搜寻：一个针对于起作用的的和有效率的寻找出租车乘客的路线建议系统

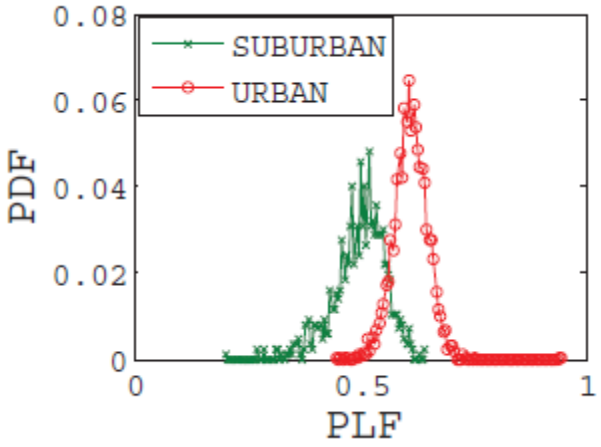
Ye Ding1, Siyuan Liu3, Jiansu Pu1, Lionel M. Ni1,2
1 计算机科学与工程系; 2 广州市香港科大霍英东研究院
香港科技大学
3 卡内基梅隆大学
1,2 {valency, jspu, ni}@cse.ust.hk 3 syliu@andrew.cmu.edu

抽象——目前，有许多出租车跑遍整个城市来寻找打车的乘客，但是他们搜寻乘客并不总是有效率的。根据交通的动力学和有偏见的乘客分布，基于繁华地段当前的离线推荐工作效果并不是很好。在这论文中，我们将此定义成一个新的问题，全局最优路线检索工具（GOTR），是在给定的实时时间内寻找一个有着高利润的和高可能性的可连接的路线来接到一个乘客的一个工具。为了解决这个富有挑战性的难题，我们研发了一个系统，叫 HUNTS，基于来自历史、在线 GPS 数据和商业数据。为了实现上述目标，首先，我们提出了一个动态得分系统通过考虑接乘客率和利润因素来评价每一个路段在不同的时间段的分数情况。第二，我们引入了一个新奇的方法，叫路线交织，基于启发式方法和天际线技巧，来产生个在实时近似的最优的路线。我们的方法是产生了一个可连接的路线而不是数个繁华地段来避免频繁的下一个查询。第三，为了避免拥挤的和其他实时的交通状况，我们会通过一个在线的处理程序经常性地更新每条路段的得分。最后，我们使用来自中国一个大城市周边的 15,000 出租车的大规模的数据验证我们的系统，并且，得出和常规的出租车搜寻乘客和国家最先进的（系统）的比较结果。

I. 序言

目前，在大城市出租车服务仍是一个主要的公众交通服务，并且经常有很庞大数量的没有搭载乘客的出租车在城市穿梭。然而，与此同时，在繁华的街道仍然很难打到车。这个问题在大型现代城市显得尤为严重（北京，纽约和伦敦）。例如，图 1 展示了大约 4000 辆出租车产生的源于我们的数据中的中国的一个大城市近一个月的客座率（ＰＬＦ）可能性分布图。ＰＬＦ是一个出租车搭载乘客走的距离除以出租车总共走的距离的商值。它显示了在远程域一个出租车的客座率。根据图１，我们能发现对于城郊的出租ＰＬＦ仅达到５０％，而对于城市内的达到６０％。由于低客座率，一个搜寻乘客建议系统就是一个出租车司机和城市当局者迫切的需求，来改善出租车的利用和减少能源的浪费。

为了制作一个乘客搜寻建议，我们能利用出租车ＧＰＳ设备搜集的地理数据，并且出租车计价器搜集的商业数据。通过分析上述数据，传统的方法能识别繁华的地方（ＰＯＩ），是一个有着很高可能性来接到乘客的地点，然后推荐给出租车司机的最优的一个地方。然而，一个ＰＯＩ推荐存在数个缺点。

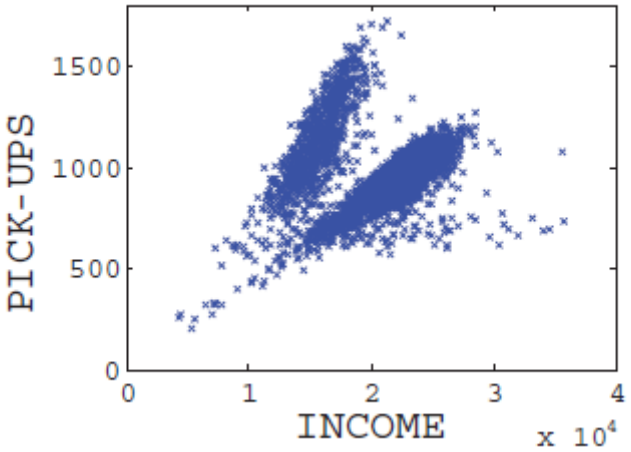


图一.出租车的 PLF 分布图，城郊的出租车仅仅为 50%，城市内的大约在 60%。

图 2.乘客尝试在长沙的中心街道打车，在发展中国家的许多城市几乎没有在 POIs 附近停靠的出租车，所以人们需要在大街的中心来打车

首先，如果一个出租车不允许在ＰＯＩs 的地方等客并且接客。ＰＯＩ ｓ将不合适作为推荐。这是因为如果一个出租车开到了ＰＯＩ但是不能接到一个乘客，这个出租车司机应该马上要求另一个ＰＯＩ推荐。在这种情况下，详尽查询的效率是不能令人接受的。事实上，出租车在大多数的路段的中心确实不被允许以接客为目的的停车。更多的，在许多发展中国家，在ＰＯＩ ｓ旁边几乎没有出租车停车的地方，并且人们需要在街道的中心打车，就像图 2 所展示的。

第二，路线搜寻ＰＯＩ场所可能不是全局最优。一个出租车可能在走过很多ＰＯＩ ｓ之后接到客。尽管每一个ＰＯＩ是每步的最优推荐，但是这个贪心ＰＯＩ搜索路线结果可能不是最好的选择。一个推荐应该是比较出租车所有可能的搜寻路线后全局中最优的，但并不是在每一步是最优的。



