Locating Lucrative Passengers for Taxicab Drivers

Haochen Tang
Stanford University
hctang@stanford.edu

Michael Kerber
Stanford University and
MPC-VCC
mkerber@mpi-inf.mpg.de

Qixing Huang
Stanford University
huangqx@stanford.edu

Leonidas Guibas
Stanford University
guibas@stanford.edu

ABSTRACT

In an urban setting, such as the city of Beijing, after a taxi driver drops the previous passenger, he/she needs to decide where to drive to find the next — preferably lucrative — passenger. Different drivers follow different strategies that are mostly based on personal experiences. In this work, we analyze large amounts of GPS location data of taxicabs to compute a high-level profit-maximizing strategy for taxi drivers. Formally, we model the problem of finding a passenger as a Markov Decision Process (MDP) whose parameters are estimated from the GPS data. For this MDP, we compute an optimal policy using dynamic programming. We show that the proposed strategy captures meaningful rules for finding a passenger and we demonstrate that taxi drivers whose behaviors agree with our proposal generate more profit than average drivers.

Categories and Subject Descriptors

G.3 [**Probability and Statistics**]: Markov processes; I.6.5 [**Simulation and Modeling**]: Model Development

General Terms

Experimentation, Algorithms

Keywords

taxi transportation system, GPS data analysis, Markov Decision Process, optimal passenger-seeking strategy

1. INTRODUCTION

In a big city like Beijing, there are more than 10,000 taxis operating every day, and the majority of taxi passengers find their taxis by standing beside the street and waiting for a vacant one to come by. Hence, for taxicab drivers, every time after they drop their previous passengers, they have to make a decision about where to search for the next passenger. The objective of a driver is to maximize the daily income – a natural strategy is to search within an "attractive" area where the chance of finding a passenger is high. However, just finding *any* passenger is not sufficient: taxi drivers prefer long trips, since they are more profitable. On the other hand, long trips may force the driver into a remote area of the city where finding the following passenger will be difficult. As every taxi driver faces this problem several times per day, he/she develops a — perhaps unconscious — strategy based on personal experience.

Contribution. We pose the question of whether a good strategy for finding a passenger can be computed from data; such a strategy could be used, for instance, as a basic guideline for an inexperienced taxi driver. For that purpose, we model the problem of finding a lucrative passenger as a Markov Decision Process (MDP). All parameters of the MDP are obtained by analyzing a collection of GPS data of 1000 taxis over one month in Beijing, China. We compute an optimal policy for the MDP using dynamic programming; that policy can be represented as a directed acyclic graph (DAG) where an edge from location a to location b represents a recommendation to drive from a to b when looking for a passenger. In particular, a sink in the DAG is a locally optimal area to find a pickup and the taxi should stay at this location. We compute such policies for weekday daylight hours, weekday night hours, and weekends, demonstrating that the policies are changing in a meaningful way. We validate the computed policies using the same GPS data: we identify instances where the search path of the taxi drivers agrees with our proposed policy and show that such instances generate more income than the average trip.

Comparison with related work. Using trajectory data, Yuan et al. [8] learn the taxi parking areas and derive their waiting queue length, picking-up probability and next trip length statistics to provide recommendations. Zheng et al. [9] use a non-homogeneous Poisson process to model the taxi arriving rate at a given location and derive the waiting time given the arriving rate. Both results focus on the problem of which "hotspot" the taxi driver should aim for, ignoring pick-ups that happen outside of these locations. Moreover, they consider the passenger-finding problem as a one-shot process, ignoring the sequential effect of driver behavior. Another branch of research is to understand and compare strategies of taxi drivers directly: Liu et al. [6] try to identify preferred locations of successful drivers depending on the day. Li et al. [4] propose a high-dimensional driving descriptor obtained by analyzing the behavior of drivers depending on time and location and compare it with the income of the driver. In contrast to these approaches, we do not propose a method for analyzing the behavior of different drivers; instead, we "distill" a strategy combining all available data without distinguishing between different drivers. In this way, we can derive a good strategy even if no "smart" driver is present – for instance, if all drivers act randomly, our approach would still find out which (random) decisions were profitable, and combines them into a coherent recommendation.

