Intention-Oriented Itinerary Recommendation Through Bridging Physical Trajectories and Online Social Networks

Xiangxu Meng, Xinye Lin, Xiaodong Wang and Xingming Zhou
College of Computer Science, University of Defense Technology
Changsha, Huna - CHINA
[e-mail: aiguokk@gmail.com]

Received June 21, 2012; revised September 15, 2012; accepted November 29, 2012; published December 27, 2012

Abstract

Compared with traditional itinerary planning, intention-oriented itinerary recommendations can provide more flexible activity planning without requiring the user's predetermined destinations and is especially helpful for those in unfamiliar environments. The rank and classification of points of interest (POI) from location-based social networks (LBSN) are used to indicate different user intentions. The mining of vehicles' physical trajectories can provide exact civil traffic information for path planning. This paper proposes a POI category-based itinerary recommendation framework combining physical trajectories with LBSN. Specifically, a Voronoi graph-based GPS trajectory analysis method is utilized to build traffic information networks, and an ant colony algorithm for multi-object optimization is implemented to locate the most appropriate itineraries. We conduct experiments on datasets from the Foursquare and GeoLife projects. A test of users' satisfaction with the recommended items is also performed. Our results show that the satisfaction level reaches an average of 80%.

Keywords: Itinerary planning, trajectory mining, location-based social networks, multi-level categories

A preliminary version of this paper appeared in the ACM SIGKDD International Workshop on Urban Computing 2012 (Held in conjunction with the 18th ACM SIGKDD 2012), August 12, Beijing, China. This version includes a more detailed recommendation algorithm design, implemention and evaluation.

Thank you to Dr. Xing Xie from Microsoft Research Asia (MSRA), who has offered me a great deal of guidance, and thank you to his team, who provided me with the Beijing taxi trajectory data. Thanks to all the students who attended the Mobile Computing course taught by Prof. Xiaodong Wang and Prof. Ming Xu at the University of Defense Technology in 2012. They participated in the questionnaire investigation and the final evaluation.

http://dx.doi.org/10.3837/tiis.2012.12.010

1. Introduction

According to observations of daily life, the nature of travel arrangements involves seeking a group of geographical locations and connecting them with the traffic paths that are most convenient best meet one's personal demands. As described in reference [1], users generally seek a set of service objects that meet their various needs. For example, a traveler may have the following needs: shopping, dining, accommodation and sightseeing. These needs can only be met through a set of different geographic locations. Of course, there are many types of criteria available to evaluate the user's final choice. Some people desire the best service, and others pursue the lowest cost. In the initial stages of itinerary planning, users often have only general intentions. Their final decisions will be made after enough information has been collected. During this collection process, they may ask friends for recommendations, conduct research on travel forums, analyze traffic data themselves, etc.

When someone plans to travel to a new place, his initial intention is usually not very clear. To analyze these initial intentions, we presented questionnaires to a group of students at the National University of Defense Technology (in Changsha, Hunan, China, approximately 1500 km from Beijing). The questionnaire includes the following three questions:

1. How much do you know about Beijing? (A. Have been there many times, very familiar; B. Only been there 1-5 times, not very familiar; C. Never been there, not familiar at all.)

2. Please describe how you would make your arrangements for a one-day trip to Beijing.

3. If someone is willing to help you to arrange your trip, what demands will you have?

Example 1: This example describes the answer of a volunteer belonging to class C (Never been there, not familiar at all):

In the morning, he wants to have breakfast in a fast-food restaurant and then visit locations of historical interest. At noon, he would like to have lunch in a restaurant serving local cuisine. In the afternoon, he wishes to walk in a park. Finally, he would like to buy souvenirs for his friends and go to a party in a bar, which should be conveniently located. Before visiting all these places, he hopes to gather as much information as possible and obtain an accurate estimate of the costs of traffic, in both time and money.

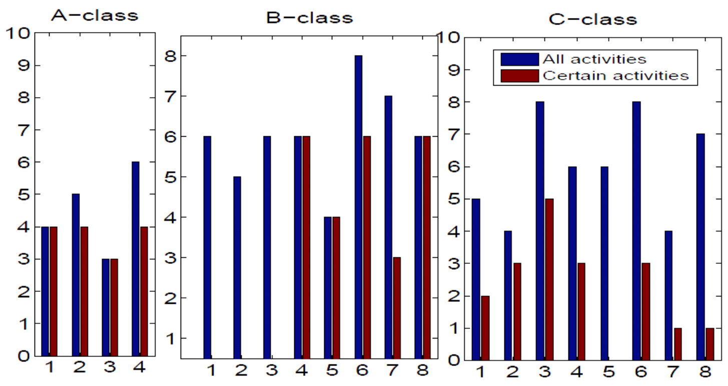


Fig. 1: Results of questionnaire

Results analysis (a total of 20 valid questionnaires were collected):

Question 1: Numbers in parentheses represent the number of volunteers belonging to each class: A. (4) B. (8) C. (8).

Question 2: For convenience of analysis, the arrangements proposed by the users are divided into two categories: 1. Specific activities: activities that have exact location requirements: for example, visiting the Imperial Palace in the morning. 2. General activities: activities with only general intentions (without particular location requirements): for example, having Sichuan cuisine for lunch. As shown in **Fig. 1**, according to their different choices in question 1, the users have been divided into 3 classes: A, B and C. Users who are familiar with Beijing (A-class) are more likely to arrange specific activities. In contrast, most of the users of the B and C classes only have general intentions. In the C-class users, the proportion of general activities is more than fifty percent. Furthermore, 4 of the 20 users have no precise preferences for their travel locations.

Question 3: The five demands with the highest occurrence rates are (the number in parentheses indicates the number of users): 1. Unique/famous places (7); 2. Least traffic time cost (5); 3. Precise location description and best route (5); 4. Avoidance of rush hour (4); 5. Lower cost (4).

After completing the analysis described above, it is reasonable to extract the following features of a common user's itinerary planning when traveling to a new place.

1. The description of the requested destination is usually a category but not a exact geographical location, such as "a famous attraction", "a well-known snack bar", etc.

2. Users demand precise information about each place they wish to visit, such as its exact geographical location, the traffic route, and the shortest and average time costs by taxi between any two locations.

3. Users are interested in hearing other people's reviews of the locations.

4. Some of the users' demands have multiple optimizing objects that are clearly expressed and can be mathematically described, such as the lowest time cost in traffic; some are very vague and difficult to describe mathematically, such as the most famous location available.

It is very difficult for a computer to implement a search or recommendation that will fully meets the user's needs with the above features because the object description is not clear, and the necessary information is too extensive. Even a human guide will find it difficult to provide such recommendations because it is impossible for him/her to have full knowledge of all the restaurants, hotels, and attractions in a city and the best traffic routes and times between them. In addition, because the GIS-based shortest route search algorithm does not consider real-time traffic changes and rush hours (weekends, people going to and from work), it is also very difficult for a route search algorithm to determine the best travel route and lowest time cost. To resolve such problems and help users make the best decisions, we jointly utilize the different users' historical trajectories and information from social networks to provide a category-based itinerary planning service. Fortunately, the current social networks and trajectory mining technologies offer effective support for such a service.

1. Currently, most mobile devices are able to provide locations based on cellular networks or GPS. This feature makes it easy to collect mobile users' trajectory data. On a much larger scale, cities' traffic control and city planning departments are also collecting the trajectory data of large numbers of vehicles and mobile devices. Some large enterprises are also doing so to conduct relative research; Microsoft's T-drive project [2][3] is one example. Because taxi drivers have the richest knowledge of a city's road systems, researchers provide a real-time navigation service by mining the driving logs of a large number of taxis.

