474	0.0153
Medical Center	3331	0.0146
Road	3207	0.0141
Others	105119	0.4622

Method	User Preference	Social Opinion from Experts	Geospatial Factor	Time-aware Sequence Pattern
MPV		✓		
LPR	✓	✓		
TLSR	✓		✓	✓
TLSR+	✓	✓	✓	✓

Table 3.4: Comparison Between Other Methods and Ours

Evaluation methods. To evaluate our recommender system, we need to test its effectiveness and efficiency. However, there are no other work focusing on time-aware location sequence recommendation in the literature, it is difficult to devise a way to carry out such experiment. To make the effectiveness evaluation, we divide a user's check-in history into two parts. 50% as a training set training set and 50% as a testing set. A check-in sequence can be generated for each day in the testing set, which we regard as the ground truth. We extract the check-in times of the sequence as the input of our recommender system. A bounding box containing the check-in location with size L will be generated for every check-in time.

Our approach will return a list of locations with size N as the result for each point-in-time in the range of bounding box. We call the recommendation for a time slot as one prediction. When the true venue is included in the list, this is a successful recommendation.

We use two criteria to evaluate the result: 1) hit rate and 2) mean reciprocal rank (MRR). The hit rate used in our experiments is defined as Formula 3.26 .

$$hit\ rate = \frac{the\ number\ of\ venues\ matched}{the\ total\ number\ of\ time\ slots} \tag{3.26}$$

One thing to note, this evaluation measurement is extremely strict. This is more like an evaluation of prediction algorithms other than recommendation algorithms – we should know that a user may like a location but won't go right away. Or, some correct prediction may be ignored – a user could has visited a location but forget to check in on app. In other words, our approach is actually more effective than our experiment result shows. However, the tests are still meaningful because it reveal the advantage of our approach beyond the comparison algorithms.

The hit rate are majorly affected by the following two factors: 1) the number of requested recommendations N, and 2) the size of the bounding box L, and 3) the length of the recommendation sequence T. Therefore, in the experiment section, we will test the effectiveness over these three factors using the data of Tokyo and New York City as illustrated above.

The mean reciprocal rank (MRR) is a statistic measure for evaluating any process that produces a list of possible responses to a sample of queries, ordered by probability of correctness. Because our recommendation is focus on sequence, we employ a measure similar to MRR. The reciprocal rank of a sequential response is the multiplicative inverse of the position of the sequence for success predictions: 1 for first slot, 1/2 for second slot, 1/3 for third slot and so on. We normalize the result, and the equation is shown below:

$$MRR = \frac{\sum_{i=1}^{|H|} \frac{1}{position_i}}{\sum_{i=1}^{L} \frac{1}{i}} \tag{3.27}$$

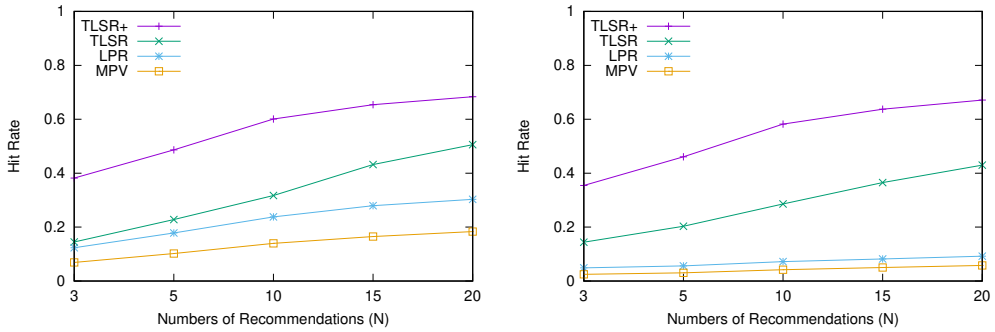
where H is the set of successful hit points, and $position_i$ is the position of the i-th query point of the sequence.

The efficiency of the online recommendation also depends on these two aspects: recommendations numbers N and range size L miles. Therefore, we test the efficiency for our algorithm over these two factors. The will show the benefit of our method for real time application.

3.5.2 Recommendation Effectiveness

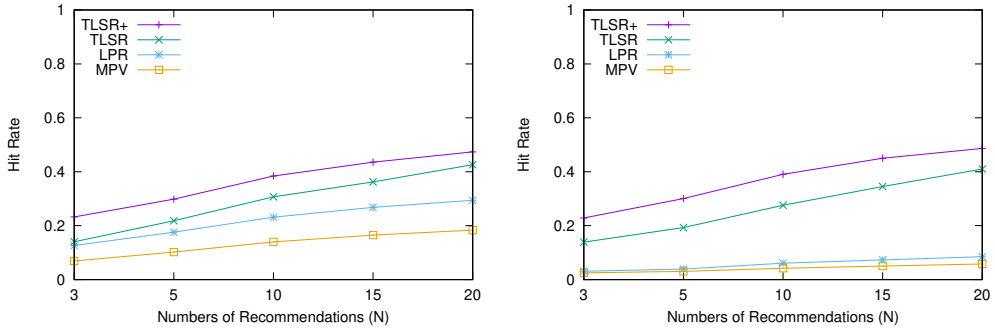
We test the users in the data bases with randomly selected dates and cases. 2293 users are tested in Tokyo dataset with total 22194 cases, and 1083 users are tested in New York City with total 9292 cases.

Figures 3.9 and 3.10 show the average hit rates and the MMRs of different methods varying in the number of recommendation locations (N) in Tokyo data set and New York City data set. Obviously, our algorithm (TLSR+) is superior to the other three approaches. As a relatively simple algorithm, MPV drops behind other methods. The result is as expected because MPV neither consider the user's own preference, nor utilize extra information from the input. LPR fully considers the social opinion and the category trees, so it outperforms MPV. The lack of adequate utilization of the user's local data makes it unsuitable for our reference scenario, therefore it keeps at a relatively low hit rate. TLSR exceeds MPV and LPR due to the advantages of spatial clustering and pattern learning. Finally, TLSR+ performs amazingly. Born out of our old approach TLSR, the new approach makes good use of the time factor, and deeply mines the social knowledge. The accuracy of our results proves the practical application value of our algorithm.



(a) Tokyo at weekdays

(b) NYC at weekdays



(c) Tokyo at weekends

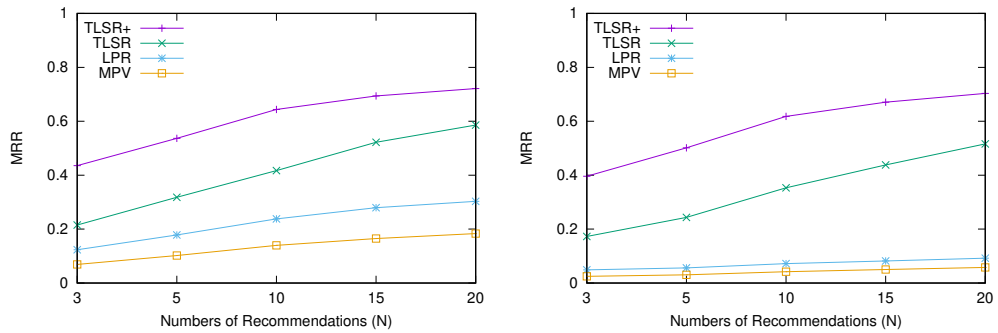
(d) NYC at weekends

Figure 3.9: Impact on Recommendation Numbers of Hit Rate (L=10, T=5)

As shown in Figure 3.9 and 3.10, the hit rate of our method increases as the number of recommendation increases. It can be seen that continuing increasing the number of recommendations does not yield high returns, because the trend of increase is slowing down.

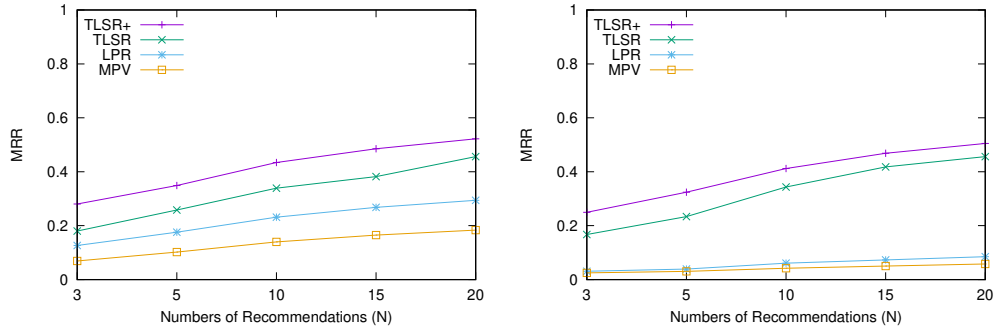
Figures 3.11 and 3.12 present the hit rates and MMRs of different methods changing over the range of the bounding box. As a result, the larger the search range is, the lower precision for our predictions. We can see the recommendation range (L) is largely affect the hit rate. However, Our algorithm remains highly efficient even L increase to 20 miles.

Figures 3.13 and 3.14 presents the hit rates and MRRs of different methods varying the length of recommendation sequence. There is some decline of our method's while the sequence length grows. This is very normal because the longer it is from now, the harder for us to predict a user. MPV and LPR has nothing to do with the order of users' visiting, so their hit rate stay the same.



