Backward Path Growth for Efficient Mobile Sequential Recommendation

Jianbin Huang, Xuejun Huangfu, Heli Sun, Hui Li, Peixiang Zhao, Hong Cheng, and Qinbao Song

Abstract—The problem of mobile sequential recommendation is to suggest a route connecting a set of pick-up points for a taxi driver so that he/she is more likely to get passengers with less travel cost. Essentially, a key challenge of this problem is its high computational complexity. In this paper, we propose a novel dynamic programming based method to solve the mobile sequential recommendation problem consisting of two separate stages: an offline pre-processing stage and an online search stage. The offline stage pre-computes potential candidate sequences from a set of pick-up points. A backward incremental sequence generation algorithm is proposed based on the identified iterative property of the cost function. Simultaneously, an incremental pruning policy is adopted in the process of sequence generation to reduce the search space of the potential sequences effectively. In addition, a batch pruning algorithm is further applied to the generated potential sequences to remove some non-optimal sequences of a given length. Since the pruning effectiveness keeps growing with the increase of the sequence length, at the online stage, our method can efficiently find the optimal driving route for an unloaded taxi in the remaining candidate sequences. Moreover, our method can handle the problem of optimal route search with a maximum cruising distance or a destination constraint. Experimental results on real and synthetic data sets show that both the pruning ability and the efficiency of our method surpass the state-of-the-art methods. Our techniques can therefore be effectively employed to address the problem of mobile sequential recommendation with many pick-up points in real-world applications.

Index Terms—Mobile sequential recommendation, potential travel distance, backward path growth, sequence pruning

1 Introduction

With the wide utilization of wireless sensors and information infrastructures such as GPS, Wi-Fi, Blue Tooth and RFID, we can easily get access to the location trace data for a large number of moving objects. Finding useful knowledge from these trajectory data will provide strong support for the real-time decision and intelligence services in the related applications [1]. The problem of reducing cruising costs for taxicabs is a typical example [2], [3]. An unloaded taxi cruising on the road not only leads to waste of fuel and time, but also may result in traffic congestion. However, there exist many pick-up hotspots in taxi trajectories of those high-yield taxi drivers, which can be effectively leveraged to guide new drivers to pick up passengers in a more economical and energy-efficient way. Therefore, highly efficient mobile pattern mining and recommendation algorithms can significantly improve business performance and reduce the energy consumption. This problem possesses considerable theoretical significance and practical value [3], [4].

In previous studies, Ge et al. have proposed a novel problem of mobile sequential recommendation (MSR) [2], which is to suggest a route connecting a series of pick-up points for an empty cab so that the driver is more likely to get passengers with less travel cost starting from his current position. It is a challenging task, because we need to enumerate and compare all possible routes derived from the given set of pick-up points which involves a rather high computational complexity. To solve the MSR problem, they proposed a function of potential travel distance (PTD) for evaluating the cost of a driving route. Essentially, the PTD value of a suggested route is the expectation of the cruising distance for an empty cab before it successfully gets new passengers when it travels along the route. To reduce the computational cost, two effective potential sequence pruning algorithms LCP and SkyRoute, which are based on the monotone property of the PTD function, have been proposed in [2]. However, the time and space complexities of these two algorithms both grow exponentially with the number of pick-up points and the lengths of the suggested driving routes, so they can only be applied to the driving route recommendation with a limited length constraint and a small number of pick-up points.

In this paper, we propose a novel and efficient solution to the MSR problem. Our method includes an offline stage and an online stage. The offline stage effectively prunes the search space and generates a small set of candidate sequences. The online stage is aimed at obtaining the optimal driving route given the current position of an unloaded taxi as

- *J. Huang and X. Huangfu are with the School of Software, Xidian University, Xi'an 710071, China. E-mail: jbhuang@xidian.edu.cn, huangfuxj@foxmail.com.*
- *H. Sun and Q. Song are with the Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, Shaanxi, China. E-mail: helisun.sunny@gmail.com, qbsong@mail.xjtu.edu.cn.*
- *H. Li is with the School of Computer Science and Technology, Xidian University, Xi'an 710071, China. E-mail: hli@xidian.edu.cn.*
- *P. Zhao is with the Department of Computer Science, Florida State University, James Love Building, 1107 Academic Way, Tallahassee, FL 32306. E-mail: zhao@cs.fsu.edu.*
- *H. Cheng is with the Department of Systems Engineering and Engineering Management, Chinese University of Hong Kong, Hong Kong, Shatin, N.T., China. E-mail: hcheng@se.cuhk.edu.hk.*

Manuscript received 23 Apr. 2013; revised 29 Dec. 2013; accepted 31 Dec. 2013. Date of publication 8 Jan. 2014; date of current version 1 Dec. 2014. Recommended for acceptance by J. Freire.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2014.2298012

the starting point. Specifically, for the offline pre-computation, we have examined the nature of the PTD function and found that it satisfies the iterative calculation property. This property allows us to incrementally construct a potential driving route backward from the terminal point to the starting point. Furthermore, we have also found that a set of potential sequences with the same length and the same source point satisfy the incremental and batch pruning properties. To this end, we designed a mobile sequential recommendation method taking full advantage of the iterative nature of the PTD function. It incrementally generates potential sequences and removes a lot of unfertile search space, which greatly enhances the efficiency and reduces the memory consumption of our method. Among the generated potential sequences with the same length, we can further prune a large number of potential sequences that fail to form the optimal route by using a batch pruning policy. It can dramatically reduce the number of the remaining sequence candidates. In our method, the original MSR problem can be generalized effortlessly to handle the case when a travel distance or a destination constraint is imposed. Given the current position of an unloaded taxi, our method can efficiently search for the optimal driving route online. Experimental results demonstrate that the offline pruning effectiveness and the online search efficiency of our method are significantly better than the state-of-the-art methods.

The main contributions of the paper are given below. 1) Our algorithm can generate all possible candidate sequences of arbitrary lengths. 2) The recursive formula of the PTD function is proposed which makes the incremental generation of the potential sequences possible. 3) A backward incremental sequence generation algorithm with less time complexity is proposed. 4) An efficient method for comparing the PTD cost of different potential sequences and driving routes is presented. 5) An effective sequence pruning method combining incremental pruning with batch pruning is adopted which significantly improves the offline pruning effect. 6) Our method can handle the optimal driving route search problem with a maximum cruising distance or a destination constraint that has never been examined by previous studies.

The rest of the paper is organized as follows. Section 2 introduces the MSR problem and the related work. Section 3gives the detailed presentation of our proposed method. In Section 4, the offline sequence generation and online search algorithms are described in detail. Section 5 gives the experimental results and analysis. Section 6 discusses some extensions of our method. Finally, Section 7 concludes the paper.

2 Background

In this section, we first introduce the MSR problem and then describe the related work.

2.1 Problem Statement

Given a large number of GPS traces of taxi drivers collected in a period of time, the information about when a cab is empty or occupied is also available. In this data set, we can geometrically cluster the pick-up points of all taxi drivers in a certain period of time. The centers of these clusters can be used as the potential pick-up points. Let c_i be a potential pick-up position and $C = \{c_1, c_2, \ldots, c_N\}$ be a set of N pick-up points. The probability that a taxi can successfully carry passengers at the pick-up point c_i is denoted by $P(c_i)$, and the set of mutually independent probability is $P = \{P(c_1),\ P(c_2), \ldots, P(c_N)\}$. Successful probability $P(c_i)$ can be estimated by the number of pick-up events having happened in c_i divided by the number of cabs which have no customers before passing c_i. The MSR problem introduced by Ge et al. [2] is for finding a driving route that can lead to the minimum cost of picking up a new passenger when a taxi travels all or part of the pick-up points in C starting from its current location. The problem can also be found in other scenarios such as recommending tourist routes, searching for parking places, etc. In the following, we introduce some concepts of the MSR problem and all the symbols used in this paper are described in Table 1.

TABLE 1
Adopted Symbols

Symbols	Definition
C	A set of potential pick-up points.
c_i	A location point. It represents the current location of the cab when $i = 0$ and a pick-up point in C with $i > 0$.
N	The number of pick-up points in C.
$P(c_i)$	The probability of successfully taking passengers at c_i.
D	The distance matrix of pairs of location points.
D_{c_i,c_j}	The distance from c_i to c_j.
$\vec{r}$	The potential mobile sequence containing one or more different pick-up points.
$\|\vec{r}\|$	The length of potential sequence $\vec{r}$ (i.e., the number of different pick-up points in $\vec{r}$).
$C_{\vec{r}}$	The set of pick-up points in the potential sequence $\vec{r}$.
$P_{\vec{r}}$	The probability vector of the pick-up points of the potential sequence $\vec{r}$.
$\langle c, \vec{r}\rangle$	A driving route that travels the potential sequence $\vec{r}$ starting from the location point c.
$DL(\vec{d})$	the cruising distance of $\vec{d}$ from its starting point to the destination point.
$s(\vec{r})$	The source point of the potential sequence $\vec{r}$.
$\overrightarrow{R}$	A set of all potential sequences.
$\overrightarrow{R^L}$	A set of potential sequences with length L.
$\overrightarrow{R^L_{\vec{r}}}$	A set of potential sequences that have the same length, source point and pick-up point set as $\vec{r}$.
$\overrightarrow{R^L_c}$	A set of potential sequences with length L and source point c.

Let $\vec{r} = \langle c_1, c_2, \ldots, c_L\rangle$ be a potential sequence of length L derived from the pick-up points set C, where each c_i in $\vec{r}$ is different from one another. c_1 is called the source point of $\vec{r}$ and c_L is called the destination point. $C_{\vec{r}} = \{c_1, c_2, \ldots, c_L\}$ denotes the set of pick-up points of the potential sequence $\vec{r}$. $\overrightarrow{R} = \{\vec{r} | C_{\vec{r}} \subseteq C \wedge C_{\vec{r}} \neq \emptyset\}$ is the set of all possible potential sequences derived from C. $|\overrightarrow{R}| = M$ is the number of potential sequences in $\overrightarrow{R}$. $P(\vec{r}) = \langle P(c_1), P(c_2), \ldots, P(c_L)\rangle$ is the probability vector of the potential sequence $\vec{r}$ consisting of the probabilities of all the pick-up points in $\vec{r}$. $\vec{d} = \langle c_0, \vec{r}\rangle$ is a driving route, where c_0 is the starting point, $\vec{r}$ the sequence of pick-up points, and $\|\vec{d}\| = \|\vec{r}\| = L$ the length of $\vec{d}$.