2. Location-based social networks such as Jiepang and Foursquare support users by tagging, rating and reviewing places they have visited. These user-generated data have embraced the daily experience of millions of users and are highly valuable in the recommendation of itineraries.

All these technologies provide us with the opportunity to realize accurate itinerary recommendations. To the best of our knowledge, no commercial entity has yet provided an itinerary planning recommendation service based on intentions. This paper will focus on the realization of a general itinerary recommendation service and make four contributions:

1. Build a category-based activity scheduling model (Section 2);

2. Design and implement a category tree-based POI (point of interest) query strategy and algorithm (Section 4);

3. Propose a Voronoi graph-based GPS trajectory analysis method to build traffic information networks (Section 3);

4. Combine social networks with traffic information networks to implement category based-recommendations using an ant colony algorithm (Section 4).

Section 5 describes the demo and provides a detailed analysis of the query performance. Finally, section 7 concludes the paper with a discussion of future work.

2. Model and Framework

2.1 Model Of Itinerary Planning

It is clear that any activity must be related to a location. Therefore, an activity can be denoted as a tuple with three elements:

$$A=<P,T,Condition>$$

P (Place) denotes the geographical position, which indicates an exact location, such as the Imperial Palace, the Olympic Park, etc.

T (Time) denotes the time when the activity takes place.

C (Condition) represents a constraint on any aspect of an activity: for example, costing less than \$100/person, traveling by taxi, etc.

It is important to notice that the locations are normally hierarchically organized. According to the results of our analysis of the questionnaires, users generally do not provide an exact description of the destinations they demand. Instead, they are more likely to request a category, such as a bar, shopping mall, etc. To solve this problem, this paper makes use of the categorization of different locations on Foursquare. A category (C) is a set of different location points:

$$C=\{P_1,P_2,P_3,\ldots,P_n\}$$

The categories are hierarchically organized. The locations belong to categories.

$$C_{\mathrm{i}} \subset C_j \qquad P_i \in C_{\mathrm{j}}$$

A *path* indicates the connecting route between two locations; i is the description, such as the time cost.

$$Path=<P_s,P_e,i>$$

A trip is a time-ordered description of several activities and the paths between them.

$$Trip=\{A_1,Path_1,A_2,Path_2,\ldots,A_n\}$$

In this paper, the question mark (?) is used to indicate a single place, and the asterisk (*) is used to indicate all the places in a category. With the use of these two marks, a user's vague itinerary-planning demands could be described as follows:

[illegible]

optimizer denotes the optimization goal, such as the shortest travel time or the hot spots of all the POIs. According to the model, example 1 can be described as follows:

[illegible]

2.2 Traffic Information Networks

To identify the best possible trip, we need to build a cost function. The cost could be the time cost, the service evaluation, or the total length of the trip. If we want to identify the lowest time cost, information on the driving time between any two places becomes a necessary value. Thus, we build a semantic traffic map G of a city based on its taxis' trajectories:

$$G=\{P,Path\}$$

P is the set of all the geographical places, and *Path* is the set of all the existing edges between these locations. To build a map G, we need to obtain information on all the locations and the paths between them. The detailed method of building this semantic traffic map is presented in Section 3.

2.3 Framework Of Joint Itinerary Planning

This paper accessed information about geographical locations from social networks and built a semantic traffic map by mining the GPS trajectories, as shown in **Fig. 2**. According to different cost functions, the system will present different recommendations for itinerary planning with detailed explanations. These explanations include valuable information for the user, such as the popularity of each location and the driving times between adjacent places on the trip. The entire system consists of three repositories:

1. **Repositories of location points:** We crawled all the location points provided by various users and evaluated by millions of users in the target city on location-based social networks. This collaboratively evaluated information can more objectively indicate the degrees of popularity of the location points;

2. **Categories of location points:** These categories save locations' classification information. In this paper, we use the hierarchy categories provided by Foursquare;

3. **Traffic information networks:** This repository contains information on the shortest travel times and the best paths between any two arbitrary semantic location points in a city. We obtain the semantic locations to which users pay close attention from public transit agencies, and we mine taxi history data to obtain accurate information on the shortest paths (section 3.2).

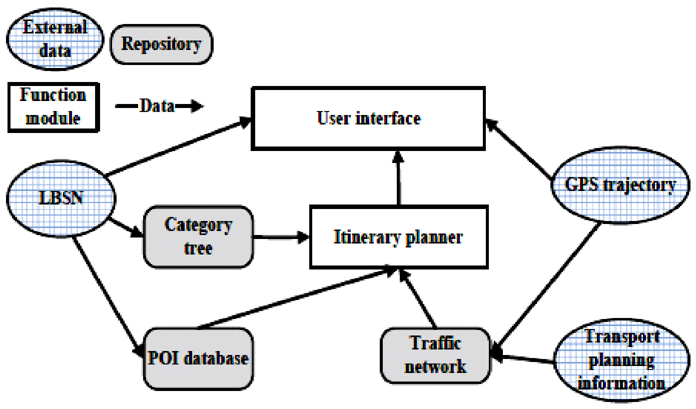


Fig. 2. Framework of joint itinerary planning

Furthermore, the itinerary planner is responsible for recommending reasonable itineraries based on the information above, according to each user's requirements. The UI module calls an online map system (such as Google Map) to display itineraries visually and supports users' efforts to modify their requirements interactively.

2.4 Flowchart Of Itinerary Recommendation

The recommendation process is described in Fig. 3. First, the user sends his itinerary needs to the itinerary recommendation server. The server then calls for the LBSN and the trajectory processor (responsible for the maintenance of traffic information networks) to obtain data for the recommendation and used the category-based itinerary recommendation algorithm (section 4) to compute candidate itineraries. The users can select a special candidate itinerary to view more detailed information obtained from location-based social networks.

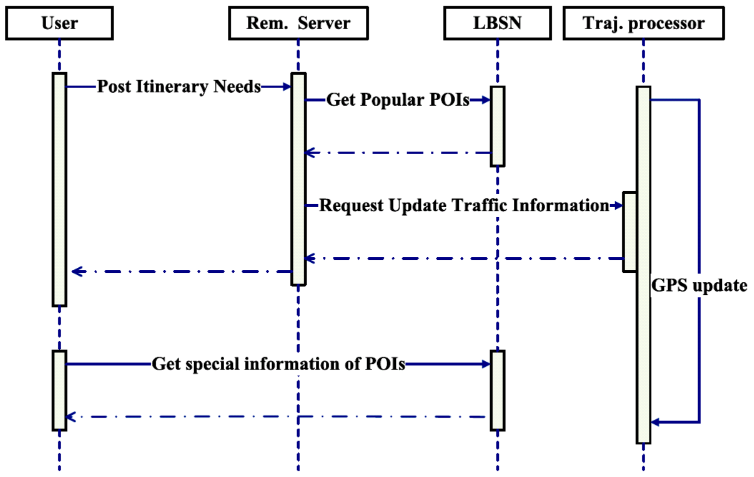


Fig. 3. Flowchart of itinerary recommendation

3. Voronoi Graph-Based Traffic Information Networks

Traffic departments in many countries and cities are collecting the trajectories of public transportation vehicles. These GPS data have high frequencies and large time spans; therefore, they are very difficult to process. In addition, these trajectory data only contain spatial and time stamps, so they are unable to provide semantic information [4], which the user generally understands better and finds more useful. There are two main obstacles to converting the physical trajectory into feasible semantic knowledge: 1. Determining the physical location that the user understands best and knows well; 2. Identifying the shortest path and driving statistics between any two location points. In the next part of this section, we propose a semantic position-determining algorithm based on public transportation stops and a semantic traffic map-building algorithm based on a Voronoi diagram. These two algorithms make it possible to obtain the shortest time and optimal path between any two locations.

