CrowdPlanner: A Crowd-Based Route Recommendation System

Han Su [#1], Kai Zheng [#2], Jiamin Huang [*1], Hoyoung Jeung [†1], Lei Chen [‡1], Xiaofang Zhou [#3]

[#]*The University of Queensland, Australia* [*]*Nanjing University, China* [†]*SAP Australia* [‡]*HKUST, Hong Kong*

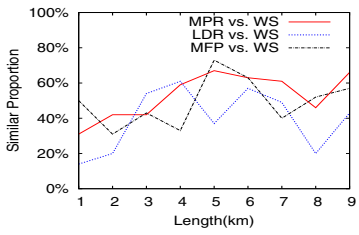
[#1,2,3]{h.su1,uqkzheng,uqxzhou}@uq.edu.au [*1]hjm10@software.nju.edu.cn

[†1]hoyoung.jeung@sap.com [‡1]leichen@cse.ust.hk

***Abstract*— As travel is taking more significant part in our life, route recommendation service becomes a big business and attracts many major players in IT industry. Given a pair of user-specified origin and destination, a route recommendation service aims to provide users with the routes of best travelling experience according to criteria, such as travelling distance, travelling time, traffic condition, etc. However, previous research shows that even the routes recommended by the big-thumb service providers can deviate significantly from the routes travelled by experienced drivers. It means travellers' preferences on route selection are influenced by many latent and dynamic factors that are hard to model exactly with pre-defined formulas. In this work we approach this challenging problem with a very different perspective– leveraging crowds' knowledge to improve the recommendation quality. In this light, CrowdPlanner – a novel crowd-based route recommendation system has been developed, which requests human workers to evaluate candidate routes recommended by different sources and methods, and determine the best route based on their feedbacks. In this paper, we particularly focus on two important issues that affect system performance significantly: (1) how to efficiently generate tasks which are simple to answer but possess sufficient information to derive user-preferred routes; and (2) how to quickly identify a set of appropriate domain experts to answer the questions timely and accurately.Specifically, the task generation component in our system generates a series of informative and concise questions with optimized ordering for a given candidate route set so that workers feel comfortable and easy to answer. In addition, the worker selection component utilizes a set of selection criteria and an efficient algorithm to find the most eligible workers to answer the questions with high accuracy. A prototype system has been deployed to many voluntary mobile clients and extensive tests on real-scenario queries have shown the superiority of CrowdPlanner in comparison with the results given by map services and popular route mining algorithms.**

I. Introduction

Travelling plays a vital role in our daily life. Thanks to the rapid development of GPS technologies and a number of navigation service providers (e.g., Google Map, Bing Map, TomTom), we can now travel to unfamiliar places with much less effort, by simply following the recommended routes. While the detailed mechanisms that are adopted to recommend routes are different, travelling distance and time are the most important criteria and factors in those recommendation algorithms, which result in the shortest route and/or fastest route. With increasing numbers of users who rely on these map services to travel, a natural question arises: *are these routes always good enough to be the best choice when people travel?* Ceikute et al [3] are the first to assess the routing service quality by comparing the popular routes, the ones most drivers prefer, and the ones recommended by a big thumb map service provider. The study concludes that there are substantial differences between popular routes and recommended routes, in which experienced/frequent drivers' preferences do not always correspond to the routes recommended by the navigation service. The primary reason for the route differences is that drivers' preferences are influenced by lots of factors in addition to distance and time, such as the number of traffic lights, speed limitation, road condition, weather, amongst many others.



(a) Similarity between routes obtained from web services and popular route mining algorithms

(b) Recommended routes from different sources

Fig. 1. Efficiency of landmark selection algorithms

In order to take into account the diversity of the preference factors simultaneously, some previous studies propose to use popular routes mined from historical trajectories as recommended routes. This approach, however, has significant drawbacks. First, it is not always possible to have a sufficient amount of historical trajectories to derive reliable route recommendation. Second, there exists a number of popular route mining algorithms. The definitions of popularity in those algorithms slightly differ from each other, which can suggest different routes for users. As a result, it is still difficult for users to select one particular route as a best choice. For example, Fig. 1(a) shows different popular routes mined using different algorithms. In this experiment, we first randomly select 5000 source-destination pairs as the testing queries. For each of them, we test the similarity between the route recommended by a big thumb Web map service (WS) and the route obtained from three popular route mining algorithms, namely Most Popular Route (MPR) [4], Local Driver Route (LDR) [3] and Most Frequent Path (MFP) [14], all of which perform reasonably well according to their reported results. The results of average similarity are shown in Fig. 1(a). One

can see that, the similarities are at best around 60%, which means that different sources recommend quite different routes. Fig. 1(b) demonstrates the recommended routes from different sources on map, where two routes recommended by WS are different from those by MPR and MFP respectively.

Going beyond the limitation of the route recommendation based on popular routes, we take the emerging concept of crowd sourcing that explicitly leverages human knowledge to resolve complex problems. Specifically, we propose a novel crowd-based route recommendation system, CrowdPlanner, which can effectively blend domain-expert knowledge for route recommendation. Instead of proposing new or optimizing existing routing algorithms, our work takes an entirely different approach by consolidating candidate routes from different sources (e.g., map service providers, popular routes) and requesting experienced drivers to select amongst them. Our system returns the most promising one according to the selection of drivers.

Taking domain-expert's knowledge to evaluate the route quality is a very challenging task. The first yard stone to be placed is how to automatically generate a user-friendly task so that domain experts can do the job more comfortably and accurately. As the performance of system largely relies on the quality of an answer given by each worker, how to choose a set of suitable worker for a given task is another problem we need to solve.

CrowdPlanner tackles the above challenges by two carefully designed core components. More specifically, *task generation* component utilizes a set of significant and discriminative landmarks to generate a binary question set by analysing the given candidate route set. Then those questions are presented to the workers with optimized orders based on the informativeness of each question (whether it is more likely to lead to the final answer) and the response of the worker. In *worker selection* component, we identify a few key attributes of workers that mostly affect their performance on a given task and propose an efficient search algorithm to find the most eligible workers.

Our key contributions in this work can be summarized as follows.

- We identify the intrinsic difficulties in the route recommendation task by solely relying on computational methodologies, and propose an entirely new approach that actively involves human to improve the recommendation quality.
- We design and develop a novel crowd-based route recommendation system, CrowdPlanner, which is able to generate concise yet informative task intelligently and assign it to the selected worker who can accomplish the task with high accuracy and low latency.
- We deploy the system and conduct extensive experiments with a large number of workers, users and queries in real scenarios. The results demonstrate that CrowdPlanner can recommend the most satisfactory routes efficiently in most cases.

The rest of this paper is organized as follows. Section II introduces the preliminary concepts and overviews the CrowdPlanner system. The two core components, task generation and worker selection, are discussed in Section III and Section IV respectively. The experimental observations are presented in Section V, followed by a brief review of related work in Section VI. Section VII concludes the paper and outlines some future work.

II. Problem Statement

In this section, we present some preliminary concepts and give an overview of the CrowdPlanner system. Table I summarized the major notations used in the rest of the paper.

TABLE I
SUMMARIZE OF NOTATIONS

Notation	Definition
R	a recommended route
$\mathbb{R}$	candidate set of recommended routes
p	a place in the space
l	a landmark in the space
$l.s$	significance of landmark l
$\mathbb{L}$	a landmark set
$\mathbb{L}_{\mathbb{R}}$	the questioned landmark set of route set $\mathbb{R}$
$d(l_i, l_j)$	Euclidean distance between landmarks l_i and l_j
w	a worker of the system
$\mathbb{W}$	a worker set
$\mathbb{W}_{\mathbb{R}}$	the selected workers of routes set $\mathbb{R}$

A. Preliminary Concepts

Definition 1: Route: A route R is a continuous travelling path. We use a sequence $[p_1, p_2, \cdots, p_n]$, which consists of a source, a destination, and a sequence of consecutive road intersections in-between, to represent a route.

Definition 2: Landmark: A landmark is a geographical object in the space, which is stable and independent of the recommended routes. A landmark can be either a point (i.e., Point Of Interest), a line (i.e., street and high way) or a region (i.e., block and suburb) in the space.

Definition 3: Landmark-based Route: A landmark-based route $\bar{R}$ is a route represented as a finite sequence of landmarks, i.e., $\bar{R} = [l_1, l_2, ..., l_n]$.
In this paper, we will also use $\bar{R}$ as the set $\{l_1, l_2, \cdots, l_n\}$ without ambiguity.