For a driving route $\vec{d} = \langle c_0, \vec{r}\rangle$, a PTD function is defined in [2] to evaluate its travel cost. Let $D(\vec{d}) = \langle D_{c_0,c_1}, (D_{c_0,c_1} + D_{c_1,c_2}), \ldots, \sum_{i=1}^{L} D_{c_{i-1},c_i}, D_{\max}\rangle$ be the distance vector of $\vec{d}$ and probability vector $P(\vec{d}) = \langle P(c_1), \overline{P(c_1)} \cdot P(c_2), \ldots, \prod_{i=1}^{L-1} \overline{P(c_i)} \cdot P(c_L), \prod_{i=1}^{L} \overline{P(c_i)}\rangle$, then the PTD cost of $\vec{d}$ can be calculated by

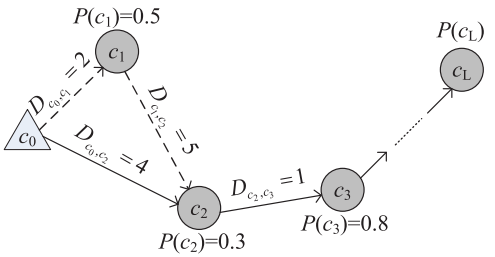


Fig. 1. An example of the driving route and its PTD cost.

$$F(\vec{d}) = F(c_0, \vec{r}, P(\vec{r})) = D(\vec{d}) \cdot P(\vec{d}), \quad (1)$$

where the manually specified parameter $D_{\max}$ represents the desired maximum cruising distance of a driver for picking up new passengers. In [2], the $D_{\max}$ is set to a large enough constant value. The PTD value of a driving route $\vec{d}$ represents the expected travel distance of an empty cab for picking up new passengers when it is cruising along the route. The smaller the PTD cost of a route, the shorter travel distance and the less energy and cost required for the cab to take new guests. Based on the above definitions, the MSR problem is given as follows:

The MSR problem
Given:
A set of potential pick-up points $C = \{c_1, c_2, ..., c_N\}$;
A probability set $P = \{P(c_1), P(c_2), ..., P(c_N)\}$;
A potential sequence set $R = \{\vec{r}_1, \vec{r}_2, ..., \vec{r}_M\}$;
The position c_0 of a cab which needs the service;
Objective: Recommending an optimal driving route $\vec{d} = \langle c_0, \vec{r} \rangle$, s.t.

$$\min_{\vec{r} \in \vec{R}} F(c_0, \vec{r}, P(\vec{r})).$$

An illustrative example is shown in Fig. 1, there are two different driving routes $\vec{d}_1 = \langle c_0, c_1, c_2 \rangle$ and $\vec{d}_2 = \langle c_0, c_2, c_3 \rangle$, both of length 2. If we set $D_{\max} = 10$, then $F(\vec{d}_1) = 5.55$ and $F(\vec{d}_2) = 5.4$. It can be seen that the PTD cost of $\vec{d}_2$ is smaller than that of $\vec{d}_1$ and then $\vec{d}_2$ should be preferentially recommended to the driver.

The computational complexity of the MSR problem is proved to be $O(N!)$ [2]. So a simple brute-force method is inefficient when tackling this problem.

2.2 Related Work

In recent years, intelligent transportation systems and trajectory data mining have attracted widespread attention [1], [6], [7], [8]. Mobile navigation and route recommendation have become a hot topic in this research field [2], [9], [10], [11], [12], [13], [14], [23], [24], [25], [26].

Recently, Ge et al. studied a novel problem of mobile sequential recommendation by using the historical trajectory data. The proposed MSR problem is rather different from the traditional problems such as shortest-path problem (SPP) [15], [16], traveling-salesman problem (TSP) [17] and vehicle-scheduling problem (VSP) [18]. In [2], the authors focused on the MSR problem with a limited length constraint due to the high computational complexity of the MSR problem. To reduce the search space, they proposed a route dominance based sequence pruning algorithm named LCP. A novel skyline based algorithm SkyRoute is also introduced to search for the optimal route which can service multiple cabs online. However, the proposed algorithms have difficulty in handling the problem with a large number of pick-up points.

A problem similar to MSR is shortest-path queries in spatial databases [19], [20], [22]. Funke and Storandt [19] studied the shortest paths with multi-criteria objectives based on contraction hierarchies [21], which makes the complexity of their algorithms reduce to polynomial time. However, the final objective needs to be translated to a linear combination of multiple criteria. For MSR problem, it is difficult to translate the PTD function to a linear combination of the distances and probabilities. In [20], the optimal route planning problem considering the energy consumption is studied, in which they adopted contraction hierarchies [21] to speed up the execution. A single-hop algorithm based on hub labeling using a technique called double-hub indexing has been proposed [22], which can be applied to the POI prediction scenario. However, the proposed techniques for shortest path are difficult to be directly adopted to solve the MSR problem.

Yuan et al. proposed a probabilistic model for detecting pick-up points [4]. It finds a route with the highest pick-up probability to the parking position constrained by a distance threshold instead of the minimal cost of the route and provides location recommendation service both for the cab drivers and for the passengers. In contrast, the problem solved in [24], [25] is different from the MSR problem which is to recommend a fastest route to a destination with a starting position and time constraints.

Powell et al. [3] proposed a grid-based approach to suggesting profit locations for taxi drivers by constructing a spatio-temporal profitability map, on which, the nearby regions of the driver are scored according to the potential profit calculated by the historical data. However, this method only finds a parking place with the largest profit in a local scope instead of a set of pick-up points with overall consideration. Wang [10] introduced a problem of finding the optimal trip route with time constraints. They also proposed an efficient trip planning method considering the current position of a user. The difference is that their method uses the score of attractions to measure the quality of a route.

3 Proposed Method

To address the computational challenges of the MSR problem, we first identify the iterative property of the PTD function. Then we propose the pruning principle which can effectively reduce the search space for MSR.

3.1 The Iterative Property of the PTD Function

As described in Section 2, the PTD function gives a computable measure for the cost of a route. Actually, an iterative computational formula of the PTD function can be obtained without considering the driving distance beyond the last pick-up point of a driving route. For this purpose, we first introduce the concept of PTD sub-function.

Definition 1 (PTD Sub-function $F1$). *Given a driving route $\vec{d} = \langle c_0, c_1, c_2, \ldots, c_L\rangle$, c_0 is its starting point and $\vec{r} = \langle c_1, c_2, \ldots, c_L\rangle$ is its pick-up sequence. Let the distance sub-vector of $D(\vec{d})$ be*

$$\widetilde{D(\vec{d})} = \left\langle lD_{c_0,c_1}, (D_{c_0,c_1} + D_{c_1,c_2}), \ldots, \sum_{i=1}^{L} D_{c_{i-1},c_i} \right\rangle$$

and the probability sub-vector of $P(\vec{d})$ be

$$\widetilde{P(\vec{d})} = \left\langle P(c_1), \overline{P(c_1)} \cdot P(c_2), \ldots, \prod_{i=1}^{L-1} \overline{P(c_i)} \cdot P(c_L) \right\rangle.$$

The PTD sub-function $F1$ of the driving route $\vec{d}$ is defined as

$$F1(\vec{d}) = \widetilde{D(\vec{d})} \cdot \widetilde{P(\vec{d})}. \tag{2}$$

Compared to the distance vector $D(\vec{d})$ and probability vector $P(\vec{d})$, the sub-vectors $\widetilde{D(\vec{d})}$ and $\widetilde{P(\vec{d})}$ of a driving route $\vec{d} = \langle c_0, \vec{r}\rangle$ both only lack the last component. Therefore, the PTD cost of a driving route $\vec{d}$ can be expressed using its PTD sub-function by the following equation:

$$F(\vec{d}) = F1(\vec{d}) + D_{\max} \cdot \prod_{i=1}^{L} \overline{P(c_i)}. \tag{3}$$

In fact, we do not have the starting point of a cab in the stage of offline processing. For enhancing the online search efficiency, we pre-compute the costs of all the potential sequences. The involved concept of probability summation function PE is introduced as follows.

Definition 2 (Probability summation function PE). *Let $\vec{r} = \langle c_1, c_2, \ldots, c_L\rangle$ be a potential sequence with length L and $\vec{d} = \langle c_0, \vec{r}\rangle$ be a driving route derived from $\vec{r}$. The probability summation of $\vec{r}$ is the sum of all the dimensions in the probability sub-vector $\widetilde{P(\vec{d})}$, and it is given as*

$$PE(\vec{r}) = P(c_1) + \overline{P(c_1)} \cdot P(c_2) + \cdots + \prod_{i=1}^{L-1} \overline{P(c_i)} \cdot P(c_L). \tag{4}$$

Since the sum of all the components in the probability vector $P(\vec{d})$ is equal to 1, the value of the probability summation function PE of $\vec{r}$ has the following property:

$$PE(\vec{r}) = PE(c_1, c_2, \ldots, c_L) = 1 - \prod_{i=1}^{L} \overline{P(c_i)}. \tag{5}$$

The value of the function PE can also be calculated recursively. Given a potential sequence $\vec{r}_1 = \langle c_1, c_2, \ldots, c_k\rangle$ and its postfix sub-sequence $\vec{r}_2 = \langle c_2, c_3, \ldots, c_k\rangle$, $PE(\vec{r}_1)$ can be iteratively calculated by

$$PE(\vec{r}_1) = P(c_1) + \overline{P(c_1)} \cdot PE(\vec{r}_2). \tag{6}$$

According to the above definitions, we can obtain the iterative computation theorem of the potential sequences.

Theorem 1. *Let $\vec{r} = \langle c_1, c_2, \ldots, c_L\rangle$ be a potential sequence with length L. The distance sub-vector of $\vec{r}$ is*

$$\widetilde{D(\vec{r})} = \left\langle D_{c_1,c_2}, (D_{c_1,c_2} + D_{c_2,c_3}), \ldots, \sum_{i=2}^{L} D_{c_{i-1},c_i} \right\rangle,$$

and its probability sub-vector is

$$\widetilde{P(\vec{r})} = \left\langle P(c_2), \overline{P(c_2)} \cdot P(c_3), \ldots, \prod_{i=2}^{L-1} \overline{P(c_i)} \cdot P(c_L) \right\rangle.$$

Then the PTD sub-function $F1$ of $\vec{r}$ is

$$F1(\vec{r}) = \widetilde{D(\vec{r})} \cdot \widetilde{P(\vec{r})}.$$

Given a potential sequence $\vec{r}_1 = \langle c_1, c_2, \ldots, c_k\rangle$ $(1 \le k \le N)$ and its postfix sub-sequence $\vec{r}_2 = \langle c_2, c_3, \ldots, c_k\rangle$, $F1(\vec{r}_1)$ can be iteratively calculated as

$$F1(\vec{r}_1) = \overline{P(c_2)} \cdot F1(\vec{r}_2) + D_{c_1,c_2} \cdot PE(\vec{r}_2). \tag{7}$$

Proof. Based on the definition of the PTD sub-function $F1$, we have

$$\begin{aligned}
F1(\vec{r}_1) &= \widetilde{D(\vec{r}_1)} \cdot \widetilde{P(\vec{r}_1)} \\
&= D_{c_1,c_2} \cdot P(c_2) + \left(D_{c_1,c_2} + D_{c_2,c_3}\right) \cdot \overline{P(c_2)} \cdot P(c_3) \\
&\quad + \cdots + \sum_{i=2}^{k} D_{c_{i-1},c_i} \cdot \prod_{i=2}^{k-1} \overline{P(c_i)} \cdot P(c_k) \\
&= D_{c_2,c_3} \cdot \overline{P(c_2)} \cdot P(c_3) + \cdots + \sum_{i=3}^{k} D_{c_{i-1},c_i} \cdot \prod_{i=2}^{k-1} \overline{P(c_i)} \cdot P(c_k) \\
&\quad + D_{c_1,c_2} \cdot \left(P(c_2) + \overline{P(c_2)} \cdot P(c_3) + \cdots + \prod_{i=2}^{k-1} \overline{P(c_i)} \cdot P(c_k) \right) \\
&= \overline{P(c_2)} \cdot \left(D_{c_2,c_3} \cdot P(c_3) + \cdots + \sum_{i=3}^{k} D_{c_{i-1},c_i} \cdot \prod_{i=3}^{k-1} \overline{P(c_i)} \cdot P(c_k) \right) \\
&\quad + D_{c_1,c_2} \cdot \left(P(c_2) + \overline{P(c_2)} \cdot P(c_3) + \cdots + \prod_{i=2}^{k-1} \overline{P(c_i)} \cdot P(c_k) \right) \\
&= \overline{P(c_2)} \cdot F1(\vec{r}_2) + D_{c_1,c_2} \cdot PE(\vec{r}_2).
\end{aligned}$$

□

According to Formulas (6) and (7), we can get the backward recursive formula for calculating the PTD sub-function $F1$ of the potential sequence as follows:

The initial value:
$\forall c \in C$, $F1(c) = 0$, $PE(c) = P(c)$

Iterative formula:

$$\begin{aligned}
F1(c_1, c_2, \ldots, c_L) &= \overline{P(c_2)} \cdot F1(c_2, c_3, \ldots, c_L) \\
&\quad + D_{c_1,c_2} \cdot PE(c_2, c_3, \ldots, c_L) \\
PE(c_1, c_2, \ldots, c_L) &= P(c_1) + \overline{P(c_1)} \cdot PE(c_2, \ldots, c_L)
\end{aligned}$$

In the stage of offline analysis, we only have the set of potential pick-up points C, but the locations of the cabs are unknown. Therefore, we can construct short postfix sequences and then incrementally add new pick-up

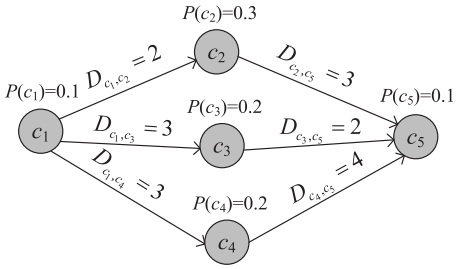


Fig. 2. An example of the sequence dominance and precedence.

points in front of them, and it will lead to longer potential sequences. Once we get the current location c_0 of a cab online, we can obtain the driving routes by inserting the current location of the cab c_0 to the head of the potential sequences as the starting point.