3.1 Voronoi-Based Semantic Point Building

Every city contains thousands of geographical locations or more; it would be very difficult for a resident to memorize all of them, let alone a stranger. Traditionally, the most frequently used method of describing a geographical location combines its “traffic network” and its landmarks: for example, No. 163, Haidian West Road, Beijing; or 400 m east of the south gate of the Olympic Park. Fortunately, when planning a stop or station on public transportation, the related departments always take the population, physical distance, traffic situation and other factors into consideration. These stops are commonly named according to semantic markers that the residents know well. Concerning the planning principle of these stops, it is reasonable to assume that for any important location A of a city, there is always a stop B so the time cost from B to A remains smaller than a given threshold: for example, 10 minutes. Based on this assumption, this paper tries to build a semantic map of a city based on the stops of its public transportation. As shown in **Fig. 4**, the entire city area is decomposed using a Voronoi diagram, taking these stops as seeds. Each cell has a unique ID named after the seed stop inside it. Thus, each cell generates a semantic point with a constant ID and a constant name (***SemiPoint***). The Voronoi diagram insures that for any geographical point (GPS-Point, ***Point***), there is a corresponding nearest semantic point (in our case, a stop).

Therefore, a GPS trajectory can be described by a semantic trajectory (Semi-trajectory, ***SemiList***), which is a serial of SemiPoints. The information about all the stops can be accessed on the traffic department's website. The Voronoi diagram is built using the classic plane sweep algorithm [5].

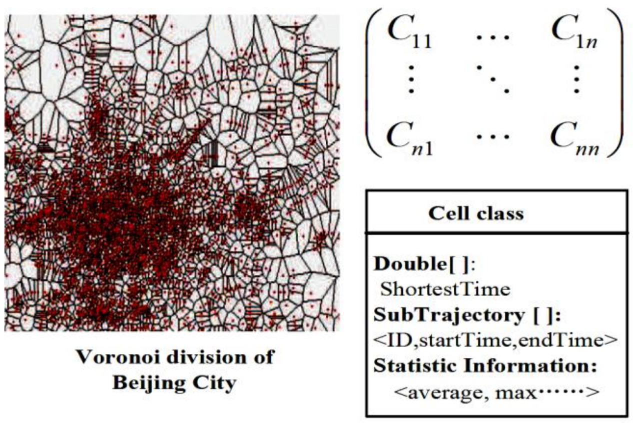

$\begin{pmatrix} C_{11} & \cdots & C_{1n} \\ \vdots & \ddots & \vdots \\ C_{n1} & \cdots & C_{nn} \end{pmatrix}$
Voronoi division of Beijing City
Cell class
Double[ ]:
ShortestTime
SubTrajectory [ ]:
<ID,startTime,endTime>
Statistic Information:
<average, max……>

Fig. 4. Voronoi based semantic point

The upper-right corner of **Fig. 4** is an n*n matrix; each element cell$_{ij}$ stores the sub-trajectory and statistical information of the path that runs from the i-th semantic stop to the j-th semantic stop. For example, cell$_{ij}$ can save the following information: all the trajectories from the i-th stop to the j-th stop (***SubTrajectory[]*** array of the right lower sub-figure), the shortest travel time (***Double []*** array) on the basis of the trajectories calculated from i to j, and other detailed statistical information, such as the average running time and the shortest path according to the statistical hour. Furthermore, the cell data structures can be modified according to the system's needs to provide more advanced traffic information.We treat each stop point separately as a start and a destination to build the semantic traffic map, which is stored in the form of an adjacent matrix (***Time-Matrix[Stop Num][StopNum]***). One element of this matrix describes the traffic statistics between two semantic points. The specific method used to generate these traffic statistics is determined by the administrator according to different demands and implemented using different statistical methods (***StatisticFunction()***: for example, during a seven-day week, calculating the shortest time and the best path every two hours. Considering that there are only seven days of data in the demo system, we only calculate the shortest time and best path between any two semantic points for rush hours on weekdays and weekends.

3.2 Building Run Through Model-Based Traffic Information Networks

After decomposing the city map, we must calculate the statistics between any two cells. In a traditional GIS system, the best path is simply the shortest path that can be located easily, and the shortest time could be estimated according to its Euclidean distance. However, in real life, traffic restrictions, one-way roads, dynamic traffic flows, and many other factors generate a very complicated situation in which the shortest path usually is not the one with the lowest time cost. Furthermore, no model or algorithm exists that can identify the path with the shortest travel time both precisely and independently.

A GPS trajectory contains a detailed log of each vehicle's travel history. Each GPS point includes latitude and longitude coordinates and the corresponding time stamp. We can make use of these data to directly obtain the travel path and the time cost between two locations. The original trajectory data are enormous, difficult to handle, and hard for users to understand. Therefore, many trajectory-based intelligent applications utilize a semantic trajectory model [4][6][7][8]. The semantic trajectory model greatly reduces the computational and storage costs of processing trajectories and contains more semantic information; it is also easier to combine with other information for processing and mining. Our work is a type of intelligent service; therefore, the semantic trajectory model can also reduce the calculation cost and enhance the user's experience.

It is understood that nearly all the semantic trajectory-building methods [9][10][11] are based on a "stop and move" model. **Stops** are important places on the trajectory from an application point of view; they represent where a moving object has stayed for a minimal amount of time. **Moves** are sub-trajectories between two consecutive stops. As **Fig. 5** shows, a trajectory can be described as a series of "stops" (clusters of yellow points that indicate that an object has stayed there beyond a time threshold [9]).

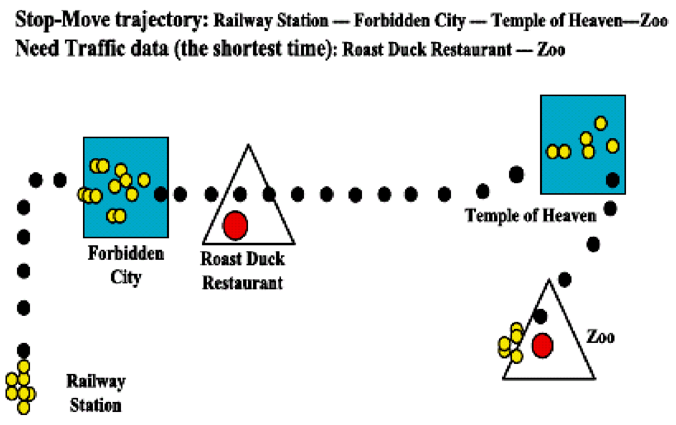

Stop-Move trajectory: Railway Station — Forbidden City — Temple of Heaven—Zoo
Need Traffic data (the shortest time): Roast Duck Restaurant — Zoo
Forbidden City
Roast Duck Restaurant
Temple of Heaven
Zoo
Railway Station

Fig. 5. Semantic trajectory building methods

We need to determine the shortest time from "Roast Duck Restaurant" to "Zoo" (red circles, where the triangles present the Voronoi cell corresponding to the two bus stops in **Fig. 4**) to conduct the itinerary recommendation, as shown in **Fig. 5**. However, we cannot obtain this information from the "Stop-Move trajectory" because there is no "stop" near "Roast Duck Restaurant".

As previously discussed, the semantic city map must cover all the "bus stops" through Voronoi division. The "stop and move" model does not meet our needs because some points have no meaning from the user's perspective but are valuable as traffic information statistics. Therefore, we designed a "run through" model for the traffic statistics, as shown below.