(a) Tokyo at weekdays

(b) NYC at weekdays



(c) Tokyo at weekends

(d) NYC at weekends

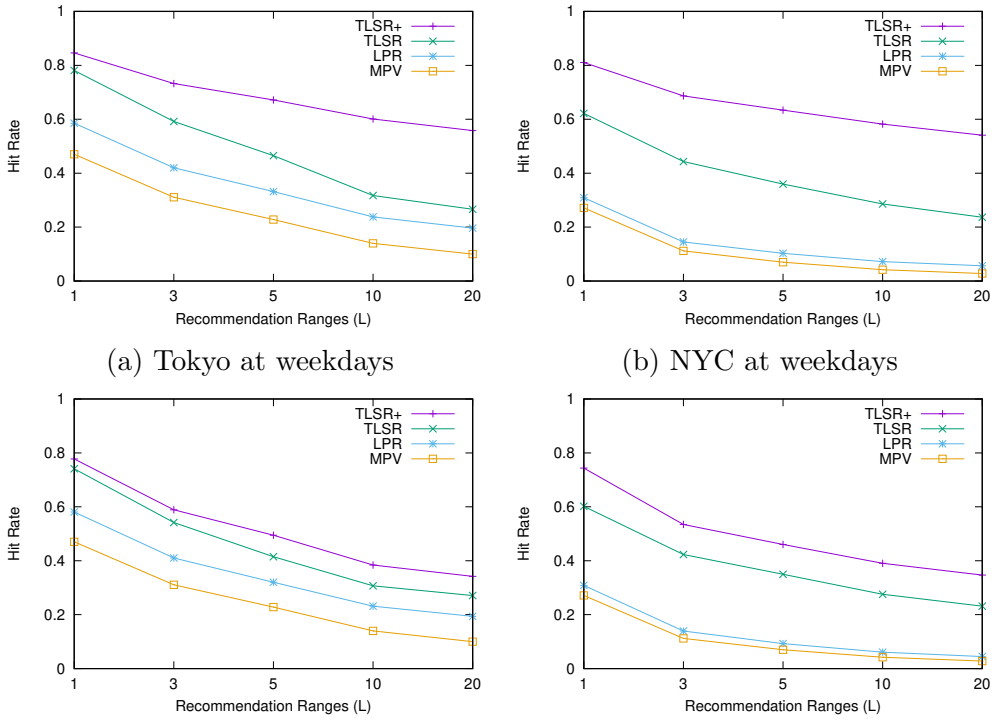
Figure 3.10: Impact on Recommendation Numbers of MRR (L=10, T=5)

Except our method, the accuracy of other approach is heavily influenced by the city. That is because the venue density and the user's range of activity are different. Consideration of multiple factors allows our algorithm to avoid these effects.

3.5.3 Recommendation Efficiency

In the efficiency study, the experiments were evaluated on a computer running Windows10 with an I7-7500U CPU 2.90GHz processor and 16 GB RAM.

Figure 3.15 shows the average online efficiency of the 4 methods varying in the number of recommendations, while the recommendation range L is 10 miles. As it turns out, MPV is the fastest method, because it only does an online selection. And it is not surprised that our method is slower than MPV. The HMM calculation and the venue search process are the parts that take time. However, we are excited to see that our algorithm is still significantly



(a) Tokyo at weekdays

(b) NYC at weekdays

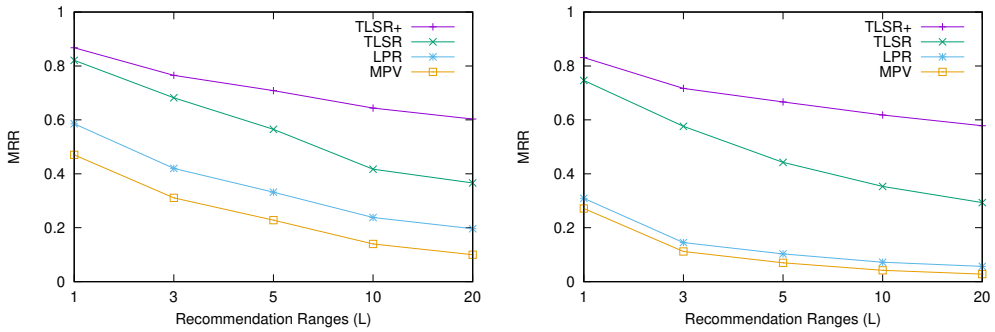
(c) Tokyo at weekends

(d) NYC at weekends

Figure 3.11: Impact on Spatial Ranges of Hit Rate (N=10, T=5)

faster than TLSR and LPR. As presented by the figure, the cost time of TLSR grows slowly as the number of recommendations increases, while the other methods performs the same.

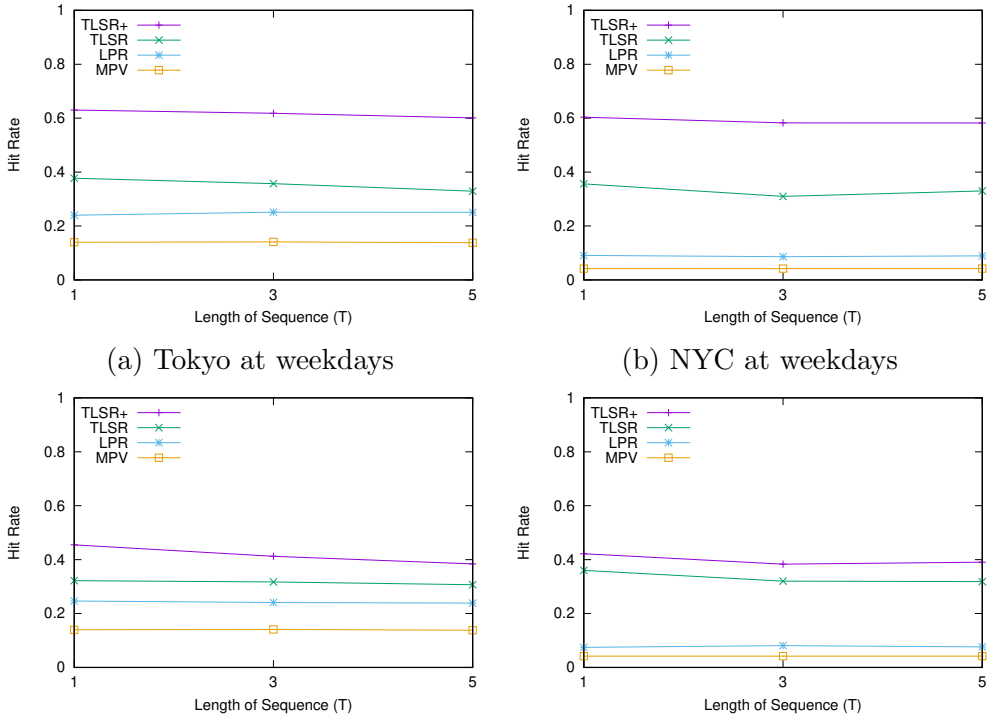
Figure 3.16 shows the average online efficiency of different methods varying in the range of recommendations. A larger range will incorporate more venues and user candidates, leading to a heavier computational load. MPV is the most affected algorithm. Conversely, the range of bounding box have limited impact on our algorithm, which is the advantage of our approach.



(a) Tokyo at weekdays (b) NYC at weekdays

(c) Tokyo at weekends (d) NYC at weekends

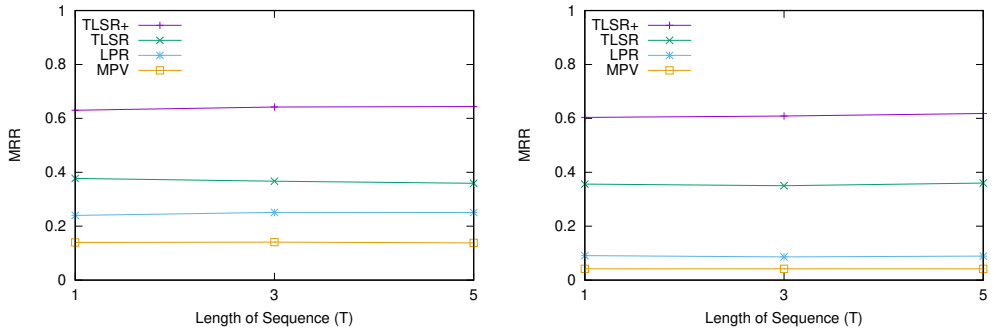
Figure 3.12: Impact on Spatial Ranges of MRR (N=10, T=5)



(a) Tokyo at weekdays (b) NYC at weekdays

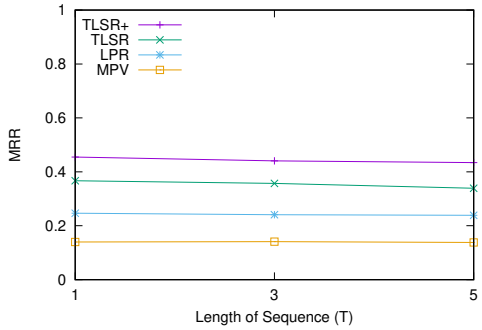
(c) Tokyo at weekends (d) NYC at weekends

Figure 3.13: Impact on Sequence Length on Hit Rate (N=10, L=10 miles)

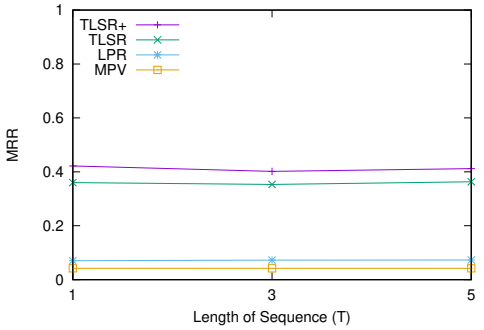


(a) Tokyo at weekdays

(b) NYC at weekdays



(c) Tokyo at weekends



(d) NYC at weekends

Figure 3.14: Impact on Sequence Length on MRR (N=10, L=10 miles)