图三.在一个月内出租车司机的收入和载客量的关系图

第三，一个出租车的目标是为了挣钱，但是寻找到更多的乘客可能并不导致挣到更多的钱。在图三中，我们能发现两个不同的挣钱模式：１）载更多的乘客，和２）载比较少的乘客但是更高的收入。产生这种情况的原因是，一个出租车可能必须在站台或者机场等很长的一段时间来载到一个乘客，但是一旦这个出租车载到一个乘客，收入会非常的高，因为这些乘客经常会到很远的目的地去。相反的情形，载客率可能在商业街非常高，但是乘客在购物之后可能去一个比较近的地方因为他们仅仅是不能拿那么多东西。因此，一个推荐应该实现不仅能有更高的载客率，并且更高的利润。

为了解决ＰＯＩ推荐的缺点。在这篇论文中首先，我们创建了一个有联系的搜寻路线而不是数个ＰＯＩｓ作为一个推荐，并且出租车司机可以开到一个路线直到接到一个乘客。第二，路线搜寻大概是个全局最优，可以比一系列的ＰＯＩｓ推荐获得更高的利润。第三，我们认为载客率和经过推荐的平均收入来使搜寻更加有效。更多的，由于实时交通和客人的分布的经常性改变，我们的系统能更加有效地处理网上数据作为输入，和产生实时的推荐。

下列方面对这篇论文做出了很大贡献，

.首先，我们创建了一个系统，叫 HUNTS，能产生为出租车司机的搜寻路线推荐让他们挣更多的钱。

.第二,为了制作这个推荐，我们定义成一个新的问题，全局最优路线检索工具，作为在给定的实时时间内寻找一个有着高利润的和高可能性的可连接的路线来接到一个乘客。为了解决这个问题，我们引入了一个新奇的方法，叫路线交织，基于一个启发式方法和天际线技巧，来利用每一个路段并且提供一个推荐。

.第三，我们使用真实生活中庞大的出租车数据来评价和比较不同的方法，并且我们的实验展示了从效率和效果上面看我们的推荐比传统的 POI 和常规的出租车搜寻推荐效果好得多。而且，我们通过我们的学习已经发现了一些有意思的结果。

TABLE I
SPECIFICATIONS OF BUSINESS DATA

Data Type	Description
Taxi ID	Taxi registration plate number.
Begin / End Time	Begin / end timestamp of the deal.
Distance	Total distance of the deal.
Time Cost	Total time of the deal.
Price	Taxi fare of the deal.
Free Distance	The distance between two deals while the taxi is unoccupied.
Free Time	The amount of time between two deals while the taxi is unoccupied.
Taxi Company	The company id of the taxi. One company can only have one color of taxis.
Taxi Color	The color of the taxi, green or red.

TABLE II
SPECIFICATIONS OF TRACE DATA

Data Type	Description
Taxi ID	Taxi registration plate number.
Timestamp	Timestamp of the sample point.
Latitude / Longitude	GPS location of the sample point.
Speed	Current speed of the taxi.
Angle	Current driving direction of the taxi.
Occupied Status	Indicator of whether the taxi is occupied by a passenger.

在下列的部分，我们将在第二部分首先定义一个问题，然后展示我们包含一个动态评分系统的系统，一个推荐。然后在第三部分展示一个在线数据处理。在第四部分，我们将评估我们系统的精度和性能。在第五部分，我们将会讨论一有趣的结果。最后在第六部分和第七部分，将会提供相关工作和结论。

II. 数据描述与问题定义

A. 数据描述

我们的数据是 2009 年 9 月在中国的一个大城市采集的，它包括两个数据集：一个是利用 GPS 设备采集的出租车轨迹，另一个是描述每笔交易的业务的商业信息。我们的数据形式在表 1 和表 2 中已经给出。我们的轨迹信息包括大约 15，000 辆出租车，样本比率大约 20 秒。我们的商业信息包括 4,197 辆出租车的商业信息，平均每辆车每天 44 笔交易。

B. 术语

路段：路段 τ_i 是两个十字路口之间的路。一些路，比如高速公路，可能有两个路段在两个

十字路口之间，因为他们有不同的方向。路网 $R=\{\tau_i\}_{i=1}^{n}$ 是由路段组成的。

交易：交易是从接到乘客到将乘客送下车的过程。

打车费：打车费是乘客在交易结束时付的钱。在这篇论文中，打车费是出租车唯一的收入。

出租车状态：出租车的状态有被占用和未被占用两种。被占用指出租车正在送乘客，否则，是未被占用的。

接载率：接载率是在某路段接到乘客的可能性。

平均收入：平均收入是某地某时间段内交易的平均打车费。

得分：对于路网 $\mathrm{R}=\{\tau_i\}_{i=1}^{n}$，路段 τ_{i}，我们定义 $\delta_{\mathrm{i}}^{\mathrm{t}}$ 作为 τ_{i} 在时间 t 上的得分，这是路段 τ_{i} 的收益的衡量。$\delta_{\mathrm{i}}^{\mathrm{t}}$ 也许根据时间变化。正如之前提过的，$\delta_{\mathrm{i}}^{\mathrm{t}}$ 的设计应该既考虑平均收入又考虑 τ_{i} 的接载率。在这篇论文中，$\delta_{\mathrm{i}}^{\mathrm{t}}$ 是通过动态的得分系统如图 5 所示计算的，细节将在第三部分展示。

三.

C.问题定义

全球最佳路线检索是使包含连接的路段路线得分增大的问题。此问题的正式定义在定义 1 中给出。

定义 1.全球最佳路线检索（GOTR）

给出一个出发点 S 和一个出发时间 t_s，寻找一个路线 $\mathrm{T}=\{\tau_i\}_{i=1}^{m}\subseteq R$，最大值为 $\sum_{\mathrm{i}=1}^{m}\delta_i^t$ 在时间 $\hat{t}$ 中。

问题的定义基于我们之前提过的设计良好的 δ_i^t，假定我们有一个合适的 δ_i^t 的设计，问题可以被模型化为一个有时间约束的最长路径问题。采用以时间为依据的最短路径问题的定义，我们定义以时间为依据的最长路径如定义 2 所示。

定义 2.以时间为依据的最长路径问题（TDLP）

给出没有方向的 FIFO 图 $G=(\mathrm{V},\mathrm{E},\Delta,\mathrm{T})$，使 $\mathrm{e}_{i,j}\in E$ 成为 $\mathrm{v_i}$ 和 $\mathrm{v_j}$ 的边缘事件。$\delta_{i,j}(\mathrm{t})\in\Delta$ 是 $e_{i,j}$ 的宽度，t 是一个时间范围内的时间变量，$\mathrm{t}_{i,j}\in T$ 是 $e_{i,j}$ 的时间延迟。寻找路径 $\mathrm{P}=(\mathrm{s},\mathrm{v}_1,\mathrm{v}_2,\ldots,\mathrm{v}_n)$ 从 S 点在 $\mathrm{t_s}$ 出发，在 $\sum t_{i,j}\le\hat{t}$ 时，最大宽度 $\sum\delta_{i,j}$。