SIGSPATIAL'13, Nov 05-08 2013, Orlando, FL, USA ACM 978-1-4503-2521-9/13/11. http://dx.doi.org/10.1145/2525314.2525471 .

2. THE DATA

We analyze the trajectory data of 1000 taxis operating in the city of Beijing, China, from May 1–31, 2009.[1] For every taxi and every day, the data consists of a sequence of *GPS points*; every point is a triple (p,t,o), where $p \in \mathbb{R}^2$ is the location of the taxi, t is the time, and o is the flag denoting whether the taxi had a passenger at that time or not. If $o = \text{TRUE}$, we call the taxi *occupied*, and otherwise *vacant*. The average sampling frequency of the data points is about 1 GPS point per minute which is fairly sparse compared to other available GPS data.

We split the sequence of GPS points of a taxi into occupied and vacant subsequences. More precisely, we define a *cruise trip* to be a maximal consecutive subsequence of GPS points such that the taxi is vacant in all except the last GPS point. Likewise, an *occupied trip* is a maximal consecutive subsequence where the taxi is occupied except for the last element. We define the *length* of a cruise or occupied trip to be the Euclidean length of the polyline defined by connecting consecutive elements of the trip. We call the location of the last element of a cruise trip a *pick-up location* and the location of the last element of an occupied trip a *drop-off location*. We call the time difference between the first and last element of a cruise or occupied trip its *duration*.

We preprocess the data by splitting it into cruise trips and occupied trips, and remove all cruise trips that consist of only 2 GPS points or with a duration of more than 2 hours. After that data cleaning, we have about 540,000 cruise trajectories and roughly same number of occupied trajectories.

Although the underlying street network of Bejing is available, we refrain from mapping the GPS trajectories on the street network; the reason is the relative sparsity and noise in the data which turns this task non-trivial; we refer to [5] for a recent algorithm for that problem operating on the same data set.

3. OUR MODEL

We describe the problem of finding a passenger in terms of a single player game. This game has several parameters which we determine by analyzing the (preprocessed) GPS data from Section 2.

Finding a passenger – the game. The game is played on a directed graph $G = (V,E)$ where every vertex represents a set of possible pick-up locations. We emphasize that self-loops in G are allowed. The goal of the game is to maximize the *profit*, initially set to 0. The game is played in iterations, called *rounds*. In every round, the player is located at a certain vertex v. Associated to this vertex is a *pick-up probability* $P_{\text{pu}}(v) \in [0,1]$. With probability $P_{\text{pu}}(v)$, a pick-up will happen in this round; in this case, the player will move to some other vertex in the graph (not necessarily adjacent to v) according to a *transfer probability distribution* $P_{\text{tr}}(v,\cdot)$ associated to v. Let w be the destination vertex. The player receives a *reward* $R(v,w)$ which is added to the profit and proceeds to the next round located at w. If no pick-up happens, the player decides on an *action* which is simply choosing an outgoing edge from v, say to vertex w (note that $w = v$ is possible). The player pays the *cruise cost* $c(v,w)$ (that is, subtracts it from the profit) and proceeds to the next round located at w.

To play the described game, the player has to fix a *policy* $A(v,\cdot)$ for each vertex, which is a probability distribution over all endpoints of outgoing edges from v and denotes the probability that the corresponding edge is used as action. If for every vertex v, there is some vertex w with $A(v,w) = 1$, the policy is called *deterministic*; in that case, we can represent A as a subgraph of G with exactly one outgoing edge per vertex.

[1]The data set is available on request at `http://sensor.ee.tsinghua.edu.cn/datasets.php`

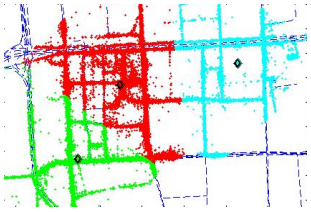

Figure 1: The street map of a part of Beijing. Pick-up locations are drawn as dots, cluster centers as diamonds. Every pick-up is colored according to the cluster that it belongs to.