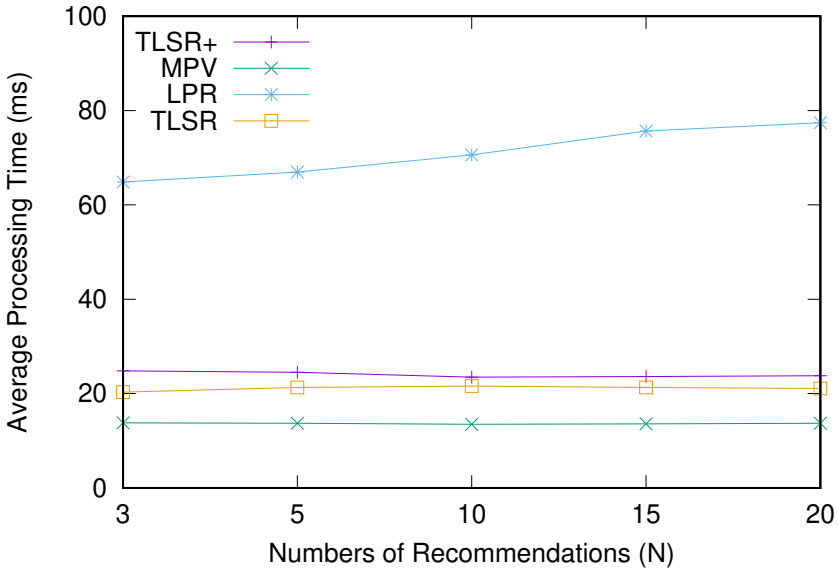


Figure 3.15: Average Processing Time w.r.t Recommendation Numbers (L=10)

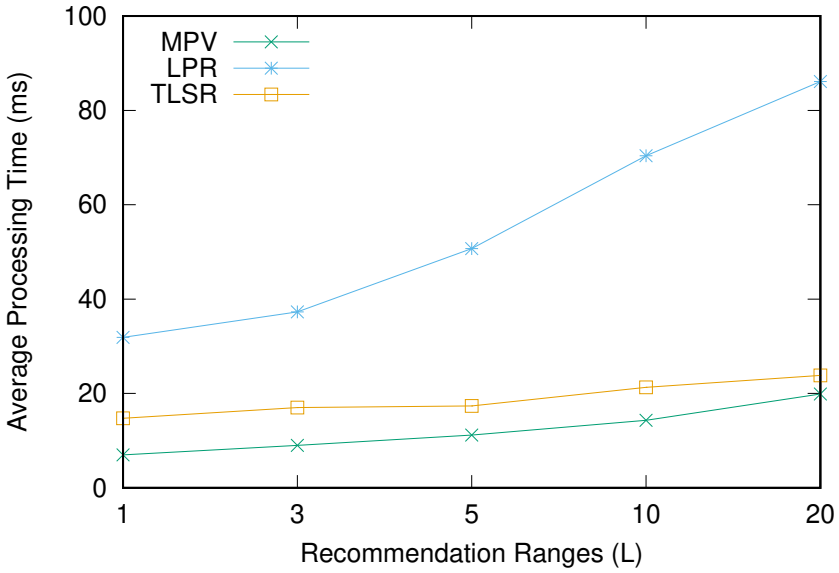


Figure 3.16: Average Processing Time w.r.t Spatial Ranges (N=10)

Chapter 4

A VLOS Compliance Solution to Ground/Aerial Parcel Delivery Problem

4.1 Overview

4.1.1 Problem Definition

Given a road network G =(V, E), where V denotes a set of vertices and E is a set of edges, a spatial object is defined by a tuple <x_1, x_2>in a Euclidean space R. Every vertex v=<x_1, x_2>is also in R. $D^R(.,.)$ and $D^G(.,.)$ represent the distance between two spatial objects in the Euclidean space R and on the road network G, respectively.

Definition Given a set of destinations $D = \{d_1, ..., d_n\}$, there are a truck t and a UAV u used for parcel delivery. Every destination is served exactly once either by the truck or the UAV. The Ground/Aerial Parcel Delivery Problem (GAPDP) targets at finding an optimal route that minimizes the delivery time of serving all destinations once and returning to the distribution center. The delivery time can be calculated by

$$A = A_{h,d_1} + A_{d_1,d_2} + ... + A_{d_i,d_{i+1}} + ... + A_{d_{n-1},d_n} + A_{d_n,h} \tag{4.1}$$

where $A_{d_i,d_{i+1}}$ represents the time between the completion of delivery to d_i and to d_{i+1}, and A_{h,d_1} and $A_{d_n,h}$ are the delivery time of d_1 from the distribution center, and the time returning to the distribution center from d_n.

To make the proposed problem more realistic, there are assumptions in the problem as follows.

- The truck travels on road network at a fixed speed t^v and the UAV moves in Euclidean space at a fixed speed u^v. The truck can stop at any locations on the road for delivery.

Table 4.1: Symbolic notations.

Symbol	Meaning
D	A destination set
R, G	A Euclidean space and road network
V, E	A set of vertices and edges
D^R, D^G	The distance in R and G
t^v, u^v	The speed of trucks and UAVs
$d_i.V$, $d_i.E$	The set of vertices and edges in the VLOS area of d_i
$V \times E$	The Cartesian product of two sets V and E

- The driver is allowed to operate both truck and UAV for delivery. However, the truck and UAV cannot move simultaneously.
- UAV carries one package at a time. UAV returns to the truck immediately upon the completion of delivery.
- There is only one package for a destination. If there are two or more packages to a destination, it is conceptually equivalent to setting up a destination at the same location for each package.
- The delivery method can be explicitly selected by customers (because they may prefer using the truck for delivery, the package may exceed the payload capacity of the UAV, or a signature may be required), which indicates that a package could be delivered only by the truck or the UAV. If the delivery method is not explicitly selected, both the truck and the UAV can be used by default.
- The power of the UAV is sufficient for a round trip of delivery (a round trip from the truck to a destination); the time of battery replacement is negligible in this research.
- Due to the requirements of the FAA, the distance between the UAV and the driver/operator cannot be greater than a threshold, which is called "Visual-Line-Of-Sight" (VLOS) distance.
- All destinations can be at any locations on the road network.

4.1.2 preliminary

To achieve an exact solution to the travelling salesman problem, all permutations of destinations have to be visited, and the route that provides the shortest travelling time is the optimal route as an exact solution. The computational complexity of this intuitive method is $O(n \cdot n!)$, where n indicates the number of destinations. An improvement by utilizing dynamic programming techniques had been proposed, which could reduce the computational cost to $O(n^2 \cdot 2^n)$ [25]. Both approaches assume that the shortest distance between any two destinations can be calculated in constant time ($O(1)$). However, this assumption is not applicable to our proposed GAPDP problem because the location where the truck stops for delivery may greatly vary by the delivery order of destinations, speeds of the truck and the UAV, the VLOS distance, and more. Additionally, the optimal delivery route may consist of the fastest paths that connect destinations in Euclidean spaces and on road networks, which makes the GAPDP problem more challenging than the TSP problem.

Therefore, we propose an exact solution to address the GAPDP problem. Because GAPDP is NP-hard, shown in Theorem 4.1, our solution follows a fundamental idea that finds an optimal delivery route by checking all permutations of destinations. The details are described in Alg. 1, which receives a set of destinations $\mathbb{D}$ and a starting distribution center h, and returns the fastest delivery time for $\mathbb{D}$. For a specific permutation of $\mathbb{D}$ in the FOR loop at lines 2 to 4, the function ShortestTime(D', h) calculates the shortest delivery time of destinations D' in a specific order. The variable $shortestDeliveryTime$ always maintains the shortest time of the delivery. To find the fastest route of destinations in a given order, we propose an index-based approach, which pre-computes the fastest delivery route in VLOS areas of destinations. Consequently, we could "jump" from an entrance of the VLOS area of a destination to its exit without route searching at run-time.

Theorem 4.1. ***GAPDP is NP-hard.***

Algorithm 1 GAPDP($\mathbb{D}$, h)

```
1. shortestDeliveryTime = ∞
2. for D' ∈ {all permutation of 𝔻} do
3.    shortestDeliveryTime = min(shortestDeliveryTime, ShortestTime(D', h));
4. end for
5. return shortestDeliveryTime;
```

Proof. GAPDP is NP-hard, since the Travelling Salesman Problem (TSP) is NP-hard [38], and TSP is a special case of the GAPDP problem when all packages are restricted to be delivered by the truck. □

4.2 Exact Solution

A major challenging problem in GAPDP is that the fastest delivery route of destinations in a given order may greatly vary under different settings of speeds of the truck and the UAV, delivery methods of destinations, and the VLOS distance. In this subsection, we illustrate an approach that finds the fastest delivery route for destinations in a given order. The fastest route is represented by a sequence of stop locations, where the truck is stopped and UAV is launched for delivery.

Before the illustration of the proposed method, we provide a formal definition of the sub-problem as follows.

Definition Given an ordered set of destinations D=$\{d_1...d_n\}$, these destinations are required to be served exactly once in the order as they are in the set. Every destination is assigned with an applicable delivery method (by truck only, by UAV only, or by either of the two). A truck and a UAV are used to deliver n packages to these n destinations. This sub-problem is to find the fastest delivery route and time for the truck and the UAV, which initially start from a distribution center h, and finally return to the center. All assumptions in Definition 4.1.1 are also applied to this sub-problem.