To match the semantic city map, the cell-ID of each GPS is allocated to it as its Semantic Point. We then mapped millions of POI points to tens of thousands of stops, significantly diminishing the storage and computation costs. Considering the way the semantic city map is generated, we can confirm that the time errors introduced by this mapping strategy will not be greater than the time cost of a vehicle passing through a cell. For the example of Beijing, the travel time of buses between any two stops in Beijing is strictly less than 10 minutes; this means that the resulting shortest time cost will include an error of less than 10 minutes, which is acceptable in the scenario of itinerary planning. In a practical sense, the difficulty is that, there are not enough trips (a trip, as previously defined, is a complete trajectory from start to end) for any pair of Semantic Points to calculate the statistical driving time.

Therefore, we propose that we treat all the cells that are passed as statistical sources, even if they are not the start or end points of the path. Therefore, if we denote a trip As－Ae, after we take the run through cells into consideration, it can be denoted as follows:

$$\langle A_s A_1 A_2 A_3 \cdots A_n A_e \rangle$$

Algorithm 1 Voronoi-based Time Matrix Building

```
    Input: Trajectory:traj; Voronoi-Map: map
    Output: Time-Matrix[CellNum][CellNum]: matrix
 1: Point p=traj.next();//Fetch a new GPS point
 2: SemiList tmpList=new SemiList();
 3: while (P!=null) do
 4:    SemiPoint Sn=M.semi(P); //Get the semi-point
       based on physical location
 5:    if (Sn is different from temList.last() ) then
 6:       for each SemiPoint Si ∈ temList do
 7:          Matrix[Si.ID][Sn.ID]=StatisticFuction();
 8:       end for
 9:       tmpList.add(semiPoint);
10:    end if
11:    p=traj.next();
12: end while
13: return matrix;
```

Algorithm 1 shows that when calculating the path statistics between cell A1 and A3, the sub-trip A1－A2－A3 of As－Ae, which runs through both A1 and A3, is also included in the statistics. In particular, this method can determine potential shortest paths within a longer trip. After entering a new cell, the algorithm will traverse all the historical semantic points in the trip, store the traveling time between each historical semantic point and the new semantic point, and update the statistics (line 5-8).

4. Category-Based Itinerary Recommendation Algorithm

4.1 Hierarchy Of Location Points

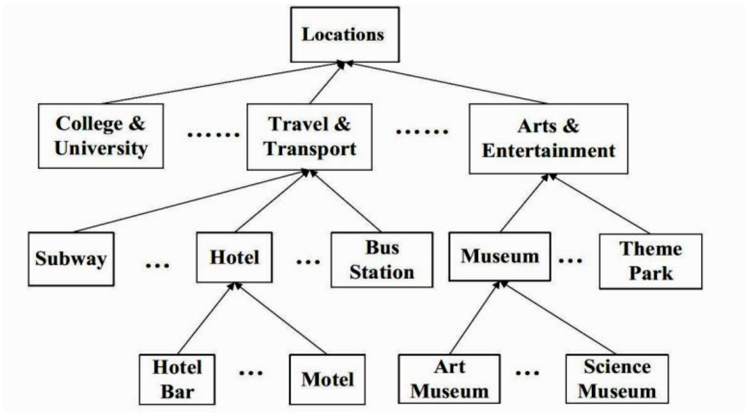


Fig. 6. Hierarchy of the location points

There are many methods of the hierarchical categorization of locations. This paper makes us of the categorizations of Foursquare, one of the earliest and largest LBSN websites today. Fig. 6 partially depicts the hierarchy of this categorization in the form of a tree. As shown in Algorithm 2, when a user provides his/her requested location category, all the locations belonging to that category and the corresponding subcategories are taken into consideration.

4.2 Least-Time Itinerary Planning Recommendation

A good itinerary is created according to many factors, which make it a constraint optimization problem. These factors include the users' needs and other potential conditions. Hyoseok Yoonetc [12] has proposed three factors that are very reasonable:

Elapsed Time Ratio: An itinerary that uses as much available time as possible is considered better; because time is a limited resource, users wish to utilize most of their available time.

Stay Time Ratio: People should spend more time at the locations than they do traveling. An itinerary with less traveling time and more on-site staying time on is considered a better choice.

Interest Density Ratio: Visitors to a new region would like to visit as many highly interesting locations as possible, i.e., popular locations and locations of cultural importance.

This subsection proposes an itinerary planning recommendation algorithm that uses a cost function considering time cost, aiming to provide recommendations with the lowest time costs. In section 4.3, we will simultaneously consider the time cost and popularity of the POIs.

4.2.1 Baseline: Single-object optimization

Algorithm 2 Category-based Activity Scheduling

```
    Input: needs; ct; PI; matrix;k
    Output: Tuple(POI[n], Path[n-1],Cost):scheduling
 1: for all need needs[i] do
 2:    CategorySet cs[i]=ct.subCategory(needs[i].category);
 3:    get top-k famous POIs POIs[i][k] belongs to cs[i];
 4: end for
 5: for all composition POI[n] built by geting one ele-
    ments from each column of POIs[i][k] do
 6:    for all $i < n - 1$ do
 7:       build Path[i] using POI[i],POI[i+1];
 8:       cost=CostFunction(matrix.pathCost[i]) ;
 9:    end for
10:    scheduling.add(POI[n],Path[n-1],cost);
11: end for
12: scheduling.sortBy(cost);
13: return scheduling;
```

We implement the baseline recommendation algorithm in three steps. The algorithm enumerates all the possible locations, traverses all the feasible trips and attempts to identify the best one to recommend. In the previous section, we received the user's needs (***needs***), built a traffic graph matrix (matrix) with each element presenting the costs between two semi-points (***Cost*** calculated by ***CostFunction( )***), obtained a category tree from Foursquare (CategoryTree, ***ct***) and built an index of POIs (POIsIndex, ***PI***) using ***CategorySet*** to describe the categories to which each POI belongs. As **Algorithm 2** shows, in the first step (line 3), the k most popular geographical locations are given to each activity instance as candidates; in the second step (lines 5-8), one of the k locations is chosen for each activity instance, the path connecting them is found by checking the semantic traffic map and a recommendation item is generated; in the third step (line 10), the set of all recommendation items is ranked according to the cost of each item, and the final recommendation list is generated.

4.2.2 Challenges Of The Baseline Algorithm

The baseline algorithm is simple and intuitive, but it faces the following challenges:

Combinatorial explosion: The algorithm needs to traverse all the possible combinations of all the activities' candidate locations, which is at the level of k dim (dim is the number of activities). When k increases, the computing cost increases immensely. For example, when dim is 6 and k is set to 10, the time cost of the baseline algorithm reaches 2328 ms, which cannot meet the real-time needs of an online service.

Overfitting of a single object: When the optimized object must obtain the least time cost and k is set to a large value (which means more candidate locations), the algorithm may be overfit to a single object, with the result that each location's popularity is ignored, and the final recommendation will be a group of locations geographically assembled in a small area.

Therefore, we conclude that the itinerary planning recommendation problem should be treated as a multi-objective optimization problem that does not strictly require an optimized result; an approximate optimization result is acceptable. In the following section, we propose a multi-objective optimization algorithm aimed at finding approximate optimizations in a short time.