The game graph. Intuitively, every vertex of the graph represents a region of the city with geometrically related pick-up locations. Although finding such meaningful clusters has been addressed as a research problem in its own [3][2], we use a comparably simple heuristic: we consider the set of all pick-up locations (over all taxi drivers and days) and apply the mean-shift clustering technique from [1] with a bandwidth parameter of 2 km. This results in a set of 195 cluster centers (Fig. 1), after removing clusters with less than 200 pickups. The vertices of G represent these cluster centers; we will from now on identify a cluster, its center, and the corresponding vertex of G. By construction, every pick-up location can be associated to a cluster. We extend this assignment (partially) to GPS locations that are not pick-up locations: let c be the nearest neighbor to a GPS point location p among all cluster centers. If the Euclidean distance $\|p-c\|_2$ is less than 2 km, we say that p *belongs* to c; otherwise, p remains unassigned. We define a *pruned cruise trip* to be a cruise trip after removing all unassigned GPS points. Now, we define the edge set E by the following procedure: start with a complete graph on V and give every edge weight 0. If any pruned cruise trip has two consecutive GPS points such that the first belongs to cluster c_1 and the second belongs to cluster c_2, we increase the weight of (c_1,c_2) by one. Finally, for a cluster c, let Σ_c denote the sum of the weights of all its outgoing edges; if an edge (c,c') has weight at least $0.05 \cdot \Sigma_c$, we add the edge to E. We also include all self-loops (v,v) to E, regardless of their weight. Roughly speaking, we only allow actions which are observed in the data frequently enough. We ignore the weights of the chosen edges in the remainder of the paper.

The pick-up probability. For every cluster, we compute two values: the *number of pickups* and the *total cruise time*. The former is simply the number of pick-up locations that belong to the corresponding cluster. For the latter, we consider two consecutive GPS points g_1 and g_2 of a pruned cruise trip and we let Δ denote the time difference between them. If g_1 and g_2 belong to the same cluster, then Δ is added to the total cruise time of that cluster. If g_1 belongs to c and g_2 belongs to another cluster, then $\Delta/2$ is added to the total cruise time of c.

Now, fix a cluster c and let P and C denote the number of pickups, and the total cruise time, respectively. Moreover, let T be a global parameter which denotes the average length of stay within a vertex. It is not easy to specify T from the data, so we leave it as a free parameter and choose it later in Section 4. Given P, C, and T, the pick-up probability at c is given by

$$P_{\text{pu}}(c) := 1 - \exp\left(-\frac{PT}{C}\right). \tag{1}$$

Further parameters. The remaining parameters are derived from the data in a straight-forward manner: For the transfer prob-

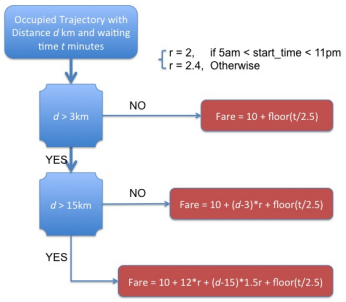

Figure 2: The flow chart for the computation of taxi fares. This is a translation of the (Chinese) description on the regulations of fare prices issued by the city of Beijing `http://www.beijing.gov.cn/ggfw/lyz/cxzn/t662370.htm`**.**

ability distribution, we consider all occupied trips which start in a cluster c. Let n denote the number of such trips. For a cluster c' in G, let $m \leq n$ be the number of occupied trips that started in c and ended in c'. We set $P_{\text{tr}}(c,c') := \frac{m}{n}$.