The PTD sub-function of driving route $\vec{d} = \langle c_0, \vec{r} \rangle$ can be calculated using the values of $F1(\vec{r})$ and $PE(\vec{r})$ via

$$F1(\vec{d}) = \overline{P(c_1)} \cdot F1(\vec{r}) + D_{c_0,c_1} \cdot PE(\vec{r}). \tag{8}$$

By combining Formula (3) with Formula (5), the PTD cost of the driving route $\vec{d}$ can be calculated using the formula

$$\begin{aligned} F(\vec{d}) &= \overline{P(c_1)} \cdot F1(\vec{r}) + \mathrm{D}_{c_0,c_1} \cdot PE(\vec{r}) + D_{\max} \cdot \prod_{i=1}^{L} \overline{P(c_i)} \\ &= \overline{P(c_1)} \cdot F1(\vec{r}) + D_{c_0,c_1} \cdot PE(\vec{r}) + D_{\max} \cdot (1 - PE(\vec{r})). \end{aligned} \tag{9}$$

For the driving route $\vec{d} = \langle c_0, c_1, c_2, \ldots, c_L \rangle$ with length L, we can efficiently calculate the values of the PTD sub-function $F1$ and the probability summation function PE of its pick-up sequence $\vec{r} = \langle c_1, c_2, \ldots, c_L \rangle$ in advance. When the current location c_0 of the cab is received online, we can calculate the PTD value of $\vec{d}$ based on Formula (9).

Based on the iterative property of the PTD function, we give a recursive calculation formula of the PTD function as well as an incremental backward path growth method, which avoid calculating the PTD value for each possible driving route from scratch. As a result, the calculation cost of the PTD values of the routes can be significantly reduced.

3.2 Sequence Pruning

In [2], Ge et al. proposed a sequence pruning algorithm LCP based on route dominance. The DP vector and the route dominance defined in [2] are introduced below.

Definition 3 (DP Vector). *Given a potential sequence $\vec{r} = \langle c_1, c_2, \ldots, c_L \rangle$ with length L, its DP vector is defined as a $2L-2$ vector $DP(\vec{r}) = \langle D_{c_1,c_2}, \overline{P(c_2)}, D_{c_2,c_3}, \overline{P(c_3)}, \ldots, D_{c_{L-1},c_L}, \overline{P(c_L)} \rangle$. Note that $DP_k (1 \le k \le 2L-2)$ denotes the value of the kth component of the vector DP.*

Definition 4 (Route dominance). *Given two potential sequences $\vec{r}_1$ and $\vec{r}_2$ with the same source and destination points as well as length L, $DP(\vec{r}_1)$ and $DP(\vec{r}_2)$ are two associated DP vectors. $\vec{r}_1$ dominates $\vec{r}_2$ iff $\exists k (1 \le k \le 2L-2)$, $DP_k(\vec{r}_1) < DP_k(\vec{r}_2)$ and $\forall k (1 \le k \le 2L-2), DP_k(\vec{r}_1) \le DP_k(\vec{r}_2)$.*

As shown in Fig. 2, two potential sequences $\vec{r}_1 = \langle c_1, c_2, c_5 \rangle$ and $\vec{r}_2 = \langle c_1, c_4, c_5 \rangle$ have the same source and destination pick-up points. The associated DP vectors are defined as $DP(\vec{r}_1) = \langle D_{c_1,c_2}, \overline{P(c_2)}, D_{c_2,c_5}, \overline{P(c_5)} \rangle$ and $DP(\vec{r}_2) = \langle D_{c_1,c_4}, \overline{P(c_4)}, D_{c_4,c_5}, \overline{P(c_5)} \rangle$. Because $(D_{c_1,c_2} \le D_{c_1,c_4}) \wedge (\overline{P(c_2)} \le \overline{P(c_4)}) \wedge (D_{c_2,c_5} \le D_{c_4,c_5}) \wedge (\overline{P(c_5)} \le \overline{P(c_5)})$ and $(D_{c_1,c_2} < D_{c_1,c_4}) \vee (\overline{P(c_2)} < \overline{P(c_4)}) \vee (D_{c_2,c_5} < D_{c_4,c_5}) \vee (\overline{P(c_5)} < \overline{P(c_5)})$ are both valid, we can infer that $\vec{r}_1$ dominates $\vec{r}_2$. Thus, $\vec{r}_2$ will be pruned in advance by the algorithm LCP.

In algorithm LCP, all possible potential sequences should be generated. Since a route being dominated by another route depends on the value of every dimension of the DP vector, the pruning effect is not significant. If we can identify and remove some non-optimal potential sequences incrementally in the stage of sequence generation, the pruning effect can be improved. Along this line, we introduce the sequence pruning principle adopted in our method.

Definition 5 (Sequence precedence). *Given two potential sequences $\vec{a} = \langle c_{a_1}, \ldots, c_{a_k} \rangle$ and $\vec{b} = \langle c_{b_1}, \ldots, c_{b_k} \rangle$ with equal length k ($1 \le k \le N$), for a starting position c_0, we will get two driving routes $\vec{d}_1 = \langle c_0, c_{a_1}, \ldots, c_{a_k} \rangle$ and $\vec{d}_2 = \langle c_0, c_{b_1}, \ldots, c_{b_k} \rangle$ with equal length k derived from $\vec{a}$ and $\vec{b}$ respectively. If $F(\vec{d}_1) < F(\vec{d}_2)$ holds for any possible c_0, then $\vec{a}$ precedes $\vec{b}$, denoted as $\vec{a} \prec \vec{b}$.*

If $\vec{a} \prec \vec{b}$, the potential sequence $\vec{b}$ cannot form an optimal driving route and it should be removed from the collection of the candidate sequences in advance. For example, as shown in Fig. 2, there is another potential sequence $\vec{r}_3 = \langle c_1, c_3, c_5 \rangle$. Since for any possible starting position c_0 the PTD value of the driving route $\vec{d}_1 = \langle c_0, c_1, c_2, c_5 \rangle$ must be smaller than that of the driving route $\vec{d}_3 = \langle c_0, c_1, c_3, c_5 \rangle$, $\vec{d}_3$ is not an optimal driving route. Therefore, we can prune the potential sequence $\vec{r}_3$ in the stage of offline processing.

Let $\vec{r}$ and $\vec{r}'$ be two potential sequences with the same source point and the equal length. It is easy to see that if $\vec{r}$ dominates $\vec{r}'$, then $\vec{r} \prec \vec{r}'$ is also valid. On the contrary, if $\vec{r} \prec \vec{r}'$, $\vec{r}$ does not necessarily dominate $\vec{r}'$. For example, as shown in Fig. 2, even though $\vec{r}_1$ does not dominate $\vec{r}_3$, $\vec{r}_1 \prec \vec{r}_3$ is still valid. It shows that sequence dominance is only a special case of sequence precedence.

In order to efficiently evaluate the costs of the potential sequences and prune some candidate sequences in the incremental process of sequence generation, we provide the iterative precedence as follows.

Definition 6 (Iterative precedence). *Let $\vec{a} = \langle c_{a_1}, \ldots, c_{a_k} \rangle$ and $\vec{b} = \langle c_{b_1}, \ldots, c_{b_k} \rangle$ be two potential sequences derived from the set of pick-up points C. If $(F1(\vec{a}) \le F1(\vec{b})) \wedge (1 - PE(\vec{a}) < 1 - PE(\vec{b}))$ or $(F1(\vec{a}) < F1(\vec{b})) \wedge (1 - PE(\vec{a}) \le 1 - PE(\vec{b}))$, then $\vec{a}$ takes iterative precedence over $\vec{b}$, denoted by $\vec{a} \propto \vec{b}$.*

Note that since the PTD function F and the iterative calculation of the PTD sub-function $F1$ are both relevant to the source point of the sequence, we only compare the PTD costs of the potential sequences with the same source point. The following theorem shows that we can determine the precedence relationship between pairs of potential sequences through iterative precedence.

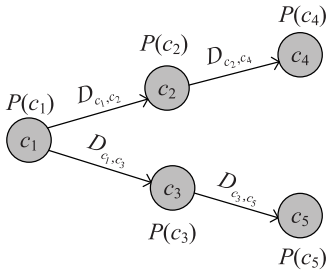


Fig. 3. An example of batch sequence pruning.

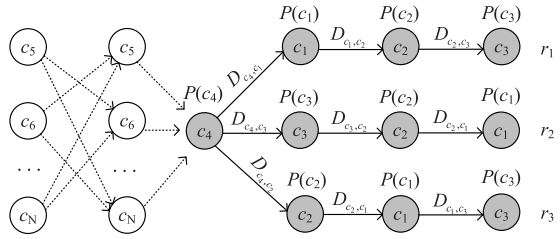


Fig. 4. An example of incremental sequence growing and pruning.

Theorem 2. *Let* $1 \le k \le N-1$, $\vec{a} = \langle c_s, c_{a_1} \ldots, c_{a_k}\rangle$ *and* $\vec{b} = \langle c_s, c_{b_1} \ldots, c_{b_k}\rangle$ *be two potential sequences with the same source point and the equal length* $k+1$. $\vec{a}' = \langle c_1, c_2, \ldots, c_m, c_s, c_{a_1}, \ldots, c_{a_k}\rangle$ *and* $\vec{b}' = \langle c_1, c_2, \ldots, c_m, c_s, c_{b_1}, \ldots, c_{b_k}\rangle$ *are two potential sequences derived from* $\vec{a}$ *and* $\vec{b}$ *by adding the same prefix sequence* $\vec{r} = \langle c_1, c_2, \ldots, c_m\rangle$ $(0 \le m \le N-k-1, c_m \in C)$ *respectively. If* $\vec{a} \propto \vec{b}$, *then* $\vec{a}' \prec \vec{b}'$.

Proof. Let c_0 be an arbitrary starting point, $\overrightarrow{d_a} = \langle c_0, c_1, c_2, \ldots, c_m, c_s, c_{a_1}, \ldots, c_{a_k}\rangle$ and $\overrightarrow{d_b} = \langle c_0, c_1, c_2, \ldots, c_m, c_s, c_{b_1}, \ldots, c_{b_k}\rangle$ be two driving routes associated with the potential sequences $\vec{a}'$ and $\vec{b}'$ respectively. $\vec{d} = \langle c_0, c_1, c_2, \ldots, c_m\rangle$ is a driving route with pick-up point sequence $\vec{r} = \langle c_1, c_2, \ldots, c_m\rangle$.