4.3 Ant Colony Optimization (ACA)-Based Multi-Objective Itinerary Planning

The multi-objective optimization problem is very common in scientific research and engineering practice. In general, it includes a final object that is composed of several objects. This type of optimization is usually high-dimensional and large in scale, and it must consider the weight allocations of different sub-objects. However, we cannot precisely design the weight allocations for the different sub-objects of the problem discussed in this paper. For example, it is difficult to determine whether the travel time or the location's popularity is more important. We hope that the planning algorithm developed here will have the following characteristics: **1) Strong availability:** it should provide each user request with an approximate solution as quickly as possible, and it should converge quickly; **2) Extensibility:** it should be easily joined with other methods for further optimization; **3) Parallelism:** it should be easy to implement in parallel to make full use of the computing resources of multi-core systems or a cluster system. Through the comparison of various optimization methods, we found that the genetic algorithm (Genetic Algorithm, GA), the ant colony algorithm (Ant Colony Optimization, ACO) and the simulated annealing algorithm (Simulated Annealing, SA) and other heuristic algorithms are suitable to meet our needs. According to the conclusions presented by the reference [13], the ant colony algorithm has a faster convergence speed than the GA and the simulated annealing algorithm. Furthermore, the results obtained when solving combinatorial optimization problems using the ant colony algorithm with suitable parameters is better than those obtained with the genetic algorithm, the evolutionary algorithm and the simulated annealing algorithm when the number of nodes is lower than 100 [13]. Therefore, we choose the ant colony algorithm for itinerary recommendation because it can reduce the time and space costs of the combinatorial explosion problem. This paper proposes a modified version of the ACA to solve the multi-objective optimization of itinerary planning recommendation. The modified ACA focused primarily on the weight allocations of different sub-objects. Furthermore, it is convenient to adjust the number of ants and running times according to the system load, improving the performance of the system.

4.3.1 Building the ant colony system

ACA is a bionic algorithm [14] whose main idea is to simulate the food-seeking behavior of ants. It initially places a large number of ants that randomly wander in the search space. Once an ant finds some "food", it puts pheromone on the path to increase its attraction to other ants; the pheromone evaporates over time. This positive feedback strategy leads the algorithm to a global optimization. ACA is intrinsically parallel, which makes it easy to program in parallel. This paper introduces an ACA and makes the following modifications: it retains the global optimization object unchanged, i.e., finding the shortest path; and it introduces a heuristic rule that merges the locations' popularity with the pheromone updating phase so the ants tend to choose locations with higher popularity. Activity scheduling is denoted as an ordered list with elements from n non-intersecting sets.

$$Path = \langle S_1, S_2, \ldots, S_n \rangle$$

p_{ij} denotes the j-th location in the i-th set, assuming that each set has k elements. For any two elements from two adjacent subsets, respectively, a connecting edge exists.

$$E(p_{ij}, p_{(i+1)l})$$

To describe the ant colony system, the following symbols are defined:

$A^t(k)$ --describes the state of the k-th ant in the t-th visit.

$D(p_{ij}, p_{(i+1)l})$-- describes the minimal delay between two POIs.

$I^t(p_{ij}, p_{(i+1)l})$-- describes the information between two POIs of the t-th visit.

$P_k^t(p_{ij}, p_{(i+1)l})$-- describes the probability of the k-th ant transfer from p_{ij} to $p_{(i+1)l}$.

$H(p_{ij})$ -- describes the popularity of a POI.

4.3.2 Itinerary planning based on a modified ACA

The procedure of the modified ACA is as follows:

1. Initialization: randomly place each ant on the locations of the first set. (Algorithm 3, lines 1-4)

2. The ants traverse the n sets in order: the transfer probability between two sets is as follows. (Algorithm 3, lines 6-10)

$$P_k^t(p_{ij}, p_{(i+1)l}) = \frac{I^t(p_{ij}, p_{(i+1)l}) \times H(p_{(i+1)l})}{D(p_{ij}, p_{(i+1)l})} \qquad (1)$$

3. After all the ants have completed their travel, calculate the total cost of each path and save the one with the smallest cost. In the meantime, evaluate the current state and determine whether the ending condition has been satisfied. If it is satisfied, return the best path; else, update the pheromone of each edge. When an ant completes a round, the pheromone of each edge is changed according to the following equation. (Algorithm 3, lines 11-18)

$$\Delta I_k^t(P_{ij}, P_{(i+1)l}) = \begin{cases} \frac{Q}{L_k^t} & \text{[illegible]} \\ 0 & \text{[illegible]} \end{cases}$$

$$I^{t+1}(P_{ij}, P_{(i+1)l}) = (1-\alpha) \cdot I^t(P_{ij}, P_{(i+1)l}) + \sum_{k=1}^{m} \Delta I_k^t(P_{ij}, P_{(i+1)l}) \quad (2)$$

4. Re-place all the ants and start a new period. (Algorithm 3, line 19)

According to the procedure described above, the pseudo-code of the modified ACA is given below.

Algorithm 3 Ant Colony based Activity Scheduling

```
1:  for all edge, ant do
2:      $I^1(P_{ij}, P_{(i+1)l}) = I_{initial}$;
3:      Place ant on a randomly choose POI of $S_1$ ;
4:  end for
5:  Let $Path_{minDelay}$ be the best path and $L_{min}$ its delay;
6:  for $t = 1$ to $t_{max}$ do
7:      for $k = 1$ to $m$ do
8:          Build tour $Path_k^t$ by applying n-1 times the following step;
9:          Choose next POI with probability computed by Formula (1);
10:     end for
11:     for $k = 1$ to $m$ do
12:         Compute the delay $L_k^t$ of $Path_k^t$ produced by ant k;
13:     end for
14:     if an faster path found then
15:         Update the $Path_{minDelay}$ and $L_{min}$;
16:     end if
17:     for all edge do
18:         Update the $I^{(t+1)}(P_{ij}, P_{(i+1)l})$ by Formula (3);
19:         Replace ant on a randomly choose POI of $S_1$ ;
20:     end for
21: end for
22: Return the $Path_{minDelay}$ and $L_{min}$;
```

4.3.3 Configuration of pertinent parameters in ACA

ACA is a randomized parallel searching algorithm, which is similar to other bionic algorithms, that searches for the optimized solution by promote several candidate solutions to evolve and change. This evolution process needs both the individual ant's adaptability and the interactions between different ants. Therefore, the performance and convergence of ACA are affected by the ant number m, the pheromone decay factor ((1-α) in Eq.2) and many other parameters at the same time. According to the analysis of many existing application cases of ACA in [15], the following conclusion holds: theoretically, larger number of ants can improve the overall searching ability and stability of ACA. However, as in practical applications, when the ants number is very large, the pheromone left on paths that have already been searched tends to be the same, which will decrease the positive feedback of the pheromone and thus slow down the convergence of the algorithm. On the contrary, when the number of ants is too small, and the search space is large, pheromone on those paths that seldom been searched will disappear (become zero) very soon, which decreases the randomness of the search process and makes the algorithm halt too early leading to instability.

It's a very complicate problem to decide ants number and other parameters for ACA. There're no perfect theories that can present fully support yet, and these parameters are usually decided empirically. In general case, the parameters can be set up by trial, but this will deteriorate the performance and convergence of ACA. [15] has conducted a detailed analysis both theoretically and experimentally on how to set up parameters for ACA, and a practical scheme is proposed. In this paper, we take use of this scheme and decide the ants number m and the pheromone decay factor 1-α. Firstly, we use her experimental conclusion that (all the candidate POI number)/(ants number) is approximately 1.5. In the evaluation in Section 5.2, we assume that the number of user's activities each day is no more than 10 (based on the practical data), and lies from 3 to 9, the candidate set size of each activity is about 10 (we choose 3, 6, 9 in the experiment separately). Thus the ants number should be within 20 to 60 (derived from 3*10/1.5 and 9*10/1.5 separately). Based on this analysis, we set the ants number to 20, 40, 60 separately in the experiment. Based on several rounds of experiments with different parameters, we conclude that the algorithm achieves the best performance when α is set to 0.5 while keeping other parameters the same. Therefore, experiments of Section 5.2 are all conducted with α=0.5.

Through these experiments, the effects of ACA algorithm, such as the time delay and recommendation results, had been evaluated. As for the future work, the optimized configuration of ACA algorithm with different parameters will be discussed and evaluated in details.