It remains to define the cruise costs and the reward for a pair of clusters (c,c'). Let d denote the Euclidean distance between the centers. We set the cruise cost to $c(c,c') := c_{\text{fuel}} \cdot d$, where $c_{\text{fuel}} = 0.8$ is a constant reflecting the *fuel cost* per kilometer. For the rewards, recall that the length of a trip is the length of the polyline induced by its GPS locations. Let ℓ denote the length of such a trip and t_{start} denote its starting time. We compute the *profit* as $-c_{\text{fuel}}\ell + F(\ell, t_{\text{start}})$, where the *fare* F is computed according to Figure 2 – we ignore the fare caused by waiting time (i.e., we set $t = 0$ in Fig. 2) because it seems impossible to get a reliable estimate of the waiting time from our sparse trajectory data. We set $R(c,c')$ to be the average profit of all occupied trips from c to c'.

4. RESULTS AND VALIDATION

Having fixed all parameters of our model in Section 3, the question is what is the optimal policy for playing the game. We will describe an algorithm to compute the optimal policy, present its solution when applied to the given data set and present some evidence that the computed policy is helpful in a real-life context.

Computation. We rephrase our game as a Markov Decision Process (MDP) [7, §17.1]: The set of states are the vertices of the game graph and the actions are determined by a policy A. The probability of going from vertex v to vertex w in an iteration is given by

$$P_A(v,w) = P_{\text{pu}}(v)P_{\text{tr}}(v,w) + (1 - P_{\text{pu}}(v))A(v,w).$$

It remains to define the expected immediate reward IR when going from v to w. Let X_{pu} denote the random variable denoting whether a pick-up has happened, and $X_{v\to w}$ be the random variable denoting whether a transition from v to w has happened. Then

$$\begin{aligned}\text{IR}_A(v,w) &= R(v,w)P(X_{pu} \mid X_{v\to w}) - c(v,w)P(\neg X_{pu} \mid X_{v\to w}) \\ &= \frac{R(v,w)P_{\text{pu}}(v)P_{\text{tr}}(v,w) - c(v,w)(1-P_{\text{pu}}(v))A(v,w)}{P_A(v,w)}\end{aligned}$$

With this MDP formulation, we can find the optimal policy with dynamic programming: Assign a value $V(v)$ to every vertex v, initially set to zero. Also, let $v \to v'$ denote a deterministic policy which always uses the edge (v,v') when located in v (note that $v = v'$ is allowed). For every v, we set

$$\pi(v) \leftarrow \operatorname*{arg\,max}_{\{v' \in V' \mid (v,v') \in E\}} \sum_{w \in V} P_{v\to v'}(v,w)(\text{IR}_{v\to v'}(v,w) + \gamma \cdot V(w))$$

with $\gamma < 1$ a *discount factor* chosen to be 0.99 throughout our experiments. Clearly, $\pi(v)$ is the best choice of the player when located at v. That means, π induces a deterministic policy (which we also call π). We update V as

$$V(v) \leftarrow \sum_{w \in V} P_\pi(v,w)(\text{IR}_\pi(v,w) + \gamma \cdot V(w))$$

and repeat until π remains stable (we stop after 2000 iterations).

Results. Applying the approach above to all available data yields an "average" policy that ignores the temporal aspect of the problem: of course, finding a passenger on a Monday morning is different from finding one on a Saturday night, and the best strategies might differ significantly. To mind this differences, we have split the data into three categories, *(weekday) rushhour*, consisting of all data attained Mon-Fri 8am-8pm, *(weekday) offpeak* (Mon-Fri 8pm-8am), and *weekend* (Sat and Sun). More precisely, we computed the 195 clusters using all available data for an easier comparison of the policies; any further parameter (edges of the graph, pick-up probability etc.) is obtained only from the restricted data.

We visualize the computed policies in Fig. 3. Throughout the experiments, we fix T (the parameter that controls the pick-up probability) to 5 min. The optimal policy can be written as a directed graph with exactly one outgoing edge per vertex; if this edge is a self-loop, we call the vertex a *sink*. Except for self-loops, our computed solutions are cycle-free,[2] hence π can be represented as a directed acyclic graph. Clearly, when following π starting at any vertex, we end up in a sink; therefore, we can color the vertices of the graph according to the sinks that they *flow* into. We extend that coloring to the whole plane by assigning a point the color of its nearest vertex (or in other words, we color the Voronoi region of a vertex in the corresponding color). The plots in Fig. 3 show the DAG together with the induced coloring.