Since the operating range of UAVs is restricted by the Visual-Line-Of-Sight (VLOS) distance from operators [1], the truck must stop and the UAV must be launched at a location

in the VLOS area of a destination for delivery. Thus, an intuitive algorithm visits all roads in the VLOS areas of destinations, and finds a sub-optimal stop location on each road. Then, the best route for serving destinations can be easily derived from the best sub-optimal stop locations. It is worth noting that there is a special case, in which the delivery time of a destination by truck is equal to the delivery time of stopping at the destination and using UAV for delivery because the delivery distance for UAV is 0. With this observation, our algorithm assumes that the UAV is used for all delivery for simplifying the problem. If the delivery by truck is the optimal method or the truck is a preferred method for a destination, our algorithm returns an equivalent case of using the UAV (the delivery distance of the UAV is 0), and the fastest delivery route and the shortest delivery time are the same with the methods considering delivery by truck.

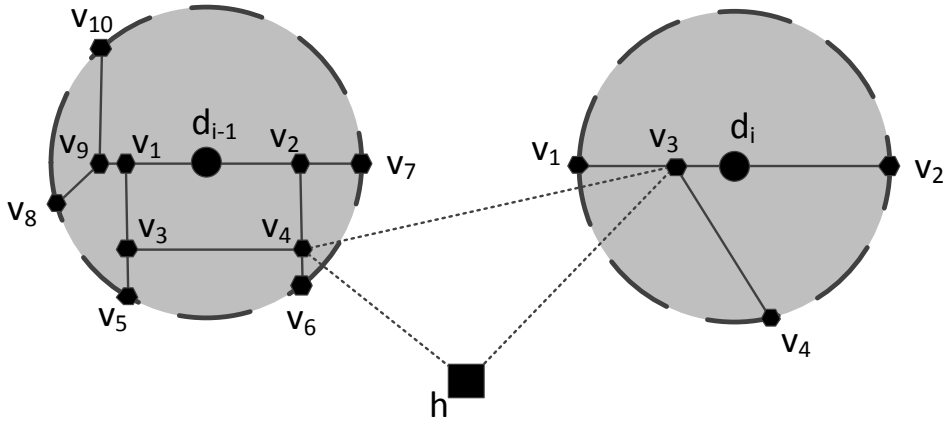


Figure 4.1: An example of selecting the fastest delivery route for two consecutive destinations.

Figure 4.1 displays an example of serving two consecutive destinations d_{i-1} and d_i. We only display one route $\{h, d_{i-1}.v_4, d_i.v_3, h\}$ for better illustration. Initially, the truck starts from h, and we calculate the travelling time from h to all vertices in the VLOS area of d_{i-1}, where $d_{i-1}.\mathbb{V} = \{d_{i-1}.v_1, ..., d_{i-1}.v_{10}\}$. Then, we set those vertices as starting points, and calculate the delivery route for d_i. Each of these routes ends at a vertex in the VLOS area of d_i. So, the fastest delivery route to a vertex of d_i can be found by examining all combinations

of vertex of d_{i-1} and d_i. Then, in the next iteration, the vertex of d_i will be a starting point used for calculating the delivery time of the next destination.

In addition to checking all permutations of destinations, the intuitive approach also needs to search the network space for calculating the shortest network distance between two locations, and iterate all roads to find an optimal stop location in VLOS areas, the computational cost of which would be considerably high due to large number of possible stop locations and complex road networks. To avoid the computation at run-time, we turn to an alternative that pre-computes optimal delivery routes in the VLOS areas. We observe that the truck has to enter a VLOS area before delivery, and leave the area for the next if the VLOS areas of two destinations do not overlap with each other. If this is the case, the optimal delivery routes can be calculated at a pre-processing stage, and we can "jump" over the VLOS areas by retrieving the optimal routes from index.

Therefore, we develop an index-based approach, in which routes from entrances of destinations to their exits are pre-built. We denote $d_i.Exit$ to be the set of intersections of the road network and the VLOS circle of destination d_i. Here $d_i.Exit$ is also called the entrances or exits of d_i in this paper.

The details of the index-base approach are described in Alg. 2. We iterate all destinations in the given order in the FOR loop from lines 1 to 13. Then, given a specific destination d_i, if d_i is the first destination d_1, we calculate the shortest travelling time from the distribution center h to every exit of $d_1.e$ at lines 4 to 6. $D^R(h, e')/t^v$ indicates the travelling time from h to an exit $d_1.e'$, and $d_1.map[e'][e]$ is the shortest delivery time from the entrance $d_1.e'$ to the exit $d_i.e$ after serving d_1. So, the shortest time of serving d_1 and stopping at $d_1.e$ is kept in $d_1.time[e]$. Then, we calculate the shortest delivery time of an intermediate destination d_i in the FOR loop at lines 7 to 12. For a specific exit $d_i.e$, the shortest delivery time after d_i is served and the truck stops at $d_i.e$ is calculated in line 10, where $d_i.map[e'][e]$ represents the shortest delivery time of the route from the entrance $d_i.e'$ to $d_i.e$. Here $d_{i-1}.time[e'']$ denotes the shortest time when the truck stops at an exit of the last destination d_{i-1}, and $D^R(e', e'')/t^v$

Algorithm 2 ShortestTime($\mathbb{D}$, h)

1. **for** $d_i \in \mathbb{D} = \{d_1, ..., d_n\}$ **do**
2. Let $d_i.map$ be the mapping from entrances to exits;
3. **if** $d_i = d_1$ **then**
4. **for** $e \in d_1.Exit$ **do**
5. $d_1.time[e] = \min(D^R(h, e')/t^v + d_1.map[e'][e] \text{ — } e' \in d_1.Exit)$;
6. **end for**
7. **else**
8. Let d_{i-1} be the last visited destination;
9. **for** $e \in d_i.Exit$ **do**
10. $d_i.time[e] = \min(d_i.map[e'][e] + D^R(e', e'')/t^v + d_{i-1}.time[e''] \text{ — } e' \in d_i.Exit,\ e'' \in d_{i-1}.Exit)$;
11. **end for**
12. **end if**
13. **end for**
14. Let d_n be the last destination in D;
15. **for** $e \in d_n.Exit$ **do**
16. $d_n.time[e]$ += $D^R(e, h)/t^v$;
17. **end for**
18. **return** $\min(d_n.time[e])$;

is the travelling time from an exit $d_{i-1}.e'$ to an entrance $d_i.e''$. Finally, the travelling time from exits of the last destination to the distribution center is added to $d_n.time[e]$, and the shortest delivery time of $d_n.time[e]$ is the solution. It is easy to see that we have to check all exits of every destination. The computational complexity of Algorithm 2 is $O(n \cdot |d_i.Exit|^2)$, where $|d_i.Exit|$ denotes the number of exits of a destination d_i and n is the number of destinations. The computational complexity of our VLOS-index-based solution is $O(n^2 \cdot 2^n \cdot |d_i.Exit|^2)$.

Moreover, there is a special case that the Euclidean distance between two continuous destinations is shorter than the VLOS distance. In this case, the two VLOS areas overlap; the exit(s) of the VLOS area of a destination may be in the VLOS area of the second destination. This indicates that the parcel truck may already be in the VLOS area of the second destination after exiting the VLOS area of the first destination. The VLOS-based index over single VLOS area cannot be used for route searching in this case. This problem can be solved by merging the VLOS areas of two or more destinations if they overlap. For example, given two destinations d_{i-1} and d_i, a VLOS-based index can be constructed to

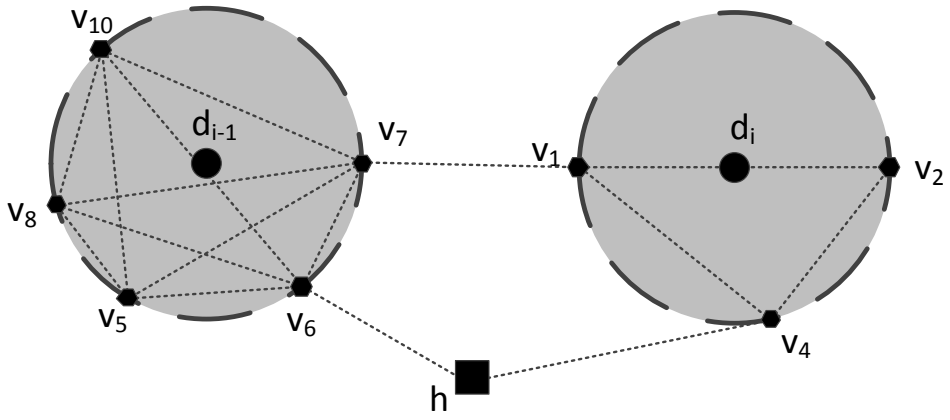


Figure 4.2: An example of selecting delivery routes in Alg.2.

contain all optimal routes from entrances to exits of the union of VLOS areas of d_{i-1} and d_i. All these routes start from an entrance of d_{i-1} and stop at an exit of d_i.

An example of our second approach is displayed in Figure 4.2. The road networks in the VLOS areas of destinations are not accessed in our approach. Instead, the fastest delivery route and time have been calculated and are available in VLOS-based index. Accordingly, take the delivery route in Figure 4.2 for example, we enter the VLOS area of d_{i-1} from entrance $d_{i-1}.v_6$, and jump to the exit $d_{i-1}.v_7$ to leave the VLOS area. Similarly, we jump over from the entrance $d_i.v_1$ to the exit $d_i.v_4$ in the VLOS area of d_i.