In order to obtain the landmark-based route from a raw route, we employ our previous research results on anchor-based trajectory calibration [24] to rewrite the continuous recommend routes into landmark-based routes, by treating landmarks as anchor points.

Definition 4: Discriminative landmarks: A landmark set $\mathbb{L}$ is called *discriminative* to a set of landmark-based routes $\bar{\mathbb{R}}$ if for any two routes $\bar{R}_1$ and $\bar{R}_2$ of $\bar{\mathbb{R}}$, the joint sets $\bar{R}_1 \cap \mathbb{L}$ and $\bar{R}_2 \cap \mathbb{L}$ are different.

For example, $\mathbb{L}_1 = \{l_3, l_4\}$ is discriminative to $R_1 = \{l_1, l_2, l_3\}$ and $R_2 = \{l_1, l_2, l_4\}$, since the joint sets $R_1 \cap \mathbb{L}_1 = \{l_3\}$ and $R_2 \cap \mathbb{L}_1 = \{l_4\}$ are different, but $\mathbb{L}_2 = \{l_1, l_2\}$ is not discriminative to R_1 and R_2.

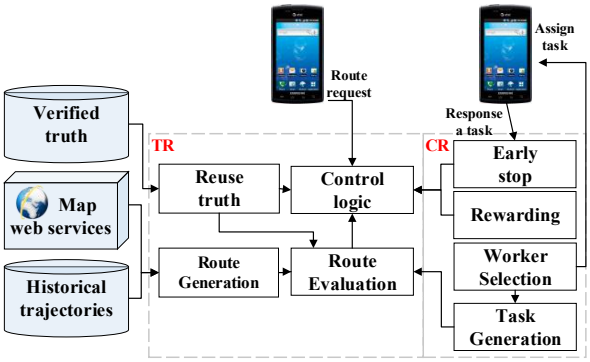


Fig. 2. System Overview

B. Overview of CrowdPlanner

CrowdPlanner is a two-layer system (mobile client layer and server layer) which receives user's request from mobile client specifying the source and destination, processes the request on the server and finally returns the verified best routes to the user. Fig. 2 shows the overview of the proposed CrowdPlanner system, which comprises two modules: traditional route recommendation (TR) and crowd-based route recommendation (CR). The workflow of CrowdPlanner is as follows: the TR module firstly processes user's request by trying to evaluating the quality of candidate routes obtained from external sources such as map services and historical trajectory mining; the CR module will generate a crowdsourcing task when the TR module can not judge the quality of candidate routes, and return the best route based on the feedbacks of human workers of the system.

1) Traditional Route Recommendation Module: This module processes the user's request by generating a set of candidate routes from external sources (route generation component) and evaluating the quality of those routes automatically without involving human effort (route evaluation component).

Control logic component: This component receives the user's request and controls the workflow of the entire system. It also coordinates the interactions between the TR module and CR module. Once a user's request is received by the control logic component, it will invoke *reuse truth component* to match the request to the verified routes (truth) between two places at his departure time. If the new coming request is a hit of the truth, the system will return result immediately. Otherwise the component will invoke the *route evaluation component* to automatically generate some candidate routes and evaluate the qualities of these candidate routes using the verified truth.

Route evaluation component: This component evaluates the routes using computer power and it provides an efficient way to reduce the cost of CrowdPlanner, since it can largely reduce the amount of tasks generated. The component will firstly build up a candidate route set by invoking *route generation component*. If some of these routes agree with each other to a high degree, one of them will be selected as the best recommended route and added into a truth database with the corresponding time tag. If a best recommended route can not be determined, the system will assign each candidate route a confidence score, which is generated by the verified truths and illustrates the possibility of the route to be the best recommended route. A route with the highest confidence score that is greater than a threshold η will be regarded to be the best recommended and returned to the user; otherwise the logic control will hand over the request to the CR module.

Route generation component: This component generates two types of candidate routes, the one provide by web services such as Google Map and the one generated from historical trajectories by using popular route mining algorithms, i.e., MPR, LDR and MFP.

2) Crowd Route Recommendation Module: Crowd route recommendation module will take over the route recommendation request when the traditional route recommendation module cannot provide the best route with confidence high enough. The module will generate a Crowdsourcing task consisting of a series simple but informative binary questions (task generation component), and assign the task to a set of selected worker who are most suitable to answer these questions (worker selection component).

Task generation component: As the core of CrowdPlanner, this component generates a task by proposing a series of questions for workers to answer. It is beneficial to have these questions as simple and compact as possible, since both the accuracy and economic effectiveness of the system can be improved. The design of this component will address two important issues: *what to ask in questions* and *how to ask the questions*. We will discuss the detailed mechanism of this part in Section III.

Worker selection component: This is another core component of CrowdPlanner. In order to maximize the effectiveness of the system, we need to select a set of eligible workers who are most suitable to answer the questions in a given task, by estimating the worker's familiarity with the area of request. Technical details of this component will be presented in Section IV.

Early stop component: In most cases, we do not to need to wait for all the answers of the assigned workers. When partial feedbacks have been collected, this component will evaluate the confidence of the answer and return the result to the user as early as possible when the confidence is high enough.

Rewarding component: This component rewards the workers according to their workload and the quality of their answers. The reward points can be used later when they request a route recommendation in CrowdPlanner.

In the following two sections, we will present the design and technical details of the two core components of CrowdPlanner: task generation and worker selection.

III. Task Generation

Almost everyone has the experience of being unable to explicitly describe a route even you know the directions clearly, which implies that this kind of job is hard for humans in its nature. Therefore, we cannot simply publish a task to workers and expect them to describe the best route in a

turn-by-turn manner. In an alternative and more friendly way, we may provide several pictures, which demonstrate several candidate routes on a map, as a multiple-choice question for workers to choose. Take the route recommendation request in Fig. 3 as an example, we publish a multiple-choice question to workers by showing four routes on a map, and asking them to pick the route they most prefer. Even when all the routes have been visualized on a map, it is still effort-demanding to tell the subtle differences between candidate routes, and especially so if doing it on a small-screen device, say a smartphone. To make the question easier to answer, we take into consideration that it is human nature to utilize significant locations, i.e., landmarks, to help describe a route in high-level, whereas a computer sees a route as a sequence of continuous roads indifferently. Thus, we choose to proactively present the differences in candidate routes to the workers using landmarks , instead of waiting for them to find out. Besides, how the questions are presented can also affect the complexity of a task. For example, a multiple-choice question with all candidate routes presented at the same time would be more difficult to answer than an equivalent combination of binary questions such as "do you prefer the route passing landmark A at 2:00pm?". Actually, [23] has pointed out that several binary choice questions are easier and more accurate than a multiple-choice question. Based on the above analysis, we will generate a task as a sequence of binary questions, each relating to a landmark that can discriminate some of the candidate routes from the others.

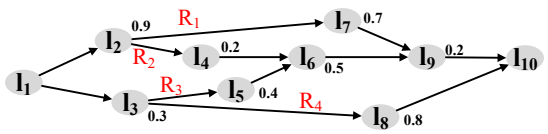


Fig. 3. An example of landmark-based recommended routes between l_1 and l_{10}

Next we will present in detail our task generation process, which can be divided into three phases: inferring landmark significance, landmark selection and question ordering. In specific, the first phase infers the significance of each landmark which indicates people's familiarity. The second phase tries to use a set of most significant landmarks to summarize the difference among the candidate routes. The third phase generates the final task by ordering the questions in a smart way so that the expected number of issued questions is as small as possible.

A. Inferring Landmark Significance

It is common sense that landmarks have different significances. For instance, the White House is world famous, but Pennsylvania Ave, where the White House is located, is only known by locals of Washington DC. People tend to be more familiar with the landmarks that are frequently referred to by different sources, e.g., public praise, news, bus stop, yellow pages. In this work, we utilize the online check-in records from a popular location-based social network (LBSN) and trajectories of cars in the target city to infer the significance of landmarks, for these two datasets are large enough to cover most areas of a city. By regarding the travellers as authorities, landmarks as hubs, and check-ins/visits as hyperlinks, we can leverage a HITS-like algorithm such as [27] to infer the significance of a landmark. Readers who are interested in the technical details can refer to [27].