Let $D_0 = D_{c_0,c_1} + D_{c_1,c_2} + D_{c_2,c_3} + \cdots + D_{c_m,c_s}$ and $\overline{P_0} = \overline{P(c_1)} \cdot \overline{P(c_2)} \cdot \overline{P(c_3)} \ldots \overline{P(c_m)}$, then

$$F(\overrightarrow{d_a}) = F1(\vec{d}) + D_0 \cdot \overline{P_0} + (D_{\max} - D_0) \cdot \overline{P_0} \cdot \overline{P(c_s)} \cdot \overline{P(c_{a_1})} \cdot \overline{P(c_{a_2})} \cdot \cdots \cdot \overline{P(c_{a_k})} + F1(\vec{a}) \cdot \overline{P_0} \cdot \overline{P(c_s)},$$

$$F(\overrightarrow{d_b}) = F1(\vec{d}) + D_0 \cdot \overline{P_0} + (D_{\max} - D_0) \cdot \overline{P_0} \cdot \overline{P(c_s)} \cdot \overline{P(c_{b_1})} \cdot \overline{P(c_{b_2})} \cdot \cdots \cdot \overline{P(c_{b_k})} + F1(\vec{b}) \cdot \overline{P_0} \cdot \overline{P(c_s)}.$$

As we know,

$$\overline{P(c_s)} \cdot \overline{P(c_{a_1})} \cdot \overline{P(c_{a_2})} \cdot \cdots \cdot \overline{P(c_{a_k})} = 1 - PE(\vec{a}),$$

$$\overline{P(c_s)} \cdot \overline{P(c_{b_1})} \cdot \overline{P(c_{b2})} \cdot \cdots \cdot \overline{P(c_{b_k})} = 1 - PE(\vec{b}).$$

Then

$$F(\overrightarrow{d_a}) = F1(\vec{d}) + \mathrm{D}_0 \cdot \overline{P_0} + \overline{\mathrm{P}_0} \cdot (F1(\vec{a}) \cdot \overline{P}(c_s) + (D_{\max} - D_0) \cdot (1 - PE(\vec{a}))),$$

$$F(\overrightarrow{d_b}) = F1(\vec{d}) + \mathrm{D}_0 \cdot \overline{P_0} + \overline{P_0} \cdot (F1(\vec{b}) \cdot \overline{P(c_s)} + (D_{\max} - D_0) \cdot (1 - PE(\vec{b}))).$$

So,

$$F(\overrightarrow{d_a}) - F(\overrightarrow{d_b}) = \overline{P_0} \cdot \overline{P(c_s)} \cdot (F1(\vec{a}) - F1(\vec{b})) + \overline{P_0} \cdot (D_{\max} - D_0) \cdot (PE(\vec{b}) - PE(\vec{a}))).$$

Actually, we do not know the desired maximum cruising distance of a driver at the offline stage. In order to obtain the potential sequences of all possible lengths, we set $D_{\max}$ to be a value large enough here which satisfies $D_{\max} > D_0$ in all the cases. Thus, if $\vec{a} \propto \vec{b}$, i.e., $(F1(\vec{a}) \le F1(\vec{b})) \wedge (1 - PE\ (\vec{a}) < 1 - PE(\vec{b}))$ or $(F1(\vec{a}) < F1(\vec{b})) \wedge (1 - PE(\vec{a}) \le 1 - PE(\vec{b}))$, then $F(\overrightarrow{d_a}) < F(\overrightarrow{d_b})$. That is to say, $\vec{a}' \prec \vec{b}'$. □

According to the feature of the sequence precedence, we introduce the theory of batch and incremental sequence pruning as follows.

Corollary 1 (Batch pruning). *Given two potential sequences* $\vec{a} = \langle c_s, c_{a_1}, \ldots, c_{a_k}\rangle$ *and* $\vec{b} = \langle c_s, c_{b_1}, \ldots, c_{b_k}\rangle$ *with equal length* $k+1 (1 \le k \le N-1)$ *and the same source point* c_s, *if* $\vec{a} \propto \vec{b}$, *then* $\vec{a} \prec \vec{b}$.

The above corollary shows that if $\vec{a} \propto \vec{b}$, the driving route $\langle c_0, \vec{b}\rangle$ derived from the potential sequence $\vec{b}$ is not an optimal one. Thus, $\vec{b}$ should be pruned from the candidate sequences with length $k+1$.

In the batch pruning, we can compare the PTD cost among potential sequences with length $L (2 \le L \le N)$ using the values of $F1$ and PE calculated in the iterative process. However, the batch pruning cannot be applied during the process of incremental sequence generation. For example, as shown in Fig. 3, there are two potential sequences $\vec{r} = \langle c_1, c_2, c_4\rangle$ and $\vec{r}' = \langle c_1, c_3, c_5\rangle$. We can generate a longer potential sequence $\langle c_3, c_1, c_2, c_4\rangle$ taking $\vec{r}$ as its postfix. However, we cannot add c_3 before $\vec{r}'$, because the pick-up point c_3 has existed in $\vec{r}'$. Even though $\vec{r} \propto \vec{r}'$, we cannot prune $\vec{r}'$ in advance in the process of the incremental backward path growth. For example, if $\langle c_2, c_1, c_3, c_5\rangle \prec \langle c_3, c_1, c_2, c_4\rangle$, we may miss the optimal route due to the improper pruning of $\vec{r}'$ in advance. Therefore, if we can add the same prefix for all the potential sequences with the same source point and length, then it is suitable for incrementally pruning potential sequences. Along this line, we proposed a new corollary suitable for pruning potential sequences incrementally.

Corollary 2 (Incremental pruning). *Given two potential sequences* $\vec{a}$ *and* $\vec{b}$ *with the same source point and the same set of pick-up points. If* $F1(\vec{a}) < F1(\vec{b})$, *then none of the driving routes having the postfix sub-sequence* $\vec{b}$ *can be an optimal driving route.*

Let us study the example shown in Fig. 4. There are three sequences with length 4: $\vec{r}_1 = \langle c_4, c_1, c_2, c_3\rangle$, $\vec{r}_2 = \langle c_4, c_3, c_2, c_1\rangle$ and $\vec{r}_3 = \langle c_4, c_2, c_1, c_3\rangle$. For any pick-up point $c \in C \setminus \{c_1, c_2, c_3, c_4\}$, it can be added before the three sequences to construct three new potential sequences with length 5. If $F1(\vec{r}_1) < F1(\vec{r}_2) < F1(\vec{r}_3)$, then $\vec{r}_1 \propto \vec{r}_2 \propto \vec{r}_3$, i.e., $\vec{r}_1 \prec \vec{r}_2 \prec \vec{r}_3$. That is to say, $\vec{r}_2$ and $\vec{r}_3$ can be pruned in advance. Because any possible driving

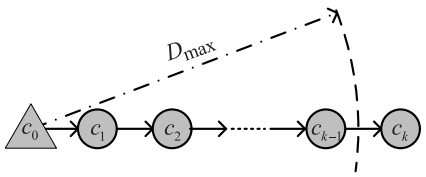


Fig. 5. An example of a driving route and the desired maximal cruising distance $D_{\max}$.

routes with a postfix sequence of the pruned sequence are not the optimal routes, they can be removed incrementally. However, $\vec{r}_1$ remains as a candidate sequence with length 4 and it is considered as the possible postfix of other longer potential sequences.

3.3 The Analysis of Pruning Effect

In this section, we analyze the pruning ratio of our incremental and batch pruning methods. Let the total number of potential sequences be M and the number of the remaining sequences after pruning be M', the pruning ratio $\eta = (M - M')/M$.

Given a set of potential pick-up points C with $|C| = N$, the number of the potential sequences with length L $(3 \leq L \leq N)$ is $M = \binom{N}{L} \cdot L!$. In the process of incremental pruning, we only consider a group of potential sequences with the same source point and the same set of pick-up points. Based on Corollary 2, the precedence relationships of these sequences are only related to the $F1$ values of them. Actually, in most cases we choose an optimal sequence with the minimum $F1$ value from all these potential sequences. Since the number of the permutations of $L-1$ pick-up points except for the same source point is $(L-1)!$, the number of the remaining sequences $M' = \binom{N}{L}.L!/(L-1)! = \binom{N}{L}.L$. Then the pruning percentage is $\eta = (M - M')/M = 1 - (\binom{N}{L}.L)/(\binom{N}{L}.L!) = 1 - 1/(L-1)!$

In summary, for all possible potential sequences with length L $(3 \leq L \leq N)$, the incremental pruning ratio η is equal to $1 - 1/(L-1)!$.

Note that the incremental pruning method is only applied to deal with the potential sequences with the length $L \geq 3$. According to the above analysis, the incremental pruning ratio sharply increases along with the increase of the sequence lengths.

In order to remove more non-optimal sequences, we need to use the batch pruning method on the remaining sequences after the incremental pruning process. In batch pruning, we compare the precedence relationship among the remaining potential sequences with the same source point $c \in C$ and the same length L. As we know, whether a potential sequence is removed by the batch pruning method is related to the values of $F1$ and PE. However, it is hard to precisely calculate the pruning ratio of batch pruning.

3.4 The Distance Constraint Parameter $D_{\max}$

In the sequence generation and pruning process above, we set $D_{\max}$ to be a value large enough for producing all the candidate sequences in arbitrary lengths. However, in real applications the driver may plan to pick up new passengers within a given distance. In this section, we show that the parameter $D_{\max}$ can be specified online to achieve this purpose.

At the online stage, assume that the user specify a reasonable desired maximal cruising distance $D_{\max}$, as shown in Fig. 5. Let $\vec{d}_1 = \langle c_0, c_1, \ldots, c_{k-1} \rangle$ be a driving route with length $k-1$ and $\vec{d}_2 = \langle c_0, c_1, \ldots, c_{k-1}, c_k \rangle$ be its super-route by adding another new pick-up point c_k. According to Formula (9), we get $F(\vec{d}_1) = F1(\vec{d}_1) + D_{\max} \cdot \prod_{i=1}^{k-1} \overline{P(c_i)}$ and $F(\vec{d}_2) = F1(\vec{d}_2) + D_{\max} \cdot \prod_{i=1}^{k} \overline{P(c_i)} = F1(\vec{d}_1) + \sum_{i=1}^{k} D_{c_{i-1},c_i} \cdot \prod_{i=2}^{k-1} \overline{P(c_i)} \cdot P(c_k) + D_{\max} \cdot \prod_{i=1}^{k} \overline{P(c_i)}$. Then

$$F(\vec{d}_2) - F(\vec{d}_1) = \prod_{i=1}^{k-1} \overline{P(c_i)} \cdot P(c_k) \cdot \left(\sum_{i=1}^{k} D_{c_{i-1},c_i} - D_{\max} \right). \tag{10}$$

Let $DL(\vec{d}_2) = \sum_{i=1}^{k} D_{c_{i-1},c_i}$ be the cruising distance of route $\vec{d}_2$. We can see that if $DL(\vec{d}_2) < D_{\max}$, then $F(\vec{d}_2) < F(\vec{d}_1)$. That is to say, the PTD value of a driving route will monotonously decrease with its length when its cruising distance is less than the value of $D_{\max}$. On the contrary, if $DL(\vec{d}_2) > D_{\max}$, then $F(\vec{d}_2) > F(\vec{d}_1)$. It means that the PTD value of a driving route will increase with its length when its cruising distance is larger than the value of $D_{\max}$. Based on the above analysis, we can infer that a driving route $\vec{d} = \{c_0, c_1, \ldots, c_k\}$ cannot be an optimal driving route if $DL(\vec{d}) > D_{\max}$. Actually, we can consider $D_{\max}$ as a distance constraint parameter which can be specified by the user. Therefore, when searching the optimal driving route online, we only consider the driving routes with cruising distance no more than $D_{\max}$.