不同于其他路径寻找问题，在这篇论文中，我们不限制结果的路径是一个简单的路径，这是因为如果这里存在一个高收益的循环，司机可以重复的开过这个循环，因为这里有一个时间界限 $\hat{t}$，我们总是可以找到一个可行的方案去解决这个问题。此外，与以时间为依据的最短路径问题相似，我们认为图是 FIFO 图。FIFO 图的定义在定义 3 中。

定义 3.FIFO 图

使 d(I,j,t)是边界 $e_{i,j}$ 的延迟函数。对于任何 $t,t' \geq 0, t \leq t'$,我们有 $t + d(\mathrm{i,j,t}) \leq \mathrm{t}' + \mathrm{d(i,j,t')}$.

如定义 2 所示，我们认为延迟函数 $d(\mathrm{i,j,t}) = \mathrm{t}_{i,j}$ 是与 t 无关的，因此它是一个 FIFO 图。因为最长路径的决定问题是 NP complete，最长路径问题是 NP hand。因此以时间为依据的最长路径问题和全球最佳路径检索问题，都是 NP hand。

为了清楚地说明全球最佳路径检索问题，请参考图表 4。在图表 4 中，使 $\hat{t}$ =2，R={a,b,c,d}，我们目的是找到{a,b}而不是{a,c}因为后一个有一个较低的得分 $\sum \delta$ ，并且也不是{a,d}因为它超过了时间限度 $\hat{t}$ 。

III.狩猎：狩猎系统

A.概述

在本文中，我们解决问题的方法主要基于图 5 中展示的体系结构。接下来的几个小节将一步一步具体的介绍这个结构，其中包括动态的评分系统，推荐和在线数据处理程序。

B.动态评分系统

在这个部分，我们将讨论为每个路段 τ_i.打分 δ_l^t 的打分系统。因为对于每个路段的计算方法是一样的，出于方便，在这一节我们将用 δ^t 代替 δ_l^t 在本文中，我们定义 δ 为：

$$\delta^t = \frac{p^t w^t}{d} \qquad (1)$$

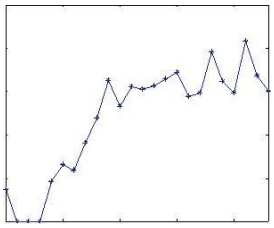


表 6：单一某天单一路段的乘客搭载率。时间间隔的大小是一个小时，其中 0 代表 00:00 - 00:59,1 代表 01:00 - 01:59 等。

其中 p^t 代表在 t 时间的乘客搭载率，w^t 代表平均收入，d 则是路段的长度。由于 p^t 和 w^t 都是统计的数值，所以上标 t 其实是一个很小的时间段。

我们设计的分数基于以下几个方面：1）p^t 和 w^t 的值越高，我们越倾向于选择那个路段，因为这样会有更高的几率接到更高收入的乘客；2）我们更倾向于选择较短的路段，因为在一段较长的路段上 δ 处处不等，而 δ 较高也只是因为其中部分路段。总的来说，我们在设计在一段时间中路段的分数时，主要考虑的是较高的 p^t 和 w^t 以及较低的 d。在接下来的几小节中，我们将就如何生成这 3 个值做出介绍。

1）乘客搭载率

我们将用在一个路段上接到乘客的出租车的数量比上空车途经这一路段的出租车总数，以得到乘客搭载率。由于这一指标在不同的时间会有区别，比如晚上 19:00 人们下班时间的乘客搭载率要明显高于凌晨 3:00 街道上几乎没有行人的时候，因此我们对每一时间间隔的各个路段做以下运算：

$$p_t = \frac{|status(0 \rightarrow 1)|_t}{|status(0)|_t} \quad (2)$$

其中|status(0 → 1)|代表了经过这一路段后从空车状态变成载客状态的车辆数目，|status(0)|代表了空车经过这一路段的车辆数。在这里我们将状态 0 定义为空车状态，状态 1 则是载客状态。表 6 就是用公式 2 生成的一天中的乘客载客率。

在表 6 中，我们可以看出，在午夜（01:00 - 03:00）的乘客搭载率几乎为 0，因为司机们不太可能会在人们睡觉的时候接到乘客；但在晚上（21:00 - 24:00）的乘客搭载率则一直保持较高水平，这大概是因为尽管经过的出租车数量较少，但这些车总能成功的拉到乘客。

在计算乘客搭载率之前，我们应该先做对不同路段上的样本点进行地图匹配。在本文中，我们采用了如算法 1 中展示的用以地图匹配的贪心算法[6][7]。

Algorithm 1 Greedy Map Matching Algorithm

//算法 1：地图匹配贪心算法

1:Separate the map into a set of subspaces $\{s_i\}_{i=1}^{n}$ of equal size, where each si contains at least one road segment;

//把地图分成一系列等大的子空间 $\{s_i\}_{i=1}^{n}$ ，每个 s_i 都包括至少一个路段

2:Find the subspace si where the sample point p ∈ si;

//找到样本点所在的子空间 s_i

3:si ← si∪ neighbors of si;

4:dmin ← ∞;

5:r ← ∅;

6:for each road segment $\tau_j \in s_i$ do

//对于任意满足 $\tau_j \in s_i$ 的路段

7:Find the minimum perpendicular distance d from p to τ j ;

//找到从样本点 p 到 τ j 的最短垂直距离

8:if d < dmin then

9:dmin ← d;

10:r ← τ j ;

11:end if

12:end for

13:Resulting r is the matching of p.

//结果 r 就是 p 的最近匹配

Algorithm 2 Algorithm of Minimum Perpendicular Distance

//算法 2：最短垂直距离求法

1:dmin ← ∞;

2:for each component segments γ i ∈ road segment τ j do

//对于路段 τ j 的任意组成部分 γ i，计算：

3:d ← perpendicular distance of p to γ i;

//d 为 p 到 γ i 的垂直距离

4:if d < dmin then

5:dmin ← d;

6:end if

7:end for

8:Resulting dmin is the minimum perpendicular distance from p to τ j .