B. Landmark Selection

Although any landmark can be used to generate a question, not all of them are suitable for the purpose of generating easy questions for a certain candidate route set $\bar{\mathbb{R}}$ (notably throughout this section we use the landmark-based routes $\bar{\mathbb{R}}$, which is generated by rewriting all the routes in $\mathbb{R}$ as described in Section II). First, the selected landmark set $\mathbb{L}$ should be discriminative to the candidate routes $\bar{\mathbb{R}}$, which ensures that the difference between any two routes can be presented. Second, the landmarks of $\mathbb{L}$ should have high significance, so that more people can answer the question accurately. Third, in order to reduce the work load of workers, the selected landmark set $\mathbb{L}$ should be as small as possible. Therefore the problem of landmark selection is to find a small set of highly significant landmarks which are discriminative to all the candidate routes. It can be formally represented as an optimization problem as below:

Given n landmark-based candidate routes $\bar{\mathbb{R}}$, and the significance of each landmark,

Select a landmark set $\mathbb{L}$ with the size of k ($\lceil \log_2(n) \rceil \leq k \leq n$) which is discriminative to $\bar{\mathbb{R}}$,

Maximize $|\mathbb{L}|^{-1} \cdot \sum_{l \in \mathbb{L}} l.s$

Here the target function aims to maximize the total significance of selected landmarks ($\sum_{l \in \mathbb{L}} l.s$), normalized by the size of $\mathbb{L}$ ($|\mathbb{L}|$).

It is a non-trivial task to trade-off between maximizing accumulate significance of the selected landmark set $\mathbb{L}$ and minimizing the size of $\mathbb{L}$, while guarantees the restriction that $\mathbb{L}$ must be discriminative to $\bar{\mathbb{R}}$. A straightforward method is to enumerate all combinations of the landmarks from $\bar{\mathbb{R}}$, and find a discriminative landmark set with the maximized target value. However, the time cost of this algorithm grows exponentially with the size of landmark set, making this method impractical. To speed up this process, we propose a greedy algorithm, called GreedySelect. The main idea is to enumerate all the possible landmark combinations in a smart order so that it enables pruning early in the enumeration process. Let S denote the current testing landmark set and $\mathbb{L}_{best}$ denote the best landmark set which is discriminative and has the highest target value. The landmark selection process can be divided into three steps:

Preparation step: During preparation, we filter out some non-beneficial landmarks, i.e., the ones which cannot discriminate any routes of $\bar{\mathbb{R}}$. A straightforward way is to filter out all landmarks which are shared by all the candidate routes, and those which do not appear on any candidate route. Thus the beneficial landmarks set of $\bar{\mathbb{R}}$ can be generated as following: $\mathbb{L} = \bigcup_{\bar{R} \in \bar{\mathbb{R}}} \bar{R} - \bigcap_{\bar{R} \in \bar{\mathbb{R}}} \bar{R}$. We sort $\mathbb{L}$ in descending order of their significances in order to enable our pruning technique later.

We still use $\mathbb{L}$ to refer to the sorted beneficial landmarks.

Expansion step: This step generates the test landmark set S. We recursively generate and test the test landmark set S as shown in Algorithm 1. The test step will be explained below. In each recursion step, for each discriminative S we find all the landmarks not in S, pick non-added biggest landmark of them, and add the landmark to S. For example as shown in Fig. 6(a), the algorithm starts by adding l_2 to S. Since $S = \{l_2\}$ is not discriminative, the S will be expanded by adding l_8 to S which is shown in Fig. 6(b), so as adding l_7 to $S = \{l_2, l_8\}$, shown in Fig. 6(c). Once S is discriminative, we stop adding landmark to it, no longer visit supersets of S, and roll back to upper layer recursion. E.g. in Fig. 6(c), $S = \{l_2, l_8, l_7\}$ will not be expanded, and the system will roll back l_7 and expand $S = \{l_2, l_8\}$ with l_6. Due to the same S may be generated in different order, to eliminate duplication, we only consider those landmarks with a lower significance than any element in S. The process stops when all the possible combinations have been visited.

Test step: Each time a new S is generated, we conduct a test to see whether S is discriminative. If S is not discriminative, return false. Otherwise, we use $GetMaxSet(S)$ to get maximum superset of S, i.e., the set which contains all the points in S, and maximizes the target function. We compare the superset got with the current best set $\mathbb{L}_{best}$. If the target value of the superset is bigger than that of $\mathbb{L}_{best}$, then the superset is current best landmark combination, and we assign the superset to $\mathbb{L}_{best}$. Note that since the landmarks in $\mathbb{L}$ are sorted, the time complexity of $GetMaxSet(S)$ is $O(k)$, where k is no larger than n (the number of candidate routes).

Algorithm 1: Expand and Test

```
if |S| = n then
    stop or S ← landmark with the next biggest significance;
else
    SetOfS ← all the landmarks has a lower sinificance than
    any landmark of S;
    Sort SetOfS in descending order of the significances of
    landmarks;
    for each l ∈ SetOfS do
        isDiscriminative = test(S ∪ {l})
        if isDiscriminative is false then
            expand(S ∪ {l});
```

However, the above process can be very time consuming when the sizes of $\mathbb{L}$ and n are large, since there will be a large amount of landmark sets to be tested. In order to improve the efficiency, we need to filter out more non-beneficial landmarks in the preparation step, test less landmark combinations in test step and generate less landmark combination in expansion step. Next we will present the optimizations for each step.

1) **Optimization at preparation step:** Each landmark l of $\mathbb{L}$ can divide the routes set $\mathbb{R}$ into two parts: the set of routes that pass l, and the set of routes that dose not. The divided two parts are defined as the discriminative information of l. For instance, the discriminative information of all the landmarks of Fig. 3 are shown in Fig. 4. We can see that l_2 has the same discriminative information of l_3, so as l_8 and l_9. For each discriminative combination S containing l_3, there must exist an discriminative combination $(S - \{l_3\}) \cup \{l_2\}$ according to the following theorem:

Landmark	Discriminative information	Significance	Keep or drop
l_2	{R1, R2}, {R3, R4}	0.9	Keep
l_3	{R1, R2}, {R3, R4}	0.3	Drop
l_4	{R2}, { R1,R3, R4}	0.2	Keep
l_5	{R3}, { R1,R2, R4}	0.4	Keep
l_6	{R1, R4}, {R2, R3}	0.5	Keep
l_7	{R1}, { R2,R3, R4}	0.7	Keep
l_8	{R4}, { R1,R2, R3}	0.8	Keep
l_9	{R4}, { R1,R2, R3}	0.2	Drop

Fig. 4. Discriminative information of landmarks

Theorem 1: Given two landmarks l_i and l_j sharing the same discriminative information and a landmark combination S, the two combinations $S \cup \{l_i\}$ and $S \cup \{l_j\}$ are either both discriminative or both non-discriminative to $\bar{\mathbb{R}}$.

Proof: Consider any two routes $\bar{R}$, $\bar{R}'$ from $\bar{\mathbb{R}}$. If S is discriminative to $\bar{R}$ and $\bar{R}'$, i.e. $\bar{R} \cap S \neq \bar{R}' \cap S$, clearly $\bar{R} \cap (S \cup \{l_i\}) \neq \bar{R}' \cap (S \cup \{l_i\})$, so as $S \cup \{l_j\}$, that is, $S \cup \{l_i\}$ $(S \cup \{l_j\})$ is discriminative to $\bar{R}$, $\bar{R}'$. Otherwise, $\bar{R} \cap S = \bar{R}' \cap S$. There are two cases:
(1) Both l_i and l_j are discriminative to $\bar{R}$ and $\bar{R}'$. W.l.o.g., assume $l_i, l_j \in \bar{R}$ but $l_i, l_j \notin \bar{R}'$. Then $l_i \in \bar{R} \cap (S \cup \{l_i\})$ but $l_i \notin \bar{R}' \cap (S \cup \{l_i\})$, so as l_j. Thus $S \cup \{l_i\}$ and $S \cup \{l_j\}$ are discriminative to $\bar{R}$ and $\bar{R}'$.
(2) Both l_i and l_j are not discriminative to $\bar{R}$ and $\bar{R}'$. W.l.o.g., assume $l_i, l_j \in \bar{R}, \bar{R}'$. Then $\bar{R} \cap (S \cup \{l_i\}) = (\bar{R} \cap S) \cup \{l_i\} = (\bar{R}' \cap S) \cup \{l_i\} = \bar{R}' \cap (S \cup \{l_i\})$, so as l_j. Thus $S \cup \{l_i\}$ and $S \cup \{l_j\}$ are not discriminative to $\bar{R}$ and $\bar{R}'$. ■

According to Theorem 1, and since the target value of $(S - \{l_3\}) \cup \{l_2\}$ is no less than S, all the combinations containing l_3 can be pruned. So for landmarks which share the same discriminative information, we keep the landmark with the highest significance and drop others. Thus we drop l_3 and l_9, while keep l_2 and l_8.