3.5 Recommendation with Destination Constraint

In real life, a taxi driver may have a targeted destination in mind when cruising. To meet this business requirement, in this section, we discuss how to deal with the MSR problem with the destination constraint. We only consider the case that the destination is one pick-up point in C. The process of generating candidate sequences is a little difference from the process described above. Actually, for each $c_i \in C$, we can produce a set of potential sequences with destination point c_i respectively by using the method proposed above. However, the cost comparison and pruning are performed among the potential sequences with the same source point and the same destination point. When the destination c_d is specified by a driver online, we only need to search for the optimal route from the set of remaining candidate sequences with the destination c_d.

4 The Algorithm

Based on the analysis above, we first present the offline algorithm generating the potential candidate sequences and the online route query algorithm. Afterwards, we analyze the time and space complexity of our algorithms.

4.1 The Offline Processing Algorithms

The detail of our dynamic programming based algorithm, called BP-Growth, is given in Algorithm 1. It generates the potential candidate sequences in the offline stage when the

position of a cab is not involved. In order to construct all possible potential candidate sequences incrementally and efficiently, a backward path growth procedure and an incremental sequence pruning process are employed which contain the iterative calculation of the $F1$ and PE values of the potential sequences.

Algorithm 1 BP-Growth

Input: A set of potential pick-up points C, the probability set P for all pick-up points, the pairwise driving distance matrix D of pick-up points.
Output: A set of potential sequences $\overrightarrow{R}$ with length L ranging from 1 to N

$\overrightarrow{R^1} \leftarrow \emptyset$;
for each $c_i \in C$ **do**
  $\vec{r} \leftarrow \langle c_i \rangle$; $F1(\vec{r}) \leftarrow 0$; $PE(\vec{r}) \leftarrow P(c_i)$; $\overrightarrow{R^1} \leftarrow \overrightarrow{R^1} \cup \{\vec{r}\}$;
end for
for $L = 2$ to N **do**
  $\overrightarrow{R^L} \leftarrow \emptyset$;
  for each $\vec{r} \in \overrightarrow{R^{L-1}}$ **do**
    for each $c_i \in (C - C_{\vec{r}})$ **do**
      //Potential Sequence Generation
      $\vec{p} \leftarrow \langle c_i, \vec{r} \rangle$; $c \leftarrow s(\vec{r})$;
      $F1(\vec{p}) \leftarrow F1(\vec{r}) \cdot \overline{P(c)} + D_{c_i,c} PE(\vec{r})$;
      $PE(\vec{p}) \leftarrow PE(\vec{r}) \cdot \overline{P(c_i)} + P(c_i)$;
      //Incremental Sequence Pruning
      $\overrightarrow{R^L_{\vec{p}}} = \{\vec{q} | \vec{q} \in \overrightarrow{R^L}, s(\vec{q}) = s(\vec{p}), C_{\vec{q}} = C_{\vec{p}}\}$;
      if $\overrightarrow{R^L_{\vec{p}}} = \emptyset$ **then**
        $\overrightarrow{R^L} \leftarrow \overrightarrow{R^L} \cup \{\vec{p}\}$;
      else
        if $\forall \vec{q} \in \overrightarrow{R^L_{\vec{p}}} (\mathrm{F1}(\vec{p}) = F1(\vec{q}))$ **then**
          $\overrightarrow{R^L} \leftarrow \overrightarrow{R^L} \cup \{\vec{p}\}$;
        else
          if $\forall \vec{q} \in \overrightarrow{R^L_{\vec{p}}} (F1(\vec{p}) < F1(\vec{q}))$ **then**
            $\overrightarrow{R^L} \leftarrow \left(\overrightarrow{R^L} - \overrightarrow{R^L_{\vec{p}}}\right) \cup \{\vec{p}\}$;
          end if
        end if
      end if
    end for
  end for
end for
return $\overrightarrow{R} = \bigcup_{L=1}^{N} \overrightarrow{R^L}$;

Algorithm 2 BatchPruning

Input: A set of the potential sequences $\overrightarrow{R^L}$ with length L.
Output: A set of the remaining candidate sequences $\overrightarrow{R'^L}$ with length L.

for each $c \in C$ **do**
  $\overrightarrow{R^L_c} \leftarrow \emptyset$;
end for
for each $\vec{r} \in \overrightarrow{R^L}$ **do**
  $c \leftarrow s(\vec{r})$;
  $\overrightarrow{R^L_c} \leftarrow \overrightarrow{R^L_c} \cup \left\{\overrightarrow{r}\right\}$;
  for each $\overrightarrow{q} \in \overrightarrow{R^L_c} \wedge \overrightarrow{r} \neq \overrightarrow{q}$ **do**
    if $\vec{q} \propto \vec{r}$ **then**
      $\overrightarrow{R^L_c} \leftarrow \overrightarrow{R^L_c} - \left\{\overrightarrow{r}\right\}$;
      break;
    else
      if $\vec{r} \propto \vec{q}$ **then**
        $\overrightarrow{R^L_c} \leftarrow \overrightarrow{R^L_c} - \left\{\overrightarrow{q}\right\}$;
      end if
    end if
  end for
end for
return $\overrightarrow{R'^L} = \bigcup_{c \in C} \overrightarrow{R^L_c}$;

We will obtain a set of candidate sequences with length ranging from 1 to N in Algorithm 1. For the generated potential candidate sequences, we adopt the batch pruning algorithm to further reduce the number of candidate sequences. As we know, after the candidate sequences are produced offline, the $F1$ and PE values of these sequences have also been calculated iteratively. Therefore, we can directly compare the $F1$ and PE values of the potential candidate sequences to prune the non-optimal ones during the batch pruning process described in Algorithm 2.

4.2 The Online Search Algorithm

Our method is able to provide real-time driving route recommendation services for the empty cabs at various positions. When a cab at the position c_0 requests the recommendation service, an online search algorithm, called RouteOnline, is adopted to find an optimal driving route from the remaining potential sequences generated in the offline stage. Algorithm 3 shows the online search procedure of optimal routes in detail.

Algorithm 3 RouteOnline

Input: : A set of the candidate sequences $\overrightarrow{R}$, the current position of a cab c_0, the desired maximum cruising distance $D_{\max}$.
Output: : A set of the optimal driving routes $\overrightarrow{D_{op}}$.

$\overrightarrow{D_{op}} \leftarrow \emptyset$; $F_{\min} \leftarrow +\infty$;
for $L = 1$ to N **do**
  for each $\vec{r} \in \overrightarrow{R^L}$ **do**
    $c = s(\vec{r})$;
    $\vec{d} = \langle c_0, \vec{r} \rangle$;
    if $DL(\vec{d}) \leq D_{\max}$ **then**
      $F(\vec{d}) = F1(\vec{r}) \cdot (1 - P(c)) + D_{c_0,c} \cdot PE(\vec{r}) + D_{\max} \cdot (1 - PE(\vec{r}))$;
      if $\overrightarrow{D_{op}} = \emptyset \vee F(\vec{d}) = F_{\min}$ **then**
        $\overrightarrow{D_{op}} \leftarrow \overrightarrow{D_{op}} \cup \{\vec{d}\}$;
      else
        if $F(\vec{d}) < F_{\min}$ **then**
          $\overrightarrow{D_{op}} \leftarrow \{\vec{d}\}$; $F_{\min} \leftarrow F(\vec{d})$;
        end if
      end if
    end if
  end for
end for
return $\overrightarrow{D_{op}}$;

In Algorithm 3, for each $L (1 \leq L \leq N)$, we first generate the potential driving routes $\overrightarrow{D^L}$ of length L by connecting c_0 with each potential sequence candidate in the set $\overrightarrow{R^L}$. Then for each driving route with cruising distance no more than $D_{\max}$, we calculate their PTD values using Formula (9). Finally, the routes with the minimal PTD value are selected and returned to the users.

4.3 Time and Space Complexity Analysis

In this section, we analyze the time and space complexities of the proposed algorithms.

4.3.1 Computational Complexity Analysis

We first analyze the computational complexity of our offline algorithm BP-Growth. The key step in algorithm BP-Growth is the incremental process of the sequence growing and pruning. As we know, in order to generate the potential sequences with length L, we add each pick-up point c before the candidate sequences with length $L-1$ that do not contain c. When the length of the potential sequence $L = 1$, all pick-up points will be enumerated, so the computational complexity is N. When $L = 2$, as we know, the number of candidate sequences with length 1 is N. Since each pick-up point only appears once in a potential

sequence, we still have $N-1$ possible pick-up points for each sequence candidate. Therefore, the loop execution time of the key step for $L=2$ is $N(N-1)$. When we generate the potential sequences with length $L>2$, the number of the remaining candidate sequences with length $L-1$ after the incremental pruning process is $\binom{N}{L-1}\cdot (L-1)!(L-2)!$. Nevertheless, we still have $N-L+1$ pick-up points to be added at the beginning of these candidate sequences, and the computational time is $(\binom{N}{L-1}\cdot (L-1)!/(L-2)!)\cdot (N-L+1)=\binom{N}{L}\cdot L\cdot (L-1)$. It can be seen that the time complexity of the process with length $L=1$ is $O(N)$. It increases gradually and reaches the peak when $L=[N/2]$. After that, the computational complexity decreases. It drops to $O(N^2)$ when $L=N$.

We then present the computational complexity analysis of our algorithm BP-Growth for generating all possible sequences of length varying from 1 to N.

Given a set of pick-up points C with $|C|=N$, as we know, the total execution time for generating all the potential sequences with length $L\leq N$ are

$$f(N)=N+N\cdot (N-1)+\sum_{L=3}^{N}(L-1)\cdot L\binom{N}{L}.$$

$f(N)$ can be transformed to

$$f(N)=N+2\binom{N}{2}+2\cdot 3\binom{N}{3}+\cdots +(N-2)\cdot (N-1)\binom{N}{N-1}+(N-1)\cdot N\binom{N}{N}.$$

Since $L\cdot \binom{N}{L}=(N-L+1)\binom{N}{L-1}$, then $f(N)$ can also be calculated from the following equation:

$$f(N)=N+(N-1)\binom{N}{1}+2(N-2)\binom{N}{2}+3(N-3)\binom{N}{3}+\cdots +(N-1)\binom{N}{N-1}.$$

If we add above two equations, we will obtain the following deduction:

$$\begin{aligned}2f(N)&=2N+(N-1)\binom{N}{1}+2(N-1)\binom{N}{2}+3(N-1)\binom{N}{3}+\cdots +N(N-1)\binom{N}{N}\\&=2N+(N-1)\left(\binom{N}{1}+2\binom{N}{2}+3\binom{N}{3}+\cdots +N\binom{N}{N}\right)\\&=2N+N(N-1)\cdot 2^{N-1}.\end{aligned}$$

Then $f(N)=N+N(N-1)\cdot 2^{N-2}$. Thus, $O(f(N))=O(N^2\cdot 2^N)$.

In summary, the computational complexity of incremental generation of the potential sequences with all possible lengths $L(1\leq L\leq N)$ via BP-Growth is $O(N^2\cdot 2^N)$. As for the time complexity of our online search algorithm, it directly depends on the number of the remaining candidate sequences and is equal to the space complexity which will be given a detailed analysis in the following.