//结果 d_{min} 就是从 p 到 τ j 的最短垂直距离

算法 1 将样本点匹配到了在、拥有最短垂直距离的最近路段上。在本文中，由于一条路段其实是一条折线，我们就将最短垂直距离定义为一个样本点 p 与该路段上所有部分之间距离的最小值，计算最短垂直距离的算法如算法 2 所示。由于算法 1 列举了所有的路段并计算了样本点到每个路段的垂直距离，所以它具有关于 $O(n^2)$的时间复杂性。

为了提升算法的效率，在计算最短垂直距离之前，我们先将地图分割成了网格状的子空间，对于任意的样本点，我们只用出于同一子空间的路段与之匹配，。但是这样就会出现图 7 中所示的边境问题。在图 7 中，尽管 p 点离 B 更近，但是由于 p 与 A 属于同一空间，p 就会被匹配到 A 路段上。在本文中，我们通过将子空间的 8 个相邻空间也纳入考虑的方式避免这一问题的发生，参见算法 1 中第 3 行。

图 8 展示的是通过算法 1 计算的我们的路线数据生成的城区交通状况。可见，在下班高峰期的平均车流量是 14.85 辆车每路段，但在半夜，则只有 7.11 辆车每路段。同时图 8 中的由红到绿的颜色梯度表现了交通状况从拥挤到空闲，拥挤就意味着路段上会有更多的车辆。

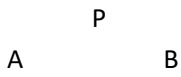
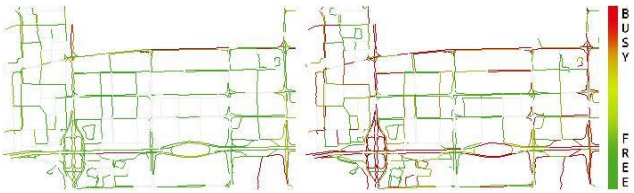


图 7：地图匹配上的边界问题，p 离 B 更近，但却会被匹配到 A 上。



(a) 午夜 03:00-04:00　　(b) 下班时间 18:00-19:00

图 8：城区内一天的交通状况，下班时间的车流量明显高于午夜。

我们明显能够看出，针对不同路段，下班高峰期的车流量都明显高于午夜，这也就解释了时间约束的重要性。

2）平均收入：

在这一部分中，我们将就对各个路段的平均收入如何计算进行讨论。如之前提到的一样，我们通过观察得出，总是奔着具有最高乘客搭载率的路段并不是最好的选择，因为在终点站或者飞机场的乘客打车的距离可能会远远超过那些普通街道或者商业区的乘客。因此，我们在为路段打分时也应该考虑平均收入。

和计算乘客搭载率相似，我们用时间段来衡量每个路段的平均收入。在本文中，我们按每笔交易的开始时间判断它属于哪一时间段。如：一笔从 9 月 1 日 23:50 开始持续到 9 月 2 日 00:10 的交易，我们就会把它算作 9 月 1 日的交易，如果时间分段是 1 小时的话，它就将属于 23:00 - 23:59 的时间段。

本文中，我们只考虑在特定路段接到乘客的出租车的平均收入，计算公式为：

$$w^t = \frac{fare(0 \rightarrow 1)_t}{|status(0 \rightarrow 1)|_t} \qquad (3)$$

其中，|status(0 → 1)|代表了经过这一路段后从空车状态变成载客状态的车辆数目，$fare(0 \rightarrow 1)$表示这些出租车收到的费用。图 9 画出了一天中不同时间某路段上的平均收入，从中不难发现，午夜的平均收入为 0,因为几乎没有人在这个时间内打车；然而白天的平均收入约为 25 美元，这符合我们的实际经验。

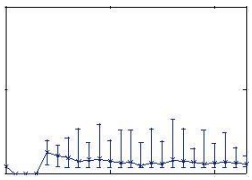
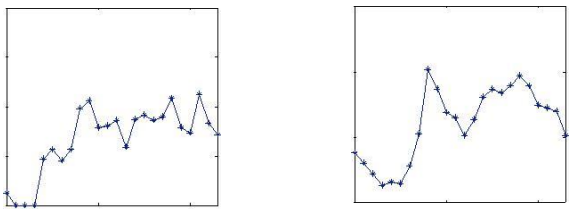
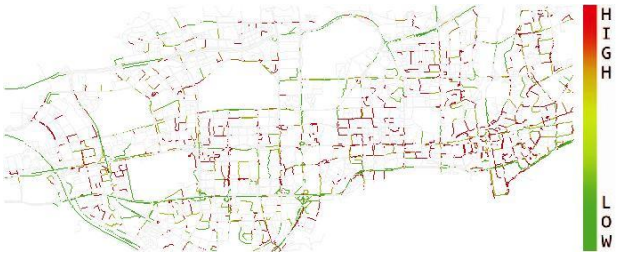


图 9：一天中同一路段的平均收入



（a）一天中同一路段的得分　　（b）一天中所有路段的平均得分



（c）18:00 - 19:00 城区路段的平均得分。用颜色从红到绿表示得分从高到低，灰色代表没有得分，即，在这一路段上没有出租车接到乘客。

图 10：一天中单一路段的得分和所有路段的平均得分。

3）打分：既然我们得到了乘客搭载率和平均收入，我们就可以按照公式 1 计算出路段的得分 δ^t。图 10 中画出的是一个路段在不同时间的得分和一天中不同时间所有路段的平均得分。从中我们可以看出，在上下班高峰期各路段的平均得分很高，但在午夜的得分却很低。

C. 推荐

当所有道路段的分数都被包含进来时，我们就可以基于出租车当前位置推荐一个路线和时间。在这部分里，我们将介绍我们的推荐系统，这个系统将基于对先前采集到的得分进行计算产生一个连接路线。

重看定义一，推荐系统旨在从开始时间 s 到结束时间 ts 的时间ˆt 内，利用最大值 $\sum_{i=1}^{m} \delta_i^t$ 找到一个路线 $T = \{\tau_i\}_{i=1}^{m} \subseteq R$。就如之前定义二提到的，我们可以把检索最优路线的问题模型化成依赖于时间的最长路径问题。尽管依赖于时间的最长路径问题看起来像一个最长或最短路径问题，但上述路径问题的解决方案不能直接适用于解决依赖于时间的最长路径问题。这主要是因为一下三个主要的限制：

1）传统的最长路径问题往往认为图像的权重是静止不变的。但由于此例中图像的权重随时间变化，因为在一个动态权重图中，我们不能比较不同时间的权重，因此传统的算法可能并不可行。例如图 11。