2) **Optimization at expansion step:** In each recursion step within the expansion step, given the current selected landmark set S, there are a set of routes $ND(S)$ where for any $\bar{R} \in ND(S)$, there exists some other route $\bar{R}' \in ND(S)$, such that S is non-discriminative to $\bar{R}$ and $\bar{R}'$, i.e., $\bar{R} \cap S = \bar{R}' \cap S$. We call $ND(S)$ the non-discriminative route set of S. Depending on $ND(S)$, there are two special set of landmarks in the set of landmarks to explore (*SetOfS* in Algorithm 1): *contributive set* and *conflict set*. A contributive set is the set of landmarks where each landmark can discriminate some pair of routes in $ND(S)$, A *conflict set* is the set of landmarks where adding any of the landmarks to S will form a superset of some discriminative set that has already been pruned.

Next we will introduce how to generate the contributive set of S. For each non-discriminative set S, $ND(S)$ is not empty. A landmark l is a contributive landmark for S if there exists two routes $\bar{R}_i$ and $\bar{R}_j$ from $ND(S)$ such that l is only on one route of $\bar{R}_i$ and $\bar{R}_j$. So the contributive set $\mathbb{L}_{contri}$ can be generate by the following equation:

$$\mathbb{L}_{contri} = \bigcup_{\bar{R}_i,\bar{R}_j \in ND(S), \bar{R}_i \cap S = \bar{R}_j \cap S} (\bar{R}_i - \bar{R}_j) \cup (\bar{R}_j - \bar{R}_i)$$

As the discriminative landmarks of two routes are fixed, we can pre-compute all the discriminative landmarks between any two routes in $\bar{\mathbb{R}}$. Fig. 5 demonstrates the discriminative landmarks of routes in Fig. 3.

Routes combinations	Discriminative landmarks
R_1, R_2	l_7, l_6, l_4
R_1, R_3	l_2, l_7, l_6, l_5
R_1, R_4	l_2, l_8, l_7
R_2, R_3	l_2, l_5, l_4
R_2, R_4	l_2, l_8, l_6, l_4
R_3, R_4	l_8, l_6, l_5

Fig. 5. Discriminative landmarks of any two routes

A landmark l is an element of the conflict set $\mathbb{L}_{conflict}$ of a non-discriminative set S if and only if $S \cup \{l\}$ is a superset of an discriminative set already being pruned. In other words, a landmark l is an element of $\mathbb{L}_{conflict}$ of S if there exists a pruned discriminative set S' satisfying $l \in S' - S \wedge |S' - S| = 1$. Therefore, during processing we keep track of all the pruned discriminative sets in $\mathbb{S}_{record}$. The $\mathbb{L}_{conflict}$ of S can be generated as follows:

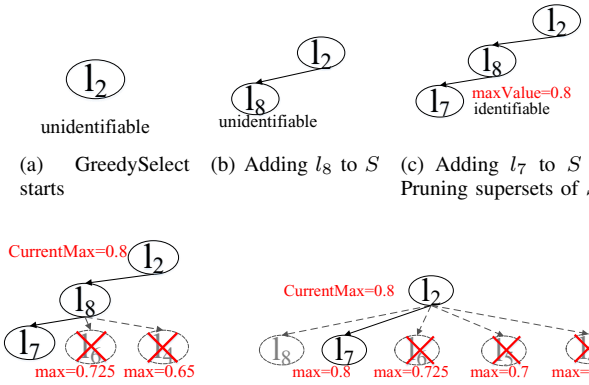
$$\mathbb{L}_{conflict} = \bigcup_{S' \in \mathbb{S}_{record}} \{l | l \in S' - S \wedge |S' - S| = 1\}$$

However, the above equation needs to compare with all the pruned discriminative sets, which is costly when there are a large amount of pruned discriminative sets. To speed up the conflict set generating, we build inverted index for each landmark to indicate which pruned discriminative sets contain it.

3) **Optimization at test step:** Our optimization for the test step comes from this important observation:

Observation 1: For any set S and S', where $S \subset S'$, if $\forall l_i \in S, l_j \in S' - S$, $l_i.s > l_j.s$, then the target value of $GetMaxSet(S')$ is always smaller than the target value of $GetMaxSet(S)$.

Based on this observation, during testing, we eagerly retrieve the maximum super of the current S. If the maximum target value is less than the target value of the current $\mathbb{L}_{best}$, then we stop expanding S, as all the following added landmark will have a lower significance than the elements in S, and following Observation 1, the following expansion cannot generate a better landmark set than the current $\mathbb{L}_{best}$. For example in Fig. 6(d), the target value of current $\mathbb{L}_{best}$ equals to 0.8 which is given by $S = \{l_2, l_8, l_7\}$. Since the possible maximum target values given by landmark sets containing $\{l_2, l_6\}$ and $\{l_2, l_4\}$ are 0.725 (given by $\{l_2, l_6, l_8, l_7\}$) and 0.65 (given by $\{l_2, l_4, l_8, l_7\}$) respectively, then the landmark sets containing $\{l_2, l_6\}$ or $\{l_2, l_4\}$ will be not be expanded, so as the landmark sets containing $\{l_6\}$, $\{l_5\}$ or $\{l_4\}$ in Fig. 6(e).



(a) GreedySelect starts (b) Adding l_8 to S (c) Adding l_7 to S and Pruning supersets of S

(d) Use upper bound to prune landmark combinations (e) Use upper bound to prune landmark combinations

Fig. 6. ILS algorithm

C. Question Ordering

In the previous step we select questions (landmarks), which can be regarded as the *question library*. However, presenting those question to workers with random order is unwise because of the following two reasons: 1) it is not necessary to ask all the questions in most cases. For example, in Fig. 3 if a worker indicates that she prefers the routes passing l_2 from l_1 to l_{10}, we do not need to ask whether he recommend to pass l_8 since all the routes passing l_2 do not pass l_8; 2) each time we ask a question, we would like to obtain the most informative feedback, which is more likely to identify the final answer. This implies that 1) the next question to be asked depends on the result of the previous question, so the question order is a tree-like structure; 2) the informativeness of a question (landmark) l is proportional to people's familiarity of the landmark (the significance of the landmark), and how many routes the landmark can prune (the information gain if we ask the question).

In order to arrange the questions into a tree-like structure, we first give the formula to calculate the strength of a question. Here we use $\bar{\mathbb{R}}_{l_{k_1}^{+/-} l_{k_2}^{+/-} \cdots l_{k_i}^{+/-}}$ to denote the subset of $\bar{\mathbb{R}}$ in which each route satisfies the answers of questions $l_{k_1}, l_{k_2} \cdots, l_{k_i}$ and the $l_{k_i}^+$ denotes that the answer of k_i is yes and $l_{k_i}^-$ denotes that the answer of k_i is no. Thus the informativeness $IS(l_{k_i})$ of question l_{k_i} is defined as following:

$$IS(l_{k_i}) = l_{k_i}.s[H(\bar{\mathbb{R}}_{k_{i-1}}) - \frac{\bar{\mathbb{R}}_{k_i}^+}{\bar{\mathbb{R}}_{k_i}^+ + \bar{\mathbb{R}}_{k_i}^-} H(\bar{\mathbb{R}}_{k_i}^+) - \frac{\bar{\mathbb{R}}_{k_i}^-}{\bar{\mathbb{R}}_{k_i}^+ + \bar{\mathbb{R}}_{k_i}^-} H(\bar{\mathbb{R}}_{k_i}^-)]$$

where $H(*)$ is the empirical entropy of $*$, $\bar{\mathbb{R}}_{k_{i-1}}$ stands for $\bar{\mathbb{R}}_{l_{k_1}^{+/-} l_{k_2}^{+/-} \cdots l_{k_{i-1}}^{+/-}}$, while $\bar{\mathbb{R}}_{k_i}^+$ and $\bar{\mathbb{R}}_{k_i}^-$ represent $\bar{\mathbb{R}}_{l_{k_1}^{+/-} \cdots l_{k_{i-1}}^{+/-} l_{k_i}^+}$ and $\bar{\mathbb{R}}_{l_{k_1}^{+/-} \cdots l_{k_{i-1}}^{+/-} l_{k_i}^-}$ respectively.