5. Demo System and Evaluation

Foursquare is the largest location-based social network, with more than 20 million active users. We have built an experimental system based on the Foursquare and T-drive [3] datasets for Beijing. Information on the experiment platform and dataset is listed as follows:

Hardware configuration: Our experiments were conducted on a quad-core 2.27 GHz Intel i3 CPU with 2G RAM and a 320G 5400 RPM disk driver. The disk page size is 4 KB (4096 Byte).

Software configuration: Our OS is Ubuntu 11.04 (Linux 2.6.16). All the indexes are developed using JAVA 1.6 without optimization by multi-threads technology; the available memory of JVM is set to 1 G.

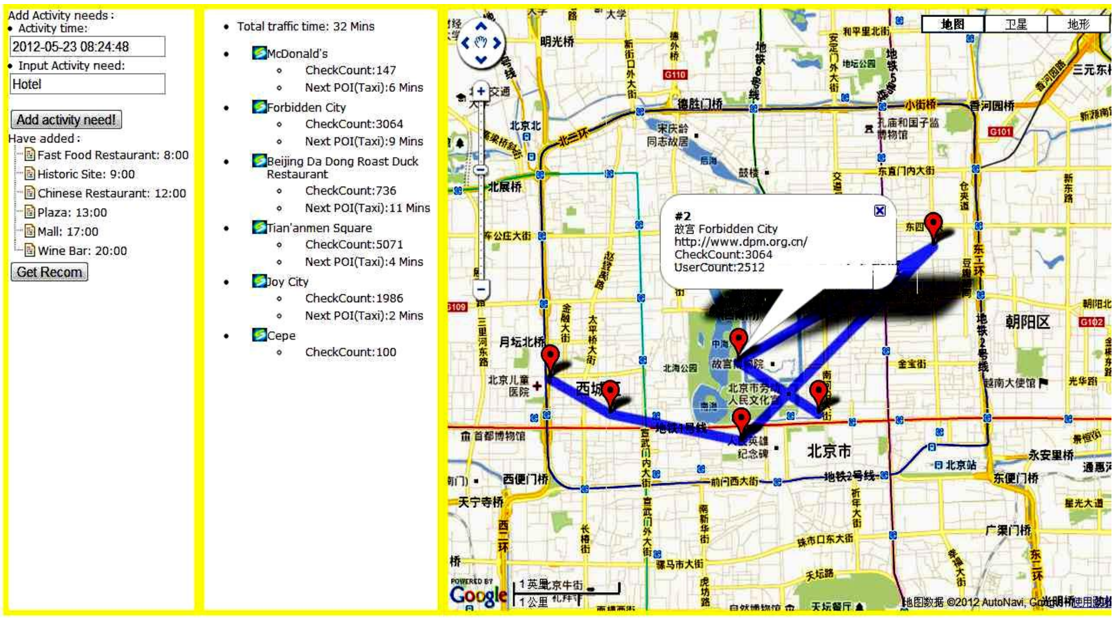
Public transportation stops data: We collected 10,684 bus stops from the Beijing public transportation network, which covers all Beijing’s urban and suburban areas. The important characteristic of this dataset is that it comes from the real traffic system. Users are familiar with the stop names, which are easy to remember, because they use public transportation every day. At the same time, the distributions of these sites are reasonable because they were chosen by the city planning department according to information about the population, environment and culture.

Foursquare dataset: We collected 30,784 effective POIs and the category information by calling the open API provided by the Foursquare Company. Each POI has a variety of information, such as its total “check-in” times, its number of historical visitors, user-submitted reviews, website addresses and so on. To quickly identify the entire candidate POIs belonging to the same categories, we built an inverted index [16] for every category and sort each POI list according to “check-in” times. The dataset is the result of crowd intelligence coming from online social networks. Therefore, it includes more comprehensive and richer information than any commercial dataset.

GPS trajectory dataset: The real dataset is provided by the T-drive project of Microsoft Research Asia, which includes all the trajectories of 10,357 taxis from 2008-02-02 to 2008-02-08 in Beijing [3].

5.1 Recommendation Results and User Interface

Table 1 lists the recommendation results computed by the baseline algorithm and modified ACA with k=3 and k=6, respectively, for example 1. The user interface is presented in Fig. 7. The users input their travel intentions through selecting a category in the left-hand input area one by one with expected start time. Next, these intentions are submitted to the recommended server for itinerary planning. The server calculates the candidate recommendation paths, as shown in section 2.4, and the UI displays them in the middle region sorted by start time. The recommendation results include the location of each activity and its popularity, as indicated by "check-in" times (shown after the POI name). Users can also get the traffic information form a location to the next one in a trip (the number in brackets). Furthermore, users can click the desired elements to browse the homepage of every POI and review the information on the Foursquare website, including reviews and other users' evaluations. At the same time, recommendations results are drawn in the Google map include all the POIs connected through a line according time. And you can click every POI to get its detail information which will shown by a tip in Google map.. From the above information, the users can develop an overall understanding of the recommended itinerary.


Add Activity needs:
- Activity time:
2012-05-23 08:24:48
- Input Activity need:
Hotel

Have added:
- Fast Food Restaurant: 8:00
- Historic Site: 9:00
- Chinese Restaurant: 12:00
- Plaza: 13:00
- Mall: 17:00
- Wine Bar: 20:00

- Total traffic time: 32 Mins
- McDonald's
  - CheckCount:147
  - Next POI(Taxi):6 Mins
- Forbidden City
  - CheckCount:3064
  - Next POI(Taxi):9 Mins
- Beijing Da Dong Roast Duck Restaurant
  - CheckCount:736
  - Next POI(Taxi):11 Mins
- Tian'anmen Square
  - CheckCount:5071
  - Next POI(Taxi):4 Mins
- Joy City
  - CheckCount:1986
  - Next POI(Taxi):2 Mins
- Cepe
  - CheckCount:100

#2
故宫 Forbidden City
http://www.dpm.org.cn/
CheckCount:3064
UserCount:2512

Fig. 7. User interface

The evaluation results in Table 1 present the average values of all the evaluations (a value between 1-10 is used to indicate the degree of satisfaction) completed in the questionnaire by the 14 volunteers belonging to classes A and B. The satisfaction levels of all four recommendations are, on average, greater than 7. At the same time, we find that although the ant colony algorithm carries a larger time cost, its recommendation results are better accepted by the users than those of the baseline algorithm. It is interesting that when more candidate locations (larger k) are taken in the recommendation, the user satisfaction decreases. This demonstrates that the users care more about places' popularity than about several minutes' additional cost. In the next step, we will provide an online service that is more sensitive to the recommendation algorithm's performance. In the next subsection, the time costs of the two algorithms are evaluated

Table 1. Itinerary recommendations for Example 1 in Beijing City

Category	Activity scheduling	Time (Mins)	Score
K=3 (Baseline)	McDonald's (143)-->6(Mins)--> Forbidden City(2914)-->9(Mins)-->Beijing Da Dong Roast Duck Restaurant (720)-->5(Mins)-->Jianwai SOHO(277)-->4(Mins)-->Sanlitun Village (5688)-->5(Mins)-->CJW The Place(167)	25	8.1
K=6 (Baseline)	McDonald's (143) -->3(Mins)--> East Gate: Temple of Heaven(193) -->3(Mins)-->Duck de Chine (287) -->3(Mins)-->Jianwai SOHO(277) -->2(Mins)--> Silk Street Market(1017) -->0(Mins)-->CJW The Place(167)	11	7.4
K=3 (Ant colony)	McDonald's (143) -->6(Mins)--> Forbidden City(2914) -->9(Mins)-->Beijing Da Dong Roast Duck Restaurant (720) -->11(Mins)--> Tian'anmen Square(4908) -->4(Mins)--> Joy City(1933) -->2(Mins)-->Cepe(94)	32	8.9
k=6 (Ant colony)	McDonald's (143) -->6(Mins)--> Forbidden City(2914) -->9(Mins)-->Beijing Da Dong Roast Duck Restaurant (720) -->5(Mins)-->Jianwai SOHO(277) -->0(Mins)-->The Place (1798) -->5 Mins)-->Enoterra(338)	25	8.2

5.2 Performance Analysis

Recommendation results offer an important value to travelers. However, any discussion of an online service must also be sensitive to the algorithms' time cost, which should not exceed two seconds. The baseline algorithm must traverse all the possible combinations among all the candidate activity sets. When the demanded itinerary contains more activities, the system will set k to a larger number, leading to a combinatorial explosion. If each activity has k candidate elements, and the itinerary contains n activities, the combination space will be k^n. When k is fixed, the cost increases exponentially with n. When n is fixed, the value is a power function of k. Neither of these cases is tolerable for real-time applications.