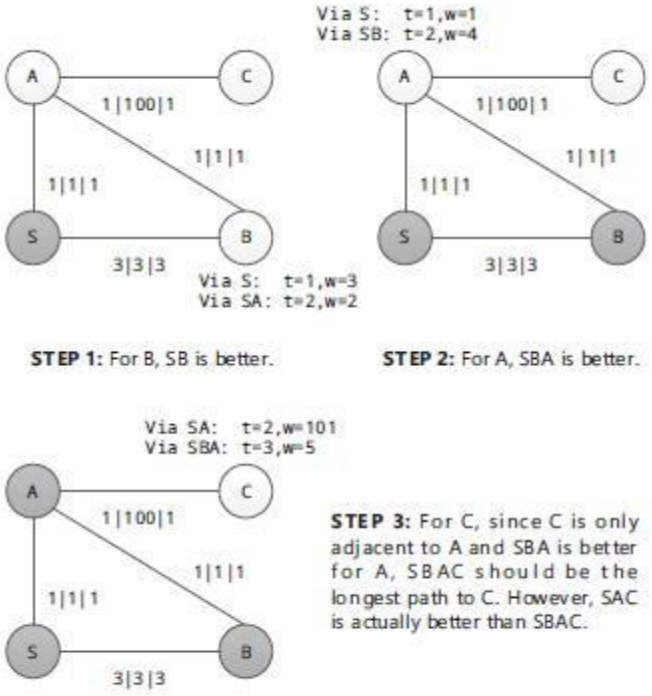


Fig. 11. Dynamic weighting problem.

在图 11 中，假设我们从 S 处开始并尝试在途中寻找最长的路径，在这条路径上，每条边的时间延迟是一，每条边上分配的权重如图所示。第一步，因为权重为 3 的 SB 比权重为 2 的 SAB 要好，我们认为 SB 是到达 B 的最长路径。与之类似，第二步，对于从 S 到 A 这条路线，权重为 4 的 SBA 比权重为 1 的 SA 要好。现在，考虑到在静态的权重图中动态编程的想法，第三步，因为 C 只和 A 相邻，到达 C 的最长路径应该与到达 A 的最长路径加 AC 间的权重的和相等，即 SBAC，权重为 5。但是，因为 AC 的权重随时间变化，我们可能会漏掉更长的路径，即权重为 101 的 SAC。动态权重问题是最长/最短路径和依赖于时间的最长/最短路径之

间的主要区别。就如同基于静态图片的最短路径算法不能用于解决依赖于时间的最短路径问题一样，传统的最长路径问题也不能被直接用于于解决依赖于时间的最长路径算法。

2）传统的找寻路径的问题经常假设图中的每一条边只有一个权值（即物理意义上的距离），但在本文提到的情况中，需要考虑两个权值：得分和时间延迟。因为我们不得不在这两者间做出妥协，所以问题变得十分困难，而传统算法一般不会考虑到类似的情况。

3）就如在定义二中提到的，和传统寻找路径问题不同的是，在实际生活中，我们并不把路径限制在简单路径范围内，我们也允许周期性的路线。因此传统算法不能被采用。

为了解决检索最优路线的问题，一个直接的办法是详尽列举所有可行的轨迹并找到一个得分最高的轨迹。算法 3 展示了具体的算法。

Fig. 11. Dynamic weighting problem.

Algorithm 3 Exhaustive Algorithm

```
1: T_0 ← {s};
2: while there exists an open T_i where T_i(t) < t̂ do
3:    for each subsequent road segment τ_j of τ_m ∈ T_i do
4:       Create T' ← T_i ∪ τ_j;
5:       if T'(t) > t̂ then
6:          Drop T';
7:       end if
8:    end for
9:    if all T' are dropped then
10:      Mark T_i as closed;
11:   else
12:      Remove T_i;
13:   end if
14: end while
15: Resulting T_i with max(T_i(δ)) is the optimal selection.
```

1）穷举算法：

在算法 3 中，我们创建了一系列路线 $\{Ti\}$，同时列举出从时间 s 开始在 t 时间内的所有可能路线。最终算法返回得分最高的路线 $Ti(\delta)$。正如在定义一和定义二中提到的那样，每个路段 τi 都有相应的得分 δt 和时间延迟 ti。类似地，对任何的 Ti，$Ti(\delta)$是路线的总得分，$Ti(t)$是总时间延迟。每个 Ti 都包含很多个按到达时间分类的路段 $\{\tau 1, \tau 2, \ldots, \tau m\}$，他们依次相连。

在这篇论文中，如果两个路段公用一个交叉路口，即两个边在图中有相同的顶点，我们说这两个路段是邻居。因为每条边都有两个顶点，我们可以把一条路的邻居分成两个部分 NL 和 NR。对于路段 $\tau j \in Ti$，假设 $\tau j-1 \in NL$ 和 $\tau j+1 \in NR$，我们说 NR 中的路段是 τj 对 $\tau j-1$ 的后续路段。引入后续路段的原因是，一个人不能够驾车返回先前的路段，否则就会违反先入先出的限制。相应的细节在图 12 中展示。

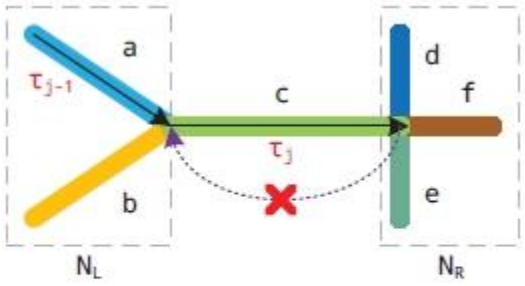


Fig. 12. An example of the subsequent road segments. Let $\tau_j = c$ and $\tau_{j-1} = a$, then $N_L = \{a, b\}$, $N_R = \{d, e, f\}$. Clearly the taxi cannot drive back to N_L, or the time delay of c will double. Hence the recommender can only recommend the road segments in N_R, which are the subsequent road segments of τ_j with respect to τ_{j-1}.