Next, we will present our method of finding the optimal route for one destination. This sub-problem can be formally defined as follows.

Definition Given two destinations d_{i-1} and d_i, d_{i-1} has been served, and d_i is the next destination for delivery. Let $d_i.\mathbb{V}$ be the set of vertices in the VLOS area of d_i, then this problem finds an optimal delivery route for d_i, which could start from any vertex in $d_{i-1}.\mathbb{V}$ and end at any vertex in $d_i.\mathbb{V}$. Here d_i must be served by the truck or the UAV exactly once on every route. All assumptions in Definition 4.1.1 are also applied to this sub-problem.

The fundamental idea of our method is to check every road in the VLOS area of d_i, and find an optimal stopping location on a road as a sub-optimum. The global optimum to serve d_i is the best of these sub-optima. Given a starting location v and a road r where the

truck stops, there are four possible delivery routes starting from v and ending at either of the two end points of road r. As listed below, o denotes the optimal stop location on road r for delivery, and l_1 and l_2 represent the two end points of road r.

- Route 1: the truck first arrives at o through l_1, and then goes back to l_1 after delivery.
- Route 2: the truck first arrives at o through l_1, and then moves to l_2 for the next delivery.
- Route 3: the truck first arrives at o through l_2, and then moves to l_1 for the next delivery.
- Route 4: the truck first arrives at o through l_2, and then goes back to l_2 after delivery.

The total delivery time of these four candidate routes are

$$\begin{cases} \frac{D^R(v,l_1)+2\cdot D^R(l_1,o)}{t^v} + \frac{2\cdot D^G(o,d_i)}{u^v} & Route1 \\ \frac{D^R(v,l_1)+D^R(l_1,l_2)}{t^v} + \frac{2\cdot D^G(o,d_i)}{u^v} & Route2 \\ \frac{D^R(v,l_2)+D^R(l_1,l_2)}{t^v} + \frac{2\cdot D^G(o,d_i)}{u^v} & Route3 \\ \frac{D^R(v,l_2)+2\cdot D^R(l_2,o)}{t^v} + \frac{2\cdot D^G(o,d_i)}{u^v} & Route4 \end{cases} \tag{4.2}$$

where

$$D^G(o,d_i) = \sqrt{D^G(l_1,d_i)^2 + D^R(l_1,o)^2 - 2\cdot D^G(l_1,d_i)\cdot D^R(l_1,o)\cdot\cos\alpha} \\ 0 \le D(l_1,o), D(l_2,o) \le D(l_1,l_2) \tag{4.3}$$

It is easy to observe from Equ. 4.2 that the delivery times of Routes 2 and 3 only depend on $D^G(o,\ d_i)$, which is the distance between o and d_i in Euclidean space. $D^R(v,l_1)$, $D^R(v,l_2)$ and $D^R(l_1,l_2)$ are the network distance among s, l_1, and l_2, which are constants in the equation. Therefore, for the shortest delivery times of Routes 2 and 3, o should be selected at a location closest to d_i on the road r. If d_i is on r, then o should be at the location of d_i. If d_i is not on the road r and the projection of r from d_i is on r, then the projection is the optimal stop location for the delivery. If the projection is out of r, the end point of r closer to the projection is the optimal stop location.

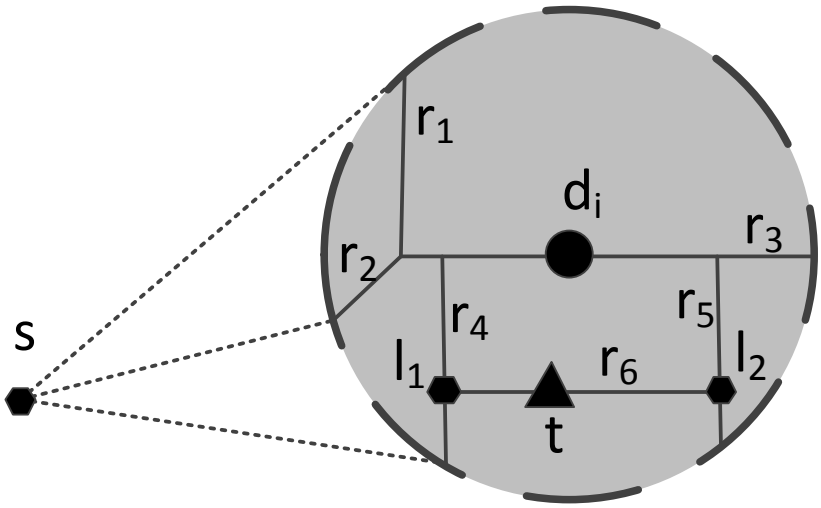


Figure 4.3: An example of calculating the fastest delivery from from s to d_i.

For Routes 1 and 4, let $x = D^R(l_1,\ o)$, then the delivery times of the two routes can be represented by

$$\begin{cases} \frac{D^R(v,\ l_1)+2\cdot x}{t^v} + \frac{2\cdot\sqrt{D^G(l_1,d_i)^2+x^2-2\cdot\cos\alpha\cdot x\cdot D^G(l_1,d_i)}}{u^v} & Route\ 1 \\ \frac{D^R(v,\ l_2)+2\cdot(D^R(l_1,\ l_2)-x)}{t^v} + \frac{2\cdot\sqrt{D^G(l_1,d_i)^2+x^2-2\cdot\cos\alpha\cdot x\cdot D^G(l_1,d_i)}}{u^v} & Route\ 4 \end{cases} \tag{4.4}$$

where α is the angle of the road r and the line segment from l_1 to d_i, and $0 \leq x \leq D(l_1, l_2)$. $D^R(v,\ l_1)$, $D^G(l_1, d_i)$, $D^R(l_1,\ l_2)$, t^v, and u^v are constants. Due to the space limit, the optimal locations for the two routes can be found at [43].

It is worth noting that if there exist two or more optimal stopping locations on one road, the total delivery time in Equ. 4.2 includes the time of moving to an end of the road twice. Theoretically, in this case, we have to partition the road into two or more road segments in such a manner that there is at most one stopping location on one road segment.

Fig. 4.3 displays an example of six road segments $\{r_1,\ r_2,\ r_3,\ r_4,\ r_5,\ r_6\}$ in the VLOS area of a destination d_i and v is a vertex in the VLOS area of d_{i-1}. Fig. 4.4 displays the method of calculating the optimal stop location o on r_6. We set x to be the distance between $r_6.l_1$ and o, and other variables are constants in Equ. 4.4.

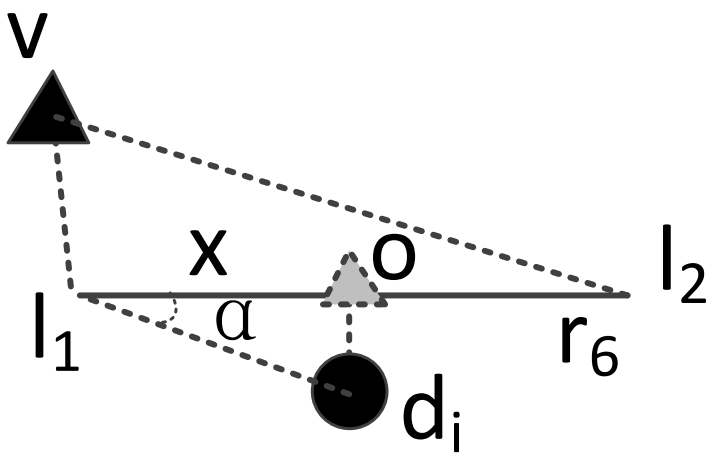


Figure 4.4: An example of finding the optimal stop location on a road segment.

The computational complexity of our solution is $O(n^2 \cdot 2^n \cdot |V|^2)$, where $|V|$ denotes the average number of vertices in the VLOS areas of destinations. If all destinations are restricted to truck-delivery only, then $|V|$ becomes 1, and the proposed GAPDP becomes the travelling salesman problem, the computational complexity of which is $O(n^2 \cdot 2^n)$.

4.3 A Heuristic for GAPDP

From our experimental results (see Section 4.4), the proposed exact solutions require hundreds of seconds to determine an optimal delivery route of 15 destinations. The execution time grows exponentially due to the nature of NP-hard problems. In this section, we propose a heuristic approach for solving the GAPDP problem of practical size.

In general, our approach is derived from heuristic approaches for the travelling salesman problem (TSP). If all destinations are restricted to truck delivery, a heuristic TSP approach can be used to address the GAPDP problem. While, if UAV delivery is allowed at one or more destinations, our heuristic method prefers 1) serving all these destinations by UAV and 2) stopping at a location to serve as many destinations as possible.

Specifically, our method first generates destination groups, each of which contains destinations that can be served by UAV from one location. The VLOS areas of the destinations in a destination group must overlap on road networks, so that the truck can stop at a location in the overlapping area, and each destination can be served by UAV in a round trip between the stopping location and the destination. If a destination is served by truck only or its VLOS area does not overlap with the ones of others, the destination forms a destination group of itself. The details of the first step are described in Alg. 3. DS denotes the set of delivery group. In the while loop (lines 3-15), delivery groups are produced in iterations until $\mathbb{D}$ is an empty set.