In order to get more information after each question, we employ the Iterative Dichotomiser 3 (ID3) algorithm [20], which recursively selects the question with the largest informativeness as the next question, to build tree-like question format $\mathcal{T}$. The algorithm consists of four steps: 1) Calculate the informativeness of every question using the whole routes set $\bar{\mathbb{R}}$. 2) Split the routes set $\bar{\mathbb{R}}_{k_{i-1}}$ into two subsets $\bar{\mathbb{R}}^+_{k_i}$ and $\bar{\mathbb{R}}^-_{k_i}$.3) make a decision node of $\mathcal{T}$ containing question l_{k_i}.4) perform the above steps recursively on routes subsets $\bar{\mathbb{R}}^+_{k_i}$ and $\bar{\mathbb{R}}^-_{k_i}$ using remaining questions until all the subsets have only one route.

For example, the question ordering result of the routes in Fig. 3 is shown in Fig 7. The system will issue questions according to the workers' answers to each question. Here workers' only need to answer two questions till the system getting their preference.

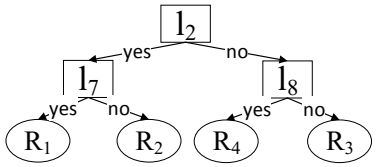


Fig. 7. Discriminative landmarks of any two routes

IV. Worker Selection

Some Crowdsourcing platforms such as AMT and CrowdFlower give workers the freedom to choose any questions. However this may cause some problems. For example, many workers choose to answer a same question while some other questions are not picked by anyone; workers have to view all the questions before they choose; workers may answer questions that they are not familiar with. CrowdPlanner avoids these problems by designing a dedicated component to assign each task to a set of eligible workers. In order to judge whether a worker is eligible for a task, many aspects of the worker have to be taken into consideration, i.e., number of outstanding tasks, worker's response time and familiarity with a certain area. First, since each worker may have many outstanding tasks, in order to balance the workload and reduce the response time, we use a threshold $\eta_{\#q}$ to restrict the maximum number of tasks for each worker. Second, each user of CrowdPlanner can specify the longest time delay she allows to get an answer, so this task will not be assigned to workers who have a high probability to miss the due time. Last, a recommended route will have high confidence to be correct if the assigned workers are very familiar with this area. Again, the worker's familiarity with respect to a certain area can also be affected by several factors, such as whether the worker lives around the area, whether the worker has answered questions relating to this area correctly in the past, etc. In summary, an eligible worker should meet three conditions: 1. has quota to answer the question; 2. has high probability to answer a question before the due time; 3. has relatively high familiarity level with the query regions.

A. Response Time

Each task has a user-specified response time, by which an answer must be returned. We assume the probability of the response time t of a worker follows an exponential distribution, i.e., $f(t;\lambda) = \lambda \exp^{-\lambda t}$, which is standard assumption in estimating worker's response time. The cumulative distribution function of $f(t;\lambda)$ is $F(t;\lambda) = 1 - \exp^{-\lambda t}$. If the probability of a worker to respond a task within time $\bar{t}$, represented by $F(\bar{t};\lambda)$, is less than the threshold η_{time}, we will not assign the task to him.

B. Worker's Familiarity Score

People usually have the best knowledge for areas where they live or visit frequently. In CrowdPlanner, we develop a familiarity score f^l_w to estimate the knowledge of a worker w about a landmark l. f^l_w is mainly affected by two factors: (1) worker's profile information, including her home address p_{home}, work place p_{work} and familiar suburbs p_{fs}, which can be collected during her registration to the system, and (2) history of worker's tasks around this area. f^l_w of landmark is defined as:

$$\begin{aligned} f^l_w =& \alpha \cdot \exp\{-(d(l,p_{home}) + d(l,p_{work}) + d(l,p_{fs})\} \\ &+ (1-\alpha)\cdot(\#correct + \beta \cdot \#wrong) \end{aligned}$$

where α is a smoothing variable, $d(l,p_*)$ is the distance between l and p_*, $\#correct$ is the number of correctly answered questions of l, $\#wrong$ is the number of incorrectly answered questions of l, and β is a constant less than 1, which measures the gain of a wrong answer. Notably, to avoid one's knowledge of far away places affect the calculating of her knowledge here, we assign $+\infty$ to $d(l,x)$ if $d(l,p_*)$ is bigger than a threshold η_{dis}. With all the n workers and m landmarks in our system, a $n*m$ matrix M with $m_{ij} = f^{l_j}_{w_i}$ is built, where $f^{l_j}_{w_i}$ is worker w_i's familiarity score of landmark l_j. Since the number of landmarks a worker has answered is always small compared with the large number of landmarks in the space, M is very sparse. Hence, if task assigning is only based on the sparse M, the assigning process has a strong bias to assign tasks to only a few well-performed workers. Actually, workers who have similar profile information or have answered several similar questions are highly possible to share the similar knowledge. For example, if a worker w_1 has high familiarity score with l_1, l_2 and l_3 and another worker w_2 living nearby has high familiarity score with l_1 and l_2, w_2 is also likely to be familiar with l_3 though w_2 has not answered any question relating to l_3. Similar situations hold for landmarks. Therefore, we need to predict familiarity scores of workers on landmarks using the latent similarity between workers and that of landmarks.

The familiarity scores of different pairs of (worker, landmark) are determined by some unweighed or even unobserved factors, which are regarded as some hidden knowledge categories, e.g. certain type of landmarks. However, we do not manually specify these factors, as hard-coded factors are usually limited and biased. Instead, we assume the familiarity

score of each worker-landmark pair is a linear combination of two groups of scores, i.e. (1) how a worker is familiar with each hidden knowledge category, and (2) how a landmark is related to each hidden knowledge category. Then we employ Probabilistic Matrix Factorization (PMF) [16] to factorize M into two latent feature matrices, $W \in R^{d\times n}$ and $L^{d\times m}$, which are the latent worker and landmark feature matrices, respectively. That is, $M = W^T L$, where $W_{i,k}$ describes how familiar worker w_i is with knowledge category k, and $L_{j,l}$ describes how related landmark l_j is to knowledge category k. Further, we assume there exists observation uncertainty R, and the uncertain follows a normal distribution. Thus the distribution of a new worker-landmark familiarity matrix M', which predicts some familiarity by leveraging the similarity between different workers and landmarks, conditioned on W and L is defined as follows:

$$p(M'|W, L, \sigma^2) = \prod_{i=1}^{n}\prod_{j=1}^{m}[\mathcal{N}(M_{ij}|W_i^T L_j, \sigma^2)]^{I_{ij}} \quad (1)$$

where $\mathcal{N}(x|\mu, \theta^2)$ is the probability density function of the normal distribution with mean μ and variance θ^2, and I_{ij} is a indicator which is equal to 1 if M_{ij} is not zero, otherwise 0. The prior of W and L are defined as follows:

$$p(W|\sigma_W^2) = \prod_{i=1}^{n}\mathcal{N}(W_i|0, \sigma_W^2 I)$$
$$p(L|\sigma_L^2) = \prod_{i=1}^{m}\mathcal{N}(L_i|0, \sigma_L^2 I)$$

where I is identity matrix. The following objective function maximizes the posterior of W and L with regularization terms, which minimizes the prediction difference between our model and the observed M, and also automatically detects the appropriate number of factors d through the regularization terms:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(M_{ij} - W_i^T L_j)^2 + \lambda_W \sum_{i=1}^{n}\|W_i\|_F^2 + \lambda_L \sum_{j=1}^{m}\|L_j\|_F^2$$

where $\lambda_W = \theta^2/\theta_W^2$, $\lambda_L = \theta^2/\theta_L^2$, and $\|\cdot\|_F^2$ denotes the Frobenius norm. A local minimum of the objective function can be found by performing gradient descent in W and L. Afterwards, more familiarity scores between workers and landmarks are inferred in M.