Table 2: Time cost (ms) and Round (in brackets) to obtain optimal recommendations with different k-values

Run	20			40			60		
ant	k=3	k=6	k=9	k=3	k=6	k=9	k=3	k=6	k=9
20	47(2)	63(6)	62(5)	63(2)	78(12)	110(2)	78(14)	109(5)	172(1)
40	62(4)	63(3)	64(14)	94(1)	109(7)	188(9)	234(3)	140(1)	203(10)
60	78(1)	128(1)	172(8)	141(10)	188(11)	266(1)	187(5)	313(1)	328(6)

Table 2 lists the time cost and the serial number of each round (in brackets) in which the optimal recommendation is reached when the modified ACA's ant number and total rounds are set to 20, 40 and 60. The results demonstrate that all the optimal recommendations are generated within 20 rounds. Furthermore, the time cost is no greater than 500 ms, which is acceptable for an online service. To compare the baseline algorithm with the modified ACA, we conducted the following experiments:

Experiment 1: the activity number varies from 2 to 10 and the candidate number of POIs of each activity is 6. **Fig. 8** presents the time cost in a logarithm chart. The ant number is set to 60, the round number is set to 20 (according to the above analysis, 20 is enough for the algorithm to obtain an optimal result). As the red line shows, the time cost of the baseline algorithm increases exponentially with the activity number, which follows our mathematical analysis. The time cost of the modified ACA increases linearly, finally stabilizing at a constant value. This is because the ants come to a consensus in a relatively short time, despite the way the optional combination increases with the activity number.

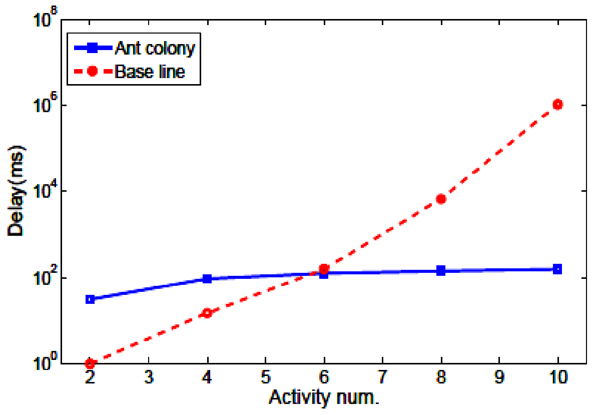


Fig. 8. Activity number vs. k-value

Experiment 2: the activity number is set to 6; the settings of the modified ACA are the same as in experiment 1. The candidate number k of each activity varies from 2 to 12, and the time cost is shown in the logarithm chart. As the red line in **Fig. 9** shows, the time cost of the baseline algorithm increases with k as the curve of a power function. When k equals 12, the time cost can be as high as 7000 ms, which cannot meet the demands of a real-time recommendation system. In contrast, the time cost of the modified ACA is much smaller, with its highest value reaching 200 ms.

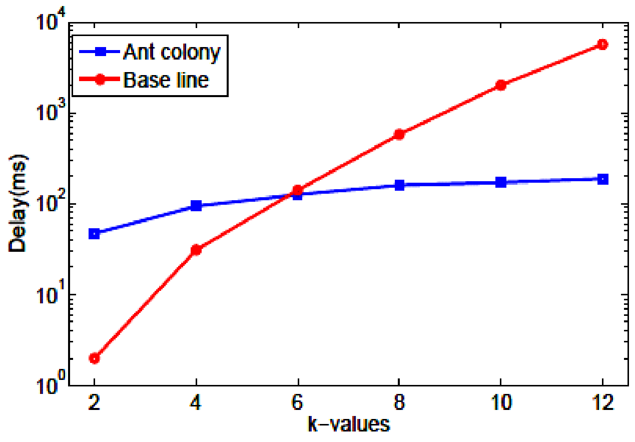


Fig. 9. Base line vs. ant colony algorithms

According to the above analysis, the recommendation time cost of the baseline algorithm increases exponentially with k and n, which cannot meet the requirements of a real-time application. On the contrary, the modified ACA is affected only slightly by these parameters and fits the demands of real-time itinerary recommendation well.

6. Related Works

6.1 Itinerary Recommendation

A number of itinerary generation and recommendation systems have previously been introduced. Hyoseok [12] provides smart itinerary recommendation services that use a simplified query composed of a start point, an end point and the duration to obtain a complete set of itineraries that are automatically generated based on real user-generated GPS trajectories. They simultaneously implement a personalized recommendation that provides the user with locations matching her travel preferences [7]. However, they have not provided an intention-oriented itinerary recommendation service.

In TripTip [17], the user selects a location to request recommendations of similar types of places using popular tags. Huang and Bian [18] have built a travel recommendation system that integrates heterogeneous online travel information based on tourism ontology and recommends tourist attractions using travel preferences estimated by a Bayesian network. Kumar et al. [19] have presented a GIS-based Advanced Traveler Information System (ATIS) for Hyderabad City in India that includes a site-tour module based on the shortest distance. Compared to these works, we bridge online social networks and knowledge from GPS trajectories that collects history data from millions of users to develop the shortest and most interesting itineraries.

6.2 Other related projects

This section lists similar studies related to our study and describes their differences from our work:

The GeoLife project of Microsoft Research Asia collected 165 users' trajectory data from 2008 to 2010. They have developed many interesting applications, such as a traffic navigation system [6], and they have studied the relationships between frequently visited locations and users. Their discoveries can answer questions such as “What are the most popular tourist attractions in Beijing city?”, “Where will a tourist who has gone to the Imperial Palace be?”, “Which is the most frequently used route to a certain place?” and so on [8].

The research of Yerach [20] attempts to discover trajectory patterns and analyze people’s lifestyles. This work associates each social network user with a specific geographic location to build a spatial social network graph. Its main contribution is that it proposes using a series of operators to query social networks and spatial networks together and has implemented them based on relational and graph databases. The work has inspired us to combine spatial and social networks to explore new applications and technologies.

The classic scenario of the spatial keyword query is to provide a physical location and a set of keywords to locate the single object that best matches the input keywords with minimal distance [21][22]. The research of [1] expands on spatial keyword search and proposed to realize the search with a set of objects that together match the input keywords with a minimum of spatial distance between the objects. The application scenarios of this type of query are restricted because they do not make full use of all the types of background knowledge in other information sources. Our work uses collective intelligence to identify more useful locations and provide reasonable travel paths for the recommended itineraries.

The work of [23] supports trajectory retrieval using k query points. This work requires the user to input k accurate location points and determines the trajectories that pass through or are nearest to all the input points. However, it does not support category-based queries and cannot guarantee that the resulting trajectory will be the shortest or have the minimum travel time.