Algorithm 3 GAPDP($\mathbb{D}$, h)

1. $DS = \emptyset$;
2. $RoD^* = \emptyset$;
3. **while** $\mathbb{D} \neq \emptyset$ **do**
4. Let d_i be an element of $\mathbb{D}$;
5. **if** d_i is served by truck only **then**
6. $DS = DS \cup \{\{d_i\}\}$; $\mathbb{D} = \mathbb{D} \backslash \{d_i\}$;
7. **else**
8. **if** $\exists\ d_j \in \mathbb{D}$, $i \neq j$, VLOS(d_i)$\cap$ VLOS(d_j) $\neq \emptyset$ **then**
9. Let $G = \{d_k \in \mathbb{D} | VLOS(d_i) \cap VLOS(d_k) \neq \emptyset\}$;
10. DS=$DS \cup \{G\}$; $\mathbb{D} = \mathbb{D} \setminus G$;
11. **else**
12. $DS = DS \cup \{\{d_i\}\}$; $\mathbb{D} = \mathbb{D} \backslash \{d_i\}$;
13. **end if**
14. **end if**
15. **end while**
16. $RoDS^* = \mathrm{TSP}(DS, h)$;
17. **for** $ds \in RoDS^*$ **do**
18. **if** $|ds| = 1$ **then**
19. $RoD^* = RoD^* \cup ds$;
20. **else**
21. $RoD^* = RoD^* \cup$ DeliveryRouteInGroup(ds);
22. **end if**
23. **end for**
24. **return** RoD^*;

A heuristic TSP approach is applied to search an optimal visiting order for the delivery groups. If a delivery group has two or more destinations, we use the central location of

the destinations as the delivery location of the delivery group. At line 16 of Alg 3, $RoDS^*$ represents the visiting order of delivery groups generated by a heuristic TSP solution. Then, we iterate every delivery group in $RoDS^*$ in order, and compute the delivery order of destinations in a specific group by using Alg. 4. Finally, the delivery order of destinations RoD^* is returned as the result, and the delivery route and cost can be computed by using Alg. 2.

Fig. 4.5 displays an example, in which the VLOS areas of d_1 and d_2 overlap. The delivery group of d_1 and d_2 and its central location d' is found and used as the delivery location of the group in Alg. 3. Then, a heuristic TSP approach is applied to a delivery problem of $\{h,$ $d',$ $d_3,$ $d_4,$ $d_5\}$, and Alg. 4 is used to compute the delivery order of $\{d_1,$ $d_2\}$ in the group.

Alg. 4 computes a delivery order in a delivery group. In each iteration, D denotes a set of destinations, each of which overlap with all others. So, D is equal to or is contained in ds. The optimal stopping location for serving all destinations in D by UAV can be computed in $O(|D| \times |R|)$, where $|R|$ indicates the number of roads in the overlapping area ($\{\bigcap_{VLOS(d_i)} | d_i \in D\}$). At line 4, we iterate all roads in the overlapping area, and find the best location among sub-optimal stopping location on each road for delivery. The sub-optimal location on each road can be determined in $O(|D|)$. Take Fig. 4.6 for example, assume that the truck stops at $v = \{x_v, 0\}$ on road l_1l_2 for serving $\{d_1,$ $d_2,$ $d_3,$ $d_4\}$. The delivery cost is $\sum_{i=1}^{4} \sqrt{(d_i.x - x_v)^2 + d_i.y^2}$ $(l_1 \leq x \leq l_2)$, the minimum value of which can be easily obtained by using the derivative of the function.

Algorithm 4 DeliveryRouteInGroup(ds)

1. $RoD^* = \emptyset$;
2. **while** $ds \neq \emptyset$ **do**
3. $\quad D = \{d_i \in ds \mid \bigcap_{VLOS(d_i)} \neq \emptyset\}$
4. $\quad$ Iterate all roads in $\bigcap_{VLOS(d_i)}$, and find an optimal stopping location for delivery to D;
5. $\quad RoD^* = RoD^* \cup D$; $ds = ds \backslash \{D\}$;
6. **end while**
7. **return** RoD^*;

4.4 Experimental Validation

In this section, we evaluated the performance of our proposed solutions to the novel GAPDP problem over the road network of Oldenburg, Germany (containing 7K roads and 6K nodes [8]). The road network was normalized to the space of $[0, 10^4]^2$. The destinations were randomly selected on the road network. Our proposed algorithms were implemented in the C programming language. In our experimental results, "Exact Solution" refers to our exact algorithm that maintains the fastest route in a VLOS-based index. We also developed our heuristic method that applies the nearest neighbor travelling salesman solution. All data were loaded into the main memory during the execution of simulations.

The VLOS-based index was pre-built and the road distance between every pair of nodes was pre-computed in a pre-processing stage. This computation time is not included in the response time reported in our experimental results.

All the experiments were conducted on a Ubuntu Linux server equipped with two Intel Xeon E5-2670 v3 2.30 GHz processors and 256 GB of memory.

4.4.1 Effect of Number of Destinations

We varied the number of destinations from 3 to 15 in the first group of experiments. We fixed the normalized VLOS distance to 10 and 50 units. All destinations can be served by either the truck or the UAV. The normalized speeds of the truck and the UAV were fixed at 40 units per second. The experimental results are displayed in Fig. 4.7 and Table I.

The cost of delivery produced by the heuristic method grows faster than our exact solution, but the key point is that the heuristic method can complete the computation in milliseconds in all cases of this group of experiments while the exact solution needs at least 60 seconds or 400 seconds in cases of 15 destinations. This verifies that GAPDP is NP-hard; the response time of the exact solution increases exponentially.

Moreover, the exact method took longer against queries with larger VLOS areas. For example, it required 54.7 seconds when the VLOS distance was equal to 10 units; while the

response time increased to 423 seconds if the VLOS distance was set to 50 units. This is because the VLOS-based index method needed to visit more entrance-exit pairs of VLOS areas. In this group of experiments, the VLOS-index can be built, on average, in 9.7 seconds.

Table 4.2 Response Time of our Heuristic Method.

VLOS=10	Number of Destinations	20	50	100
	Response Time (s)	0.003	0.007	0.019
VLOS=50	Number of Destinations	20	50	100
	Response Time (s)	0.019	0.040	0.099

Table 4.2 displays the response time of our heuristic method in cases of up to 100 destinations (A FedEx driver typically has 75-80 stops per day [30]). The results show that our heuristic method helps us solve the real delivery problems in 0.1 second, which cannot be achieved by our exact solutions.

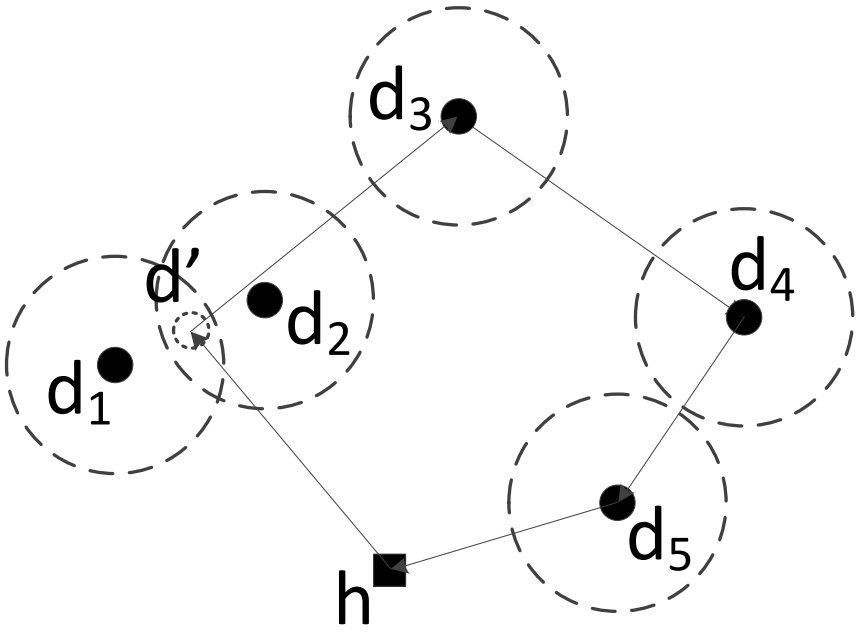


Figure 4.5: An example route of delivery group.

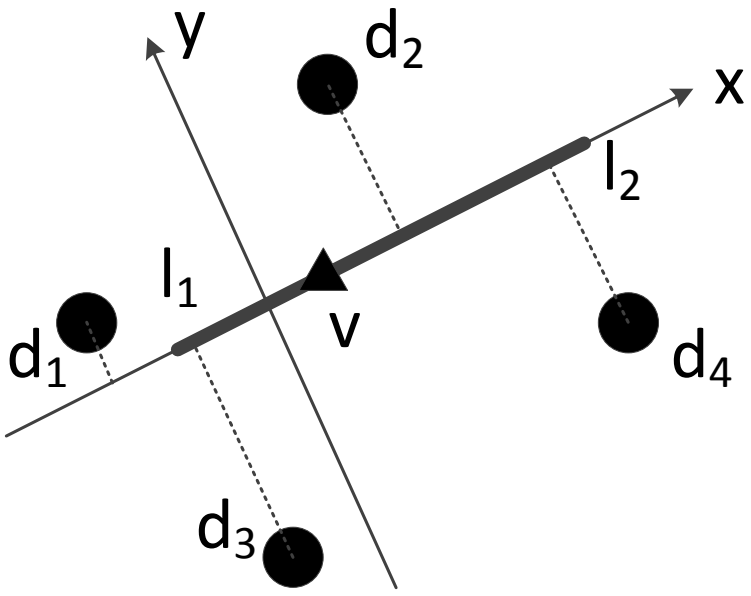
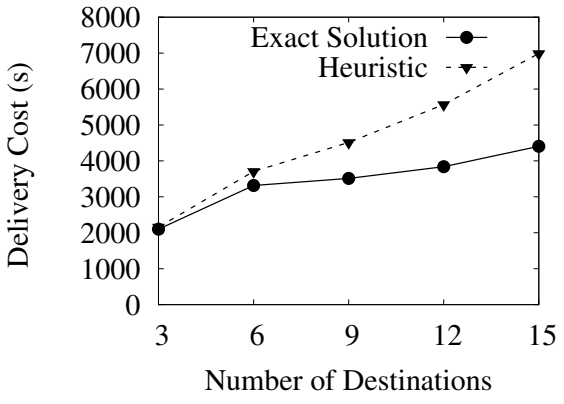
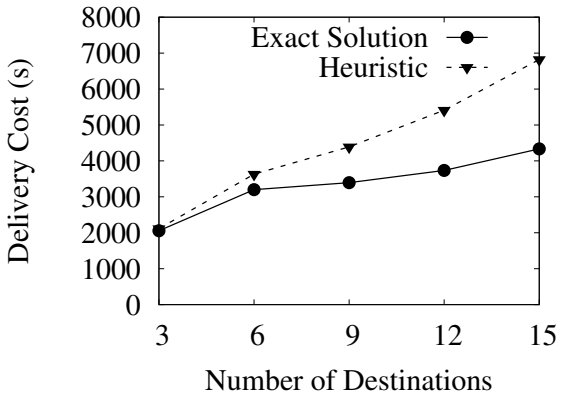


Figure 4.6: An example of computing the optimal stopping location on a road segment for delivery.