A worker with a familiarity score of a landmark means he has some knowledge about the region around the landmark, not just the landmark itself. As a result, the accumulated familiarity score $F_{w_i}^{l_j}$ of l_j of a worker w_i is a weighted sum of all the landmarks in the η_{dis} range of l_j. We assume the weight around a landmark l follows a normal distribution of the distance to l, and the region that the knowledge of l can cover is limited in a circle with the center of l and the radius of η_{dis}. Thus, $F_{w_i}^{l_j}$ is evaluated as follows:

$$F_{w_i}^{l_j} = \sum_{l\in\mathbb{L}_{near}\cup\{l_j\}} \delta_l f_{w_i}^{l}$$

where $\mathbb{L}_{near}$ is the set of landmarks in the η_{dis} range of l. The weight $\delta_l = \mathcal{N}(d(l, l_j)|0, \sigma_0^2)$ where $\sigma_0 = \eta_{dis}/3$. We use M^* to denote the worker-landmark matrix of the accumulated familiarity score, where m_{ij}^* equals to $F_{w_i}^{l_j}$.

C. Finding Top-k Eligible Workers

Next we discuss how to find the top-k eligible workers for a given task. Given a task (the selected n landmarks $\mathbb{L}_\mathbb{R}$), the worker-landmark accumulated familiarity score matrix M^*, a response time t, a positive integer k, a top-k eligible workers query returns k workers who have the most knowledge of landmarks in $\mathbb{L}_\mathbb{R}$ among all the workers and have high possibility to finish the task within time t.

For a single landmark l_j, there may be several workers, denoted as $\mathbb{W}_{l_j}$, who have non-zero accumulated familiar scores, which means these workers have some knowledge of l_j. For a task (a set of landmarks $\mathbb{L}_\mathbb{R}$), $\bigcup_{l\in\mathbb{L}_\mathbb{R}} \mathbb{W}_l$ represents workers who have knowledge of any landmark of $\mathbb{L}_\mathbb{R}$. Then we filter out workers, of who the possibility of finishing the task within time t is no more than η_{time}, from $\bigcup_{l\in\mathbb{L}_\mathbb{R}} \mathbb{W}_l$. Afterwards the remained workers in $\bigcup_{l\in\mathbb{L}_\mathbb{R}} \mathbb{W}_l$ are regarded as candidate workers denoted by $\mathbb{W}$. However, simply adding up a worker's accumulated familiarity scores on all the landmarks of $\mathbb{L}_\mathbb{R}$ may lead biased result in worker selection. For example, there are ten landmarks in a task and two candidates workers w_1 and w_2, that w_1 only has a very good knowledge of landmark l_1, say $F_{w_1}^{l_1}$=2, and knows nothing about the rest landmarks, $F_{w_1}^{l_i} = 0$, $(2 \le i \le 10)$; while w_2 has some knowledge of all the landmarks that $F_{w_2}(l_i) = 0.1$ $(1 \le i \le 10)$. Comparing the adding up sum of accumulated familiarity scores of the ten landmarks, w_1 will be selected to be assigned the task. However, the coverage of w_1's knowledge of the landmark set is too narrow, that w_1 may feel hard to answer questions about $l_2, l_3, \cdots, l_{10}$, in the knowledge coverage manner, w_2 is a better choice. Thus, when selecting workers from candidate workers, not only their sum of accumulated familiarity scores of all the landmarks, but also the knowledge coverage of all the landmarks should be considered. The choosing rules are quite similar to rated voting system [17], of which the wining option is chosen according to the voters preferences score of options and the number of voters preferring the options. In our system, we can treat each landmark of $\mathbb{L}_\mathbb{R}$ as a voter and each worker of $\mathbb{W}$ as an option. Adopting the idea of rated voting system, we can measure the landmark l_j's preference of all the candidate workers by the following two steps: 1) rank workers of $\mathbb{W}_{l_j} \cap \mathbb{W}$, who are in the candidate workers set $\mathbb{W}$ and have accumulated familiar scores $F_w^{l_j}$ bigger than zero, in descending order of $F_w^{l_j}$; 2) the preference score $p_{l_j}^w$ of l_j to each worker w in $\mathbb{W}_{l_j} \cap \mathbb{W}$ is defined as follows:

$$p_{l_j}^w = \begin{cases} 1 - \frac{rank(w)-1}{|\mathbb{W}_{l_j}\cap\mathbb{W}|}, if\ w \in \mathbb{W}_{l_j} \cap \mathbb{W} \\ 0, otherwise \end{cases}$$

where $rank(w)$ is the ranked place of w among $\mathbb{W}_{l_j} \cap \mathbb{W}$. In this way the worker with high accumulate familiarity score will

get a relatively high preference score and ensure the preference score will not result in a bias in worker selecting. Afterwards, all the landmarks will vote their preferences to the candidate workers. Then we sum up the preferences of each worker voted by landmarks, and choose the workers with the top-k biggest adding up preference scores as the query results.

V. Experiments

In this section, we conduct extensive experiments to validate the effectiveness and efficiency of the two core components of our proposed CrowdPlanner system, namely landmark selection and worker selection. All the algorithms in our system are implemented in Java and run on a computer with Intel Core i5-3210 CPU (2.50GHz) and 4 GB memory.

A. Experiment Setup

Trajectory Dataset: We use two real trajectory datasets generated by taxis, trucks and private cars in Beijing and Nanjing (big cities of China). The detail information of these trajectory datasets is shown in Table II.

TABLE II
Dataset

Id	City	#trajectory	Duration
A	Beijing	112,232	six months
B	Nanjing	35,340	three months

POI Clusters: We get two POI datasets of the Beijing and Nanjing cities from a reliable third-party company in China. After performing DBSCAN on these POI datasets, approximately 50,000 POI clusters are obtained and each POI cluster is used as a landmark.

Ground truth route: We carefully choose 1000 popular routes agreed by all three route mining algorithms in each city as the ground truth. These routes are treated as the correct answers for the route recommendation request between corresponding places.

Workers: In each of the cities we have several volunteers to answer the questions generated by CrowdPlanner.

B. Evaluation Approach

For each ground truth route, we query a big thumb map service provider to get three recommended routes from its source to its destination. The ground truth and the recommended routes form the candidate route set, based on which a task will be generated and assigned to workers. In this way, we can assess the accuracy of the answers returned by the system by comparing the answer with the ground truth.

Table III lists all the parameters we use throughout the experiments. All the parameters are assigned the default values unless specified explicitly.

TABLE III
Parameter Settings

Notation	Explanation	Default value
n	number of candidate routes	6
$\|\mathbb{L}\|$	size of landmarks on candidate routes	200
α	influence factor of people's living space to their knowledge	0.3
β	influence factor of people answering a question wrong to their knowledge	0.3
$\eta_{\#q}$	the maximum number of outstanding tasks of each worker	3
η_{dis}	radius of knowledge influence region	500m
η_t	the minimal possibility to answer a question in time	80%

C. Performance Evaluation

1) Case Study: Before conducting the quantitative performance evaluation, we give a demonstration of the CrowdPlanner system. Fig. 8 shows the system interface when a client user submit a recommendation request, which specifies she wants to get the best recommended routes from 'Nanjing Confucius Temple' to 'Nanjing Railway Station' fifteen minutes later (about 1:48am) and she awards the request for five coins (the virtual currency of CrowdPlanner). After

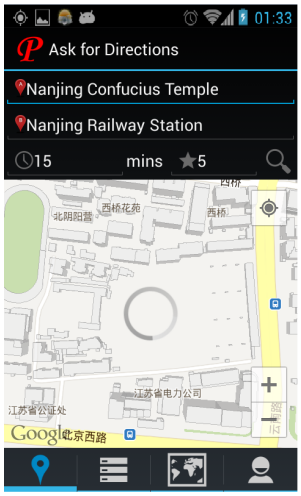
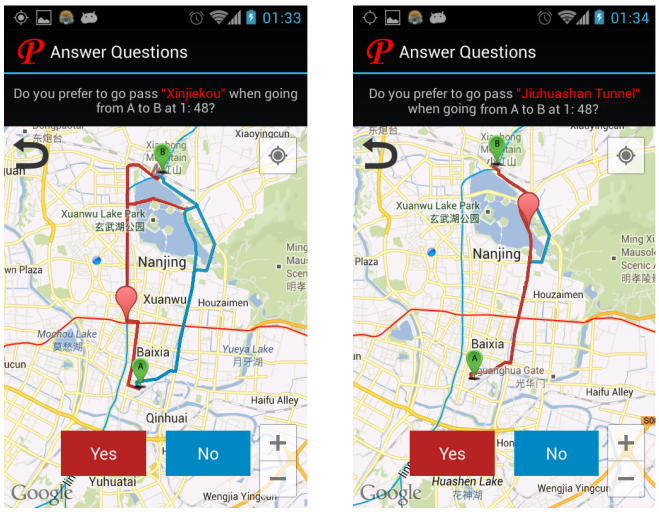