在列举过程中，算法 3 表现为广度优先搜索，它在没有更多路线的时候就会停止。假设所有路段的最短时间延迟为 t，算法 3 最多花费$\hat{}t/t$ 圈。假设所有路段的后续路段的最大数字是 r，对每一圈来说，算法 3 最多创造出 r 条新的路线。因此，算法 3 的时间复杂度最大是 $O(1+r+r2+\cdots+r\hat{}\,t/t) \leq O(r\hat{}\,t/t+1)$。鉴于所有的路线都被列举了，算法 3 是一个准确的算法。

2）贪心算法：

如同前面提到过的，传统的推荐系统经常推荐最可能拉到乘客的路段给出租车。在驾驶到推荐路段后，如果出租车不能拉到乘客，那么一个新的建议就会被生成。这样的方式会一直重复，直到出租车拉到一个乘客。如果我们考虑到这个建议有时间限制$\hat{}t$ 的话，正如算法 4 展示的那样，这种连续的跳到下一个的推荐方式实际上是贪心算法。

算法 4 是一个近似算法，它试图在每一步找到最佳的后续路段，但是这样得到的路段可能不会是最佳结果。算法 4 的精确度是我们介绍接下来两种算法的原因。

Algorithm 4 Greedy Algorithm (GA)

1: $T \leftarrow \{s\}$;
2: **while** $T(t) < \hat{t}$ **do**
3: Find the subsequent road segment τ_j of $\tau_m \in T$ with the highest score among all the subsequent road segments of $\tau_m \in T$;
4: Append τ_j to T;
5: **end while**
6: Resulting T is the approximate selection.

3）启发式算法：

受本篇文章中 *A** 算法的启发，我们介绍一种如图所示的算法 5，这种算法比算法 3 更快，比算法 4 更准确。

Algorithm 5 Heuristic Algorithm (k-HA)

```
1: Create empty stack I;
2: T ← {s};
3: Push T into I;
4: T* ← null;
5: while I is not empty do
6:    Pop a trajectory from I to T;
7:    for each subsequent road segment τ_j of τ_m ∈ T do
8:       Create T' ← T ∪ τ_j;
9:       if T'(t) ≤ t̂ and T'(δ) + (t̂ − T'(t)) * ρ ≥ T*(δ)
         then
10:         Push T' into I;
11:         if T'(δ) > T*(δ) then
12:            T* ← T';
13:         end if
14:      end if
15:   end for
16:   Reduce the size of I to k by removing lower-scored
      trajectories in I;
17: end while
18: Resulting T* is the approximate selection.
```

在算法 5 中，我们使用启发式估计去评估最优路线：$h(T) = T(\delta) + (\hat{t} - T(t)) * \rho$，

ρ是在可行域的路段的最高时间效率值，即出租车在时间$\hat{t}$内能够到达的最大区域范围。

路段的时间效率是它的得分除以时间延迟的商。

与算法 3 不同，在算法 5 的迭代中，我们通过评估新路线能得到的最高分来检查它是否是最优的。如果新的路线能达到的最高分比目前的最高分还要低，它就会被删除。此外，在每个迭代的末尾，我们通过删除栈中分数低的路线，把栈的大小减小到用户定义的更高界限 k。k越高，找到的线路越精确。如果 k 设定为 1，那么算法 5 会找到和算法 4 一样的路线。因此算法 5 的精度要高于算法 4，同时它的时间复杂度要低于算法 3。

和其他的启发式算法类似，算法 5 是一步一步得到最优解的。因此在运行这个算法时，我们可以提取一部分产生的路线给出租车。这是算法 5 的另一个优点。

4）路线串连算法：

为了提高算法5的精度，我们通过使用算法6中展现的天际线操作（Skyline operation）【8】介绍另一种新方法，叫做路线串连。

Algorithm 6 Trajectory Sewing Algorithm (TS)

```
1: ...
2: while I is not empty do
3:    ...
4:    Remove trajectories which can be dominated by others
      in I;
5: end while
6: ...
```

在本文中，我们在两个维度定义天际线操作（Skyline operation）的控制操作：时间延迟 $\pi(t)$和得分 $\pi(\delta)$。如果 *T1* 比 *T2* 有更短的时间延迟和更高的分数，我们说 *T1* 优于 *T2*。对于所有 $T \in I$，如果存在另一个 $T \in I - T$ 优于 T，T 将会被删除。由于天际线操作（Skyline operation）考虑到了两个维度而不是算法 5 中的一个维度，它比算法 5 更精确。

总之，受益于我们的评分方案，我们的推荐系统 1）考虑到了时间对接乘客的影响 2）同时考虑到了接到乘客的概率和每个路段的平均收入 3)提供了连接路线而不是一些地点 4）找到了近似的最佳路线。

D.在线数据处理程序

正如先前提到的，因为跟踪数据和业务数据总是流向系统，我们需要不断上传各个路段的评分。在这篇论文中，某个路段的得分应该总是历史得分的平均值。然而，在现实生活中，交通和城市规划可能会变化，因此就没有必要计算出所有历史得分的平均值。因此，我们使用简单动态平均数去计算平均得分：

$$SMA(n+1) = SMA(n) - \frac{v_{n+1-m}}{m} + \frac{v_{n+1}}{m}$$

$SMA(n)$是 n 个数字的简单动态平均数，vn 是第 n 个数字的数值，m 是栈 M 的大小。当我们计算 n+1 个数字的简单移动平均数时，我们从栈 M 中弹出一个数字，然后把第 n+1 个数字插入到栈 M 的底部，然后通过上述公式得到栈的平均值。

计算平均数的一个重要问题是，可能不会同时得到跟踪数据和业务数据。跟踪数据往往可以从 GPS 得到，然而业务数据只能在乘客下车并且出租车报告乘客的消费数额时才能得到。因此在本文中，我们分别上传接到乘客的概率和平均收入，并在上传之后重新通过以下的公式计算分数。

$$\delta^t = \frac{SMA(p^t)SMA(w^t)}{d}$$

简单移动平均数要求大小为 m 的栈，因此 δt 总是使用最近的 m 个记录。如果 δt 每日被上传，这意味着只有 m 天的数据被用作产生路段的得分。请注意，这里的平均值可以是日平均值，也可以是周平均值，即一个人可以给一个路段分配七个分数，同时每个分数代表着一周中的一天。用周平均值的原因是接到乘客的概率和平均收入在工作日和周末会有不同。在本文中，我们在实验中使用日平均值。

通过在线数据处理程序，我们不需要在每次推荐路线时都计算得分，而仅仅需要从数据库中查询最新的数据。这样的在线数据处理程序极大地提高了系统的效率。

IV. 评估

A. 实验数据

在这篇论文中，我们用一周的历史数据（包括 15，231 辆出租车所载的 154，000，000 条记录）来评测我们的系统。我们利用在线数据处理器输入所有记录，并以 10 分钟的时隙来计算路段的得分。我们结合地面实况数据和传统方式比较我们的推荐路线。这些比较涉及精度检验和动作性能。在线数据处理器被用于输入数据，它可被证实精准度极高，所以没有必要再去对它进行专门的评估。比较部分包括：

1）**GT**：一天当中，出租车搜索路线的地面实况数据从 t_s 开始，结束于时刻 $t_s + \hat{t}$。

2）**GA**:通过传统下一跳贪心算法方式形成该推荐路线（算法 4）。

3）TS:通过路线 sewing 算法方式形成该推荐路线（算法 6）。
4）**10-HA：**通过探试算法方式形成该推荐路线（算法 5），此处 k=10.因为探试算法执行起来较为耗时，我们只在 3 分钟内检索结果。

考虑到花需要费大量时间去执行，所以穷举算法并不被包括在实验之中。为在不同情形下比较推荐路线，我们基于不同开始时间t_s，不同地点 s ，不同时间段$\hat{t}$共实施 90 种不同的查询类型。具体细节如下：

1）t_s：09:00:00, 14:00:00, 18:00:00.