TABLE 2
Some Acronyms Used in Experimental Analysis

LCP	Sequence pruning via route dominance
SkyRoute	Sequence pruning via skyline query
SR(BNL)	SkyRoute with skyline computing method BNL
SR(D&C)	SkyRoute with skyline computing method D&C
IP	Generation of potential candidate sequences via BP-Growth with incremental pruning
IBP	Generation of potential candidate sequences via BP-Growth with Incremental and Batch Pruning
BFS	Brute-force search
LCPS	Search via LCP
SR(BNL)S	Skyline search via the algorithm SkyRoute + BNL
SR(D&C)S	Skyline search via the algorithm SkyRoute + D&C
IPS	Search on the potential sequences generated by IP
IBPS	Search on the potential sequences generated by IBP

4.3.2 Space Complexity Analysis

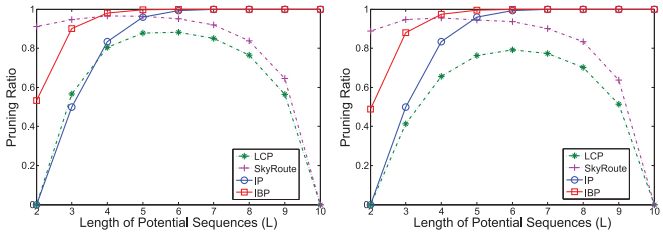
If we use the internal memory of a single PC to store and process the data, the generated candidate sequences are stored in RAM. Thus, the space complexity of our algorithm is directly determined by the number of the remaining candidate sequences.

For algorithm IP, as we know, the remaining sequences with length L is $\binom{N}{L}\cdot L$. However, it is hard to precisely evaluate the number of the remaining candidate sequences with length L generated after using batch pruning. Since the batch pruning is performed after the process of incremental pruning, the upper bound is $\binom{N}{L}\cdot L$ when none of candidate sequences precedes the others. The lower bound is N when only a candidate sequence is chosen from the candidate sequences with the same source point and the same length. In short, the range of the number of the remaining candidate sequences with length L pruned by algorithm BatchPruning is from N to $\binom{N}{L}\cdot L$.

As for the remaining candidate sequences without length constraint, it is the sum of all the remaining candidate sequences. Therefore, the set of the candidate sequences $\overrightarrow{R}$ produced by algorithm BP-Growth with incremental pruning equals $\sum_{L=1}^{N}\binom{N}{L}\cdot L$. Therefore, the overall space complexity of IP is $O(N\cdot 2^N)$. And the space complexity of IBP is from $O(N^2)$ to $O(N\cdot 2^N)$.

5 Experimental Evaluations

In this section, we evaluate the performance of our method by comparing its pruning effects, memory consumption and online search time with other state-of-the-art methods. All acronyms of evaluated algorithms are given in Table 2. LCP and SkyRoute are two route dominance based pruning algorithms proposed in [2]. In particular, SkyRoute is an online pruning algorithm, where two skyline computing methods BNL and D&C can be applied to prune potential sequences [5]. Its corresponding online search methods are denoted as SR(BNL)S and SR(D&C)S, respectively. All the algorithms in Table 2 were implemented in Visual C++ 6.0. The experiments were conducted on a PC with a Intel Pentium Dual E2180 processor and 4 GB RAM.



(a) Real-World Data (b) Synthetic Data

Fig. 6. The pruning ratio of different algorithms w.r.t. the length of potential sequences on the data sets with $|C| = 10$.

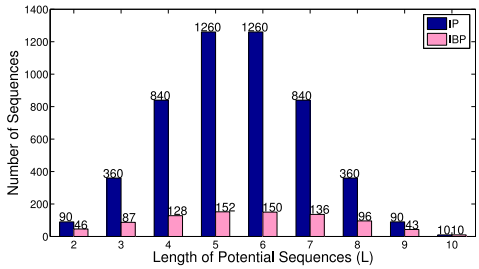


Fig. 7. The number of remaining candidate sequences after using the pruning algorithms IP and IBP respectively on the synthetic data set with $|C| = 10$.

5.1 Data Sets

The adopted experimental data sets are divided into two categories: real-world data and synthetic data.

Real-world data. In the experiments, we adopt real-world cab mobility traces used in [2], which are provided by Exploratorium—the museum of science, art and human perception. It contains GPS location traces of 514 taxis collected around 30 days in San Francisco Bay Area. 21,980 and 38,280 historical pick-up locations of all the taxi drivers are extracted on two time periods: 2PM-3PM and 6PM-7PM. We obtain 10 and 25 clusters respectively in these two data sets by using a clustering tool Cluto [29]. Each cluster has a spatial coverage. Let T denote the number of cabs which have no passenger before passing cluster c_i. For these T empty cabs, the number of pick-up events U is counted in this cluster. Then the successful probability of c_i can be estimated as $P(c_i) = \frac{U}{T}$ [2].

Synthetic data. We also generate four synthetic data sets. Specifically, potential pick-up points and their pick-up probabilities are randomly generated within a special area by a standard uniform distribution. In total, four synthetic data sets with 10, 15, 20 and 25 pick-up points are produced respectively. The euclidean distance instead of the driving distance is adopted to measure the distances between pairs of pick-up points.

For both real and synthetic data, we randomly generate the positions of the target cabs for recommendation.

5.2 The Overall Comparison of Pruning Effects

As we know, algorithms BFS and LCP need to enumerate all possible sequences of a certain length L. When the number of pick-up points N or the length of suggested route L is a little larger (e.g., $N = 20$ and $L = 6$), both BFS and LCPS cannot finish the enumeration process within a reasonable period of time. Therefore, when we analyze the variations of pruning ratio with the length of suggested driving routes on the same set of pick-up points, we make the number of pick-up points small (e.g., $|C| = 10$) in order to show the overall comparison of all concerned algorithms. When we analyze the pruning ratio with the number of pick-up points on the fixed length of driving routes, we also make the length of the routes small (e.g., $L = 3$ and $L = 5$). For the algorithms proposed in this paper, since the sequences are pruned incrementally, both the time and space complexity are better than that of BFS and LCP. Thus, we can use the synthetic data set with $|C| = 25$ to analyze the pruning effect of the proposed incremental algorithm BP-Growth in detail.

5.2.1 *The Pruning Ratio Varying with the Length of Potential Sequence*

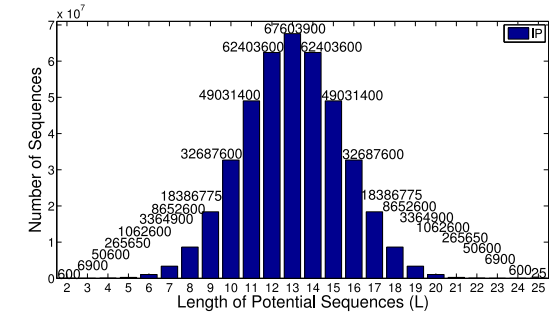
Fig. 6 shows the variations of pruning ratio of several algorithms with the length of potential sequence on both real-world and synthetic data with $|C| = 10$. Algorithms LCP, SkyRoute, IP and IBP are all able to prune some non-optimal sequences derived from C. When the length $L = 2$ or $L = 3$, the proposed algorithms IBP and IP perform worse than the algorithms SkyRoute and LCP respectively. However, as the length of the potential sequence L increases, the pruning ratios of our algorithms IP and IBP both significantly increase. It can be observed that IBP outperforms SkyRoute and IP outperforms LCP on both real and synthetic data when $L \geq 5$. Furthermore, the pruning ratio of our algorithms gradually increases and close to 1 when the length of suggested driving route $L \geq 6$. In contrast, the change of the pruning ratios of LCP shows a trend of parabola. When $L > 5$, the pruning ratios of LCP and SkyRoute both gradually drop. When the length is equal to the number of pick-up points (i.e., $L = |C|$), the pruning ratios of them decrease to 0.

To verify that our method can process the potential sequences derived from a larger number of pick-up points, we test the pruning ratios of IP and IBP on the synthetic data set with $|C| = 25$. We find that the trends of the pruning ratios of our algorithms on different data sets are consistent. Since LCP and SkyRoute are only able to deal with the driving routes with $L \leq 5$ on the data set with $|C| = 25$, we cannot obtain the complete result of them on this data set.

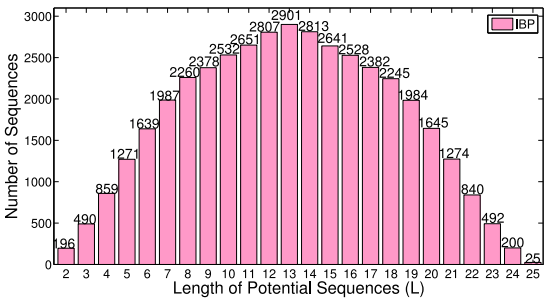
Let us analyze the reason why the pruning ratios of algorithms IP and IBP are relatively high. First, for the incremental pruning algorithm IP, its pruning ratio is equal to $1 - 1/(L-1)!$ which dramatically increases along with the increase of the length of potential sequences. When $L = 6$, the pruning percentage has reached 99.2 percent.

The algorithm IBP employs an additional batch pruning process to remove some non-optimal potential sequences. As shown in Figs. 7 and 8, the number of remaining sequences of IBP can be several orders of magnitude smaller than that of IP, especially when $|C|$ is large, which demonstrates the effectiveness of batch pruning. The overall trend of the number of the remaining candidate sequences exhibits a Gaussian distribution. It first increases with the increase of the length L, and then decreases when $L \geq |C|/2$.

In term of LCP, whether a route is dominated by another route depends on the value of each dimension of the vector DP. When the value of L is small, the pruning ratio has



(a)



(b)

Fig. 8. The number of remaining candidate sequences after using the pruning algorithms IP and IBP respectively on the synthetic data set with $|C| = 25$.

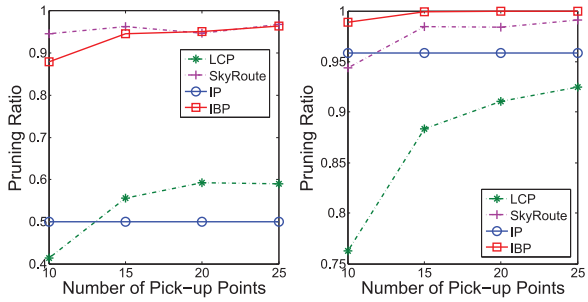
some growth with the increase of the length. However, when the sequence length becomes larger, the number of the dimensions of vector DP increases and the probability of domination in each dimension between DP vectors becomes smaller, which leads to the gradual decline of the pruning ratio. When $L = |C|$, since all pick-up points are involved, it is impossible that the probability of each pick-up point in a sequence is larger than that of the other. Thus, the pruning ratio is 0 in this case. As for SkyRoute, since the principle of it is similar to that of LCP, the overall trends of them are almost the same.

5.2.2 The Pruning Ratio Varying with the Number of Pick-Up Points

Fig. 9 shows the variations of pruning ratio with respect to the number of pick-up points on synthetic data with $|C| = 25$ when $L = 3$ and $L = 5$ respectively. It can be observed that the pruning ratios of LCP and IBP increase with the increase of the number of pick-up points, and the pruning ratio of IP is constant. When $L = 3$, the pruning ratio of IP is equal to 0.5 and the pruning ratio of IBP gradually increases with the number of pick-up points. In fact, our algorithms do not perform better than algorithms LCP and SkyRoute in this case. However, when $L = 5$, the pruning ratio of IP is more than 0.95 and the pruning ratio of IBP is close to 1 which are much higher than those of LCP and SkyRoute respectively.

5.3 The Comparison of the Memory Consumption

Fig. 10 shows the variations of consumed internal memory storage with the length of potential sequence on synthetic data sets with $|C| = 10$ and $|C| = 20$ respectively. For



(a) $L = 3$ (b) $L = 5$

Fig. 9. The pruning ratios varying with the number of pick-up points on the synthetic data with $|C| = 25$.