Fig. 8. A request for route evaluating

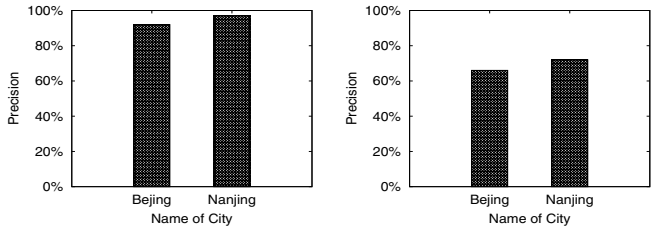
receiving the request, the server matches the request to the verified routes and generating candidates routes by invoking web services and popular route mining algorithms in turn. Since the system cannot automatically evaluate these candidate routes, it generates a route evaluation task and assigns it to some eligible workers. Fig. 9(a) illustrates the evaluation task on a client, where four candidate routes from Nanjing Confucius temple to Nanjing railway station are shown in red and blue lines. The first binary question is 'do you prefer to go past "Xinjiekou" from Nanjing Confucius temple to Nanjing railway station at 1:48?'. Xinjiekou is one of the most flushing commercial districts of Nanjing. Thus, to avoid traffic, the worker may prefer not to pass Xinjiekou, which prunes the two red routes. The second question for her is whether the route should pass "Jiuhuashan Tunnel". As shown in Fig. 9(b), "Jiuhuashan Tunnel" is the most famous tunnel under the Xuanwu Lake, which is the major difference between the two routes left.



(a) Question 1 (b) Question 2

Fig. 9. Questions of evaluating best route from Nanjing Confucius Temple to Nanjing Railway Station

2) Quality of Recommendation: The goal of CrowdPlanner is to give users the verified best routes between two places. In the first set of experiments, we evaluate the accuracy of routes recommended by CrowdPlanner by comparing with the ground truth. As shown in Fig. 10(a), the system can achieve very high accuracy ($\geq 90\%$) in the cities when suitable workers are selected, which means our system can recommend the best route from the set of candidate routes in most cases. Note that, as shown in Fig. 10(b), the system still has about 70% accuracy even if the tasks are assigned to random workers, demonstrating the robustness and tolerance to the workers' qualities of our system.

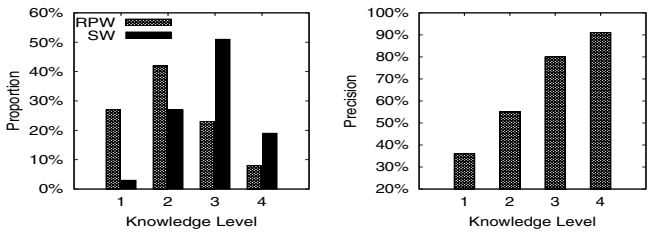


(a) Precision with worker selection (b) Precision without worker selection

Fig. 10. Accuracy of route recommendation

3) Effectiveness of Worker Selection: In this experiment we test whether the overall performance of the system can be improved by assigning tasks to suitable workers with good knowledge about the query area. As a comparison, we also assign the same tasks to random workers without any selection algorithms applied. The accuracy of the route recommendation is shown in Fig. 10, from which we can see that the overall accuracy can improve by 20% by applying the proposed worker selection methods.

We also collect statistics of the workers' knowledge about the queried area to further demonstrate why worker selection is necessary. Since a worker's knowledge is hard to quantify exactly, we propose to use four familiarity levels to assess the worker's knowledge: (1) has no idea of the area; (2) have heard the area but never been there; (3) have visited the area several times; (4) knows this area very well (local resident). We ask the workers to classify themselves into one of the four levels based on her familiarity to the query area. Fig. 11(a) shows the knowledge level of randomly picked workers (RPW) and selected workers (SW) of the querying regions. We can see that nearly 70% of the randomly picked workers have not travelled to the query regions and even 27% know nothing about the regions; on the other hand, 70% of selected workers have travelled at least once in these regions and about 20% selected workers know the area very well. This implies that the proposed worker selection algorithms can effectively find the workers with good knowledge about the query area.



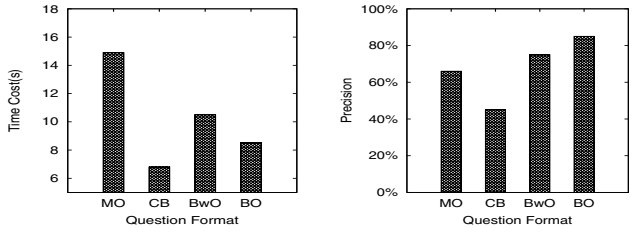
(a) Workers knowledge about querying places (b) Relationship between knowledge and precision

Fig. 11. Analysis of worker's knowledge

To further demonstrate the relationship between worker's knowledge and the accuracy of an answer, we analyze the relationship between the precision of recommended routes and workers' knowledge level. The result is shown in Fig. 11(b), from which we can see that the precision grows steadily with workers getting more familiar with the area.

4) Effect of Question Format: The question format adopted by CrowdPlanner is a series of binary questions with a certain order. In this experiment, we evaluate the effect of different question formats to the performance of the system. We compare our question format (BO) with three other format candidates: (1) map only format (MO: show the candidate routes directly on map and ask workers to choose), (2) check-box format (CB: workers need to choose all the landmarks on their preferred routes) and (3) binary question without smart ordering (BwO: the questions are asked in the descending order of the significance). We generate the same tasks using the four question formats and assign to the same set of workers. Both efficiency and accuracy are evaluated. The results are shown in Fig. 12. From Fig. 12(a) we can see that MO takes the longest time for workers to finish a task, which is because the workers need to spend lots of time to realize the differences between candidate routes. All other question formats, of which the differences are automatically summarized by the system, cost around 10s for each task. Notably, BO format costs less time than BwO since by presenting the questions in a smarter order the number of questions needed for each task

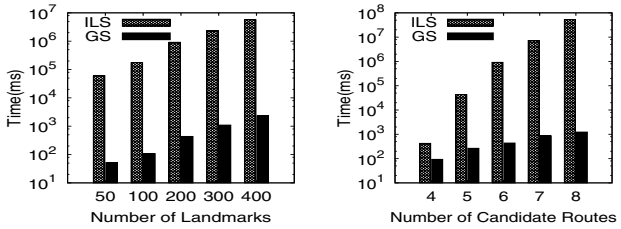
has reduced. CB takes the least time as many workers do not bother to check a lot of landmarks and simply skip the questions. Fig. 12(b) shows the results of accuracy, in which CB format has the lowest precision since people pay least time and attention on this type of question. Binary question format, both BwO and BO, outperform MO in precision since MO is hard for people to realize the difference between candidate routes on a map. Furthermore, the precision of BO is more than 10% higher than that of BwO, demonstrating that smart ordering not only reduces the time cost but also improves the accuracy of answers.



(a) Time cost of different question formats (b) Precision of different question formats

Fig. 12. Influence of question formats

5) Time Cost of Landmark Selection: We also test the time cost of landmark selection process, which is important for CrowdPlanner to respond to user request in real-time. In general two factors can influence the time cost, (a) the number of landmarks on the candidate routes and (b) the number of candidate routes. Both factors are tested in our experiments by comparing the GreedySelecting (GS) method with the Incremental Landmark Selecting (ILS) introduced by [22]. The average time cost for selecting landmarks of a candidate route set with the size of 6 is shown in Fig. 13(a), from which we observe that both the time costs of GS and ILS grow with the landmark size increasing from 50 to 400. However, GS constantly outperforms ILS by three orders of magnitudes. The average time costs for selecting landmarks with different number of candidate route set are shown in Fig. 13(b). It illustrates that the running time of GS is much more stable as the number of candidate routes grows, compared to the exponential growth of the time cost of ILS.