In general, we observe the non-surprising fact that all policies recommend to move towards the center when looking for a passenger. An exception is the sink in the upper right corner which is the major airport of the city (compare the street map in Fig. 3). We see that our model recommends to go to the airport during nights and weekends even when located in the north-east part of the city center, while it recommends to stay in the center on weekdays. This makes intuitive sense as one can find passengers more easily in that time and would like to avoid the cost of driving to the airport. Other remarkable differences are the high attraction of the central business district on weekdays compared to nights and weekends, and the increased attraction of the Zhongguancun area during nights. The latter might appear counter-intuitive on a first sight since that area is known for its IT-companies and markets which are closed during the night. However, the area "competes" for taxis against neighboring sites like Peking University which restricts its basin of attraction during business hours; on the other hand, Zhongguancun accommodates more nightlife activities (restaurants, bars) compared to its neighborhood sites which increases its *relative* attractiveness during the night.

Validation. We provide some evidence that it is indeed beneficial for drivers to follow our recommendation. For that, we define a conservative predicate for whether cruise trips agree with our policy and compare the (immediate) reward of "optimal" cruise trips with the reward of an average cruise trip.

Let us start with the test whether a cruise trip agrees with our policy π: consider a pruned cruise trip (i.e., with all unassigned GPS points removed). We represent the cruise trip as a sequence of clusters $(c_{i_1}, \ldots, c_{i_m})$ meaning that the j-th GPS point lies in cluster

[2]This property should be intuitive and can also be proven to hold in general in our setup; we skip the proof in this paper.

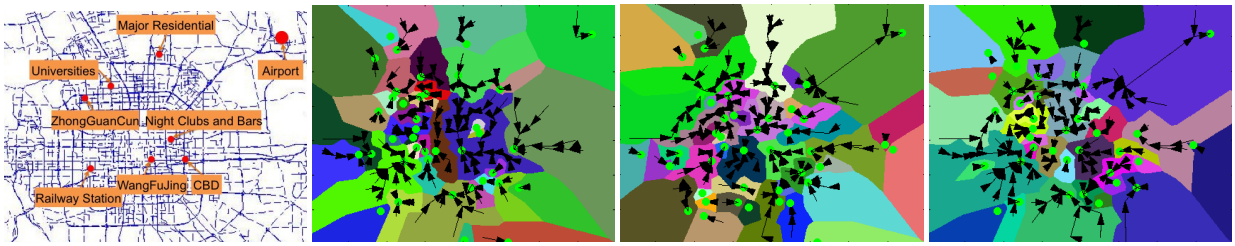

Figure 3: From left to right: A street map of Beijing with annotation of landmark areas for better orientation; the optimal policies for $T = 5\,\text{min}$ rushhour, offpeak, and weekend. In all plots, sinks are displayed as green dots, and regions are colored randomly.

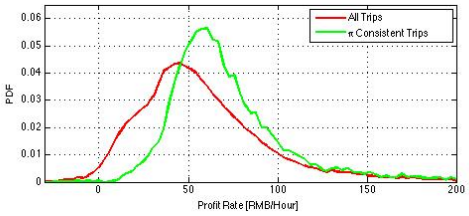

Figure 4: (Normalized) distribution of offpeak trip-pairs (red) and π-consistent offpeak trip-pairs (green) with respect to their profitability (rushhour and weekend results are similar).

i_j. Intuitively, the cruise trip agrees with π if $c_{i_{j+1}} = \pi(c_{i_j})$ for all $j = 1,\ldots,m-1$. However, this will rarely be the case, since clusters represent regions of the city, and the taxi needs time to traverse them, causing several GPS points in the same cluster. For that reason, we use the following heuristic: if the pruned cruise trip contains up to 5 consecutive repetitions of the same cluster, we contract that subsequence to one element (e.g, (c_1,c_2,c_2,c_2,c_3) is transformed to (c_1,c_2,c_3)). We call a cruise trip π-*consistent* if after this contraction, $c_{i_{j+1}} = \pi(c_{i_j})$ holds everywhere. In our data set, about 7% of all trajectories were π-consistent.