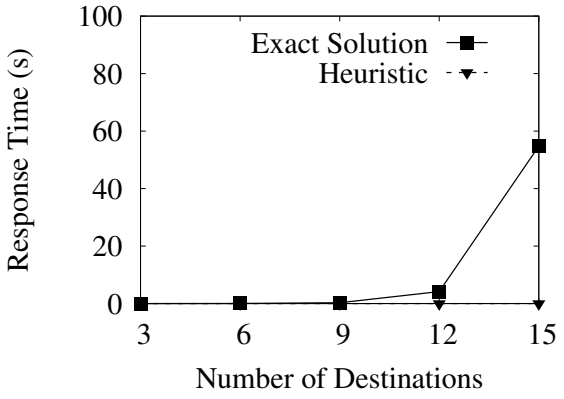
(a) VLOS=10



(b) VLOS=50



(c) VLOS=10



(d) VLOS=50

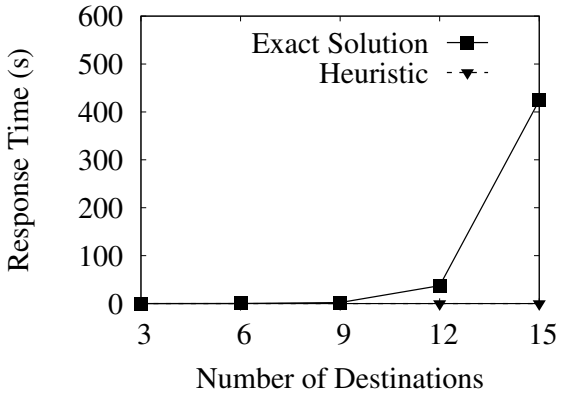


Figure 4.7: Vary the number of destinations.

Chapter 5

Conclusion

In previous Chapters, we propose a novel approach for the time-aware location sequence recommendation for cold-start mobile users, and provide either exact solution or heuristic method for the GAPDP problem. In this chapter, we will summarize our contribution and demonstrate our future work.

5.1 Main Contributions

we propose a new framework, TLSR+, which enables time-sensitive location sequence recommendation in support of sparse user-location matrix. TLSR+ makes recommendation by considering individual preference, social opinions, geospatial factors, and users' temporal behavior patterns. In the semantic similarity modeling, TLSR+ identifies the most semantically similar local experts (i.e.,users with rich local knowledge) by constructing a venue categorical semantic tree while in the spatial similarity modeling, TSLR+ retrieves the most spatially similar local experts by employing Gaussian Mixture Model (GMM) to capture users' geographical movement pattern. A Hidden Markov Model (HMM) is trained to predict the likelihood for each user to visit each location category by using the check-ins from both her semantically and spatially similar local experts. Extensive experiments on four real datasets demonstrate that TLSR+ significantly outperforms the baseline methods in terms of all effectiveness metrics.

In the GAPDP research, we formulated a novel problem that utilizes both a truck and a UAV for parcel delivery. We also considered the VLOS regulation that restricts the UAV operating range. Then, we proposed two exact solutions for the problem. One approach searches optimal stop locations for delivery by visiting all roads in the VLOS

areas of destinations. The other approach pre-computes the fastest delivery routes in the VLOS areas, and maintains them in an index. Due to the high computational cost of exact solutions, we also develop a heuristic solution for the proposed problem in practical size. We demonstrated the performance of the proposed solutions through extensive simulations.

5.2 Future Work

Considering deep learning We employ Hidden Markov Model (HMM) to model the users behavior pattern. Since neural networks are very powerful and they can be used for almost any statistical learning problem with great results, we may try to replace the existing model with it.

More efficient heuristic solution for GAPDP Due to the high computational cost of exact solutions, we develop a heuristic algorithm which can reduce the amount of calculation. However, it's still too time consuming for mobile platforms. Our future goal is to propose a algorithm friendly to mobile phones.

More practical restrictions/assumptions for delivery problem For the future work, we will apply more practical restrictions/assumptions, such as UAV regulations, in problems in order to make it closer to real applications.

Bibliography

[1] F. A. Administration. Summary of Small Unmanned Aircraft Rule (Part 107). https://www.faa.gov/uas/media/Part_107_Summary.pdf.

[2] N. Agatz, P. Bouman, and M. Schmidt. Optimization Approaches for the Traveling Salesman Problem with Drone. *ERIM Report Series*, 2016.

[3] J. A. Álvarez-García, J. A. Ortega, L. G. Abril, and F. V. Morente. Trip destination prediction based on past GPS log using a hidden markov model. *Expert Syst. Appl.*, 37(12):8166–8171, 2010.

[4] I. Amazon.com. Amazon Prime Air. https://www.amazon.com/Amazon-Prime-Air/b?ie=UTF8&node=8037720011. Accessed on May 1, 2019.

[5] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal and Ubiquitous Computing*, 7(5):275–286, 2003.

[6] J. Bao, Y. Zheng, and M. F. Mokbel. Location-based and preference-aware recommendation using sparse geo-social networking data. In I. F. Cruz, C. A. Knoblock, P. Kröger, E. Tanin, and P. Widmayer, editors, *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, pages 199–208. ACM, 2012.

[7] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state markov chains. *Ann. Math. Statist.*, 37(6):1554–1563, 12 1966.

[8] T. Brinkhoff. A framework for generating network-based moving objects. *GeoInformatica*, 6(2):153–180, 2002.

[9] K. Chang, L. Wei, M. Yeh, and W. Peng. Discovering personalized routes from trajectories. In C. S. Jensen, W. Lee, Y. Zheng, and M. F. Mokbel, editors, *Proceedings of the 2011 International Workshop on Location Based Social Networks, LBSN 2011, November 1, 2011, Chicago, IL, USA, Proceedings*, pages 33–40. ACM, 2011.

[10] H. Chen, W.-S. Ku, M.-T. Sun, and R. Zimmermann. The partial sequenced route query with traveling rules in road networks. *GeoInformatica*, 15(3):541–569, 2011.

[11] E. Cho, S. A. Myers, and J. Leskovec. Friendship and mobility: User movement in location-based social networks. In *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '11, pages 1082–1090, New York, NY, USA, 2011. ACM.

[12] C. Chow, J. Bao, and M. F. Mokbel. Towards location-based social networking services. In X. Zhou, W. Lee, W. Peng, and X. Xie, editors, *Proceedings of the 2010 International Workshop on Location Based Social Networks, LBSN 2010, November 2, 2010, San Jose, CA, USA, Proceedings*, pages 31–38. ACM, 2010.

[13] J. C. de Freitas and P. H. V. Penna. A Randomized Variable Neighborhood Descent Heuristic to Solve the Flying Sidekick Traveling Salesman Problem. *Electronic Notes in Discrete Mathematics*, 66:95–102, 2018.

[14] K. Dorling, J. Heinrichs, G. G. Messier, and S. Magierowski. Vehicle routing problems for drone delivery. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 47(1):70–85, 2017.

[15] Y. Doytsher, B. Galon, and Y. Kanza. Storing routes in socio-spatial networks and supporting social-based route recommendation. In C. S. Jensen, W. Lee, Y. Zheng, and M. F. Mokbel, editors, *Proceedings of the 2011 International Workshop on Location Based Social Networks, LBSN 2011, November 1, 2011, Chicago, IL, USA, Proceedings*, pages 49–56. ACM, 2011.

[16] G. Ference, M. Ye, and W. Lee. Location recommendation for out-of-town users in location-based social networks. In Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi, editors, *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*, pages 721–726. ACM, 2013.

[17] S. M. Ferrandez, T. Harbison, T. Weber, R. Sturges, and R. Rich. Optimization of a truck-drone in tandem delivery network using k-means and genetic algorithm. *Journal of Industrial Engineering and Management*, 9(2):374–388, 2016.

[18] B. Fuglede and F. Topsoe. Jensen-shannon divergence and hilbert space embedding. In *International Symposium onInformation Theory, 2004. ISIT 2004. Proceedings.*, pages 31–, June 2004.

[19] H. Gao, J. Tang, X. Hu, and H. Liu. Modeling temporal effects of human mobile behavior on location-based social networks. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, CIKM '13, pages 1673–1678, New York, NY, USA, 2013. ACM.