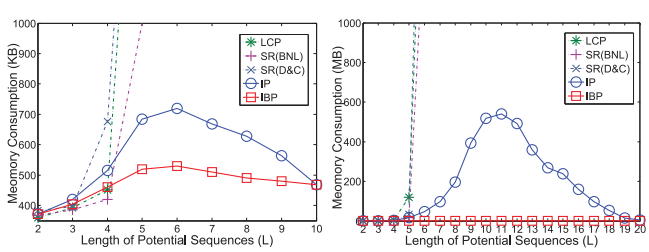
algorithms BFS, LCP and SkyRoute, it is necessary to enumerate all possible sequences of a certain length, so the internal memory consumption is huge. As the length of potential sequence L increases, the memory consumption of LCP, SR(BNL), and SR(D&C) dramatically increase. As shown in Fig. 10a, they can only deal with the potential sequences with length $L \leq 5$ on the data set with $|C| = 10$. The space cost of SR(D&C) increases at the fastest pace due to its recursive calculation process.

It can be observed that the space performances of our algorithm IP are better than those of BFS and LCP, because it uses an incremental method to generate the potential sequences. When the length $L \leq 4$, the memory cost of IP is a little bit higher than those of the other three algorithms due to the storage of some iterative calculation values, such as $F1$ and PE. Our algorithm IP presents an overall trend of parabola, which reaches the peak of 700K and 600M RAM on these two data sets respectively. Nevertheless, when the number of pick-up points is larger (e.g., $|C| = 25$), the generation process of IP cannot be performed in the internal memory of a PC with 4 G RAM. In this case, we have to utilize external memory to store the generated potential sequences. For algorithm IBP, since the number of the remaining candidate sequences generated by IBP is far less than that generated by IP, its space consumption has been reduced further.

5.4 The Comparison of Online Search Time

In this section, we compare the efficiency of various online route search algorithms. Note that all the search time is the average of 10 running tests.

Tables 3 and 4 show the online search time consumed by algorithms BFS, LCPS, SR(D&C)S, SR(BNL)S, IPS and



(a) $|C| = 10$ (b) $|C| = 20$

Fig. 10. The internal memory consumption varying with the length of suggested driving routes on the synthetic data sets with $|C| = 10$ and $|C| = 20$.

TABLE 3
A Comparison of Search Time (Millisecond) on the Real-World Data Set (2-3PM)

	$L=2$	$L=3$	$L=4$	$L=5$	All
BFS	0.008	0.029	0.198	1.434	763.246
LCPS	0.007 (1.14)	0.016 (1.81)	0.037 (5.35)	0.118 (12.15)	487.899 (1.56)
SR(D&C)S	10.327 (0.0)	41.679 (0.0)	182.052 (0.0)	1520.310 (0.0)	- (-)
SR(BNL)S	1.931 (0.0)	24.354 (0.0)	139.060 (0.00)	2333.120 (0.00)	- (-)
IPS	0.007 (1.14)	0.011 (2.64)	0.017 (11.65)	0.025 (57.36)	0.132 (5782.17)
IBPS	0.007 (1.14)	0.007 (4.14)	0.008 (24.75)	0.009 (159.33)	0.060 (12720.80)

The speedup values compared to BFS are given in the round brackets.

IBPS on real-world and synthetic data sets with various numbers of pick-up points and lengths of suggested driving routes. In particular, the numbers inside the round brackets represent the speedups compared to BFS. It can be observed that the search time consumed by our algorithm IBPS is the least and the speedup values are the highest. Even when $C = 25$, the search time for all possible driving routes with $1 \leq L \leq N$ is still less than 1.3ms. The online route search of IPS is slightly slower than that of IBPS. However, both of them always have a better performance over the other four algorithms. For IPS and LCPS, as we know, when L is small (e.g., $L = 3$), the pruning ratio of IP is slightly lower than that of LCP. However, our algorithm IP outperforms LCP benefited from

TABLE 4
A Comparison of Search Time (Millisecond) on the Synthetic Data Set

	$L=2$	$L=3$	$L=4$	$L=5$	All
			$\|C\| = 15$		
BFS	0.013	0.093	1.303	17.958	-
LCPS	0.012 (1.08)	0.046 (2.02)	0.300 (4.34)	1.987 (9.04)	- (-)
SR(D&C)S	24.708 (0.00)	154.679 (0.00)	1962.810 (0.00)	- (-)	- (-)
SR(BNL)S	3.371 (0.00)	109.556 (0.00)	3612.410 (0.00)	- (-)	- (-)
IPS	0.009 (1.44)	0.056 (1.66)	0.232 (5.62)	0.732 (24.53)	8.700 (-)
IBPS	0.009 (1.44)	0.010 (9.30)	0.011 (118.46)	0.012 (1496.50)	0.152 (-)
			$\|C\| = 20$		
BFS	0.019	0.224	4.612	89.780	-
LCPS	0.019 (1.00)	0.109 (2.06)	0.833 (5.54)	7.916 (11.34)	- (-)
SR(D&C)S	40.135 (0.00)	617.766 (0.00)	11858.100 (0.00)	- (-)	- (-)
SR(BNL)S	11.026 (0.00)	769.158 (0.00)	49492.500 (0.00)	- (-)	- (-)
IPS	0.015 (1.27)	0.058 (3.86)	0.348 (13.25)	1.533 (58.56)	210.256 (-)
IBPS	0.012 (1.58)	0.014 (16.00)	0.017 (271.29)	0.020 (4489.00)	0.641 (-)
			$\|C\| = 25$		
BFS	0.026	0.448	11.940	305.583	-
LCPS	0.026 (1.00)	0.192 (2.33)	2.020 (5.91)	22.632 (13.50)	- (-)
SR(D&C)S	52.182 (0.00)	1362.730 (0.00)	36792.000 (0.00)	- (-)	- (-)
SR(BNL)S	17.772 (0.00)	1559.320 (0.00)	144124.000 (0.00)	- (-)	- (-)
IPS	0.019 (1.37)	0.138 (3.25)	0.814 (14.67)	5.126 (59.61)	190900.000 (-)
IBPS	0.015 (1.73)	0.018 (24.89)	0.024 (497.50)	0.045 (6790.73)	1.267 (-)

The speedup values compared to BFS are given in the round brackets.

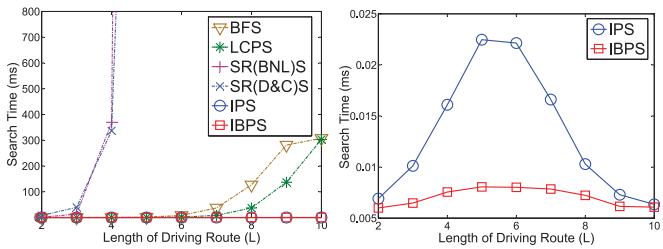


Fig. 11. The search time varying with the length of suggested driving routes on the synthetic data set with $|C| = 10$: (a) an overall comparison of various algorithms; (b) the comparison between IPS and IBPS.

the recursive computation of the PTD cost. The search time of the two skyline methods SR(D&C)S and SR(BNL)S is much longer. The major reason is that skyline query is processed online which needs a rather long time. Therefore, such methods are not suitable for recommending driving routes for a single cab. Actually, it performs better in providing multiple optimal driving routes for different cabs at the same place and time.

Fig. 11 shows the curves of the search time by varying the length of suggested routes on the synthetic data set with $|C| = 10$. A comparison of the search time of all the five algorithms above on the driving routes with various length L is given in Fig. 11a. Obviously, the search time of SR (D&C)S and SR(BNL)S dramatically increases along with the increase of the length of the suggested route. Since the number of remaining candidate sequences is very small, the search time consumed by our algorithms IBPS and IPS is always shorter than those of other four algorithms, and it becomes more and more obvious as the length of suggested driving routes increases.

In order to have a clearer illustration of the trend of search time, the curves of our algorithms IPS and IBPS on the synthetic data set with $|C| = 10$ are shown in Fig. 11b. We can see that the search time of IPS on the driving routes of a certain length L also shows a parabola trend, which is the same as the trend of the remaining candidate sequences. After the batch pruning, the number of remaining sequences becomes so small and the search time of IBPS is almost constant.

In addition, we also conduct several significance tests throughout t-test. Table 5 shows the average search time for LCPS, IPS and IBPS. It also shows the p values of paired t-test for IPS and IBPS compared to LCPS. It can be observed that the search time consumed by our methods is always significantly shorter than that of LCPS.

TABLE 5
The Paired t-Test between Our Methods and LCPS

	$L=2$	$L=3$	$L=4$	$L=5$
mean(LCPS)	0.0079	0.0191	0.0717	0.3381
mean(IPS)	0.0069	0.0101	0.0161	0.0225
mean(IBPS)	0.0060	0.0065	0.0076	0.0081
t-test(IPS, LCPS)	$p = 0.001$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$
t-test(IBPS, LCPS)	$p = 0.000$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$	$p = 0.000$ $p \ll 0.01$

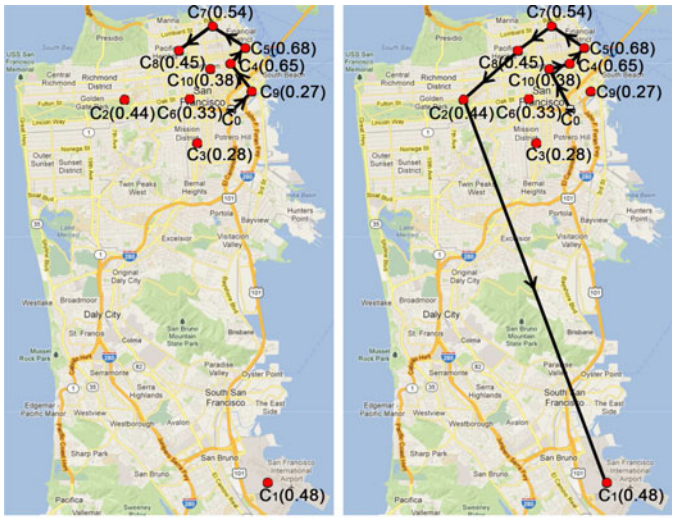


Fig. 12. Two examples of suggested route on the real-world data set of 6-7PM.

It can be observed that our online search algorithm has much shorter search time compared to other existing methods. In particular, the online search time of our method IBP is far less than its theoretical upper bound.

5.5 The Discovered Optimal Routes on Real Data Set

In Fig. 12, there are 10 pick-up points c_i $(1 \leq i \leq 10)$ with their successful probabilities revealed on the real-world data set of 6PM-7PM. A randomly generated current position c_0 is also labeled on it. When we set $D_{\max} = 6, 8, 10$ km, the optimal driving routes detected by the PTD function are $c_0 \rightarrow c_9 \rightarrow c_4 \rightarrow c_5$, $c_0 \rightarrow c_9 \rightarrow c_4 \rightarrow c_5 \rightarrow c_7$ and $c_0 \rightarrow c_9 \rightarrow c_4 \rightarrow c_5 \rightarrow c_7 \rightarrow c_8$ respectively, which are shown in Fig. 12a. If we set c_1 as the destination and $D_{\max} = 45$km, the optimal driving route is $c_0 \rightarrow c_{10} \rightarrow c_4 \rightarrow c_5 \rightarrow c_7 \rightarrow c_8 \rightarrow c_2 \rightarrow c_1$, as shown in Fig. 12b. It is observed that the optimal drive routes detected by our method are meaningful and are not always the same under different distance and destination constraints. Therefore, the proposed method can meet different service requests.

6 Discussion

As we know, the PTD function is a measure for evaluating the cost of a driving route. To meet different business requirements, we can also adopt other evaluation functions in our method to suggest different types of optimal routes to the drivers. Two examples are given as follows.

The potential travel time (PTT) [30]. Since the driving time between two pick-up points usually depends on the traffic flow on the road, the distance does not always represent the cost of a travel route properly. Thus, it is also valuable to recommend a route with the least driving time. Let us present the definition of PTT. Assume that T_{c_{i-1},c_i} is the driving time from c_{i-1} to c_i during a certain period of time. In Formula (1), if we replace the distance D_{c_{i-1},c_i} with travel time T_{c_{i-1},c_i} and $D_{\max}$ with $T_{\max}$, we can get a function of potential travel time

$$F_T(\vec{d}) = T_{c_0,c_1} \cdot P(c_1) + (T_{c_0,c_1} + T_{c_1,c_2}) \cdot \overline{P(c_1)} \cdot P(c_2) + \cdots + T_{\max} \cdot \prod_{i=1}^{L} \overline{P(c_i)}, \tag{11}$$

where $T_{\max}$ denotes the desired maximum cruising time for a driver to pick up new passengers.