To compare π-consistent cruise trip with average cruise trips, consider a trip consisting of a cruise trip and the immediately following occupied trip. Let ℓ_c denote the length of the cruise trip, ℓ_o the length of the occupied trip, t_{start} its starting time, and Δ the sum of the duration of both trips. We define the *profitability* of a pair as

$$\frac{1}{\Delta}\left(-c_{\text{fuel}}(\ell_c+\ell_o)+F(\ell_0,t_{\text{start}})\right)$$

(recall the definitions of c_{fuel} and F from Section 3). We plot the profitability of π-consistent and general trip pairs in Fig. 4 We see that π-consistent trips generally yield higher immediate rewards than average cruise trips. We have to mention, however, that our measurement does not consider the consequences of the occupied trip (that is, how attractive is the drop-off location with respect to the next pickup). Measuring these consequences from the data seems difficult because of the small number of consecutive π-consistent cruise trips.

5. CONCLUSION

We concentrated on the simplistic "waiving model" where passenger wait at the street for a vacant taxi driving by. However, our methodology is still applicable when the locations of currently waiting passengers is known in advance (an assumption that is justified by the existence of mobile phone apps for calling a taxi); still, a driver has to decide whether it is worth to pick up a passenger depending on the distance to that passenger and, if known, the drop-off location of the trip. The main difference is that the policy is not static anymore, but depends on the currently waiting passengers. The setup changes more drastically if we incorporate the interaction between taxi drivers into the model which leads into the realm of game theory.

Besides the application to taxi problems, our work can be seen as an attempt to improve the performance of an individual acting in an unknown environment by examining which actions of other individuals were successful in a comparable setup. The results obtained by our model provide meaningful high-level insights in the underlying process, even though the data set under consideration is sparse and noisy. We pose the question whether this approach could find applications in other areas of mobility analysis.

Acknowledgments.. The authors acknowledge support of ARO grant W911NF-07-2-0027, NSF grants CCF-1011228 and CCF-1161480, a Google Research award and by the Max Planck Center for Visual Computing and Communication. We also thank David Hsu for many helpful discussions.

6. REFERENCES

[1] K. G. Derpanis. Mean shift clustering. Tutorial, 2005.

[2] H. Gonzalez, J. Han, X. Li, M. Myslinska, and J. P. Sondag. Adaptive fastest path computation on a road network: a traffic mining approach. In *Proceedings of the 33rd international conference on very large data bases*, p. 794–805, 2007.

[3] J. Krumm and E. Horvitz. Predestination: Where do you want to go today? *Computer*, 40(4):105–107, 2007.

[4] B. Li, D. Zhang, L. Sun, C. Chen, S. Li, G. Qi, and Q. Yang. Hunting or waiting? Discovering passenger-finding strategies from a large-scale real-world taxi dataset. In *Pervasive Computing and Communications Workshops*, p. 63–68, 2011.

[5] Y. Li, Q. Huang, M. Kerber, L. Zhang, and L. Guibas. Large-scale joint map matching of GPS traces. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013.

[6] L. Liu, C. Andris, A. Biderman, and C. Ratti. Uncovering taxi driver's mobility intelligence through his trace. In *IEEE Pervasive Computing*, 2009.

[7] S. Russell and P. Norvig. *Artifical Intelligence: A Modern Approach*. Prentice Hall, 2009.

[8] N. J. Yuan, Y. Zheng, L. Zhang, and X. Xie. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Trans. Knowledge and Data Engineering (TKDE)*, 2012.

[9] X. Zheng, X. Liang, and K. Xu. Where to wait for a taxi? In *Proceedings of the ACM SIGKDD International Workshop on Urban Computing*, p. 149–156, 2012.