[20] D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. *Journal of Heuristics*, 20(3):291–328, 2014.

[21] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. J. Pazzani. An energy-efficient mobile recommender system. In B. Rao, B. Krishnapuram, A. Tomkins, and Q. Yang, editors, *Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, July 25-28, 2010*, pages 899–908. ACM, 2010.

[22] I. Google. Project Wing. http://www.bbc.com/news/technology-28964260. Accessed on May 1, 2019.

[23] D. P. D. Group. http://www.dhl.com/en/press/releases/releases_2014/group/dhl_parcelcopter_launches_initial_operations_for_research_purposes.html. Accessed on May 1, 2019.

[24] Q. M. Ha, Y. Deville, Q. D. Pham, and M. H. Hà. Heuristic methods for the Traveling Salesman Problem with Drone. https://pdfs.semanticscholar.org/59b4/8e77e710917d85facb5d2cebf2e2ebd5dfae.pdf, 2015. Accessed on May 1, 2019.

[25] M. Held and R. M. Karp. A dynamic programming approach to sequencing problems. *Journal of the Society for Industrial and Applied Mathematics*, 10(1):196–210, 1962.

[26] H. Hernández-Pérez, I. Rodríguez-Martín, and J.-J. Salazar-González. A hybrid heuristic approach for the multi-commodity pickup-and-delivery traveling salesman problem. *European Journal of Operational Research*, 251(1):44–52, 2016.

[27] K. L. Hoffman, M. Padberg, and G. Rinaldi. Traveling Salesman Problem. *Encyclopedia of Operations Research and Management Science*, pages 1573–1578, 2013.

[28] T. Horozov, N. Narasimhan, and V. Vasudevan. Using location for personalized POI recommendations in mobile environments. In *2006 International Symposium on Applications and the Internet (SAINT 2006), 23-27 January 2006, Phoenix, Arizona, USA*, pages 124–129. IEEE Computer Society, 2006.

[29] A. A. Hosseinabadi, M. Kardgar, M. Shojafar, S. Shamshirband, and A. Abraham. Gelsga: hybrid metaheuristic algorithm for solving multiple travelling salesman problem. In *ISDA*, pages 76–81, 2014.

[30] R. Inman. A Day in the Life of a FedEx Driver. https://www.aol.com/2010/05/28/fedex-home-delivery-driver/. Accessed on May 1, 2019.

[31] J. Jones and A. Adamatzky. Computation of the travelling salesman problem by a shrinking blob. *Natural Computing*, 13(1):1–16, 2014.

[32] Y. Kim and S. Cho. A hmm-based location prediction framework with location recognizer combining k-nearest neighbor and multiple decision trees. In J. Pan, M. M. Polycarpou, M. Wozniak, A. C. P. L. F. de Carvalho, H. Quintián-Pardo, and E. Corchado, editors, *Hybrid Artificial Intelligent Systems - 8th International Conference, HAIS 2013, Salamanca, Spain, September 11-13, 2013. Proceedings*, volume 8073 of *Lecture Notes in Computer Science*, pages 618–628. Springer, 2013.

[33] K. Kodama, Y. Iijima, X. Guo, and Y. Ishikawa. Skyline queries based on user locations and preferences for making location-based recommendations. In X. Zhou and X. Xie, editors, *Proceedings of the 2009 International Workshop on Location Based Social Networks, LBSN 2009, November 3, 2009, Seattle, Washington, USA, Proceedings*, pages 9–16. ACM, 2009.

[34] J. Krumm and E. Horvitz. Predestination: Inferring destinations from partial trajectories. In P. Dourish and A. Friday, editors, *UbiComp 2006: Ubiquitous Computing, 8th International Conference, UbiComp 2006, Orange County, CA, USA, September 17-21, 2006*, volume 4206 of *Lecture Notes in Computer Science*, pages 243–260. Springer, 2006.

[35] F. Li, D. Cheng, M. Hadjieleftheriou, G. Kollios, and S.-H. Teng. On trip planning queries in spatial databases. In *International Symposium on Spatial and Temporal Databases*, pages 273–290. Springer, 2005.

[36] B. Lindsay. *Mixture Models: Theory, Geometry, and Applications*. Conference Board of the Mathematical Sciences: NSF-CBMS regional conference series in probability and statistics. Institute of Mathematical Statistics, 1995.

[37] C. C. Murray and A. G. Chu. The flying sidekick traveling salesman problem: Optimization of drone-assisted parcel delivery. *Transportation Research Part C: Emerging Technologies*, 54:86 – 109, 2015.

[38] C. H. Papadimitriou. The Euclidean travelling salesman problem is NP-complete. *Theoretical Computer Science*, 4(3):237–244, 1977.

[39] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, pages 4–15, January 1986.

[40] L. Ramaswamy, P. Deepak, R. Polavarapu, K. Gunasekera, D. Garg, K. Visweswariah, and S. Kalyanaraman. CAESAR: A context-aware, social recommender system for low-end mobile devices. In Y. Tseng, P. Scheuermann, R. H. Güting, W. Lee, C. King, and E. Pitoura, editors, *MDM 2009, Tenth International Conference on Mobile Data Management, Taipei, Taiwan, 18-20 May 2009*, pages 338–347. IEEE Computer Society, 2009.

[41] M. Sharifzadeh and C. Shahabi. Processing optimal sequenced route queries using voronoi diagrams. *GeoInformatica*, 12(4):411–433, 2008.

[42] T. Shen, H. Chen, and W. Ku. Time-aware location sequence recommendation for cold-start mobile users. In F. B. Kashani, E. G. Hoel, R. H. Güting, R. Tamassia, and L. Xiong, editors, *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*, pages 484–487. ACM, 2018.

[43] T. Shen, J. Zhang, W. Wang, X. Jiang, W.-S. Ku, M.-T. Sun, and Y.-Y. Chiang. Optimal Stop Location Selection for two specific cases in GAPDP. *Technical Report, CSSE17-04, Auburn University*, 2017.

[44] C. Tai, D. Yang, L. Lin, and M. Chen. Recommending personalized scenic itinerarywith geo-tagged photos. In *Proceedings of the 2008 IEEE International Conference on Multimedia and Expo, ICME 2008, June 23-26 2008, Hannover, Germany*, pages 1209–1212. IEEE Computer Society, 2008.

[45] M. Terrovitis, S. Bakiras, D. Papadias, and K. Mouratidis. Constrained shortest path computation. In *International Symposium on Spatial and Temporal Databases*, pages 181–199. Springer, 2005.

[46] W. Wang and W. Ku. Recommendation-based smart indoor navigation: Poster abstract. In T. F. Abdelzaher, P. R. Kumar, A. P. Buchmann, and C. Lu, editors, *Proceedings of the Second International Conference on Internet-of-Things Design and Implementation, IoTDI 2017, Pittsburgh, PA, USA, April 18-21, 2017*, pages 311–312. ACM, 2017.

[47] X. Wang, S. Poikonen, and B. Golden. The vehicle routing problem with drones: several worst-case results. *Optimization Letters*, pages 1–19, 2016.

[48] X. Xu, H. Yuan, M. Liptrott, and M. Trovati. Two phase heuristic algorithm for the multiple-travelling salesman problem. *Soft Computing*, pages 1–15, 2017.

[49] D. Yang, D. Zhang, V. W. Zheng, and Z. Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Trans. Syst. Man Cybern. Syst.*, 45(1):129–142, 2015.

[50] M. Ye, P. Yin, and W. Lee. Location recommendation for location-based social networks. In D. Agrawal, P. Zhang, A. E. Abbadi, and M. F. Mokbel, editors, *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pages 458–461. ACM, 2010.

[51] M. Ye, P. Yin, W. Lee, and D. L. Lee. Exploiting geographical influence for collaborative point-of-interest recommendation. In W. Ma, J. Nie, R. Baeza-Yates, T. Chua, and W. B. Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 325–334. ACM, 2011.

[52] H. Yin, Y. Sun, B. Cui, Z. Hu, and L. Chen. LCARS: a location-content-aware recommender system. In I. S. Dhillon, Y. Koren, R. Ghani, T. E. Senator, P. Bradley, R. Parekh, J. He, R. L. Grossman, and R. Uthurusamy, editors, *The 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2013, Chicago, IL, USA, August 11-14, 2013*, pages 221–229. ACM, 2013.

[53] J. J. Ying, E. H. Lu, W. Kuo, and V. S. Tseng. Urban point-of-interest recommendation by mining user check-in behaviors. In O. E. Wolfson and Y. Zheng, editors, *Proceedings of the ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2012, Beijing, China, August 12, 2012*, pages 63–70. ACM, 2012.

[54] P. Zhao, X. Xu, Y. Liu, V. S. Sheng, K. Zheng, and H. Xiong. Photo2trip: Exploiting visual contents in geo-tagged photos for personalized tour recommendation. In Q. Liu, R. Lienhart, H. Wang, S. K. Chen, S. Boll, Y. P. Chen, G. Friedland, J. Li, and S. Yan, editors, *Proceedings of the 2017 ACM on Multimedia Conference, MM 2017, Mountain View, CA, USA, October 23-27, 2017*, pages 916–924. ACM, 2017.

[55] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from GPS trajectories. In J. Quemada, G. León, Y. S. Maarek, and W. Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 791–800. ACM, 2009.

[56] W.-Y. Zhu, W.-C. Peng, L.-J. Chen, K. Zheng, and X. Zhou. Modeling user mobility for location promotion in location-based social networks. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1573–1582, New York, NY, USA, 2015. ACM.