7. Conclusion and Future Works

By combining data from social networks with physical trajectories and making use of hierarchical categories of geographical locations, this paper accomplishes recommendation in itinerary planning. Based on semantic traffic information contained in historical trajectories, our recommendation algorithm provides the best path along multiple points, which satisfies the users demands better than the shortest path-searching algorithm. Meanwhile, the recommendations are presented with reviews of the recommended locations on the social network, effectively helping the user understand the system’s final recommendations. Therefore, the recommendation results are meaningful to the users as they plan their itineraries.

However, this paper has only introduced the basic theories and built an original demo system. The system still requires much improvement and further optimization. The future work includes the following elements.

1. Improve the user interface to receive user feedback.
2. Generate a more precise analysis of the traffic information map, except for the shortest time; many more statistical factors should be considered and calculated.
3. Make use of the parallelism of ACA to reduce the execution time.
4. Collect user preferences to provide more personalized recommendations.

References

[1] Xin Cao, Gao Cong, Christian S. Jensen and Beng Chin Ooi, "Collective spatial keyword querying," in *Proc. of ACM Int. Conf. on Management of Data,* pp.373-384, Jun. 2011. Article (CrossRef Link)

[2] Jing Yuan, Yu Zheng, Xing Xie and Guangzhong Sun, "Driving with knowledge from the physical world," in *Proc. of 17th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining*, pp.316-324, Aug.2011.Article (CrossRef Link)

[3] Jing Yuan, Yu Zheng, Chengyang Zhang, Wenlei Xie, Xing Xie, Guangzhong Sun and Yan Huang, "T-drive: driving directions based on taxi trajectories," in *Proc. of 18th SIGSPATIAL Int. Conf. on Advances in Geographic Information Systems*, pp.99-108, Nov.2010. Article (CrossRef Link)

[4] Zhixian Yan, Dipanjan Chakraborty, Christine Parent, Stefano Spaccapietra and Karl Aberer, "Semitri: a framework for semantic annotation of heterogeneous trajectories," in *Proc. of 14th Int. Conf. on Extending Database Technology*, pp.259–270, Mar.2011. Article (CrossRef Link)

[5] Mark de Berg, Otfried Cheong, Marc van Kreveld and Mark Overmars, *Computational Geometry: Algorithms and Applications,* Third Edition, Springer-Verlag., Heidelberg, 2008.

[6] Yu Zheng, Yukun Chen, Xing Xie and Wei-Ying Ma, "Geolife2.0: A location-based social networking service," in *Proc. of 10 Int. Conf. on Mobile Data Management: Systems, Services and Middleware*, pp.357-358, 2009. Article (CrossRef Link)

[7] Yu Zheng and Xing Xie, "Learning travel recommendations from user-generated GPS traces," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, no. 1, pp.2-12, Jan.2011. Article (CrossRef Link)

[8] Yu Zheng, Lizhu Zhang, Xing Xie and Wei-Ying Ma, "Mining interesting locations and travel sequences from gps trajectories," in *Proc. of 18th Int. Conf. on World Wide Web*, pp.791-800, Apr. 2009. Article (CrossRef Link)

[9] Luis O. Alvares, Gabriel Oliveira, Carlos A. Heuser, and Vania Bogorny, "A framework for trajectory data preprocessing for data mining," in *Proc. of 21st Int. Conf. on Software Engineering*, pp.698-702, Jul.2009.

[10] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jos A. Fernandes de Macedo, Fabio Porto and Christelle Vangenot, "A conceptual view on trajectories," *Data Knowl. Eng.*, vol. 65, no.1, pp.126-146, 2008. Article (CrossRef Link)

[11] Luis O. Alvares, Vania Bogorny, Bart Kuijpers, Jos A. Fernandes de Macedo, Bart Moelans and Alejandro A. Vaisman, "A model for enriching trajectories with semantic geographical information," in *Proc. of 15th ACM Int. Symposium on Geographic Information Systems*, pp.22, November 7-9 2007. Article (CrossRef Link)

[12] Hyoseok Yoonl, Yu Zheng, Xing Xie and Woontack Woo, "Smart itinerary recommendation based on user-generated gps trajectories," in *Proc. of 7th Int. Conf. on Ubiquitous Intelligence and Computing*, pp.19–34, October 26-29 2010. Article (CrossRef Link)

[13] Marco Dorigo, Vittorio Maniezzo and Alberto Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems*, vol. 26, no.1, pp.29-41, 1996. Article (CrossRef Link)

[14] Marco Dorigo and Luca Maria Gambardella, "Ant colony system: a cooperative learning approach to the traveling salesman problem," *IEEE Trans. Evolutionary Computation*, vol. 1, no.1, pp.53-66, 1997.Article (CrossRef Link)

[15] Haibin Duan, *Ant colony algorithms: Theory and Applications,* Science Press, Beijing, 2005

[16] Manish Patil, Sharma V. Thankachan, Rahul Shah, Wing-Kai Hon, Jeffrey Scott Vitter and Sabrina Chandrasekaran, "Inverted indexes for phrases and strings," in *Proc. of 34th Int. ACM SIGIR Conf. on Research and Development in Information Retrieval,* pp.555-564, July 25-29 2011. Article (CrossRef Link)

[17] Jinyoung Kim, Hyungjin Kim and Jung hee Ryu, "Triptip: a trip planning service with tag-based recommendation," in *Proc. of CHI Extended Abstracts on Human Factors in Computing Systems*, pp.3467-3472, 2009. Article (CrossRef Link)

[18] Yuxia Huang and Ling Bian, "A bayesian network and analytic hierarchy process based personalized recommendations for tourist attractions over the internet," *Expert Syst. Appl*., vol. 36, no.1, pp.933–943, 2009. Article (CrossRef Link)
[19] Praveen Kumar, Varun Singh and Dhanunjaya Reddy, "Advanced traveler information system for hyderabad city," *IEEE Transactions on Intelligent Transportation Systems*, vol.6, no. 1, pp.26–37, 2005. Article (CrossRef Link)
[20] Yerach Doytsher, Ben Galon and Yaron Kanza, "Querying geo-social data by bridging spatial networks and social networks," in *Proc. of Int. Workshop on Location Based Social Networks*, pp.39-46, Nov. 2010. Article (CrossRef Link)
[21] Xin Cao, Gao Cong and Christian S. Jensen, "Retrieving top-k prestige-based relevant spatial web objects," *Proc. VLDB Endow*, vol.3, no.1, pp.373-384, Sep. 2010. Article (CrossRef Link)
[22] Ian De Felipe, Vagelis Hristidis and Naphtali Rishe, "Keyword search on spatial databases," in *Proc. of 24th Int. Conf. on Data Engineering*, pp.656-665, Apr.2008. Article (CrossRef Link)
[23] Lu An Tang, Yu Zheng, Xing Xie, Jing Yuan, Xiao Yu and Jiawei Han, "Retrieving k-nearest neighboring trajectories by a set of point locations," in *Proc. of 12th Int. Symposium on Advances in Spatial and Temporal Databases*, pp.223-241, Aug.2011. Article (CrossRef Link)

Xiangxu Meng, Ph.D. candidate with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, China. He has obtained the M.D. and B.S. in 2007 and 2004, respectively, all in computer science. His research interests include Location privacy & security; Location based service system; Spatial-temporal data query and mining.

Xinye Lin, Ph.D. candidate with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, China. He has obtained the M.D. and B.S. in 2011 and 2008, respectively, all in computer science. His research interests include data mining and recommendation system.

Xiaodong Wang, Professor with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, China. He has obtained the Ph.D., M.D. and B.S. in 2002, 1998 and 1996, respectively, all in computer science. His research interests include mobile computing, social network and wireless networks.

Xingming Zhou, Professor with National Laboratory for Parallel and Distributed Processing, National University of Defense Technology, China. His research interests include high performance computing, database and wireless communication.