2）s: 35 条任意路段.

3）$\hat{t}$：300 秒，600 秒，1800 秒.

综上所述，我们每种方式形成 315 条推荐路线和 36，534 条路面实况路线。因为推荐路线的精度并不有赖于t_s，s ，$\hat{t}$，所以我们平等对待所有推荐路线。

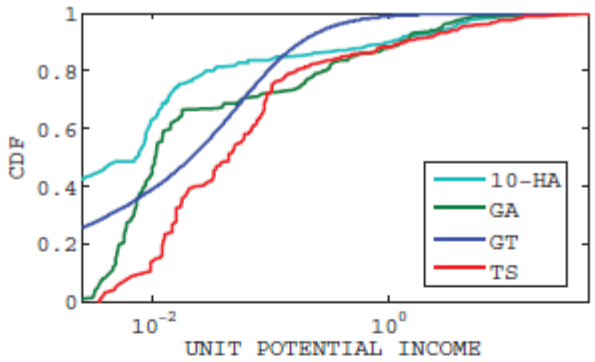
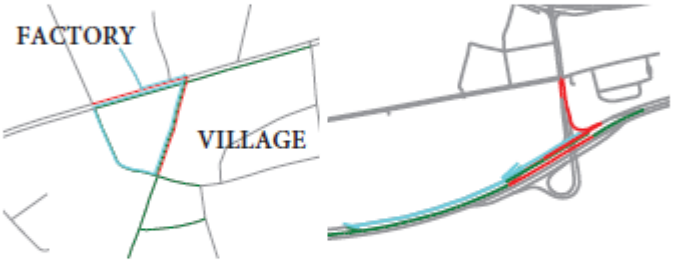


Fig. 13. CDF of the average potential income of GT, GA, TS, and 10-HA.



(a) Case 1 (b) Case 2

Fig. 14. Comparing recommendations via different methods, GA is colored green, 10-HA is colored cyan, and TS is colored red. The start point of each case is the road segment colored by all three colors.

Fig. 15. Comparing recommendations with ground truth hunting trajectories, GA is colored green, 10-HA is colored cyan, TS is colored red, and GT is colored blue.

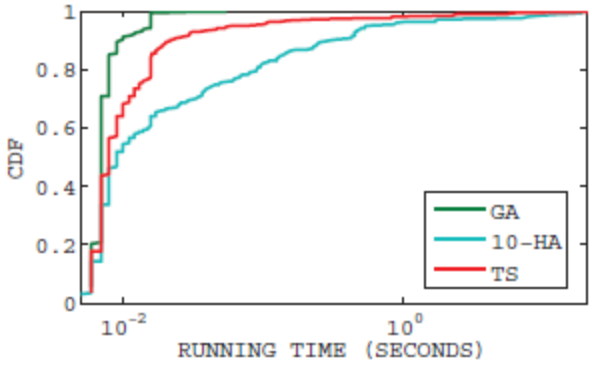


Fig. 16. CDF of execution time of 10-HA, TS, and GA.

B. 精度

在这篇论文当中，我们用单元潜在收益来评估路线的精度。正如之前提到的，乘客的接取率反映了接取乘客的几率。并且如果出租车接取一位乘客，那么它的平均收益可以反映其潜在收益。因此，我们可以通过乘客接取率来评估一项推荐路线的有效性，通过它的平均收益来评估它的效率。但是，我们不能用单独的方式评估这两种指标。因为效率依赖于有效性。例如，如果乘客的接取率比较低，那么无论潜在的均收益有多高，推荐路线都是没有多大意义的。因此在这篇论文中，我们根据每条路段的有效性和效率来评估精度，经由其潜在收益：

$$f^t = p^t w^t \qquad (7)$$

f^t 是一条路段在时刻 t 的潜在收益，p^t 是接客率，w^t 是出租车成功接取一位乘客后的平均收益。为了更客观公正地比较路线的潜在收益，我们用每单位距离潜在收益价值作为*路线的单位潜在收益*。在这篇论文中，我们用 100 米作为单位长度。比较的精度展示在**图 13** 中。

在**图 13** 中，我们可以发现在大约 70%的实验中，TS 的表现总要好于 GA。除此之外所有由 TS 形成的推荐路线要好于地面实况接取路线。这正解释了出租车司机有经验找到最短路径，但在接取乘客上经验匮乏。

为了清楚地展示推荐路线，让我们看一些**图 14** 中的例子。案例 1 中，由 TS 推荐路段是一些工厂的入口，同时在主路的另一边是一些餐馆。在主路南边是乡村（在中国，因为低劳动力成本，乡村通常距离工厂很近），并且在这儿的推荐是没有意义的。在你案例 2 中，由 GA 推荐的路段穿过高速公路，高时耗，无乘客。这是因为传统的下一跳推荐缺乏一个对真实生活情形的全面视角。这些案例解释了我们的路线 sewing 方式表现要好于传统的下一跳方式。在**图 15** 中，由 TS 推荐的路段是一个社区的入口，其他路段连接社区和主要道路。真实生活中，乘客往往等在社区入口租赁出租车。我们发现一般出租车没有意识到这些情形。这个案例解释了由 TS 形成的推荐确实优于一般的乘客接取方式。

C. 表现

在我们的实验中，我们用一台配置 Intel Core 2 Duo CPU @ 2.53GHz and 4 GB 物理内存的电脑来评估我们的系统。所有实验跑在同一个线程内。正如之前提到的，我们共实施了 90 次实验，对三种算法基于不同的 s，t_s 和 $\hat{t}$。因为 s 是一个任取的变量，我们用基于不同 s，但是相同的t_s和 $\hat{t}$ 的平均实验执行时间作为算法在t_s 和 $\hat{t}$ 上的执行时间。**图 16** 展示了三种算法的具体表现。