The potential travel and waiting time (PTW). In real life, the taxi drivers usually get passengers through two ways: cruising on a road and waiting at a place [3], [4]. Assume that a cab travels along a driving route $\vec{d} = \langle c_0, c_1, c_2, \ldots, c_L \rangle$ $(1 \leq L \leq N)$, but it has not got a passenger when arriving the last pick-up point c_L and then waits at c_L. Let the waiting time be a fixed value T_W and the probability that it successfully gets a passenger at c_L during the waiting time T_W be $P_W(c_L)$, the cruising time vector of $\vec{d}$ is $T(\vec{d}) = \langle T_{c_0,c_1}, (T_{c_0,c_1} + T_{c_1,c_2}), \ldots, \sum_{i=1}^{L} T_{c_{i-1},c_i} \rangle$ and its probability vector is $P(\vec{d}) = \langle P(c_1), \overline{P(c_1)} \cdot P(c_2), \ldots, \prod_{i=1}^{L-1} \overline{P(c_i)} \cdot P(c_L) \rangle$. Then the time cost of successfully picking up a passenger by cruising is $F_C(\vec{d}) = T(\vec{d}) \cdot P(\vec{d})$. The time cost of picking up a passenger by waiting at the last point c_L is $F_W(\vec{d}) = (\sum_{i=1}^{L} T_{c_{i-1},c_i} + T_W) \cdot \prod_{i=1}^{L} \overline{P(c_i)} \cdot P_W(c_L)$. The time cost when a driver does not get passengers after leaving the last point c_L can be set to $F_{\max} = T_{\max} \cdot \prod_{i=1}^{L} \overline{P(c_i)} \cdot \overline{P_W(c_L)}$. Then, the PTW of route $\vec{d}$ can be given as

$$F_{CW}(\vec{d}) = F_C(\vec{d}) + F_W(\vec{d}) + F_{\max}. \tag{12}$$

Actually, the driving time between two pick-up points may vary with the time within a day. We can partition a day into several periods (e.g., 1 hour a period). For every period, assume that the driving time is stable. Then the above two measurements can be applied to evaluate the costs of potential sequences in a fixed period.

7 Conclusion

This paper presents a dynamic programming based method to solve the problem of mobile sequential recommendation. The proposed method utilizes the iterative nature of the cost function and multiple pruning policies which greatly improve the pruning effect. The overall time complexity for handling mobile sequential recommendation problem without length constraint has been reduced from $O(N!)$ to $O(N^2 \cdot 2^N)$. Experimental results show that the pruning effect and the online search time are better than those of other existing methods. In the future, it will be interesting to use parallel techniques for sequence generation and recommendation.

Acknowledgments

This research was supported by the Nature Science Foundation of China (No.61173093, No.61472299 and No.61202182). The authors would like to thank Prof. Jiawei Han and Dr. Jing Gao for their thoughtful comments on this paper.

REFERENCES

[1] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory Pattern Mining," *Proc. 13th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data (KDD '07)*, pp. 330-339, 2007.

[2] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, "An Energy-Efficient Mobile Recommender System," *Proc. 16th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '10)*, pp. 899-908, 2010.

[3] J. Powell, Y. Huang, F. Bastani, and M. Ji, "Towards Reducing Taxicab Cruising Time Using Spatio-Temporal Profitability Maps," *Proc. 12th Int'l Conf. Advances in Spatial and Temporal Databases (SSTD '11)*, pp. 242-260, 2011.

[4] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to Find My Next Passenger," *Proc. 13th Int'l Conf. Ubiquitous Computing (UbiComp '11)*, pp. 109-118, 2011.

[5] S. Börzsönyi, K. Stocker, and D. Kossmann, "The Skyline Operator," *Proc. 17th Int'l Conf. Data Eng. (ICDE '01)*, pp. 421-430, 2001.

[6] H. Kargupta, J. Gama, and W. Fan, "The Next Generation of Transportation Systems, Greenhouse Emissions, and Data Mining," *Proc. 16th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '10)*, pp. 1209-1212, 2010.

[7] J.G. Lee, J. Han, X. Li, and H. Cheng, "Mining Discriminative Patterns for Classifying Trajectories on Road Networks," *IEEE Trans. Knowledge and Data Eng.*, vol. 23, no. 5, May 2011.

[8] N. Pelekis, I. Kopanakis, E. Ramasso, and Y. Theodoridis, "Segmentation and Sampling of Moving Object Trajectories Based on Representativeness," *IEEE Trans. Knowledge and Data Eng.*, vol. 24, no. 7, pp. 1328-1343, July 2012.

[9] E.H.C. Lu, C.Y. Lin, and V.S. Tseng, "Trip-Mine: An Efficient Trip Planning Approach with Travel Time Constraints," *Proc. IEEE 12th Int'l Conf. Mobile Data Management (MDM '11)*, pp. 152-161, 2011.

[10] H. Wang, "The Strategy of Utilizing Taxi Empty Cruise Time to Solve the Short Distance Trip Problem," master's thesis, the Univ. of Melbourne, 2009.

[11] K. Yamamoto, K. Uesugi, and T. Watanabe, "Adaptive Routing of Cruising Taxis by Mutual Exchange of Pathways," *Proc. 12th Int'l Conf. Knowledge-Based Intelligent Information and Eng. Systems (KES '08)*, pp. 559-566, 2008.

[12] Q. Li, Z. Zeng, B. Yang, and T. Zhang, "Hierarchical Route Planning Based on Taxi GPS-trajectories," *Proc. 17th Int'l Conf. Geoinformatics (Geoinformatics '09)*, pp. 1-5, 2009.

[13] S.F. Cheng and X. Qu, "A Service Choice Model for Optimizing Taxi Service Delivery," *Proc. 12th Int'l IEEE Conf. Intelligent Transportation Systems (ITSC '09)*, pp. 1-6, 2009.

[14] G. Hong-Cheng, Y. Xin, and W. Qing, "Investigating the Effect of Travel Time Variability on Drivers' Route Choice Decisions in Shanghai, China," *Transportation Planning and Technology*, vol. 33, no. 8, pp. 657-669, 2010.

[15] E.W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathe*, vol. 1, no. 1, pp. 269-271, 1959.

[16] E.V. Denardo and B.L. Fox, "Shortest-Route Methods: 1. Reaching, Pruning, and Buckets," *Operations Research*, vol. 27, pp. 161-186, 1979.

[17] D.L. Applegatea, *The Traveling Salesman Problem: A Computational Study*. Princeton Univ. Press, 2006.

[18] M. Dell'Amico, M. Fischetti, and P. Toth, "Heuristic Algorithms for the Multiple Depot Vehicle Scheduling Problem," *Management Science*, vol. 39, pp. 115-125, 1993.

[19] S. Funke and S. Storandt, "Polynomial-Time Construction of Contraction Hierarchies for Multi-Criteria Objectives," *Proc. Algorithm Eng. and Experimentation (ALENEX '13)*, pp. 41-54, 2013.

[20] J. Eisner, S. Funke, and S. Storandt, "Optimal Route Planning for Electric Vehicles in Large Networks," *Proc. 25th AAAI Conf. Artificial Intelligence (AAAI '11)*, 2011.

[21] R. Geisberger, P. Sanders, D. Schultes, and D. Delling, "Contraction Hierarchies: Faster and Simpler Hierarchical Routing in Road Networks," *Proc. Seventh Int'l Conf. Experimental Algorithms (WEA '08)*, pp. 319-333, 2008.

[22] I. Abraham, D. Delling, A. Fiat, A.V. Goldberg, and R.F. Werneck, "HLDB: Location-Based Services in Databases," *Proc. 20th Int'l Conf. Advances in Geographic Information Systems (SIGSPATIAL GIS'12)*, pp. 339-348, 2012.

[23] X. Cao, L. Chen, G. Cong, and X. Xiao, "Keyword-Aware Optimal Route Search," *Proc. VLDB Endowment*, vol. 5, no. 11, pp. 1136-1147, 2012.

[24] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, and Y. Huang, "T-Drive: Driving Directions Based on Taxi Trajectories," *Proc. 18th SIGSPATIAL Int'l Conf. Advances in Geographic Information Systems (GIS '10)*, pp. 99-108, 2010.

[25] J. Yuan, Y. Zheng, X. Xie, and G Sun, "Driving with Knowledge from the Physical World," *Proc. 17th ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining (KDD '11)*, pp. 316-324, 2011.

[26] Y. Ge, H. Xiong, C. Liu, and Z. Zhou, "A Taxi Driving Fraud Detection System," *Proc. IEEE 11th Int'l Conf. Data Mining (ICDM '11)*, pp. 181-190, 2011.

[27] D. Grosu and A.T. Chronopoulos, "Algorithmic Mechanism Design for Load Balancing in Distributed Systems," *IEEE Trans. Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 34, no. 1, pp. 77-84, Feb. 2004.

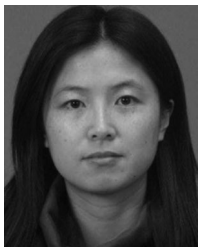
[28] Z. Xu and R. Huang, "Performance Study of Load Balancing Algorithms in Distributed Web Server Systems," *Proc. Int'l Conf. Electrical Eng. and Informatics*, 2004.

[29] http://glaros.dtc.umn.edu/gkhome/views/cluto/, 2014.

[30] Y. Xun and G. Xue, "An Online Fastest-Path Recommender System," *Proc. Sixth Int'l Conf. Intelligent Systems and Knowledge Eng.*, pp. 341-348, 2011.

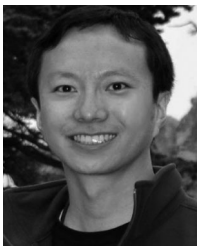
Jianbin Huang received the PhD degree in pattern recognition and intelligent systems from the Xidian University in 2007. He is a professor in the School of Software, Xidian University of China. His research interests are in data mining and knowledge discovery.

Xuejun Huangfu received the bachelor's degree in software engineering from the Xidian University in 2012. He is working toward the master's degree in the school of software in the Xidian University of China. His research interests are in data mining and knowledge discovery.

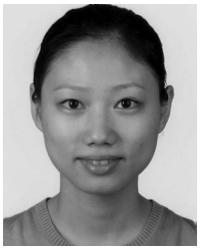


Heli Sun received the PhD degree in computer science from Xi'an Jiaotong University in 2011. She is a lecture in the Department of Computer Science & Technology, Xi'an Jiaotong University of China. Her research interests are in data mining and information retrieval.

Hui Li received the BEng degree from Harbin Institute of Technology in 2005 and the PhD degree from Nanyang Technological University, Singapore, in 2012, respectively. He is a lecturer in School of Computer Science and Technology, Xidian University, China. His research interests include data mining and knowledge discovery.



Peixiang Zhao received the PhD degree from the University of Illinois at Urbana-Champaign in 2012. He is an assistant professor in the Department of Computer Science at Florida State University. His primary research interests include database systems, data mining and data-intensive computation and analysis.



Hong Cheng received the PhD degree from the University of Illinois at Urbana-Champaign in 2008. She is an assistant professor in the Department of Systems Engineering and Engineering Management at the Chinese University of Hong Kong. Her primary research interests include data mining and database systems.

Qinbao Song is a professor in Xi'an Jiaotong University, China. He has published more than 80 referred papers in the areas of data mining and software engineering. His research interests include data mining and machine learning for software engineering.