(a) Time cost of $k = 6$ (b) Time cost of $|\mathbb{L}| = 200$

Fig. 13. Efficiency of landmark selection algorithms

6) Precision of Map Service and Popular Route Mining Algorithms: The last set of experiments are conducted to demonstrate the precision of six kind of candidate routes, i.e., three provided by the web service and the others provided by three popular routes mining algorithms, respectively. The three routes, denoted by WS1, WS2 and WS3, are recommended by the web service with different levels of recommendation, where WS1 is the best recommended route of the web service, while WS3 is the least. We randomly generate 100 route recommendation tasks in each city and assign each task to its top-5 eligible workers to get the best route of each task. Fig. 14 shows the percentage of desirable results of each kind candidate route, defined as precision. Clearly, it shows that the precision of routes provided by web services (WS1, WS2 and WS3) is about 70%, however, the best recommended route WS1 has less 40% precision. Moreover, none of these providers can have the probability high enough to provide best routes. Though the precision of MFP is the highest, about 43%, among the six kinds of routes, it is still not satisfactory.

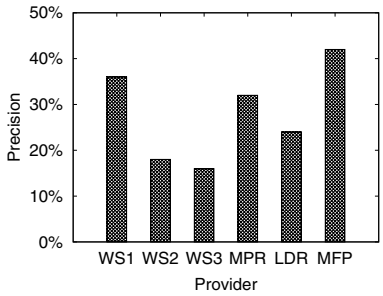


Fig. 14. Precision of routes from different sources

VI. Related Work

To our knowledge, there is no existing work on evaluating the quality of recommended routes. As the goal of this work is to evaluate the quality of recommended routes by web services and mining algorithms, the route recommendation algorithms (mining frequent path algorithms) used in this paper are reviewed first. Also we leverage the generating easy questions and finding target workers to improve the quality of evaluating and reduce the workers' workload, which share the same motivation of some research works of Crowdsourcing question designing and workers selecting. Therefore in the last of this section, we will review previous works of these two aspects.

Route Recommendation Algorithms. The popular routes mining has received tremendous research interests for a decade and a lot of works are on it, such as [21], [15], [6], [7], [27], [6], [15], [7], [27], [12], [11], [5], [10]. Among these works, [4], [26], [14], [3] are the most representative. Chen et al. [4] proposes a novel popularity function for path desirability evaluation using historical trajectory datasets. The popular routes recommended by it tends to have fewer vertices. The work in [26] provides k popular routes by mining uncertain trajectories. The recommendation routes of this work tend to be rough routes instead of correct routes. [14] claims the popular routes change with time, so it carries out a popular routes mining algorithms which can provide the recommended

routes in arbitrary time periods specified by the users. [3] provides the evidences that the routes recommended by web services are sometimes different from drivers' preference. Thus it mines the individual popular routes from his historical trajectories. The recommended routes of this method reflect certain people's preference.

Question Designing. Question designing is always an application dependent strategy, which may consider the cost of questions or the number of questions. [8], [18] propose strategies to minimize the cost of the questions designed. The question designing strategy of [25] is to minimize the number of questions. The question designing strategy of [19] is to generate the optimal set of questions. [13] builds the desired traveling plans incrementally, optimally choosing at each step the best questions so that the overall number of questions to minimize the number of the asked questions.

Worker Selecting. Selecting workers with high individual qualities for tasks always does beneficial to the final quality of answers. Thus [9] propose an algorithm to select workers fulfilling some skills with the minimized the cost of choosing them. In [1] use emails communication to identifying skillful workers. Cao et al [2] assign tasks to micro-blog users by mining users' knowledge and measuring their error rate.

VII. Conclusions

In this work we present a novel crowd-based route recommendation system – CrowdPlanner, which evaluates the quality of routes recommended from different sources by leveraging the knowledge and opinions of the crowd. Two core components, task generation and worker selection, have been carefully designed such that informative and concise questions will be created and assigned to the most suitable workers. By having the system deployed and tested in real scenarios, we demonstrate CrowdPlanner is able to recommend users the most satisfactory routes with at least 90-percent chances, much higher than either the most well-known map services or the state-of-art route mining algorithms. Besides, this research sheds light on some other crowd-based recommendation systems such as location recommendation and itinerary planning, which can be used in more application scenarios.

Acknowledgments

This research is partially supported by Natural Science Foundation of China (Grant No.61232006) and the Australian Research Council (Grants No. DP110103423, DP120102829 and LP130100164). Lei Chen's work is partially supported by Hong Kong RGC-NSFC N_HKUST637/13, National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200, Microsoft Research Asia Gift Grant and Google Faculty Award 2013

References

[1] C. Campbell, P. Maglio, A. x. Cozzi, and B. Dom. Expertise identification using email communications. In *CIKM*, pages 528–531. ACM, 2003.

[2] C. Cao, J. g. She, Y. Tong, and L. Chen. Whom to ask? jury selection for decision making tasks on micro-blog services. *PVLDB*, 5(11):1495–1506, 2012.

[3] V. Ceikut and C. Jensen. Routing service qualityłlocal driver behavior versus routing services. In *MDM*, pages 195–203. IEEE, 2013.

[4] Z. Chen, H. Shen, and X. Zhou. Discovering popular routes from trajectories. In *ICDE*, pages 900–911, 2011.

[5] S. Gaffney and P. Smyth. Trajectory clustering with mixtures of regression models. In *SIGKDD*, pages 63–72. ACM, 1999.

[6] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi. Trajectory pattern mining. In *KDD*, pages 330–339, 2007.

[7] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. Sondag. Adaptive fastest path computation on a road network: A traffic mining approach. In *PVLDB*, pages 794–805, 2007.

[8] S. Guo, A. Parameswaran, and H. Garcia-Molina. So who won? dynamic max discovery with the crowd. In *SIGMOD*, pages 385–396. ACM, 2012.

[9] T. Lappas, K. Liu, and E. Terzi. Finding a team of experts in social networks. In *SIGKDD*, pages 467–476. ACM, 2009.

[10] J. Lee, J. Han, X. Li, and H. Gonzalez. Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering. *PVLDB*, 1(1):1081–1094, 2008.

[11] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *SIGMOD*, pages 593–604. ACM, 2007.

[12] X. Li, J. Han, J. Lee, and H. Gonzalez. Traffic density-based discovery of hot routes in road networks. *Advances in Spatial and Temporal Databases*, pages 441–459, 2007.

[13] I. Lotosh, T. Milo, and S. Novgorodov. Crowdplanr: Planning made easy with crowd. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 1344–1347, 2013.

[14] W. Luo, H. Tan, L. Chen, and L. Ni. Finding time period-based most frequent path in big trajectory data. In *SIGMOD*, pages 195–203. ACM, 2013.

[15] N. Mamoulis, H. Cao, G. Kollios, M. Hadjieleftheriou, Y. Tao, and D. Cheung. Mining, indexing, and querying historical spatiotemporal data. In *KDD*, pages 236–245, 2004.

[16] A. Mnih and R. Salakhutdinov. Probabilistic matrix factorization. In *NIPS*, pages 1257–1264, 2007.

[17] L. Nordmann and H. Pham. Weighted voting systems. *Reliability, IEEE Transactions on*, 48(1):42–49, 1999.

[18] A. Parameswaran, H. Garcia-Molina, H. Park, N. Polyzotis, A. Ramesh, and J. Widom. Crowdscreen: Algorithms for filtering data with humans. In *SIGMOD*, pages 361–372. ACM, 2012.

[19] A. Parameswaran, A. Sarma, H. Garcia-Molina, N. Polyzotis, and J. Widom. Human-assisted graph search: it's okay to ask questions. *PVLDB*, 4(5):267–278, 2011.

[20] J. R. Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[21] D. Sacharidis, K. Patroumpas, M. Terrovitis, V. Kantere, M. Potamias, K. Mouratidis, and T. Sellis. On-line discovery of hot motion paths. In *EDBT*, pages 392–403, 2008.

[22] H. Su. Crowdplanner: A crowd-based route recommendation system. *CoRR*, abs/1309.2687, 2013.

[23] H. Su, J. Deng, and F. Li. Crowdsourcing annotations for visual object detection. In *AAAI*, 2012.

[24] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou. Calibrating trajectory data for similarity-based analysis. In *SIGMOD*, pages 833–844. ACM, 2013.

[25] J. Wang, T. Kraska, M. Franklin, and J. Feng. Crowder: Crowdsourcing entity resolution. *PVLDB*, 5(11):1483–1494, 2012.

[26] L.-Y. Wei, Y. Zheng, and W.-C. Peng. Constructing popular routes from uncertain trajectories. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 195–203. ACM, 2012.

[27] Y. Zheng, L. Zhang, X. Xie, and W. Ma. Mining interesting locations and travel sequences from gps trajectories. In *WWW*, pages 791–800, 2009.