较为直观地，我们知道三种算法的执行时间应该是$GA \le TS \le 10-HA$，它和**图 16** 很接近。尽管 TS 比 GA 需要更多的时间去执行，它的执行时间仍旧可以按秒计算，因此我们相信它是可被接受的。

V . 讨论

当我们在探索我们数据时发现几个有趣的事。例如，有两种类型的出租车，一种是红色的，可以被允许在全城驾驶，另一种是绿色的，只允许在郊区通行。这两类出租车的驾驶模式相当不同。绿车趋于拉取更多乘客，而红车趋于拉取少量长途的乘客。选择的理由可能是区域限制。**图 17** 展示了绿红出租车一个月内距离和接取的关系。

另一个有趣的发现是，出租车在接取乘客前后的驾车模式有很大差异。例如，平均驾车速度在接客后会有所提升。此外，尽管驾车路线和接客钱有很多共同之处，但往往不同。这也证明了，司机总是有经验“找路”，无经验“找人”。

VI. 相关工作

路线（轨迹）挖掘：路线相关的挖掘知识已被研究多年。一些工作从运动物体的路线（轨迹）来研究它们的特征；一些工作研究路线（轨迹）分析的技巧，包括校准、分类、聚类、日常侦测等等；一些工作基于这些技能作更深一步分析，像[15],他们用城市计划的观点来看待有瑕疵的路段。在这些领域发现的只是是路线规划的基础。

路线计划：路线规划是利用这些从历史路线发现的知识的一个重要应用，这包括出租车接客的路线推荐[17][3],个性化的路线规划[18],动态路线规划[19][20][21]等等。这些工作既关注确认 POIs，也结合流行的路线片段。在这篇论文中，我们关注确认 POIs 和组合这些 POIs 作为一条连接着的路线。

VII. 结论和后序工作

在这篇论文中，为得到“接取乘客的路线推荐”，从而使出租车司机收益赚取更多的钱，我们构建了一个系统，叫作“**HUNTS**”。为了得到推荐路线，我们研究了一个新问题，叫“全局路线获取优化”。为解决这个问题，我们基于得分，建议一种小说式的线索串联方式。得分参照每一条路段的乘客接取率和平均收益。推荐路线大概是全局优化路线而不是几条 POIs。最后，我们通过传统方式和地面路况的实时数据，评估了我们系统的精度检验和动作性能。

选取路线就像是在赌博，如果相同路段的所有出租车同一时间选取相同路线，那么因为过多的竞争者，乘客接取率就会变的更低。今后，我们将研究出租车竞争的游戏智能，而这也会是一个有挑战性的问题。

VIII. ACKNOWLEDGMENT

This paper was supported in part by Huawei Corp. Under Contract YBCB2009041-27; Hong Kong, Macao and Taiwan Science & Technology Cooperation Program of China under Grant No. 2012DFH10010; Science and Technology Planning Project of Guangzhou, China, under Grant No. 2012Y2-00030;

Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office; and a University Transportation Center grant (DTRT12-G-UTC11) from the US Department of Transportation.

REFERENCES

[1] Taxi number stopped growing for 10 years in beijing. Beijing Daily. [Online]. Available: http://politics.people.com.cn/n/2012/0816/c1001-18753269.html

[2] Taxi number stopped growing for 8 years in changsha. Changsha Evening. [Online]. Available: http://www.voc.com.cn/Topic/article/201109/201109061730026480.html

[3] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "Driving with knowledge from the physical world," in *ACM SIGKDD 2011*.

[4] J. Yuan, Y. Zheng, L. Zhang, X. Xie, and G. Sun, "Where to find my next passenger?" in *UbiComp 2011*, 2011.

[5] A. Ziliaskopoulos and H. Mahmassani, *Time-dependent, shortest-path algorithm for real-time intelligent vehicle highway system applications*, 1993.

[6] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang, "Mapmatching for low-sampling-rate gps trajectories," in *ACM SIGSPATIAL GIS 2009*, 2009.

[7] O. Pink and B. Hummel, "A statistical approach to map matching using road network geometry, topology and vehicular motion constraints," in *IEEE ITSC 2008*, 2008.

[8] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *IEEE ICDE 2001*, 2001.

[9] Q. He, K. Chang, and E.-P. Lim, "Analyzing feature trajectories for event detection," in *ACM SIGIR 2007*.

[10] S. Liu, C. Liu, Q. Luo, L. Ni, and R. Krishnan, "Calibrating large scale vehicle trajectory data," in *IEEE MDM 2012*, 2012.

[11] J. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: trajectory classification using hierarchical region-based and trajectory-based clustering," in *VLDB 2008*, 2008.

[12] S. Liu, Y. Liu, L. M. Ni, J. Fan, and M. Li, "Towards mobility-based clustering," in *ACM SIGKDD 2010*.

[13] J. gil Lee and J. Han, "Trajectory clustering: A partition-and-group framework," in *ACM SIGMOD 2007*, 2007.

[14] J.-G. Lee, J. Han, and X. Li, "Trajectory outlier detection: A partitionand-detect framework," in *IEEE ICDE 2008*.

[15] Y. Zheng, Y. Liu, J. Yuan, and X. Xie, “Urban computing with taxicabs,” in *UbiComp 2011*, 2011.
[16] Y. Ge, H. Xiong, A. Tuzhilin, K. Xiao, M. Gruteser, and M. Pazzani, “An energy-efficient mobile recommender system,” in *ACM SIGKDD 2010*.
[17] J. Yuan, Y. Zheng, C. Zhang, and W. Xie, “T-drive: Driving directions based on taxi trajectories,” in *ACM SIGSPATIAL GIS 2010*.
[18] J. Letchner, J. Krumm, and E. Horvitz, “Trip router with individualized preferences (trip): Incorporating personalization into route planning,” in *AAAI 2006*, 2006.
[19] H. Kanoh and K. Hara, “Hybrid genetic algorithm for dynamic multiobjective route planning with predicted traffic in a real-world road network,” in *GECCO 2008*, 2008.
[20] N. Malviya, S. Madden, and A. Bhattacharya, “A continuous query system for dynamic route planning,” in *IEEE ICDE 2011*, 2011.
[21] J. Xu, L. Guo, Z. Ding, X. Sun, and C. Liu, “Traffic aware route planning in dynamic road networks,” in *DASFAA 2013*, 2012.